合肥大學
HEFEI UNIVERSITY

Datenbanken

24. Entwicklung

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: 2025-10-10

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

Einleitung

- In unserem einführenden Fabrik-Beispiel hatten wir eine einfache Idee, wie unsere Datenbank aussehen sollten.

Einleitung

- In unserem einführenden Fabrik-Beispiel hatten wir eine einfache Idee, wie unsere Datenbank aussehen sollten.
- Diese Idee haben wir dann einfach implementiert.

Einleitung

- In unserem einführenden Fabrik-Beispiel hatten wir eine einfache Idee, wie unsere Datenbank aussehen sollten.
- Diese Idee haben wir dann einfach implementiert.
- Das ist natürlich nicht, wie so etwas funktioniert.

Einleitung

- In unserem einführenden Fabrik-Beispiel hatten wir eine einfache Idee, wie unsere Datenbank aussehen sollten.
- Diese Idee haben wir dann einfach implementiert.
- Das ist natürlich nicht, wie so etwas funktioniert.
- In der echten Welt sind Datenbanken viel komplizierter.

Einleitung

- In unserem einführenden Fabrik-Beispiel hatten wir eine einfache Idee, wie unsere Datenbank aussehen sollten.
- Diese Idee haben wir dann einfach implementiert.
- Das ist natürlich nicht, wie so etwas funktioniert.
- In der echten Welt sind Datenbanken viel komplizierter.
- Da sind nicht nur drei Tabellen.

Einleitung

- In unserem einführenden Fabrik-Beispiel hatten wir eine einfache Idee, wie unsere Datenbank aussehen sollten.
- Diese Idee haben wir dann einfach implementiert.
- Das ist natürlich nicht, wie so etwas funktioniert.
- In der echten Welt sind Datenbanken viel komplizierter.
- Da sind nicht nur drei Tabellen.
- Die Interaktionen zwischen Objekten, Prozessen, und Menschen sind viel komplizierter.

Einleitung

- In unserem einführenden Fabrik-Beispiel hatten wir eine einfache Idee, wie unsere Datenbank aussehen sollten.
- Diese Idee haben wir dann einfach implementiert.
- Das ist natürlich nicht, wie so etwas funktioniert.
- In der echten Welt sind Datenbanken viel komplizierter.
- Da sind nicht nur drei Tabellen.
- Die Interaktionen zwischen Objekten, Prozessen, und Menschen sind viel komplizierter.
- Wir haben eine grobe Idee, welche technologische Werkzeuge wir zur Verfügung haben, wenn wir mit DBs haben.

Einleitung

- In unserem einführenden Fabrik-Beispiel hatten wir eine einfache Idee, wie unsere Datenbank aussehen sollten.
- Diese Idee haben wir dann einfach implementiert.
- Das ist natürlich nicht, wie so etwas funktioniert.
- In der echten Welt sind Datenbanken viel komplizierter.
- Da sind nicht nur drei Tabellen.
- Die Interaktionen zwischen Objekten, Prozessen, und Menschen sind viel komplizierter.
- Wir haben eine grobe Idee, welche technologische Werkzeuge wir zur Verfügung haben, wenn wir mit DBs haben.
- Aber wir verstehen den praktischen Prozess des Datenbankentwurfs nicht so wirklich.

Einleitung

- In unserem einführenden Fabrik-Beispiel hatten wir eine einfache Idee, wie unsere Datenbank aussehen sollten.
- Diese Idee haben wir dann einfach implementiert.
- Das ist natürlich nicht, wie so etwas funktioniert.
- In der echten Welt sind Datenbanken viel komplizierter.
- Da sind nicht nur drei Tabellen.
- Die Interaktionen zwischen Objekten, Prozessen, und Menschen sind viel komplizierter.
- Wir haben eine grobe Idee, welche technologische Werkzeuge wir zur Verfügung haben, wenn wir mit DBs haben.
- Aber wir verstehen den praktischen Prozess des Datenbankentwurfs nicht so wirklich.
- Uns fehlt das Know-How um ein echtes Datenbankprojekt professionell zu realisieren.

Sinnbild

- Stellen Sie sich vor, dass wir Autos bauen wöllten.

Sinnbild

- Stellen Sie sich vor, dass wir Autos bauen wöllten.
- Wir wissen, was Räder sind und wie sie funktionieren.

Sinnbild

- Stellen Sie sich vor, dass wir Autos bauen wöllten.
- Wir wissen, was Räder sind und wie sie funktionieren.
- Wir wissen, dass die Batterie macht und wie wir mit ihr den Motor zum Laufen bringen.

Sinnbild

- Stellen Sie sich vor, dass wir Autos bauen wöllten.
- Wir wissen, was Räder sind und wie sie funktionieren.
- Wir wissen, dass die Batterie macht und wie wir mit ihr den Motor zum Laufen bringen.
- Wir wissen, dass normale Leute im Auto auf Sitzen sitzen wollen und wie man Sitze einbaut.

Sinnbild

- Stellen Sie sich vor, dass wir Autos bauen wöllten.
- Wir wissen, was Räder sind und wie sie funktionieren.
- Wir wissen, dass die Batterie macht und wie wir mit ihr den Motor zum Laufen bringen.
- Wir wissen, dass normale Leute im Auto auf Sitzen sitzen wollen und wie man Sitze einbaut.
- Wir wissen was ein Fahrgestellt und eine Karosserie sind, wie man sie baut, und wie man die anderen Teile mit ihnen verbindet.

Sinnbild

- Stellen Sie sich vor, dass wir Autos bauen wöllten.
- Wir wissen, was Räder sind und wie sie funktionieren.
- Wir wissen, dass die Batterie macht und wie wir mit ihr den Motor zum Laufen bringen.
- Wir wissen, dass normale Leute im Auto auf Sitzen sitzen wollen und wie man Sitze einbaut.
- Wir wissen was ein Fahrgestellt und eine Karosserie sind, wie man sie baut, und wie man die anderen Teile mit ihnen verbindet.
- Das bedeutet aber noch nicht, dass wir zulassungsfähige Autos bauen können.

Sinnbild

- Stellen Sie sich vor, dass wir Autos bauen wöllten.
- Wir wissen, was Räder sind und wie sie funktionieren.
- Wir wissen, dass die Batterie macht und wie wir mit ihr den Motor zum Laufen bringen.
- Wir wissen, dass normale Leute im Auto auf Sitzen sitzen wollen und wie man Sitze einbaut.
- Wir wissen was ein Fahrgestellt und eine Karosserie sind, wie man sie baut, und wie man die anderen Teile mit ihnen verbindet.
- Das bedeutet aber noch nicht, dass wir zulassungsfähige Autos bauen können.
- Das ist nämlich kein Puzzelspiel, wo man schrittweise Teile zusammensetzt bis nichts mehr übrig ist.

Sinnbild

- Stellen Sie sich vor, dass wir Autos bauen wöllten.
- Wir wissen, was Räder sind und wie sie funktionieren.
- Wir wissen, dass die Batterie macht und wie wir mit ihr den Motor zum Laufen bringen.
- Wir wissen, dass normale Leute im Auto auf Sitzen sitzen wollen und wie man Sitze einbaut.
- Wir wissen was ein Fahrgestellt und eine Karosserie sind, wie man sie baut, und wie man die anderen Teile mit ihnen verbindet.
- Das bedeutet aber noch nicht, dass wir zulassungsfähige Autos bauen können.
- Das ist nämlich kein Puzzelspiel, wo man schrittweise Teile zusammensetzt bis nichts mehr übrig ist.
- Das erfordet einen vernünftig dokumentierten Entwurfsprozess.

Sinnbild

- Stellen Sie sich vor, dass wir Autos bauen wöllten.
- Wir wissen, was Räder sind und wie sie funktionieren.
- Wir wissen, dass die Batterie macht und wie wir mit ihr den Motor zum Laufen bringen.
- Wir wissen, dass normale Leute im Auto auf Sitzen sitzen wollen und wie man Sitze einbaut.
- Wir wissen was ein Fahrgestellt und eine Karosserie sind, wie man sie baut, und wie man die anderen Teile mit ihnen verbindet.
- Das bedeutet aber noch nicht, dass wir zulassungsfähige Autos bauen können.
- Das ist nämlich kein Puzzelspiel, wo man schrittweise Teile zusammensetzt bis nichts mehr übrig ist.
- Das erfordet einen vernünftig dokumentierten Entwurfsprozess.
- Erst machen wir einen klaren Plan machen, was wohin gehört und wann was gemacht wird.

Sinnbild

- Stellen Sie sich vor, dass wir Autos bauen wöllten.
- Wir wissen, was Räder sind und wie sie funktionieren.
- Wir wissen, dass die Batterie macht und wie wir mit ihr den Motor zum Laufen bringen.
- Wir wissen, dass normale Leute im Auto auf Sitzen sitzen wollen und wie man Sitze einbaut.
- Wir wissen was ein Fahrgestellt und eine Karosserie sind, wie man sie baut, und wie man die anderen Teile mit ihnen verbindet.
- Das bedeutet aber noch nicht, dass wir zulassungsfähige Autos bauen können.
- Das ist nämlich kein Puzzelspiel, wo man schrittweise Teile zusammensetzt bis nichts mehr übrig ist.
- Das erfordet einen vernünftig dokumentierten Entwurfsprozess.
- Erst machen wir einen klaren Plan machen, was wohin gehört und wann was gemacht wird.
- Erst nach dem Planen beginnt ein Konstruktionsprozess.

Die Drei Modelle / Schemas

Drei Modelle

- Datenbanken werden in einem methodischen und systematischen Prozess entwickelt.

Drei Modelle

- Datenbanken werden in einem methodischen und systematischen Prozess entwickelt.
- Wir beginnen mit einer Analyse der Anforderungen und arbeiten uns über immer genauere Modelle unserer Applikation vorwärts.

Drei Modelle

- Datenbanken werden in einem methodischen und systematischen Prozess entwickelt.
- Wir beginnen mit einer Analyse der Anforderungen und arbeiten uns über immer genauere Modelle unserer Applikation vorwärts.
- Damit werden wir uns in den folgenden Einheiten beschäftigen.

Drei Modelle

- Datenbanken werden in einem methodischen und systematischen Prozess entwickelt.
- Wir beginnen mit einer Analyse der Anforderungen und arbeiten uns über immer genauere Modelle unserer Applikation vorwärts.
- Damit werden wir uns in den folgenden Einheiten beschäftigen.
- Im Kern erfordert der Prozess des Datenbankdesigns das Entwickeln von drei Modellen (auch Schemas genannt)[23].

Drei Modelle

- Datenbanken werden in einem methodischen und systematischen Prozess entwickelt.
- Wir beginnen mit einer Analyse der Anforderungen und arbeiten uns über immer genauere Modelle unserer Applikation vorwärts.
- Damit werden wir uns in den folgenden Einheiten beschäftigen.
- Im Kern erfordert der Prozess des Datenbankdesigns das Entwickeln von drei Modellen (auch Schemas genannt)[23]:
  1. das konzeptuelle Modell

Drei Modelle

- Datenbanken werden in einem methodischen und systematischen Prozess entwickelt.
- Wir beginnen mit einer Analyse der Anforderungen und arbeiten uns über immer genauere Modelle unserer Applikation vorwärts.
- Damit werden wir uns in den folgenden Einheiten beschäftigen.
- Im Kern erfordert der Prozess des Datenbankdesigns das Entwickeln von drei Modellen (auch Schemas genannt)[23]:
  1. das konzeptuelle Modell,
  2. das logische Modell

Drei Modelle

- Datenbanken werden in einem methodischen und systematischen Prozess entwickelt.
- Wir beginnen mit einer Analyse der Anforderungen und arbeiten uns über immer genauere Modelle unserer Applikation vorwärts.
- Damit werden wir uns in den folgenden Einheiten beschäftigen.
- Im Kern erfordert der Prozess des Datenbankdesigns das Entwickeln von drei Modellen (auch Schemas genannt)[23]:
  1. das konzeptuelle Modell,
  2. das logische Modell und
  3. das physische Modell.

Konzeptuelles Modell

Definition: Konzeptuelles Modell

Das *konzeptuelle Model* (oder auch *konzeputelle Schema*) einer Datenbank ist ein Model der Entitäten aus er echten Welt deren Daten das System speichert sowie die Beziehungen zwischen diesen.

Konzeptuelles Modell

Definition: Konzeptuelles Modell

Das *konzeptuelle Model* (oder auch *konzeputelle Schema*) einer Datenbank ist ein Model der Entitäten aus er echten Welt deren Daten das System speichert sowie die Beziehungen zwischen diesen.

- Das konzeptuelle Modell ist eine abstrakte Sicht auf unserer Szenario.

Konzeptuelles Modell

Definition: Konzeptuelles Modell

Das *konzeptuelle Model* (oder auch *konzeputelle Schema*) einer Datenbank ist ein Model der Entitäten aus er echten Welt deren Daten das System speichert sowie die Beziehungen zwischen diesen.

- Das konzeptuelle Modell ist eine abstrakte Sicht auf unserer Szenario.
- Der Sinn dieses Modells ist es, ein klares Konzept der Applikation bereitzustellen, dass alle am Projekt Beteiligten verstehen können.

Konzeptuelles Modell

Definition: Konzeptuelles Modell

Das *konzeptuelle Model* (oder auch *konzeputelle Schema*) einer Datenbank ist ein Model der Entitäten aus er echten Welt deren Daten das System speichert sowie die Beziehungen zwischen diesen.

- Das konzeptuelle Modell ist eine abstrakte Sicht auf unserer Szenario.
- Der Sinn dieses Modells ist es, ein klares Konzept der Applikation bereitzustellen, dass alle am Projekt Beteiligten verstehen können.
- Dieses Model kann als eine Formalisierung des Daten-bezogenen Teils der Anforderungen gesehen werden.

Konzeptuelles Modell

Definition: Konzeptuelles Modell

Das *konzeptuelle Model* (oder auch *konzeputelle Schema*) einer Datenbank ist ein Model der Entitäten aus er echten Welt deren Daten das System speichert sowie die Beziehungen zwischen diesen.

- Das konzeptuelle Modell ist eine abstrakte Sicht auf unserer Szenario.
- Der Sinn dieses Modells ist es, ein klares Konzept der Applikation bereitzustellen, dass alle am Projekt Beteiligten verstehen können.
- Dieses Model kann als eine Formalisierung des Daten-bezogenen Teils der Anforderungen gesehen werden.
- Es ist auch ein wichtiger Teil der Projektdokumentation.

Konzeptuelles Modell

Definition: Konzeptuelles Modell

Das *konzeptuelle Model* (oder auch *konzeputelle Schema*) einer Datenbank ist ein Model der Entitäten aus er echten Welt deren Daten das System speichert sowie die Beziehungen zwischen diesen.

- Das konzeptuelle Modell ist eine abstrakte Sicht auf unserer Szenario.
- Der Sinn dieses Modells ist es, ein klares Konzept der Applikation bereitzustellen, dass alle am Projekt Beteiligten verstehen können.
- Dieses Model kann als eine Formalisierung des Daten-bezogenen Teils der Anforderungen gesehen werden.
- Es ist auch ein wichtiger Teil der Projektdokumentation.
- Es ist unabhängig von konkreten Datenmodellen und Datenbanktechnologien.

Konzeptuelles Modell

Definition: Konzeptuelles Modell

Das *konzeptuelle Model* (oder auch *konzeputelle Schema*) einer Datenbank ist ein Model der Entitäten aus er echten Welt deren Daten das System speichert sowie die Beziehungen zwischen diesen.

- Das konzeptuelle Modell ist eine abstrakte Sicht auf unserer Szenario.
- Der Sinn dieses Modells ist es, ein klares Konzept der Applikation bereitzustellen, dass alle am Projekt Beteiligten verstehen können.
- Dieses Model kann als eine Formalisierung des Daten-bezogenen Teils der Anforderungen gesehen werden.
- Es ist auch ein wichtiger Teil der Projektdokumentation.
- Es ist unabhängig von konkreten Datenmodellen und Datenbanktechnologien.
- Es stellt alle Entitäten, deren Eigenschaften (Attribute) sowie die Beziehungen zwischen ihnen dar.

Logisches Modell

Definition: Logisches Modell

Das *logische Modell* (oder *logische Schema*) einer Datenbank ist ein Modell der Daten, das auf einer spezifischen Datenbanktechnologie basiert, z. B. relationale Datenbank, hierarchische Datenbank, NoSQL Datenbank, usw.

Logisches Modell

Definition: Logisches Modell

Das *logische Modell* (oder *logische Schema*) einer Datenbank ist ein Modell der Daten, das auf einer spezifischen Datenbanktechnologie basiert, z. B. relationale Datenbank, hierarchische Datenbank, NoSQL Datenbank, usw. Es representiert die Entitäten, ihre Attribute und Beziehungen sowie Einscharänkungen mit konkreten Datentypen in einem strukturierten formalen Format.

Logisches Modell

Definition: Logisches Modell

Das *logische Modell* (oder *logische Schema*) einer Datenbank ist ein Modell der Daten, das auf einer spezifischen Datenbanktechnologie basiert, z. B. relationale Datenbank, hierarchische Datenbank, NoSQL Datenbank, usw. Es representiert die Entitäten, ihre Attribute und Beziehungen sowie Einscharänkungen mit konkreten Datentypen in einem strukturierten formalen Format.

- Das logische Model ist die Ebene, auf der die Benutzer und Applikationen mit der Datenbank interagieren.
- Das konzeptuelle Model wird in eine technschie Spezifikation übersetzt.
- Es kann oft auch in einer formalen Datenbanksprache wie SQL spezifiziert werden.

Logisches Modell

Definition: Logisches Modell

Das *logische Modell* (oder *logische Schema*) einer Datenbank ist ein Modell der Daten, das auf einer spezifischen Datenbanktechnologie basiert, z. B. relationale Datenbank, hierarchische Datenbank, NoSQL Datenbank, usw. Es representiert die Entitäten, ihre Attribute und Beziehungen sowie Einscharänkungen mit konkreten Datentypen in einem strukturierten formalen Format.

- Das logische Model ist die Ebene, auf der die Benutzer und Applikationen mit der Datenbank interagieren.
- Das konzeptuelle Model wird in eine technschie Spezifikation übersetzt.
- Es kann oft auch in einer formalen Datenbanksprache wie SQL spezifiziert werden.
- Es kann auf einem bestimmten DBMS oder auf einer Art von DBMSen basieren.

Physisches Modell

Definition: Physisches Model

Das *physische Modell* (oder *physische Schema* oder *interne Schema*) ist die Spezifikation der konkreten technischen Realisierung des logischen Modells.

Physisches Modell

Definition: Physisches Model

Das *physische Modell* (oder *physische Schema* oder *interne Schema*) ist die Spezifikation der konkreten technischen Realisierung des logischen Modells. Es ist gebunden an ein konkretes DBMS und spezifiziert exakt wie die Daten gespeichert werden und wie auf sie zugegriffen wird.

Physisches Modell

Definition: Physisches Model

Das *physische Modell* (oder *physische Schema* oder *interne Schema*) ist die Spezifikation der konkreten technischen Realisierung des logischen Modells. Es ist gebunden an ein konkretes DBMS und spezifiziert exakt wie die Daten gespeichert werden und wie auf sie zugegriffen wird. Das physische Modell bestimmt die Performanz und die Systemanforderungen, aber nicht die Art, wie Benutzer und Applikationen auf die Daten zugreifen (denn das wird durch das logische Modell definiert)

Physisches Modell

Definition: Physisches Model

Das *physische Modell* (oder *physische Schema* oder *interne Schema*) ist die Spezifikation der konkreten technischen Realisierung des logischen Modells. Es ist gebunden an ein konkretes DBMS und spezifiziert exakt wie die Daten gespeichert werden und wie auf sie zugegriffen wird. Das physische Modell bestimmt die Performanz und die Systemanforderungen, aber nicht die Art, wie Benutzer und Applikationen auf die Daten zugreifen (denn das wird durch das logische Modell definiert)

- In vielen kleineren Datenbankapplikationen kann das logische Modell direkt als physisches Modell genutzt werden.

Physisches Modell

Definition: Physisches Model

Das *physische Modell* (oder *physische Schema* oder *interne Schema*) ist die Spezifikation der konkreten technischen Realisierung des logischen Modells. Es ist gebunden an ein konkretes DBMS und spezifiziert exakt wie die Daten gespeichert werden und wie auf sie zugegriffen wird. Das physische Modell bestimmt die Performanz und die Systemanforderungen, aber nicht die Art, wie Benutzer und Applikationen auf die Daten zugreifen (denn das wird durch das logische Modell definiert)

- In vielen kleineren Datenbankapplikationen kann das logische Modell direkt als physisches Modell genutzt werden.
- Das logische Modell kann in einer Sprache wie SQL definiert werden.

Physisches Modell

Definition: Physisches Model

Das *physische Modell* (oder *physische Schema* oder *interne Schema*) ist die Spezifikation der konkreten technischen Realisierung des logischen Modells. Es ist gebunden an ein konkretes DBMS und spezifiziert exakt wie die Daten gespeichert werden und wie auf sie zugegriffen wird. Das physische Modell bestimmt die Performanz und die Systemanforderungen, aber nicht die Art, wie Benutzer und Applikationen auf die Daten zugreifen (denn das wird durch das logische Modell definiert)

- In vielen kleineren Datenbankapplikationen kann das logische Modell direkt als physisches Modell genutzt werden.
- Das logische Modell kann in einer Sprache wie SQL definiert werden.
- In diesem Fall haben wir bereits eine klare Definition, wie Tabellen erstellt und wie mit Anfragen auf sie zugegeriffen wird.

Physisches Modell

Definition: Physisches Model

Das *physische Modell* (oder *physische Schema* oder *interne Schema*) ist die Spezifikation der konkreten technischen Realisierung des logischen Modells. Es ist gebunden an ein konkretes DBMS und spezifiziert exakt wie die Daten gespeichert werden und wie auf sie zugegriffen wird. Das physische Modell bestimmt die Performanz und die Systemanforderungen, aber nicht die Art, wie Benutzer und Applikationen auf die Daten zugreifen (denn das wird durch das logische Modell definiert)

- In vielen kleineren Datenbankapplikationen kann das logische Modell direkt als physisches Modell genutzt werden.
- Das logische Modell kann in einer Sprache wie SQL definiert werden.
- In diesem Fall haben wir bereits eine klare Definition, wie Tabellen erstellt und wie mit Anfragen auf sie zugegeriffen wird.
- Von einem `CREATE TABLE`-Kommando, z. B., können die meisten normalen DBMSe bereits vernünftige Standardeinstellungen ableiten, wie die Daten gespeichert werden sollen.

Physisches Modell

Definition: Physisches Model

Das *physische Modell* (oder *physische Schema* oder *interne Schema*) ist die Spezifikation der konkreten technischen Realisierung des logischen Modells.

- In vielen kleineren Datenbankapplikationen kann das logische Modell direkt als physisches Modell genutzt werden.
- Das logische Modell kann in einer Sprache wie SQL definiert werden.
- In diesem Fall haben wir bereits eine klare Definition, wie Tabellen erstellt und wie mit Anfragen auf sie zugegeriffen wird.
- Von einem `CREATE TABLE`-Kommando, z. B., können die meisten normalen DBMSe bereits vernünftige Standardeinstellungen ableiten, wie die Daten gespeichert werden sollen.
- Um eine hohe Performance zu erreichen, kann ein physisches Modell aber weitere Spezifikationen hinzufügen.

Physisches Modell

Definition: Physisches Model

Das *physische Modell* (oder *physische Schema* oder *interne Schema*) ist die Spezifikation der konkreten technischen Realisierung des logischen Modells.

- In vielen kleineren Datenbankapplikationen kann das logische Modell direkt als physisches Modell genutzt werden.
- Das logische Modell kann in einer Sprache wie SQL definiert werden.
- In diesem Fall haben wir bereits eine klare Definition, wie Tabellen erstellt und wie mit Anfragen auf sie zugegeriffen wird.
- Von einem `CREATE TABLE`-Kommando, z. B., können die meisten normalen DBMSe bereits vernünftige Standardeinstellungen ableiten, wie die Daten gespeichert werden sollen.
- Um eine hohe Performance zu erreichen, kann ein physisches Modell aber weitere Spezifikationen hinzufügen.
- Wie sollen große Datensätze gespeichert werden?

Physisches Modell

Definition: Physisches Model

Das *physische Modell* (oder *physische Schema* oder *interne Schema*) ist die Spezifikation der konkreten technischen Realisierung des logischen Modells.

- Das logische Modell kann in einer Sprache wie SQL definiert werden.
- In diesem Fall haben wir bereits eine klare Definition, wie Tabellen erstellt und wie mit Anfragen auf sie zugegeriffen wird.
- Von einem `CREATE TABLE`-Kommando, z. B., können die meisten normalen DBMSe bereits vernünftige Standardeinstellungen ableiten, wie die Daten gespeichert werden sollen.
- Um eine hohe Performance zu erreichen, kann ein physisches Modell aber weitere Spezifikationen hinzufügen.
- Wie sollen große Datensätze gespeichert werden?
- Sollen die Daten auf eine bestimmte Art sortiert werden?

Physisches Modell

Definition: Physisches Model

Das *physische Modell* (oder *physische Schema* oder *interne Schema*) ist die Spezifikation der konkreten technischen Realisierung des logischen Modells.

- In diesem Fall haben wir bereits eine klare Definition, wie Tabellen erstellt und wie mit Anfragen auf sie zugegeriffen wird.
- Von einem `CREATE TABLE`-Kommando, z. B., können die meisten normalen DBMSe bereits vernünftige Standardeinstellungen ableiten, wie die Daten gespeichert werden sollen.
- Um eine hohe Performance zu erreichen, kann ein physisches Modell aber weitere Spezifikationen hinzufügen.
- Wie sollen große Datensätze gespeichert werden?
- Sollen die Daten auf eine bestimmte Art sortiert werden?
- Soll es Indexe geben, also zusätzliche Datenstruktueren um bestimmte Anfragen zu beschleunigen?

Physisches Modell

Definition: Physisches Model

Das *physische Modell* (oder *physische Schema* oder *interne Schema*) ist die Spezifikation der konkreten technischen Realisierung des logischen Modells.

- Von einem `CREATE TABLE`-Kommando, z. B., können die meisten normalen DBMSe bereits vernünftige Standardeinstellungen ableiten, wie die Daten gespeichert werden sollen.
- Um eine hohe Performance zu erreichen, kann ein physisches Modell aber weitere Spezifikationen hinzufügen.
- Wie sollen große Datensätze gespeichert werden?
- Sollen die Daten auf eine bestimmte Art sortiert werden?
- Soll es Indexe geben, also zusätzliche Datenstruktueren um bestimmte Anfragen zu beschleunigen?
- Diese Spezifikationen sind unsichtbar für Applikationen und Benutzer.

Physisches Modell

Definition: Physisches Model

Das *physische Modell* (oder *physische Schema* oder *interne Schema*) ist die Spezifikation der konkreten technischen Realisierung des logischen Modells.

- Um eine hohe Performance zu erreichen, kann ein physisches Modell aber weitere Spezifikationen hinzufügen.
- Wie sollen große Datensätze gespeichert werden?
- Sollen die Daten auf eine bestimmte Art sortiert werden?
- Soll es Indexe geben, also zusätzliche Datenstruktueren um bestimmte Anfragen zu beschleunigen?
- Diese Spezifikationen sind unsichtbar für Applikationen und Benutzer.
- Sie greifen auf die Datenbank z. B. via SQL zu.

Physisches Modell

Definition: Physisches Model

Das *physische Modell* (oder *physische Schema* oder *interne Schema*) ist die Spezifikation der konkreten technischen Realisierung des logischen Modells.

- Wie sollen große Datensätze gespeichert werden?
- Sollen die Daten auf eine bestimmte Art sortiert werden?
- Soll es Indexe geben, also zusätzliche Datenstruktueren um bestimmte Anfragen zu beschleunigen?
- Diese Spezifikationen sind unsichtbar für Applikationen und Benutzer.
- Sie greifen auf die Datenbank z. B. via SQL zu.
- Ihre Anfragen ändern sich nicht, wenn sich das physische Modell ändert.

Physisches Modell

Definition: Physisches Model

Das *physische Modell* (oder *physische Schema* oder *interne Schema*) ist die Spezifikation der konkreten technischen Realisierung des logischen Modells.

- Sollen die Daten auf eine bestimmte Art sortiert werden?
- Soll es Indexe geben, also zusätzliche Datenstruktueren um bestimmte Anfragen zu beschleunigen?
- Diese Spezifikationen sind unsichtbar für Applikationen und Benutzer.
- Sie greifen auf die Datenbank z. B. via SQL zu.
- Ihre Anfragen ändern sich nicht, wenn sich das physische Modell ändert.
- Das physische Modell beinflusst aber die Anfrage-Performanz und den Speicherbedarf der Daten.

Physisches Modell

Definition: Physisches Model

Das *physische Modell* (oder *physische Schema* oder *interne Schema*) ist die Spezifikation der konkreten technischen Realisierung des logischen Modells.

- Sollen die Daten auf eine bestimmte Art sortiert werden?
- Soll es Indexe geben, also zusätzliche Datenstruktueren um bestimmte Anfragen zu beschleunigen?
- Diese Spezifikationen sind unsichtbar für Applikationen und Benutzer.
- Sie greifen auf die Datenbank z. B. via SQL zu.
- Ihre Anfragen ändern sich nicht, wenn sich das physische Modell ändert.
- Das physische Modell beinflusst aber die Anfrage-Performanz und den Speicherbedarf der Daten.
- Für große Datenbank kann das einen riesigen Unterschied machen.

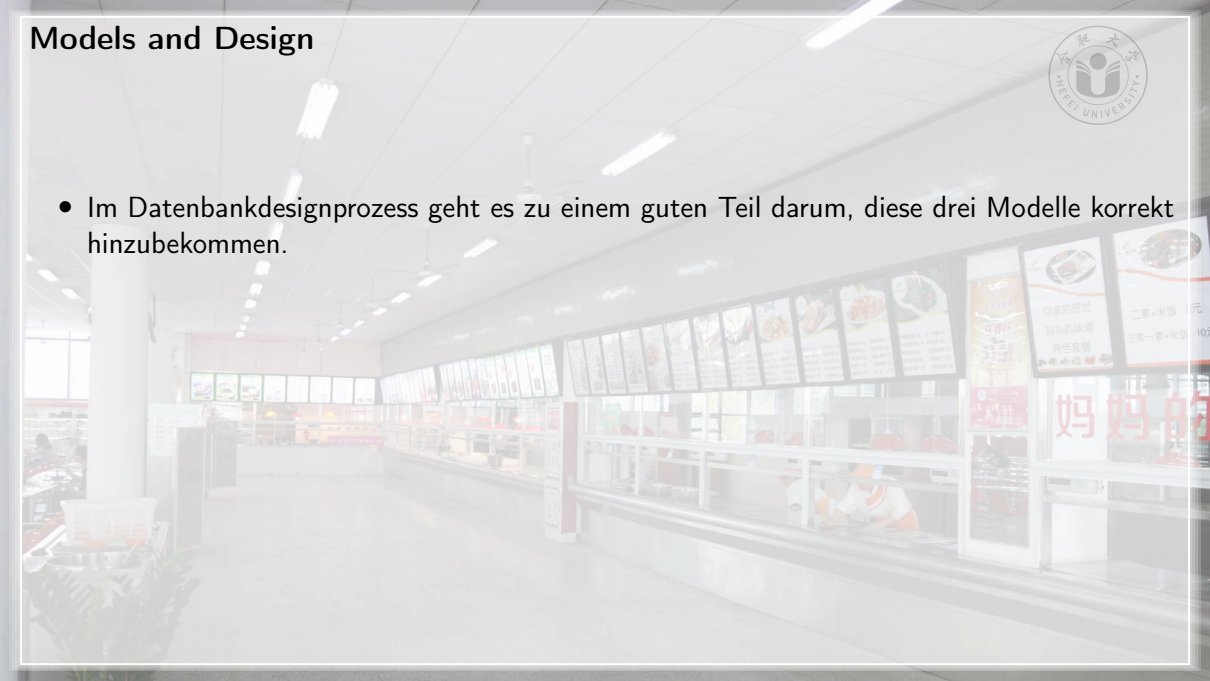
Physisches Modell

Definition: Physisches Model

Das *physische Modell* (oder *physische Schema* oder *interne Schema*) ist die Spezifikation der konkreten technischen Realisierung des logischen Modells.

- Soll es Indexe geben, also zusätzliche Datenstruktueren um bestimmte Anfragen zu beschleunigen?
- Diese Spezifikationen sind unsichtbar für Applikationen und Benutzer.
- Sie greifen auf die Datenbank z. B. via SQL zu.
- Ihre Anfragen ändern sich nicht, wenn sich das physische Modell ändert.
- Das physische Modell beinflusst aber die Anfrage-Performanz und den Speicherbedarf der Daten.
- Für große Datenbank kann das einen riesigen Unterschied machen.

Models and Design



- Im Datenbankdesignprozess geht es zu einem guten Teil darum, diese drei Modelle korrekt hinzubekommen.

Models and Design

- Im Datenbankdesignprozess geht es zu einem guten Teil darum, diese drei Modelle korrekt hinzubekommen.
- Es gibt verschiedene Herausforderungen in jedem Schritt, von der Diskussion mit zukünftigen Benutzern hin bis zum Feineinstellen das DBMS.

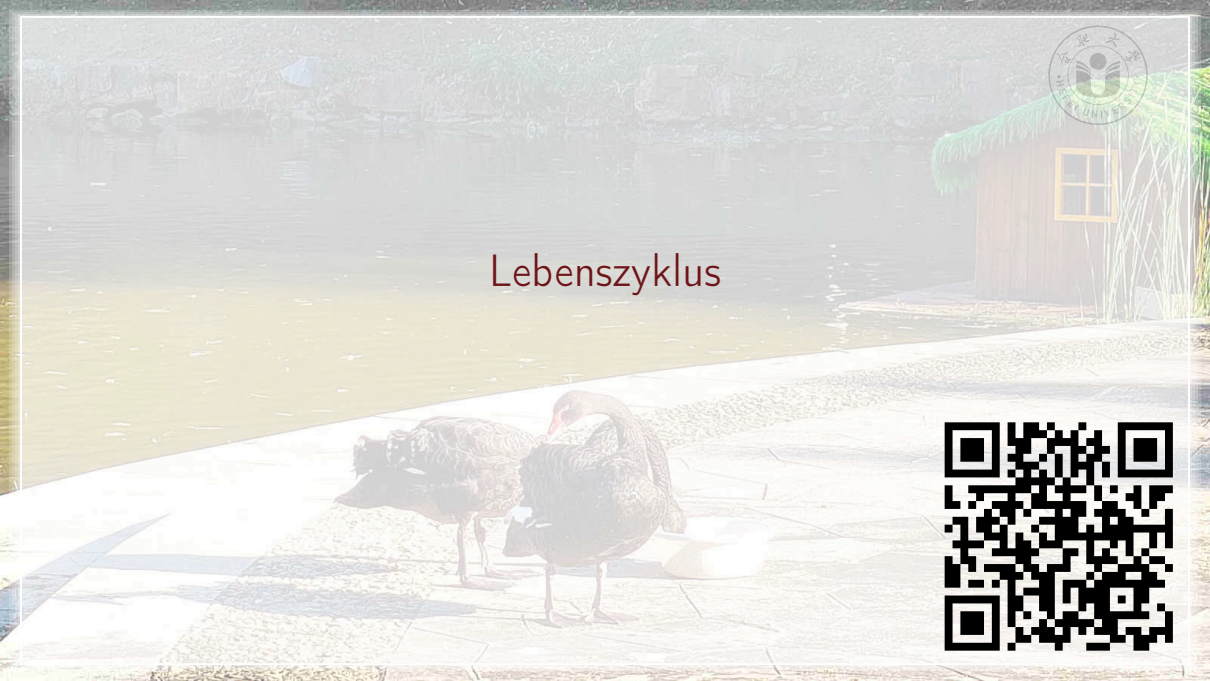
Models and Design

- Im Datenbankdesignprozess geht es zu einem guten Teil darum, diese drei Modelle korrekt hinzubekommen.
- Es gibt verschiedene Herausforderungen in jedem Schritt, von der Diskussion mit zukünftigen Benutzern hin bis zum Feineinstellen das DBMS.
- Wir werden uns jetzt diesen Prozess und die Schritte, die man braucht, um eine verünftige Datenbankapplikation zu erstellen, etwas genauer anschauen.

Models and Design

- Im Datenbankdesignprozess geht es zu einem guten Teil darum, diese drei Modelle korrekt hinzubekommen.
- Es gibt verschiedene Herausforderungen in jedem Schritt, von der Diskussion mit zukünftigen Benutzern hin bis zum Feineinstellen das DBMS.
- Wir werden uns jetzt diesen Prozess und die Schritte, die man braucht, um eine verünftige Datenbankapplikation zu erstellen, etwas genauer anschauen.
- Als übergreifendes Beispiel wollen wir eine Datenbank zum Managen von Studierenden, Lehrkräften, und Kursen an einer Universtität erstellen.

Lebenszyklus



Lebenszyklus von Datenbanken

- Die Entwicklung von Datenbanken ist kein einfacher Prozess der Form „Idee → Design → Fertig“.

Lebenszyklus von Datenbanken

- Die Entwicklung von Datenbanken ist kein einfacher Prozess der Form „Idee → Design → Fertig“.
- Stattdessen benötigt das Entwickeln von Datenbanken mehrere Schritte.

Lebenszyklus von Datenbanken

- Die Entwicklung von Datenbanken ist kein einfacher Prozess der Form „Idee → Design → Fertig“.
- Stattdessen benötigt das Entwickeln von Datenbanken mehrere Schritte.
- Diese Schritte folgen einem *Lebenszyklus*.

Was wir machen müssen

- Wir müssen die Anforderungen an die Datenbank klar verstehen.

Was wir machen müssen

- Wir müssen die Anforderungen an die Datenbank klar verstehen.
- Wir brauchen eine grobe Skizze der Dinge, die wir einbauen müssen.

Was wir machen müssen

- Wir müssen die Anforderungen an die Datenbank klar verstehen.
- Wir brauchen eine grobe Skizze der Dinge, die wir einbauen müssen.
- Wir müssen entscheiden, welche Tabellen und Beziehungen gebraucht werden.

Was wir machen müssen

- Wir müssen die Anforderungen an die Datenbank klar verstehen.
- Wir brauchen eine grobe Skizze der Dinge, die wir einbauen müssen.
- Wir müssen entscheiden, welche Tabellen und Beziehungen gebraucht werden.
- Dieses konzeptionelle Modell muss letztendlich auf ein physisches Modell abgebildet werden.

Was wir machen müssen

- Wir müssen die Anforderungen an die Datenbank klar verstehen.
- Wir brauchen eine grobe Skizze der Dinge, die wir einbauen müssen.
- Wir müssen entscheiden, welche Tabellen und Beziehungen gebraucht werden.
- Dieses konzeptionelle Modell muss letztendlich auf ein physisches Modell abgebildet werden.
- Irgendwann wird man eine Art Prototyp brauchen, wo die Benutzer Testdaten eingeben können, um auszuprobieren, ob alles mehr oder weniger so funktioniert, wie es soll.

Was wir machen müssen

- Wir müssen die Anforderungen an die Datenbank klar verstehen.
- Wir brauchen eine grobe Skizze der Dinge, die wir einbauen müssen.
- Wir müssen entscheiden, welche Tabellen und Beziehungen gebraucht werden.
- Dieses konzeptionelle Modell muss letztendlich auf ein physisches Modell abgebildet werden.
- Irgendwann wird man eine Art Prototyp brauchen, wo die Benutzer Testdaten eingeben können, um auszuprobieren, ob alles mehr oder weniger so funktioniert, wie es soll.
- Dann können wir Probleme feststellen und wir müssen vielleicht unseren Entwurf verändern.

Was wir machen müssen

- Wir müssen die Anforderungen an die Datenbank klar verstehen.
- Wir brauchen eine grobe Skizze der Dinge, die wir einbauen müssen.
- Wir müssen entscheiden, welche Tabellen und Beziehungen gebraucht werden.
- Dieses konzeptionelle Modell muss letztendlich auf ein physisches Modell abgebildet werden.
- Irgendwann wird man eine Art Prototyp brauchen, wo die Benutzer Testdaten eingeben können, um auszuprobieren, ob alles mehr oder weniger so funktioniert, wie es soll.
- Dann können wir Probleme feststellen und wir müssen vielleicht unseren Entwurf verändern.
- Irgendwann muss die fertige Applikation auch getestet werden.

Was wir machen müssen

- Wir müssen die Anforderungen an die Datenbank klar verstehen.
- Wir brauchen eine grobe Skizze der Dinge, die wir einbauen müssen.
- Wir müssen entscheiden, welche Tabellen und Beziehungen gebraucht werden.
- Dieses konzeptionelle Modell muss letztendlich auf ein physisches Modell abgebildet werden.
- Irgendwann wird man eine Art Prototyp brauchen, wo die Benutzer Testdaten eingeben können, um auszuprobieren, ob alles mehr oder weniger so funktioniert, wie es soll.
- Dann können wir Probleme feststellen und wir müssen vielleicht unseren Entwurf verändern.
- Irgendwann muss die fertige Applikation auch getestet werden.
- Wenn dann alle mit dem Design zufrieden sind, können die Datenbank und die Applikation in der Produktivumgebung eingesetzt werden.

Was wir machen müssen

- Wir müssen die Anforderungen an die Datenbank klar verstehen.
- Wir brauchen eine grobe Skizze der Dinge, die wir einbauen müssen.
- Wir müssen entscheiden, welche Tabellen und Beziehungen gebraucht werden.
- Dieses konzeptionelle Modell muss letztendlich auf ein physisches Modell abgebildet werden.
- Irgendwann wird man eine Art Prototyp brauchen, wo die Benutzer Testdaten eingeben können, um auszuprobieren, ob alles mehr oder weniger so funktioniert, wie es soll.
- Dann können wir Probleme feststellen und wir müssen vielleicht unseren Entwurf verändern.
- Irgendwann muss die fertige Applikation auch getestet werden.
- Wenn dann alle mit dem Design zufrieden sind, können die Datenbank und die Applikation in der Produktivumgebung eingesetzt werden.
- Die Datenbank wird regelmäßig gewartet und ge-backup-ed.

Langlebigkeit

- Durch die Langlebigkeit von Datenbanken werden wir irgendwann neue Features hinzufügen und unser Design überarbeiten müssen.

Langlebigkeit

- Durch die Langlebigkeit von Datenbanken werden wir irgendwann neue Features hinzufügen und unser Design überarbeiten müssen.
- Anders als normale Softwareprojekte scheinen Datenbankapplikationen überide Jahre *nicht* komplexer zu werden, sie entwickeln sich aber trotzdem immer weiter[66].

Langlebigkeit

- Durch die Langlebigkeit von Datenbanken werden wir irgendwann neue Features hinzufügen und unser Design überarbeiten müssen.
- Anders als normale Softwareprojekte scheinen Datenbankapplikationen überide Jahre *nicht* komplexer zu werden, sie entwickeln sich aber trotzdem immer weiter[66].
- Es gibt viele verschiedene Ansätze, wie man dem Lebenszyklus von Datenbanken eine Struktur geben kann[29].

Langlebigkeit

- Durch die Langlebigkeit von Datenbanken werden wir irgendwann neue Features hinzufügen und unser Design überarbeiten müssen.
- Anders als normale Softwareprojekte scheinen Datenbankapplikationen überide Jahre *nicht* komplexer zu werden, sie entwickeln sich aber trotzdem immer weiter[66].
- Es gibt viele verschiedene Ansätze, wie man dem Lebenszyklus von Datenbanken eine Struktur geben kann[29].
- Datenbanken sind im Grunde Softwareartefakte, also könnte man einen der vielen Software-Development Lifecycles (SDLC) nehmen[37,50].

Langlebigkeit

- Durch die Langlebigkeit von Datenbanken werden wir irgendwann neue Features hinzufügen und unser Design überarbeiten müssen.
- Anders als normale Softwareprojekte scheinen Datenbankapplikationen überide Jahre *nicht* komplexer zu werden, sie entwickeln sich aber trotzdem immer weiter[66].
- Es gibt viele verschiedene Ansätze, wie man dem Lebenszyklus von Datenbanken eine Struktur geben kann[29].
- Datenbanken sind im Grunde Softwareartefakte, also könnte man einen der vielen Software-Development Lifecycles (SDLC) nehmen[37,50].
- Datenbanken sind aber speziell.

Langlebigkeit

- Durch die Langlebigkeit von Datenbanken werden wir irgendwann neue Features hinzufügen und unser Design überarbeiten müssen.
- Anders als normale Softwareprojekte scheinen Datenbankapplikationen überide Jahre *nicht* komplexer zu werden, sie entwickeln sich aber trotzdem immer weiter[66].
- Es gibt viele verschiedene Ansätze, wie man dem Lebenszyklus von Datenbanken eine Struktur geben kann[29].
- Datenbanken sind im Grunde Softwareartefakte, also könnte man einen der vielen Software-Development Lifecycles (SDLC) nehmen[37,50].
- Datenbanken sind aber speziell.
- Sie haben eine vergleichsweise lange Lebenszeit[60].

Langlebigkeit

- Durch die Langlebigkeit von Datenbanken werden wir irgendwann neue Features hinzufügen und unser Design überarbeiten müssen.
- Anders als normale Softwareprojekte scheinen Datenbankapplikationen überide Jahre *nicht* komplexer zu werden, sie entwickeln sich aber trotzdem immer weiter[66].
- Es gibt viele verschiedene Ansätze, wie man dem Lebenszyklus von Datenbanken eine Struktur geben kann[29].
- Datenbanken sind im Grunde Softwareartefakte, also könnte man einen der vielen Software-Development Lifecycles (SDLC) nehmen[37,50].
- Datenbanken sind aber speziell.
- Sie haben eine vergleichsweise lange Lebenszeit[60].
- Die Hardware eines Computers bleibt für vielleicht drei bis fünf Jahre aktuell.

Langlebigkeit

- Durch die Langlebigkeit von Datenbanken werden wir irgendwann neue Features hinzufügen und unser Design überarbeiten müssen.
- Anders als normale Softwareprojekte scheinen Datenbankapplikationen überide Jahre *nicht* komplexer zu werden, sie entwickeln sich aber trotzdem immer weiter[66].
- Es gibt viele verschiedene Ansätze, wie man dem Lebenszyklus von Datenbanken eine Struktur geben kann[29].
- Datenbanken sind im Grunde Softwareartefakte, also könnte man einen der vielen Software-Development Lifecycles (SDLC) nehmen[37,50].
- Datenbanken sind aber speziell.
- Sie haben eine vergleichsweise lange Lebenszeit[60].
- Die Hardware eines Computers bleibt für vielleicht drei bis fünf Jahre aktuell.
- Softwareprogramme ändern sich oft alle fünf bis zehn Jahre.

Langlebigkeit

- Durch die Langlebigkeit von Datenbanken werden wir irgendwann neue Features hinzufügen und unser Design überarbeiten müssen.
- Anders als normale Softwareprojekte scheinen Datenbankapplikationen überide Jahre *nicht* komplexer zu werden, sie entwickeln sich aber trotzdem immer weiter[66].
- Es gibt viele verschiedene Ansätze, wie man dem Lebenszyklus von Datenbanken eine Struktur geben kann[29].
- Datenbanken sind im Grunde Softwareartefakte, also könnte man einen der vielen Software-Development Lifecycles (SDLC) nehmen[37,50].
- Datenbanken sind aber speziell.
- Sie haben eine vergleichsweise lange Lebenszeit[60].
- Die Hardware eines Computers bleibt für vielleicht drei bis fünf Jahre aktuell.
- Softwareprogramme ändern sich oft alle fünf bis zehn Jahre.
- Z. B. kommt alle zwei Jahre eine neue Long-Term Support (LTS)-Version des Ubuntu OS heraus[75].

Langlebigkeit

- Durch die Langlebigkeit von Datenbanken werden wir irgendwann neue Features hinzufügen und unser Design überarbeiten müssen.
- Anders als normale Softwareprojekte scheinen Datenbankapplikationen überide Jahre *nicht* komplexer zu werden, sie entwickeln sich aber trotzdem immer weiter[66].
- Es gibt viele verschiedene Ansätze, wie man dem Lebenszyklus von Datenbanken eine Struktur geben kann[29].
- Datenbanken sind im Grunde Softwareartefakte, also könnte man einen der vielen Software-Development Lifecycles (SDLC) nehmen[37,50].
- Datenbanken sind aber speziell.
- Sie haben eine vergleichsweise lange Lebenszeit[60].
- Die Hardware eines Computers bleibt für vielleicht drei bis fünf Jahre aktuell.
- Softwareprogramme ändern sich oft alle fünf bis zehn Jahre.
- Z. B. kommt alle zwei Jahre eine neue Long-Term Support (LTS)-Version des Ubuntu OS heraus[75].
- LibreOffice veröffentlicht neue Releases alle sechs Monate[74].

Langlebigkeit

- Anders als normale Softwareprojekte scheinen Datenbankapplikationen überide Jahre *nicht* komplexer zu werden, sie entwickeln sich aber trotzdem immer weiter[66].
- Es gibt viele verschiedene Ansätze, wie man dem Lebenszyklus von Datenbanken eine Struktur geben kann[29].
- Datenbanken sind im Grunde Softwareartefakte, also könnte man einen der vielen Software-Development Lifecycles (SDLC) nehmen[37,50].
- Datenbanken sind aber speziell.
- Sie haben eine vergleichsweise lange Lebenszeit[60].
- Die Hardware eines Computers bleibt für vielleicht drei bis fünf Jahre aktuell.
- Softwareprogramme ändern sich oft alle fünf bis zehn Jahre.
- Z. B. kommt alle zwei Jahre eine neue Long-Term Support (LTS)-Version des Ubuntu OS heraus[75].
- LibreOffice veröffentlicht neue Releases alle sechs Monate[74].
- Hauptversionen von Microsoft Windows kommen alle zwei bis sechs Jahre heraus[27].

Langlebigkeit

- Es gibt viele verschiedene Ansätze, wie man dem Lebenszyklus von Datenbanken eine Struktur geben kann[29].
- Datenbanken sind im Grunde Softwareartefakte, also könnte man einen der vielen Software-Development Lifecycles (SDLC) nehmen[37,50].
- Datenbanken sind aber speziell.
- Sie haben eine vergleichsweise lange Lebenszeit[60].
- Die Hardware eines Computers bleibt für vielleicht drei bis fünf Jahre aktuell.
- Softwareprogramme ändern sich oft alle fünf bis zehn Jahre.
- Z. B. kommt alle zwei Jahre eine neue Long-Term Support (LTS)-Version des Ubuntu OS heraus[75].
- LibreOffice veröffentlicht neue Releases alle sechs Monate[74].
- Hauptversionen von Microsoft Windows kommen alle zwei bis sechs Jahre heraus[27].
- Wenn eine neue Softwareversion erscheint, dann fallen die alten Versionen normalerweise in ca. fünf Jahren aus dem Support und müssen ersetzt werden.

Langlebigkeit

- Datenbanken sind im Grunde Softwareartefakte, also könnte man einen der vielen Software-Development Lifecycles (SDLC) nehmen[37,50].
- Datenbanken sind aber speziell.
- Sie haben eine vergleichsweise lange Lebenszeit[60].
- Die Hardware eines Computers bleibt für vielleicht drei bis fünf Jahre aktuell.
- Softwareprogramme ändern sich oft alle fünf bis zehn Jahre.
- Z. B. kommt alle zwei Jahre eine neue Long-Term Support (LTS)-Version des Ubuntu OS heraus[75].
- LibreOffice veröffentlicht neue Releases alle sechs Monate[74].
- Hauptversionen von Microsoft Windows kommen alle zwei bis sechs Jahre heraus[27].
- Wenn eine neue Softwareversion erscheint, dann fallen die alten Versionen normalerweise in ca. fünf Jahren aus dem Support und müssen ersetzt werden.
- Datenbanken bleiben für Dekaden in Benutzung.

Langlebigkeit

- Datenbanken sind aber speziell.
- Sie haben eine vergleichsweise lange Lebenszeit[60].
- Die Hardware eines Computers bleibt für vielleicht drei bis fünf Jahre aktuell.
- Softwareprogramme ändern sich oft alle fünf bis zehn Jahre.
- Z. B. kommt alle zwei Jahre eine neue Long-Term Support (LTS)-Version des Ubuntu OS heraus[75].
- LibreOffice veröffentlicht neue Releases alle sechs Monate[74].
- Hauptversionen von Microsoft Windows kommen alle zwei bis sechs Jahre heraus[27].
- Wenn eine neue Softwareversion erscheint, dann fallen die alten Versionen normalerweise in ca. fünf Jahren aus dem Support und müssen ersetzt werden.
- Datenbanken bleiben für Dekaden in Benutzung.

Langlebigkeit

- Sie haben eine vergleichsweise lange Lebenszeit[60].
- Die Hardware eines Computers bleibt für vielleicht drei bis fünf Jahre aktuell.
- Softwareprogramme ändern sich oft alle fünf bis zehn Jahre.
- Z. B. kommt alle zwei Jahre eine neue Long-Term Support (LTS)-Version des Ubuntu OS heraus[75].
- LibreOffice veröffentlicht neue Releases alle sechs Monate[74].
- Hauptversionen von Microsoft Windows kommen alle zwei bis sechs Jahre heraus[27].
- Wenn eine neue Softwareversion erscheint, dann fallen die alten Versionen normalerweise in ca. fünf Jahren aus dem Support und müssen ersetzt werden.
- Datenbanken bleiben für Dekaden in Benutzung.
- Ich habe selbst vor langer Zeit eine Datenbank entwickelt, die in einer Niederlassung einer mittelgroßen Firma über zehn Jahre in Benutzung blieb.

Langlebigkeit

- Die Hardware eines Computers bleibt für vielleicht drei bis fünf Jahre aktuell.
- Softwareprogramme ändern sich oft alle fünf bis zehn Jahre.
- Z. B. kommt alle zwei Jahre eine neue Long-Term Support (LTS)-Version des Ubuntu OS heraus[75].
- LibreOffice veröffentlicht neue Releases alle sechs Monate[74].
- Hauptversionen von Microsoft Windows kommen alle zwei bis sechs Jahre heraus[27].
- Wenn eine neue Softwareversion erscheint, dann fallen die alten Versionen normalerweise in ca. fünf Jahren aus dem Support und müssen ersetzt werden.
- Datenbanken bleiben für Dekaden in Benutzung.
- Ich habe selbst vor langer Zeit eine Datenbank entwickelt, die in einer Niederlassung einer mittelgroßen Firma über zehn Jahre in Benutzung blieb.
- Datenbanken und die Applikationen auf ihnen sind wichtige Assets einer Organisation.

Langlebigkeit

- Softwareprogramme ändern sich oft alle fünf bis zehn Jahre.
- Z. B. kommt alle zwei Jahre eine neue Long-Term Support (LTS)-Version des Ubuntu OS heraus[75].
- LibreOffice veröffentlicht neue Releases alle sechs Monate[74].
- Hauptversionen von Microsoft Windows kommen alle zwei bis sechs Jahre heraus[27].
- Wenn eine neue Softwareversion erscheint, dann fallen die alten Versionen normalerweise in ca. fünf Jahren aus dem Support und müssen ersetzt werden.
- Datenbanken bleiben für Dekaden in Benutzung.
- Ich habe selbst vor langer Zeit eine Datenbank entwickelt, die in einer Niederlassung einer mittelgroßen Firma über zehn Jahre in Benutzung blieb.
- Datenbanken und die Applikationen auf ihnen sind wichtige Assets einer Organisation.
- Sie beinhalten wichtige Informationen, sowohl historische Daten als auch die Daten, die für die aktuelle Arbeit benötigt werden.

Datenbanken als Spiegel der Realen Welt

- Datenbanken speichern nicht nur Daten, sie spiegeln Entitäten aus der realen Welt wider.

Datenbanken als Spiegel der Realen Welt

- Datenbanken speichern nicht nur Daten, sie spiegeln Entitäten aus der realen Welt wider.
- In unserem kleinen Fabrikbeispiel spiegeln die Daten die Produkte, Kunden, und Interaktionen der Kunden mit der Fabrik wider.

Datenbanken als Spiegel der Realen Welt

- Datenbanken speichern nicht nur Daten, sie spiegeln Entitäten aus der realen Welt wider.
- In unserem kleinen Fabrikbeispiel spiegeln die Daten die Produkte, Kunden, und Interaktionen der Kunden mit der Fabrik wider.
- Alle diese Aspekte können sich mit der Zeit ändern.

Datenbanken als Spiegel der Realen Welt

- Datenbanken speichern nicht nur Daten, sie spiegeln Entitäten aus der realen Welt wider.
- In unserem kleinen Fabrikbeispiel spiegeln die Daten die Produkte, Kunden, und Interaktionen der Kunden mit der Fabrik wider.
- Alle diese Aspekte können sich mit der Zeit ändern.
- Vielleicht wird die Fabrik irgendwann nicht mehr viele Produkte mit jeweils einem eindeutigen, festen Namen verkaufen, sondern stattdessen weniger Produkte, die aber konfigurierbar sind.

Datenbanken als Spiegel der Realen Welt

- Datenbanken speichern nicht nur Daten, sie spiegeln Entitäten aus der realen Welt wider.
- In unserem kleinen Fabrikbeispiel spiegeln die Daten die Produkte, Kunden, und Interaktionen der Kunden mit der Fabrik wider.
- Alle diese Aspekte können sich mit der Zeit ändern.
- Vielleicht wird die Fabrik irgendwann nicht mehr viele Produkte mit jeweils einem eindeutigen, festen Namen verkaufen, sondern stattdessen weniger Produkte, die aber konfigurierbar sind.
- Vielleicht können ja die Kunden irgendwann die Schuhgröße und -Farbei frei konfigurieren.

Datenbanken als Spiegel der Realen Welt

- Datenbanken speichern nicht nur Daten, sie spiegeln Entitäten aus der realen Welt wider.
- In unserem kleinen Fabrikbeispiel spiegeln die Daten die Produkte, Kunden, und Interaktionen der Kunden mit der Fabrik wider.
- Alle diese Aspekte können sich mit der Zeit ändern.
- Vielleicht wird die Fabrik irgendwann nicht mehr viele Produkte mit jeweils einem eindeutigen, festen Namen verkaufen, sondern stattdessen weniger Produkte, die aber konfigurierbar sind.
- Vielleicht können ja die Kunden irgendwann die Schuhgröße und -Farbei frei konfigurieren.
- Das muss dann ja auch irgendwie in die Datenbank rein.

Datenbanken als Spiegel der Realen Welt

- Datenbanken speichern nicht nur Daten, sie spiegeln Entitäten aus der realen Welt wider.
- In unserem kleinen Fabrikbeispiel spiegeln die Daten die Produkte, Kunden, und Interaktionen der Kunden mit der Fabrik wider.
- Alle diese Aspekte können sich mit der Zeit ändern.
- Vielleicht wird die Fabrik irgendwann nicht mehr viele Produkte mit jeweils einem eindeutigen, festen Namen verkaufen, sondern stattdessen weniger Produkte, die aber konfigurierbar sind.
- Vielleicht können ja die Kunden irgendwann die Schuhgröße und -Farbei frei konfigurieren.
- Das muss dann ja auch irgendwie in die Datenbank rein.
- Vielleicht wird die Datenbank auch irgendwann erweitert, so dass auch der Vorrat an Produkten und die Lagerhaltung mit erfasst werden sollen.

Datenbanken als Spiegel der Realen Welt

- Datenbanken speichern nicht nur Daten, sie spiegeln Entitäten aus der realen Welt wider.
- In unserem kleinen Fabrikbeispiel spiegeln die Daten die Produkte, Kunden, und Interaktionen der Kunden mit der Fabrik wider.
- Alle diese Aspekte können sich mit der Zeit ändern.
- Vielleicht wird die Fabrik irgendwann nicht mehr viele Produkte mit jeweils einem eindeutigen, festen Namen verkaufen, sondern stattdessen weniger Produkte, die aber konfigurierbar sind.
- Vielleicht können ja die Kunden irgendwann die Schuhgröße und -Farbei frei konfigurieren.
- Das muss dann ja auch irgendwie in die Datenbank rein.
- Vielleicht wird die Datenbank auch irgendwann erweitert, so dass auch der Vorrat an Produkten und die Lagerhaltung mit erfasst werden sollen.
- Datenbanken existieren also für eine sehr lange Zeit und können sich während ihrer Existenz weiterentwickeln.

Datenbanklebenszyklus

- All das zusammen führt zu vielen Anforderungen an den Lebenszyklus der Datanbankentwicklung.

Datenbanklebenszyklus

- All das zusammen führt zu vielen Anforderungen an den Lebenszyklus der Datanbankentwicklung.
- Dieser Lebenszyklus ist dem normalen SDLC ähnlich.

Datenbanklebenszyklus

- All das zusammen führt zu vielen Anforderungen an den Lebenszyklus der Datanbankentwicklung.
- Dieser Lebenszyklus ist dem normalen SDLC ähnlich.
- Es gibt jedoch zwei Aspekte, die ihn besonders machen.

Datenbanklebenszyklus

- All das zusammen führt zu vielen Anforderungen an den Lebenszyklus der Datanbankentwicklung.
- Dieser Lebenszyklus ist dem normalen SDLC ähnlich.
- Es gibt jedoch zwei Aspekte, die ihn besonders machen
  1. die Langlebigkeit von Datenbanken.

Datenbanklebenszyklus

- All das zusammen führt zu vielen Anforderungen an den Lebenszyklus der Datanbankentwicklung.
- Dieser Lebenszyklus ist dem normalen SDLC ähnlich.
- Es gibt jedoch zwei Aspekte, die ihn besonders machen
  1. die Langlebigkeit von Datenbanken und
  2. der Fakt, das Datenbanken oft die unterste Ebene von Applikationen sind.

Datenbanklebenszyklus

- All das zusammen führt zu vielen Anforderungen an den Lebenszyklus der Datanbankentwicklung.
- Dieser Lebenszyklus ist dem normalen SDLC ähnlich.
- Es gibt jedoch zwei Aspekte, die ihn besonders machen
  1. die Langlebigkeit von Datenbanken und
  2. der Fakt, das Datenbanken oft die unterste Ebene von Applikationen sind.
- Datenbanken sind die Grundlage auf der andere Applikationen mit Berichten, Formularen, und Webseiten entwickelt werden.

Datenbanklebenszyklus

- All das zusammen führt zu vielen Anforderungen an den Lebenszyklus der Datanbankentwicklung.
- Dieser Lebenszyklus ist dem normalen SDLC ähnlich.
- Es gibt jedoch zwei Aspekte, die ihn besonders machen
  1. die Langlebigkeit von Datenbanken und
  2. der Fakt, das Datenbanken oft die unterste Ebene von Applikationen sind.
- Datenbanken sind die Grundlage auf der andere Applikationen mit Berichten, Formularen, und Webseiten entwickelt werden.
- Deshalb sind die erste Kontaktpunkt von Entwicklern und Stakeholdern in Projekten.

Datenbanklebenszyklus

- All das zusammen führt zu vielen Anforderungen an den Lebenszyklus der Datanbankentwicklung.
- Dieser Lebenszyklus ist dem normalen SDLC ähnlich.
- Es gibt jedoch zwei Aspekte, die ihn besonders machen
  1. die Langlebigkeit von Datenbanken und
  2. der Fakt, das Datenbanken oft die unterste Ebene von Applikationen sind.
- Datenbanken sind die Grundlage auf der andere Applikationen mit Berichten, Formularen, und Webseiten entwickelt werden.
- Deshalb sind die erste Kontaktpunkt von Entwicklern und Stakeholdern in Projekten.
- Der Datenbankentwicklungsprozess ist, wo das grundlegende Verständnis der Daten und Prozesse in einer Organisation aufgebaut wird.

Datenbanklebenszyklus

- All das zusammen führt zu vielen Anforderungen an den Lebenszyklus der Datanbankentwicklung.
- Dieser Lebenszyklus ist dem normalen SDLC ähnlich.
- Es gibt jedoch zwei Aspekte, die ihn besonders machen
  1. die Langlebigkeit von Datenbanken und
  2. der Fakt, das Datenbanken oft die unterste Ebene von Applikationen sind.
- Datenbanken sind die Grundlage auf der andere Applikationen mit Berichten, Formularen, und Webseiten entwickelt werden.
- Deshalb sind die erste Kontaktpunkt von Entwicklern und Stakeholdern in Projekten.
- Der Datenbankentwicklungsprozess ist, wo das grundlegende Verständnis der Daten und Prozesse in einer Organisation aufgebaut wird.
- Hier ist die Situation „am unklarsten“.

Datenbanklebenszyklus

- All das zusammen führt zu vielen Anforderungen an den Lebenszyklus der Datanbankentwicklung.
- Dieser Lebenszyklus ist dem normalen SDLC ähnlich.
- Es gibt jedoch zwei Aspekte, die ihn besonders machen
  1. die Langlebigkeit von Datenbanken und
  2. der Fakt, das Datenbanken oft die unterste Ebene von Applikationen sind.
- Datenbanken sind die Grundlage auf der andere Applikationen mit Berichten, Formularen, und Webseiten entwickelt werden.
- Deshalb sind die erste Kontaktpunkt von Entwicklern und Stakeholdern in Projekten.
- Der Datenbankentwicklungsprozess ist, wo das grundlegende Verständnis der Daten und Prozesse in einer Organisation aufgebaut wird.
- Hier ist die Situation „am unklarsten“.
- Hier passieren die meisten Misverständnisse.

Datenbanklebenszyklus

- Dieser Lebenszyklus ist dem normalen SDLC ähnlich.
- Es gibt jedoch zwei Aspekte, die ihn besonders machen
  1. die Langlebigkeit von Datenbanken und
  2. der Fakt, das Datenbanken oft die unterste Ebene von Applikationen sind.
- Datenbanken sind die Grundlage auf der andere Applikationen mit Berichten, Formularen, und Webseiten entwickelt werden.
- Deshalb sind die erste Kontaktpunkt von Entwicklern und Stakeholdern in Projekten.
- Der Datenbankentwicklungsprozess ist, wo das grundlegende Verständnis der Daten und Prozesse in einer Organisation aufgebaut wird.
- Hier ist die Situation „am unklarsten“.
- Hier passieren die meisten Misverständnisse.
- Die offensichtlichste Anforderung an einen Datenbankentwurfsprozess ist, dass er uns bei der Übersetzung der Informationen von den Stakeholdern in eine laufende Datenbankapplikation hilft.

Datenbanklebenszyklus

- Es gibt jedoch zwei Aspekte, die ihn besonders machen
  1. die Langlebigkeit von Datenbanken und
  2. der Fakt, das Datenbanken oft die unterste Ebene von Applikationen sind.
- Datenbanken sind die Grundlage auf der andere Applikationen mit Berichten, Formularen, und Webseiten entwickelt werden.
- Deshalb sind die erste Kontaktpunkt von Entwicklern und Stakeholdern in Projekten.
- Der Datenbankentwicklungsprozess ist, wo das grundlegende Verständnis der Daten und Prozesse in einer Organisation aufgebaut wird.
- Hier ist die Situation „am unklarsten“.
- Hier passieren die meisten Misverständnisse.
- Die offensichtlichste Anforderung an einen Datenbankentwurfsprozess ist, dass er uns bei der Übersetzung der Informationen von den Stakeholdern in eine laufende Datenbankapplikation hilft.
- Er soll uns helfen, das Projekt zu planen, Zeitlinen zu entwickeln, wann und wie welcher Meilenstein erreicht werden kann, wie viel was kostet, usw.

Datenbanklebenszyklus

- Es gibt jedoch zwei Aspekte, die ihn besonders machen
  1. die Langlebigkeit von Datenbanken und
  2. der Fakt, das Datenbanken oft die unterste Ebene von Applikationen sind.
- Datenbanken sind die Grundlage auf der andere Applikationen mit Berichten, Formularen, und Webseiten entwickelt werden.
- Deshalb sind die erste Kontaktpunkt von Entwicklern und Stakeholdern in Projekten.
- Der Datenbankentwicklungsprozess ist, wo das grundlegende Verständnis der Daten und Prozesse in einer Organisation aufgebaut wird.
- Hier ist die Situation „am unklarsten“.
- Hier passieren die meisten Misverständnisse.
- Die offensichtlichste Anforderung an einen Datenbankentwurfsprozess ist, dass er uns bei der Übersetzung der Informationen von den Stakeholdern in eine laufende Datenbankapplikation hilft.
- Er soll uns helfen, das Projekt zu planen, Zeitlinen zu entwickeln, wann und wie welcher Meilenstein erreicht werden kann, wie viel was kostet, usw.
- Aber all diese Dinge werden durch die Unsicherheit beeinflusst.

Unsicherheiten

- Oft sind sich die Stakeholder selbst nicht völlig klar über ihre eigenen Prozesse und Daten.

Unsicherheiten

- Oft sind sich die Stakeholder selbst nicht völlig klar über ihre eigenen Prozesse und Daten.
- Deshalb können wichtige Aspekte bei Diskussionen einfach übersehen werden.

Unsicherheiten

- Oft sind sich die Stakeholder selbst nicht völlig klar über ihre eigenen Prozesse und Daten.
- Deshalb können wichtige Aspekte bei Diskussionen einfach übersehen werden.
- Z. B. sieht die Aussage „*Jedes Produkt hat einen Namen.*“ auf den ersten Blick sehr vernünftig aus.

Unsicherheiten

- Oft sind sich die Stakeholder selbst nicht völlig klar über ihre eigenen Prozesse und Daten.
- Deshalb können wichtige Aspekte bei Diskussionen einfach übersehen werden.
- Z. B. sieht die Aussage *„Jedes Produkt hat einen Namen.“* auf den ersten Blick sehr vernünftig aus.
- Manchmal werden wir aber auch überrascht und folgendes kann passieren: *„Ah, ja klar, wir verwenden verschiedene Produktnamen für verschiedene Kunden.“*

Unsicherheiten

- Oft sind sich die Stakeholder selbst nicht völlig klar über ihre eigenen Prozesse und Daten.
- Deshalb können wichtige Aspekte bei Diskussionen einfach übersehen werden.
- Z. B. sieht die Aussage *„Jedes Produkt hat einen Namen.“* auf den ersten Blick sehr vernünftig aus.
- Manchmal werden wir aber auch überrascht und folgendes kann passieren: *„Ah, ja klar, wir verwenden verschiedene Produktnamen für verschiedene Kunden.“*

Unsicherheiten

- Oft sind sich die Stakeholder selbst nicht völlig klar über ihre eigenen Prozesse und Daten.
- Deshalb können wichtige Aspekte bei Diskussionen einfach übersehen werden.
- Z. B. sieht die Aussage *„Jedes Produkt hat einen Namen.“* auf den ersten Blick sehr vernünftig aus.
- Manchmal werden wir aber auch überrascht und folgendes kann passieren: *„Ah, ja klar, wir verwenden verschiedene Produktnamen für verschiedene Kunden.“*
- Schon ist die Idee einer einzelnen Tabelle für Produkte dahin...

Unsicherheiten

- Oft sind sich die Stakeholder selbst nicht völlig klar über ihre eigenen Prozesse und Daten.
- Deshalb können wichtige Aspekte bei Diskussionen einfach übersehen werden.
- Z. B. sieht die Aussage „*Jedes Produkt hat einen Namen.*“ auf den ersten Blick sehr vernünftig aus.
- Manchmal werden wir aber auch überrascht und folgendes kann passieren: „*Ah, ja klar, wir verwenden verschiedene Produktnamen für verschiedene Kunden.*“
- Schon ist die Idee einer einzelnen Tabelle für Produkte dahin...
- Vielleicht haben Prozesse einen harten Kern wie „*Wir senden das Produkt an die Kunden, nachdem diese bezahlt haben.*“

Unsicherheiten

- Oft sind sich die Stakeholder selbst nicht völlig klar über ihre eigenen Prozesse und Daten.
- Deshalb können wichtige Aspekte bei Diskussionen einfach übersehen werden.
- Z. B. sieht die Aussage *„Jedes Produkt hat einen Namen.“* auf den ersten Blick sehr vernünftig aus.
- Manchmal werden wir aber auch überrascht und folgendes kann passieren: *„Ah, ja klar, wir verwenden verschiedene Produktnamen für verschiedene Kunden.“*
- Schon ist die Idee einer einzelnen Tabelle für Produkte dahin...
- Vielleicht haben Prozesse einen harten Kern wie *„Wir senden das Produkt an die Kunden, nachdem diese bezahlt haben.“*
- Manchmal gibt es aber auch unscharfe Enden, wie *„Nein, diese Firma ist ein sehr guter Kunde. Die können auch später bezahlen. Warum geht das nicht mit Ihrem Programm?“*

Unsicherheiten

- Deshalb können wichtige Aspekte bei Diskussionen einfach übersehen werden.
- Z. B. sieht die Aussage „*Jedes Produkt hat einen Namen.*“ auf den ersten Blick sehr vernünftig aus.
- Manchmal werden wir aber auch überrascht und folgendes kann passieren: „*Ah, ja klar, wir verwenden verschiedene Produktnamen für verschiedene Kunden.*“
- Schon ist die Idee einer einzelnen Tabelle für Produkte dahin...
- Vielleicht haben Prozesse einen harten Kern wie „*Wir senden das Produkt an die Kunden, nachdem diese bezahlt haben.*“
- Manchmal gibt es aber auch unscharfe Enden, wie „*Nein, diese Firma ist ein sehr guter Kunde. Die können auch später bezahlen. Warum geht das nicht mit Ihrem Programm?*“
- Manchmal nehmen die Stakeholders auch einfach an, dass bestimmte Dinge Allgemeinwissen sind und jeder die kennt: „*Wenn wir unsere Ware außerhalb der EU verschicken, dann muss Exportsteuer bezahlt werden. Das weiß doch jeder!*“

Unsicherheiten

- Deshalb können wichtige Aspekte bei Diskussionen einfach übersehen werden.
- Z. B. sieht die Aussage „*Jedes Produkt hat einen Namen.*“ auf den ersten Blick sehr vernünftig aus.
- Manchmal werden wir aber auch überrascht und folgendes kann passieren: „*Ah, ja klar, wir verwenden verschiedene Produktnamen für verschiedene Kunden.*“
- Schon ist die Idee einer einzelnen Tabelle für Produkte dahin...
- Vielleicht haben Prozesse einen harten Kern wie „*Wir senden das Produkt an die Kunden, nachdem diese bezahlt haben.*“
- Manchmal gibt es aber auch unscharfe Enden, wie „*Nein, diese Firma ist ein sehr guter Kunde. Die können auch später bezahlen. Warum geht das nicht mit Ihrem Programm?*“
- Manchmal nehmen die Stakeholders auch einfach an, dass bestimmte Dinge Allgemeinwissen sind und jeder die kennt: „*Wenn wir unsere Ware außerhalb der EU verschicken, dann muss Exportsteuer bezahlt werden. Das weiß doch jeder!*“ ... der Datenbankentwickler weiß das vielleicht nicht...

Unsicherheiten

- Z. B. sieht die Aussage *„Jedes Produkt hat einen Namen.“* auf den ersten Blick sehr vernünftig aus.
- Manchmal werden wir aber auch überrascht und folgendes kann passieren: *„Ah, ja klar, wir verwenden verschiedene Produktnamen für verschiedene Kunden.“*
- Schon ist die Idee einer einzelnen Tabelle für Produkte dahin...
- Vielleicht haben Prozesse einen harten Kern wie *„Wir senden das Produkt an die Kunden, nachdem diese bezahlt haben.“*
- Manchmal gibt es aber auch unscharfe Enden, wie *„Nein, diese Firma ist ein sehr guter Kunde. Die können auch später bezahlen. Warum geht das nicht mit Ihrem Programm?“*
- Manchmal nehmen die Stakeholders auch einfach an, dass bestimmte Dinge Allgemeinwissen sind und jeder die kennt: *„Wenn wir unsere Ware außerhalb der EU verschicken, dann muss Exportsteuer bezahlt werden. Das weiß doch jeder!“* ... der Datenbankentwickler weiß das vielleicht nicht...
- Ein Datenbankdesignprozess sollte es uns erlauben, unser Design schrittweise und in Antwort auf entdeckte Probleme anzupassen[29,82].

Unsicherheiten

- Manchmal werden wir aber auch überrascht und folgendes kann passieren: *„Ah, ja klar, wir verwenden verschiedene Produktnamen für verschiedene Kunden.“*
- Schon ist die Idee einer einzelnen Tabelle für Produkte dahin...
- Vielleicht haben Prozesse einen harten Kern wie *„Wir senden das Produkt an die Kunden, nachdem diese bezahlt haben.“*
- Manchmal gibt es aber auch unscharfe Enden, wie *„Nein, diese Firma ist ein sehr guter Kunde. Die können auch später bezahlen. Warum geht das nicht mit Ihrem Programm?“*
- Manchmal nehmen die Stakeholders auch einfach an, dass bestimmte Dinge Allgemeinwissen sind und jeder die kennt: *„Wenn wir unsere Ware außerhalb der EU verschicken, dann muss Exportsteuer bezahlt werden. Das weiß doch jeder!“* ... der Datenbankentwickler weiß das vielleicht nicht...
- Ein Datenbankdesignprozess sollte es uns erlauben, unser Design schrittweise und in Antwort auf entdeckte Probleme anzupassen[29,82].
- Gleichzeitig sollte verhindern, dass iterative Veränderungen die Projektstruktur immer weiter verändern und das Projekt durch immer neue Features immer weiter vom Originalziel wegdriftet[29,82].

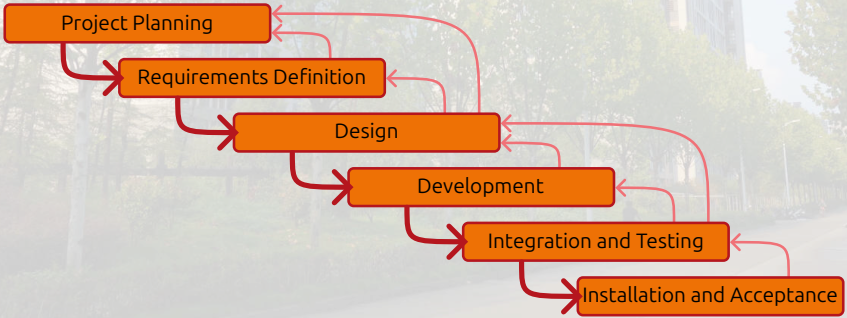
Klassische Softwarentwicklungsprozesse

Wasserfallmodell

- Im Gebiet der Softwaretechnologie wurden mehrere verschiedene Designmethoden, also SDLCs, vorgeschlagen.

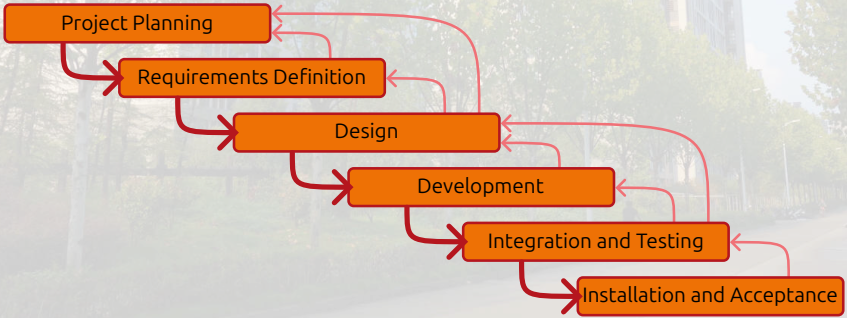
Wasserfallmodell

- Im Gebiet der Softwaretechnologie wurden mehrere verschiedene Designmethoden, also SDLCs, vorgeschlagen.
- Das einfachste ist vielleicht das Wasserfallmodell[29,37,50,54,56] von Royce 1970 veröffentlicht wurde[56].



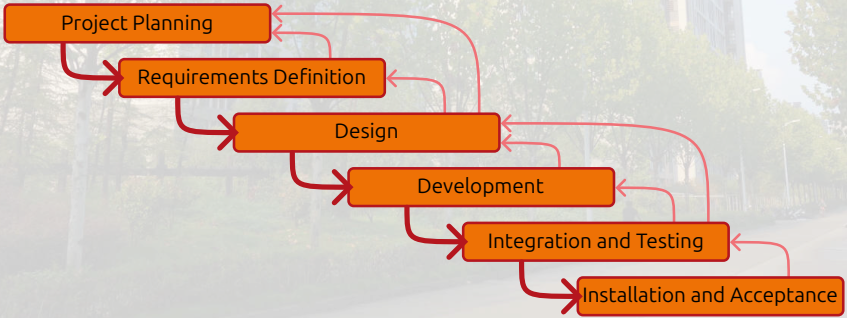
Wasserfallmodell

- Im Gebiet der Softwaretechnologie wurden mehrere verschiedene Designmethoden, also SDLCs, vorgeschlagen.
- Das einfachste ist vielleicht das Wasserfallmodell[29,37,50,54,56] von Royce 1970 veröffentlicht wurde[56].
- Das Wasserfallmodell ist ein sequentieller Prozess aus Planung, Anforderungsanalyse, Design, Entwicklung, Testen, und Installation und produktivem Einsatz.



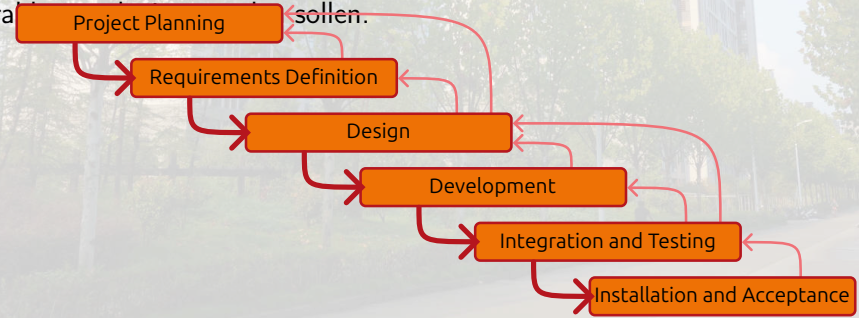
Wasserfallmodell

- Im Gebiet der Softwaretechnologie wurden mehrere verschiedene Designmethoden, also SDLCs, vorgeschlagen.
- Das einfachste ist vielleicht das Wasserfallmodell[29,37,50,54,56] von Royce 1970 veröffentlicht wurde[56].
- Das Wasserfallmodell ist ein sequentieller Prozess aus Planung, Anforderungsanalyse, Design, Entwicklung, Testen, und Installation und produktivem Einsatz.
- Jeder Schritt produziert Artefakte als Ergebnis, die dann die Eingabe für die nächste Phase werden.



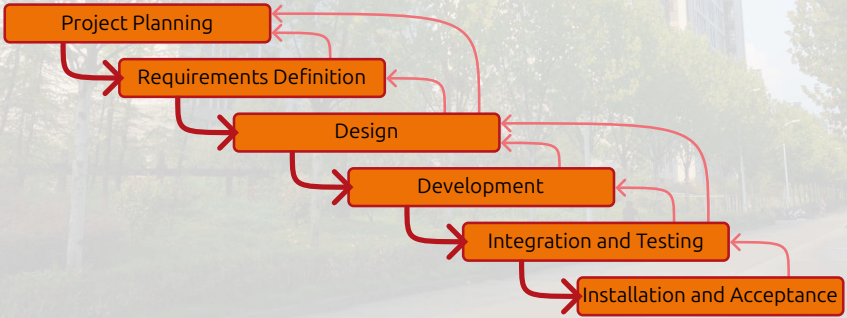
Wasserfallmodell

- Das einfachste ist vielleicht das Wasserfallmodell[29,37,50,54,56] von Royce 1970 veröffentlicht wurde[56].
- Das Wasserfallmodell ist ein sequentieller Prozess aus Planung, Anforderungsanalyse, Design, Entwicklung, Testen, und Installation und produktivem Einsatz.
- Jeder Schritt produziert Artefakte als Ergebnis, die dann die Eingabe für die nächste Phase werden.
- Das Wasserfallmodell legt fest, welche Aktivitäten in welcher Phase stattfinden und welche Delivera sollen.



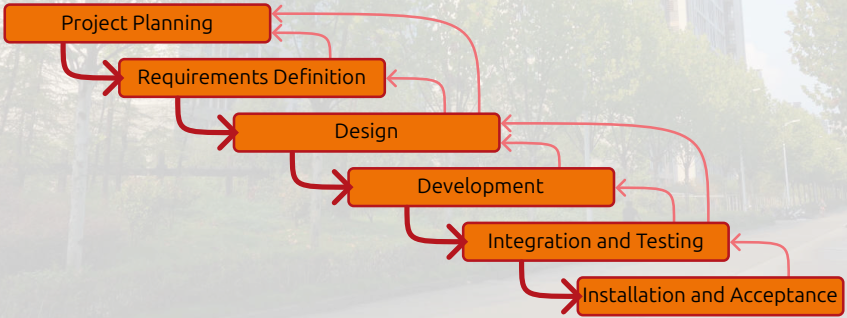
Wasserfallmodell

- Jeder Schritt produziert Artefakte als Ergebnis, die dann die Eingabe für die nächste Phase werden.
- Das Wasserfallmodell legt fest, welche Aktivitäten in welcher Phase stattfinden und welche Deliverables produziert werden sollen.
- Während der Requirements Definition (Anforderungen)-Phase wird der Umfang des Projekts basierend auf Diskussionen mit den Stakeholdern festgelegt.



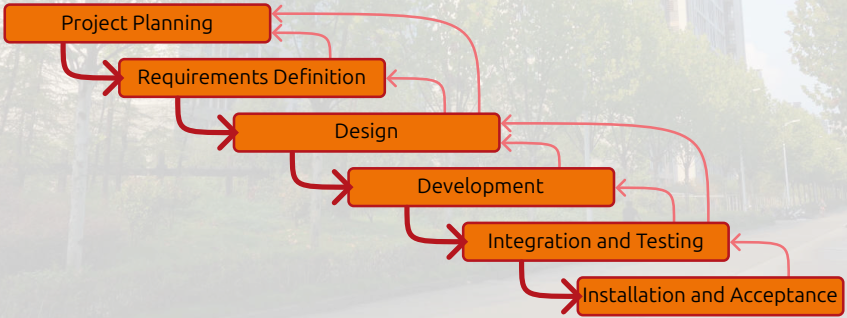
Wasserfallmodell

- Jeder Schritt produziert Artefakte als Ergebnis, die dann die Eingabe für die nächste Phase werden.
- Das Wasserfallmodell legt fest, welche Aktivitäten in welcher Phase stattfinden und welche Deliverables produziert werden sollen.
- Während der Requirements Definition (Anforderungen)-Phase wird der Umfang des Projekts basierend auf Diskussionen mit den Stakeholdern festgelegt.
- Das Wasserfallmodell ist einfach zu verwenden.



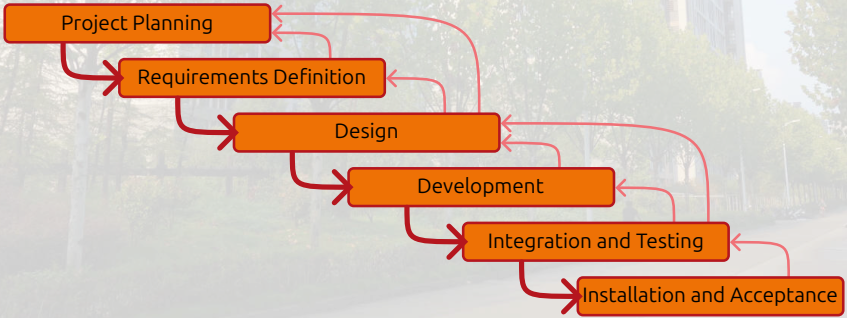
Wasserfallmodell

- Das Wasserfallmodell legt fest, welche Aktivitäten in welcher Phase stattfinden und welche Deliverables produziert werden sollen.
- Während der Requirements Definition (Anforderungen)-Phase wird der Umfang des Projekts basierend auf Diskussionen mit den Stakeholdern festgelegt.
- Das Wasserfallmodell ist einfach zu verwenden.
- Es erlaubt Entwicklern, entlang einer gut bekannten Struktur zu arbeiten.



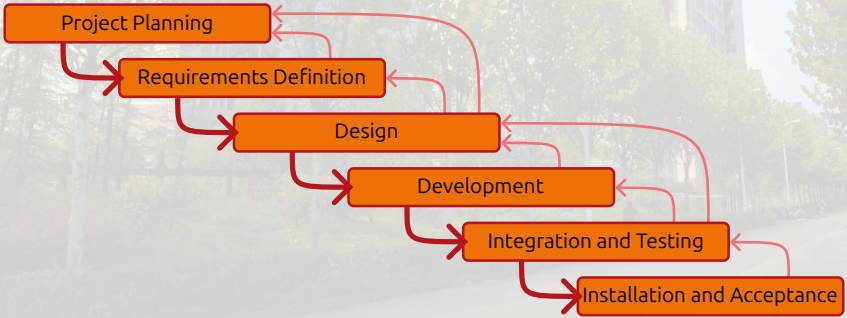
Wasserfallmodell

- Während der Requirements Definition (Anforderungen)-Phase wird der Umfang des Projekts basierend auf Diskussionen mit den Stakeholdern festgelegt.
- Das Wasserfallmodell ist einfach zu verwenden.
- Es erlaubt Entwicklern, entlang einer gut bekannten Struktur zu arbeiten.
- Wir wissen zu jeder Phase des Projekts, wo wir stehen.



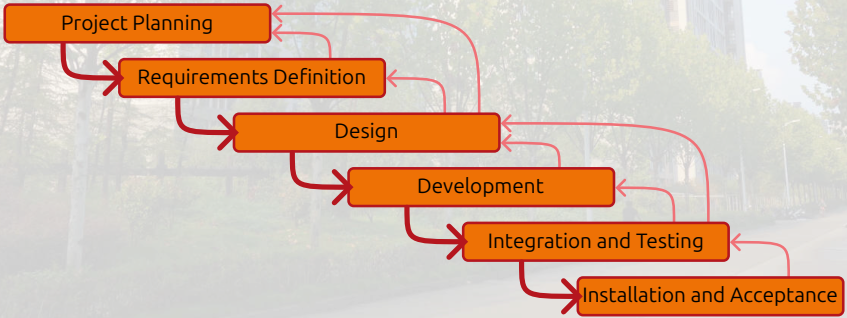
Wasserfallmodell

- Das Wasserfallmodell ist einfach zu verwenden.
- Es erlaubt Entwicklern, entlang einer gut bekannten Struktur zu arbeiten.
- Wir wissen zu jeder Phase des Projekts, wo wir stehen.
- So kommt es zu weniger Missverständnissen.



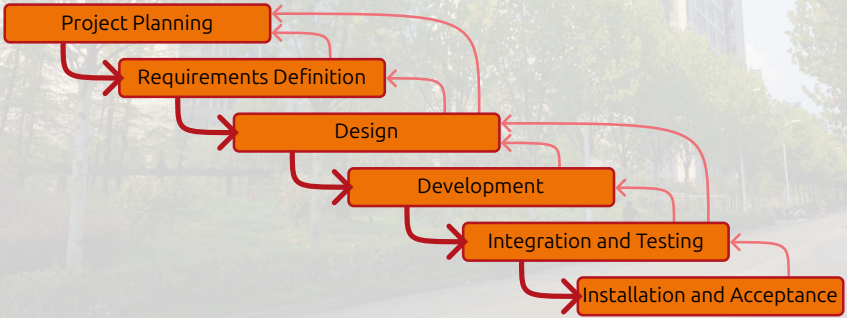
Wasserfallmodell

- Es erlaubt Entwicklern, entlang einer gut bekannten Struktur zu arbeiten.
- Wir wissen zu jeder Phase des Projekts, wo wir stehen.
- So kommt es zu weniger Missverständnissen.
- Dieses Model kann manchal as streng sequentiell missverstanden werden.



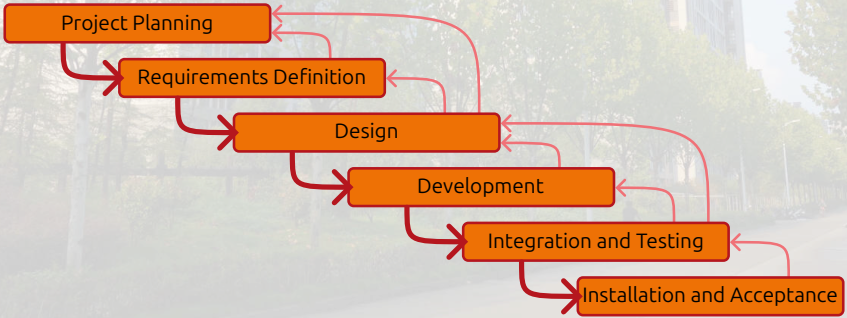
Wasserfallmodell

- Wir wissen zu jeder Phase des Projekts, wo wir stehen.
- So kommt es zu weniger Missverständnissen.
- Dieses Model kann manchal as streng sequentiell missverstanden werden.
- Wenn wir es wirklich streng sequentiell anwenden würden, dann würde wir nicht verstehen, dass wir in jeder Phase fehler machen können und dass es unerwartete Probleme geben kann.



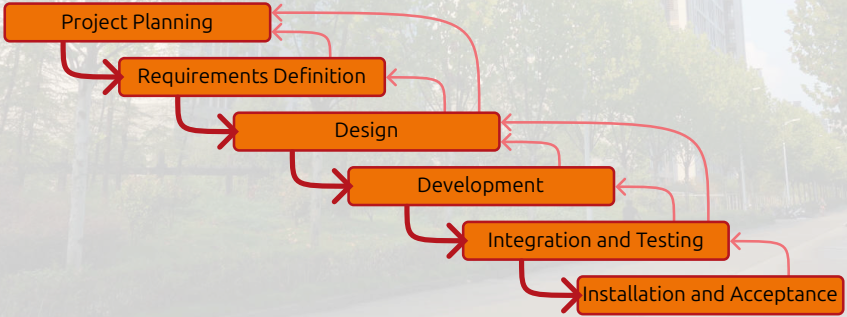
Wasserfallmodell

- So kommt es zu weniger Missverständnissen.
- Dieses Model kann manchal as streng sequentiell missverstanden werden.
- Wenn wir es wirklich streng sequentiell anwenden würden, dann würde wir nicht verstehen, dass wir in jeder Phase fehler machen können und dass es unerwartete Probleme geben kann.
- Royce hat bereits anerkannt, dass es Feedback zwischen der vorigen und der nächsten Phase gibt.



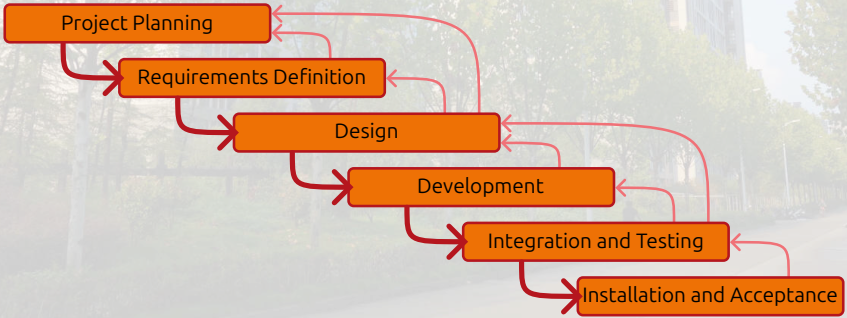
Wasserfallmodell

- Wenn wir es wirklich streng sequentiell anwenden würden, dann würde wir nicht verstehen, dass wir in jeder Phase fehler machen können und dass es unerwartete Probleme geben kann.
- Royce hat bereits anerkannt, dass es Feedback zwischen der vorigen und der nächsten Phase gibt.
- Manchmal brauchen wir auch größere Änderungen, wenn z. B. die Test-Phase nicht das erwartete Ergebnis bringt.



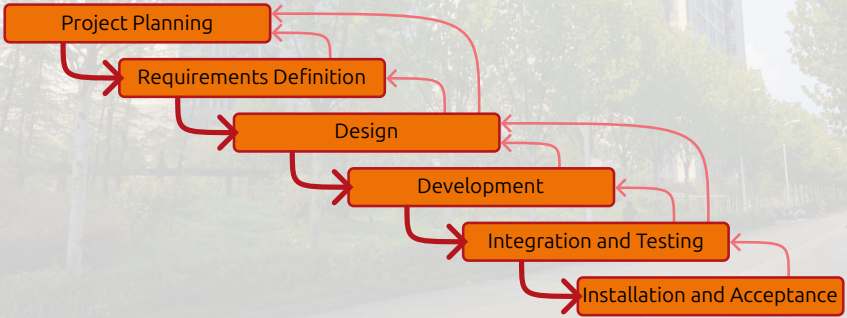
Wasserfallmodell

- Royce hat bereits anerkannt, dass es Feedback zwischen der vorigen und der nächsten Phase gibt.
- Manchmal brauchen wir auch größere Änderungen, wenn z. B. die Test-Phase nicht das erwartete Ergebnis bringt.
- Eine Idee von Royce ist, bei ganz neuen Projekten den gesamten Prozess einfach zweimal zu durchlaufen.



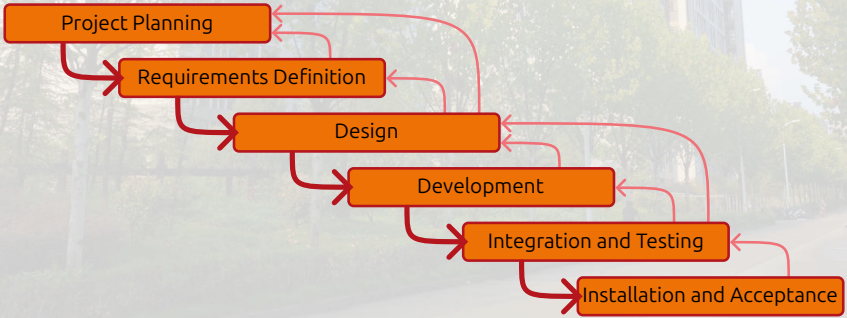
Wasserfallmodell

- Manchmal brauchen wir auch größere Änderungen, wenn z. B. die Test-Phase nicht das erwartete Ergebnis bringt.
- Eine Idee von Royce ist, bei ganz neuen Projekten den gesamten Prozess einfach zweimal zu durchlaufen.
- Der erste Durchlauf wäre eine kurze Simulation der Entwicklung, für die vielleicht ein Viertel der Projektzeit eingesetzt wird.



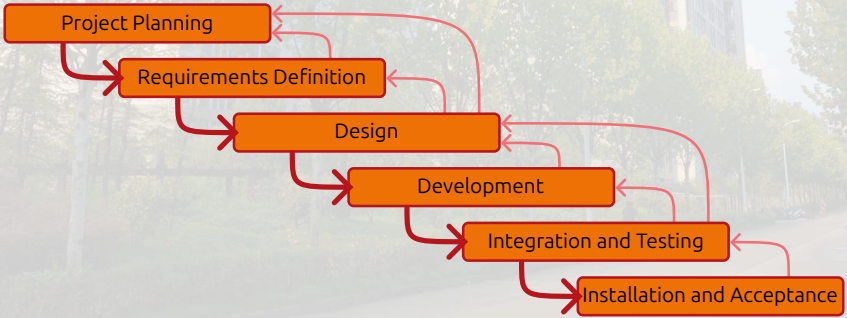
Wasserfallmodell

- Eine Idee von Royce ist, bei ganz neuen Projekten den gesamten Prozess einfach zweimal zu durchlaufen.
- Der erste Durchlauf wäre eine kurze Simulation der Entwicklung, für die vielleicht ein Viertel der Projektzeit eingesetzt wird.
- Der Sinn dieser Phase wäre es, Erfahrung zu sammeln, mit der wir dann den zweiten Durchlauf zum Erfolg bringen können.



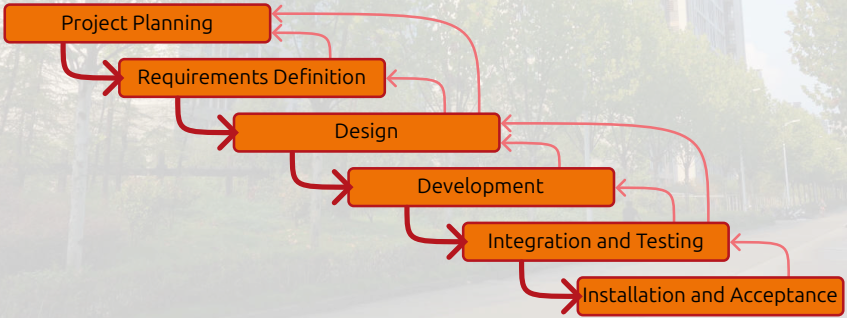
Wasserfallmodell

- Der erste Durchlauf wäre eine kurze Simulation der Entwicklung, für die vielleicht ein Viertel der Projektzeit eingesetzt wird.
- Der Sinn dieser Phase wäre es, Erfahrung zu sammeln, mit der wir dann den zweiten Durchlauf zum Erfolg bringen können.
- Ich mag diese Idee sehr.



Wasserfallmodell

- Der Sinn dieser Phase wäre es, Erfahrung zu sammeln, mit der wir dann den zweiten Durchlauf zum Erfolg bringen können.
- Ich mag diese Idee sehr.
- So oder so, eine Schwachstelle dieses Modells ist es, dass die Anforderungen früh spezifiziert werden und es keinen vordefinierten Weg gibt, später im Projekt Änderungen einzuführen.

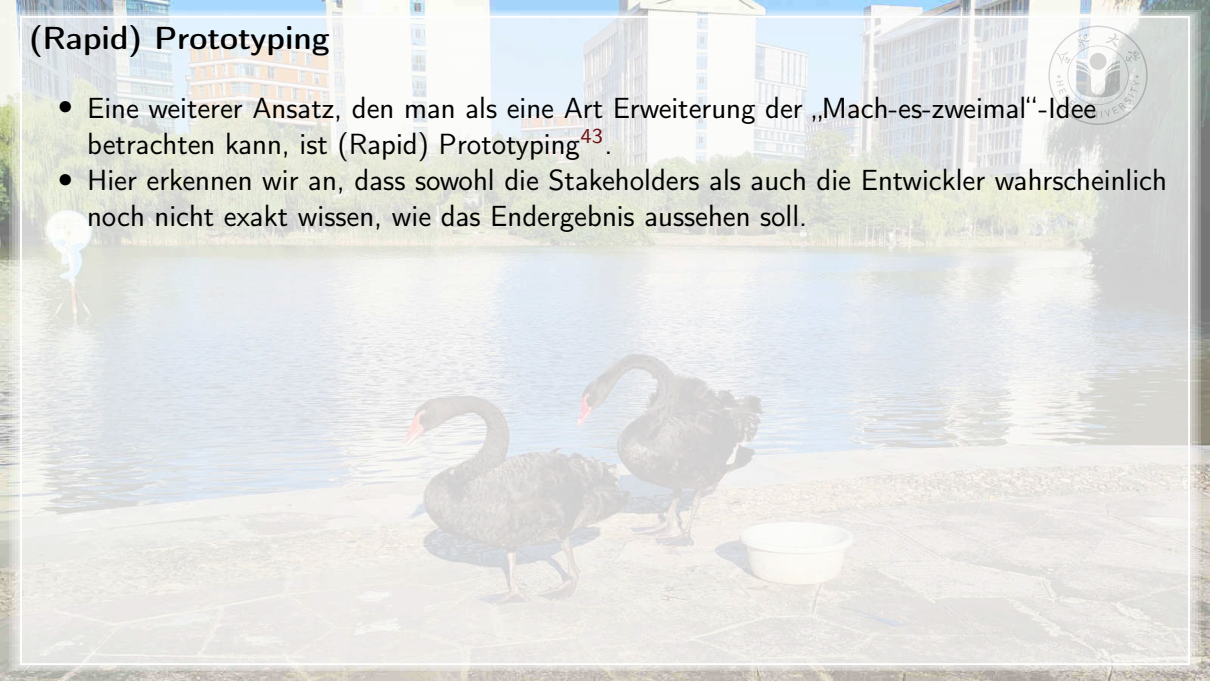


(Rapid) Prototyping

- Eine weiterer Ansatz, den man als eine Art Erweiterung der „Mach-es-zweimal“-Idee betrachten kann, ist (Rapid) Prototyping[43].

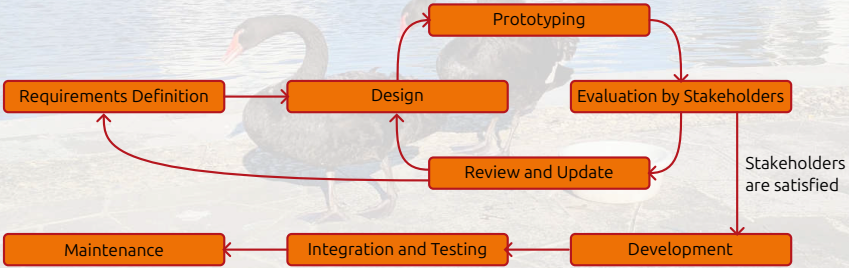
(Rapid) Prototyping

- Eine weiterer Ansatz, den man als eine Art Erweiterung der „Mach-es-zweimal“-Idee betrachten kann, ist (Rapid) Prototyping[43].
- Hier erkennen wir an, dass sowohl die Stakeholders als auch die Entwickler wahrscheinlich noch nicht exakt wissen, wie das Endergebnis aussehen soll.



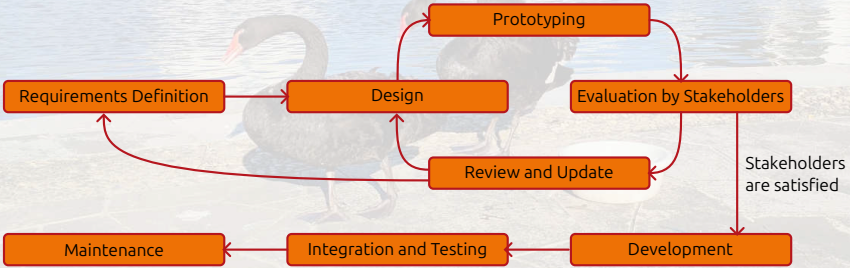
(Rapid) Prototyping

- Eine weiterer Ansatz, den man als eine Art Erweiterung der „Mach-es-zweimal“-Idee betrachten kann, ist (Rapid) Prototyping[43].
- Hier erkennen wir an, dass sowohl die Stakeholders als auch die Entwickler wahrscheinlich noch nicht exakt wissen, wie das Endergebnis aussehen soll.
- Wir platzieren einen Kreislauf aus Prototypen und Diskussionen in den Mittelpunkt des Entwicklungsprozesses[5,43].



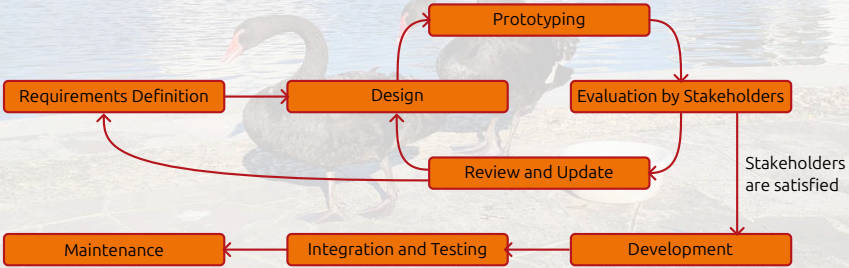
(Rapid) Prototyping

- Eine weiterer Ansatz, den man als eine Art Erweiterung der „Mach-es-zweimal“-Idee betrachten kann, ist (Rapid) Prototyping[43].
- Hier erkennen wir an, dass sowohl die Stakeholders als auch die Entwickler wahrscheinlich noch nicht exakt wissen, wie das Endergebnis aussehen soll.
- Wir platzieren einen Kreislauf aus Prototypen und Diskussionen in den Mittelpunkt des Entwicklungsprozesses[5,43].
- Zuerst wird eine eingeschränkte Version der Anforderungen aufgenommen und ein Prototyp der Datenbank und der Software wird sofort entwickelt.



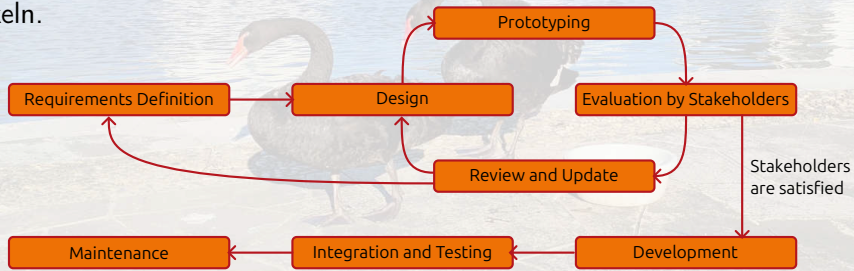
(Rapid) Prototyping

- Wir platzieren einen Kreislauf aus Prototypen und Diskussionen in den Mittelpunkt des Entwicklungsprozesses[5,43].
- Zuerst wird eine eingeschränkte Version der Anforderungen aufgenommen und ein Prototyp der Datenbank und der Software wird sofort entwickelt.
- Der Prototyp kann sogar nur ein Mock-Up sein, das nur die grundlegenden Funktionen des Projekts illustriert und nicht mit anderen Werkzeugen interagiert[5].



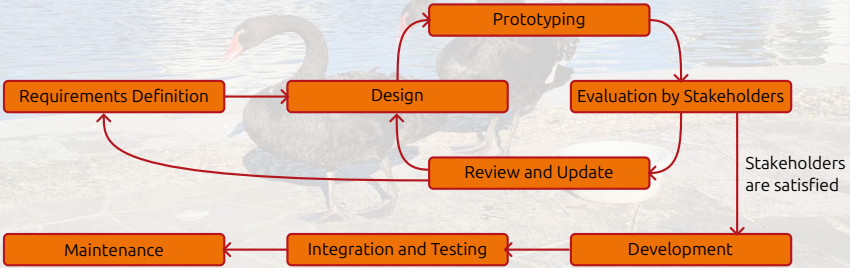
(Rapid) Prototyping

- Wir platzieren einen Kreislauf aus Prototypen und Diskussionen in den Mittelpunkt des Entwicklungsprozesses[5,43].
- Zuerst wird eine eingeschränkte Version der Anforderungen aufgenommen und ein Prototyp der Datenbank und der Software wird sofort entwickelt.
- Der Prototyp kann sogar nur ein Mock-Up sein, das nur die grundlegenden Funktionen des Projekts illustriert und nicht mit anderen Werkzeugen interagiert[5].
- Er dient als Basis für Diskussionen und wir müssen ihn wahrscheinlich mehrfach neu entwickeln.



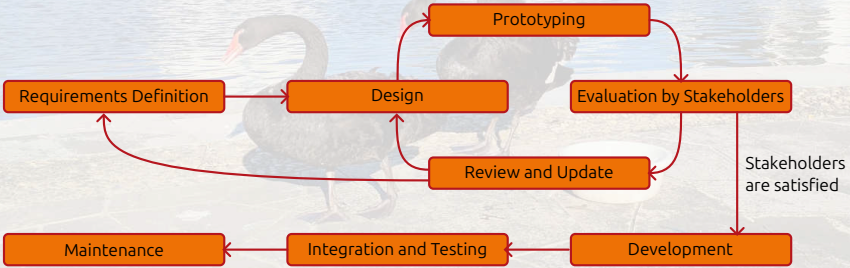
(Rapid) Prototyping

- Der Prototyp kann sogar nur ein Mock-Up sein, das nur die grundlegenden Funktionen des Projekts illustriert und nicht mit anderen Werkzeugen interagiert[5].
- Er dient als Basis für Diskussionen und wir müssen ihn wahrscheinlich mehrfach neu entwickeln.
- In manchen Varianten des Ansatzes werden die Projektanforderungen für festgelegt[29,52], in anderen können sie später bearbeitet werden[43].



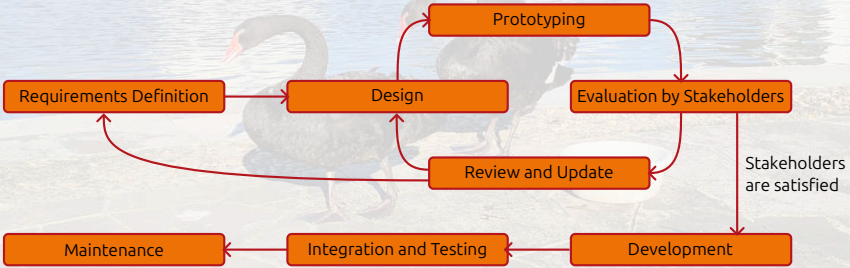
(Rapid) Prototyping

- Der Prototyp kann sogar nur ein Mock-Up sein, das nur die grundlegenden Funktionen des Projekts illustriert und nicht mit anderen Werkzeugen interagiert[5].
- Er dient als Basis für Diskussionen und wir müssen ihn wahrscheinlich mehrfach neu entwickeln.
- In manchen Varianten des Ansatzes werden die Projektanforderungen für festgelegt[29,52], in anderen können sie später bearbeitet werden[43].
- Prototyping erlaubt es uns, schrittweise die Projektspezifikation zu verbessern.



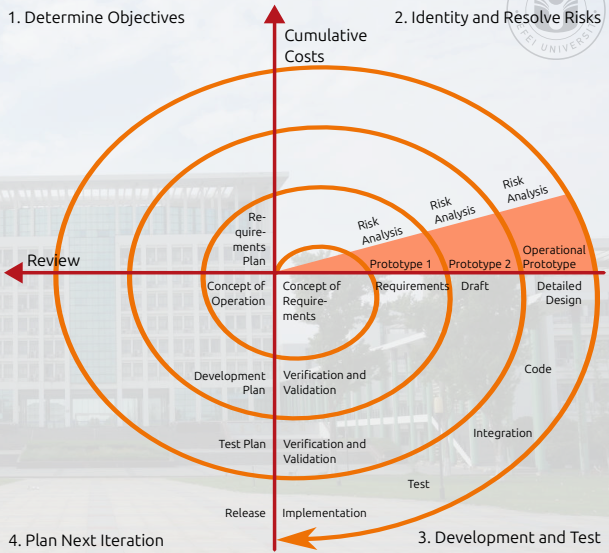
(Rapid) Prototyping

- Er dient als Basis für Diskussionen und wir müssen ihn wahrscheinlich mehrfach neu entwickeln.
- In manchen Varianten des Ansatzes werden die Projektanforderungen für festgelegt[29,52], in anderen können sie später bearbeitet werden[43].
- Prototyping erlaubt es uns, schrittweise die Projektspezifikation zu verbessern.
- Das kostet uns aber auch Klarheit im Projektablauf.



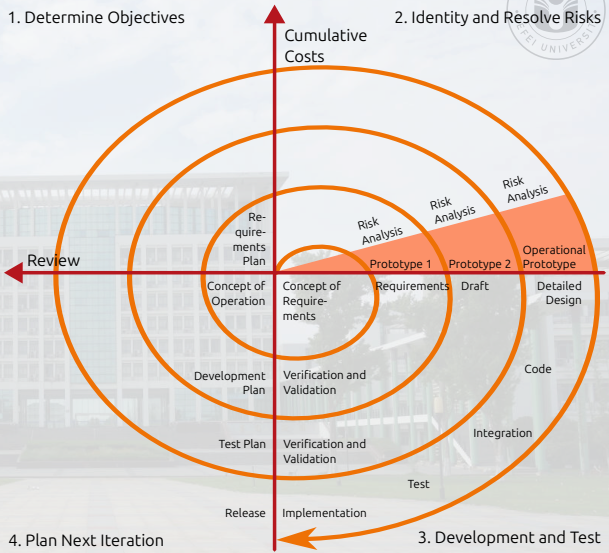
Das Spiralenmodell

- Das Spiralenmodell[7,8,29,52] kann vielleicht als eine weitere Mult-Pass Wasserfallmethode betrachtet werden.



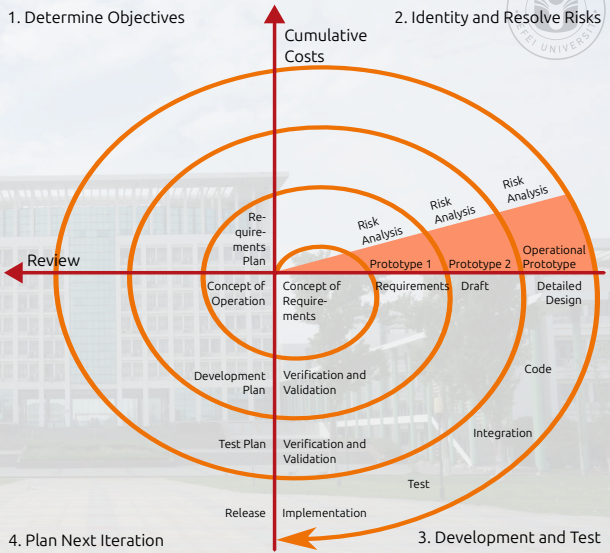
Das Spiralenmodell

- Das Spiralenmodell[7,8,29,52] kann vielleicht als eine weitere Mult-Pass Wasserfallmethode betrachtet werden.
- Der Entwicklungsprozess beginnt wieder mit einer Anforderungsanalyse und einem Entwicklungsplan.



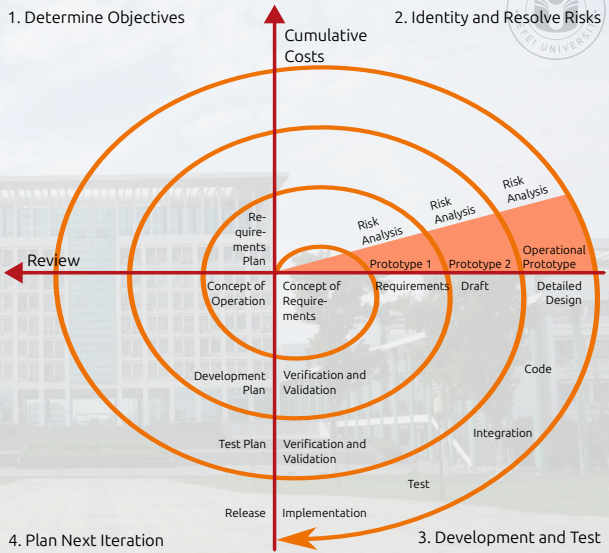
Das Spiralenmodell

- Das Spiralenmodell[7,8,29,52] kann vielleicht als eine weitere Mult-Pass Wasserfallmethode betrachtet werden.
- Der Entwicklungsprozess beginnt wieder mit einer Anforderungsanalyse und einem Entwicklungsplan.
- Ein erste Durchlauf des Wasserfallmodells wird auf Basis einer Untermenge der Anforderungen durchgeführt, um einen ersten Prototyp zu entwickeln.



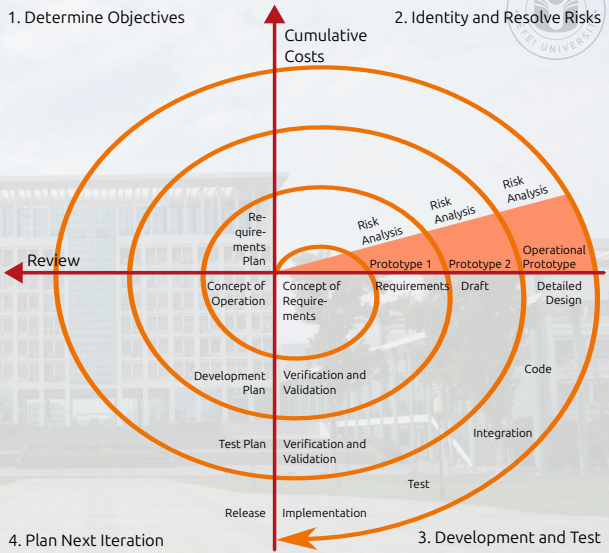
Das Spiralenmodell

- Das Spiralenmodell[7,8,29,52] kann vielleicht als eine weitere Mult-Pass Wasserfallmethode betrachtet werden.
- Der Entwicklungsprozess beginnt wieder mit einer Anforderungsanalyse und einem Entwicklungsplan.
- Ein erste Durchlauf des Wasserfallmodells wird auf Basis einer Untermenge der Anforderungen durchgeführt, um einen ersten Prototyp zu entwickeln.
- Der Prototyp wird ausgewertet und der Zykls beginnt noch einmal, um neue Funktionalität hinzuzufügen und einen neuen Prototypen zu entwickeln.



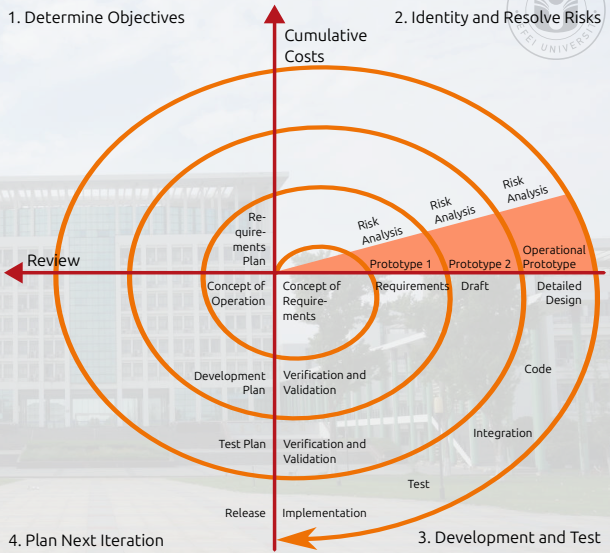
Das Spiralenmodell

- Der Entwicklungsprozess beginnt wieder mit einer Anforderungsanalyse und einem Entwicklungsplan.
- Ein erste Durchlauf des Wasserfallmodells wird auf Basis einer Untermenge der Anforderungen durchgeführt, um einen ersten Prototyp zu entwickeln.
- Der Prototyp wird ausgewertet und der Zykls beginnt noch einmal, um neue Funktionalität hinzuzufügen und einen neuen Prototypen zu entwickeln.
- In jedem Durchlauf wird eine Risikoanalyse durchgeführt, bevor der Prototyp entwickelt wird.



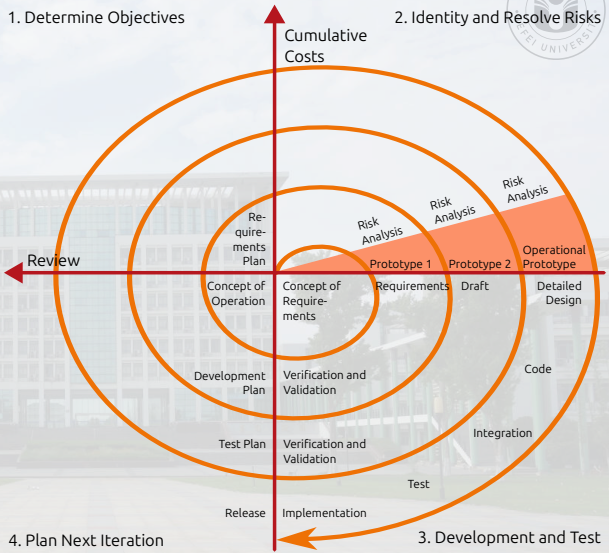
Das Spiralenmodell

- Ein erste Durchlauf des Wasserfallmodells wird auf Basis einer Untermenge der Anforderungen durchgeführt, um einen ersten Prototyp zu entwickeln.
- Der Prototyp wird ausgewertet und der Zykls beginnt noch einmal, um neue Funktionalität hinzuzufügen und einen neuen Prototypen zu entwickeln.
- In jedem Durchlauf wird eine Risikoanalyse durchgeführt, bevor der Prototyp entwickelt wird.
- Jeder Prototyp reduziert also das Risko.



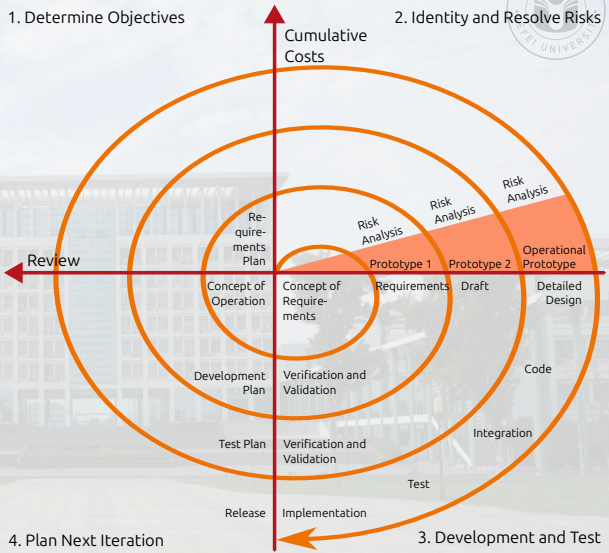
Das Spiralenmodell

- Der Prototyp wird ausgewertet und der Zykls beginnt noch einmal, um neue Funktionalität hinzuzufügen und einen neuen Prototypen zu entwickeln.
- In jedem Durchlauf wird eine Risikoanalyse durchgeführt, bevor der Prototyp entwickelt wird.
- Jeder Prototyp reduziert also das Risko.
- Eine Idee des Spiralmodells ist, das Anforderungen hierarchisch sind.



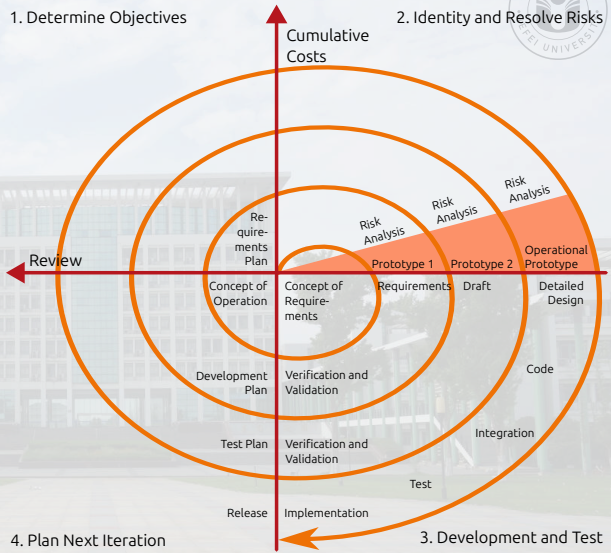
Das Spiralenmodell

- In jedem Durchlauf wird eine Risikoanalyse durchgeführt, bevor der Prototyp entwickelt wird.
- Jeder Prototyp reduziert also das Risko.
- Eine Idee des Spiralmodells ist, das Anforderungen hierarchisch sind.
- In jedem Durchlauf können zusätzliche Anforderungen zu den bereits implementierten hinzugefügt werden.



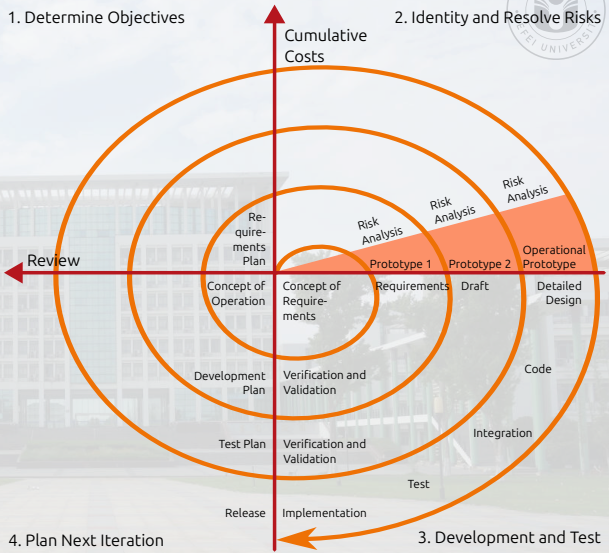
Das Spiralenmodell

- Jeder Prototyp reduziert also das Risko.
- Eine Idee des Spiralmodells ist, das Anforderungen hierarchisch sind.
- In jedem Durchlauf können zusätzliche Anforderungen zu den bereits implementierten hinzugefügt werden.
- In einer Datenbank können verschiedene Funktionen (z. B. Lagerhaltung und Bestellungsmanagement) eher unabhängig von einander sein, so dass dieser Ansatz nicht immer gut passt.



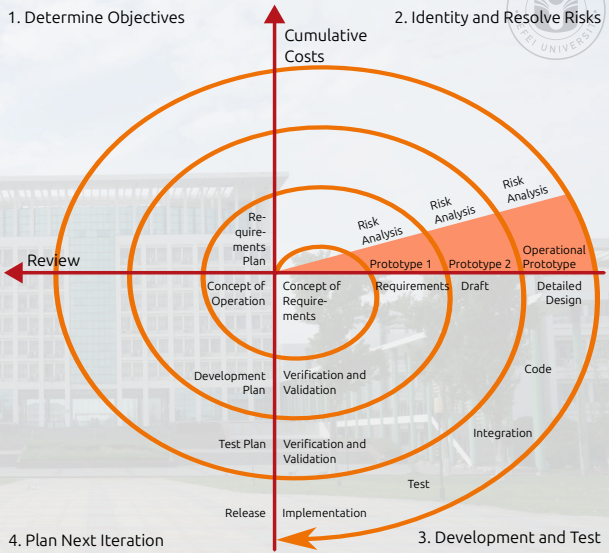
Das Spiralenmodell

- Eine Idee des Spiralmodells ist, das Anforderungen hierarchisch sind.
- In jedem Durchlauf können zusätzliche Anforderungen zu den bereits implementierten hinzugefügt werden.
- In einer Datenbank können verschiedene Funktionen (z. B. Lagerhaltung und Bestellungsmanagement) eher unabhängig von einander sein, so dass dieser Ansatz nicht immer gut passt.
- Die Anforderungen an das Projekt werden früh definiert, so dass der Projektumfang klar festgelegt ist.



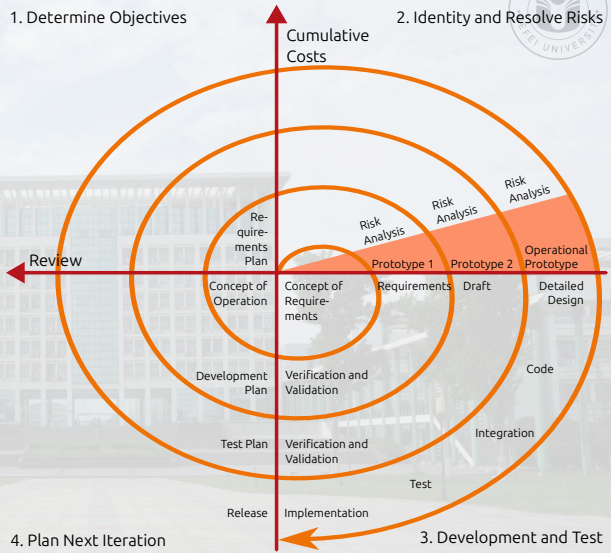
Das Spiralenmodell

- In jedem Durchlauf können zusätzliche Anforderungen zu den bereits implementierten hinzugefügt werden.
- In einer Datenbank können verschiedene Funktionen (z. B. Lagerhaltung und Bestellungsmanagement) eher unabhängig von einander sein, so dass dieser Ansatz nicht immer gut passt.
- Die Anforderungen an das Projekt werden früh definiert, so dass der Projektumfang klar festgelegt ist.
- Dadurch wird aber auch eine schrittweise Verbesserung des Designs nicht so gut unterstützt.



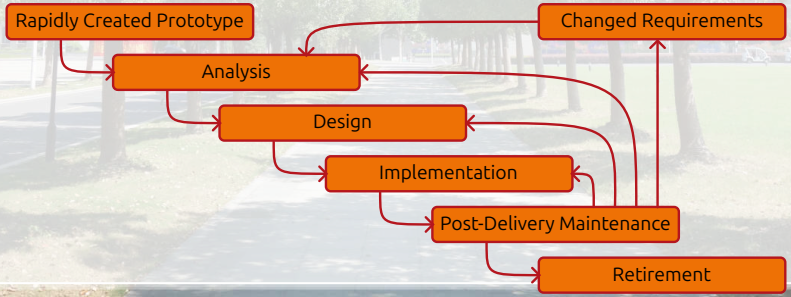
Das Spiralenmodell

- In einer Datenbank können verschiedene Funktionen (z. B. Lagerhaltung und Bestellungsmanagement) eher unabhängig von einander sein, so dass dieser Ansatz nicht immer gut passt.
- Die Anforderungen an das Projekt werden früh definiert, so dass der Projektumfang klar festgelegt ist.
- Dadurch wird aber auch eine schrittweise Verbesserung des Designs nicht so gut unterstützt.
- Dazu sind sowohl Risikoanalysen und die Struktur des Spiralmodells beide etwas komplex.[29,57].



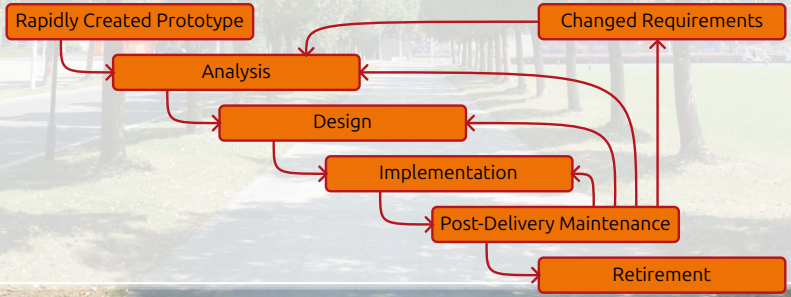
Rapid Application Development

- Rapid Application Development (RAD)[4,45] is similar to (rapid) prototyping.



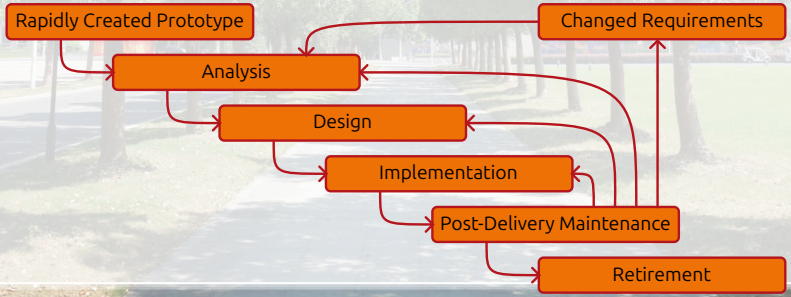
Rapid Application Development

- Rapid Application Development (RAD)[4,45] is similar to (rapid) prototyping.
- Die Benutzer können die (prototypische) Applikation ausprobieren, bevor sie ausgeliefert wird[29,46,57].



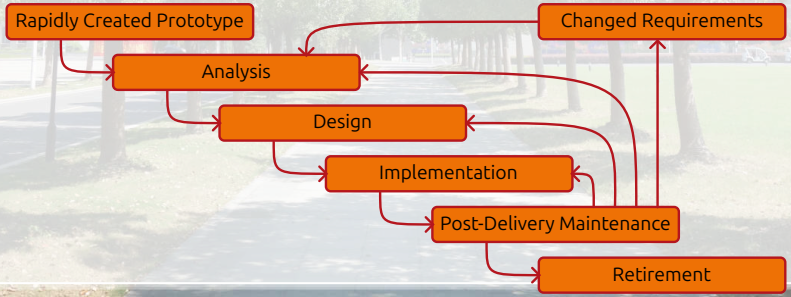
Rapid Application Development

- Rapid Application Development (RAD)[4,45] is similar to (rapid) prototyping.
- Die Benutzer können die (prototypische) Applikation ausprobieren, bevor sie ausgeliefert wird[29,46,57].
- Wenn die Stakeholders mit dem echten System herumspielen können, dann können sie wahrscheinlich besseres Feedback geben als wenn sie nur Spezifikationsdokumente zur Verfügung haben.



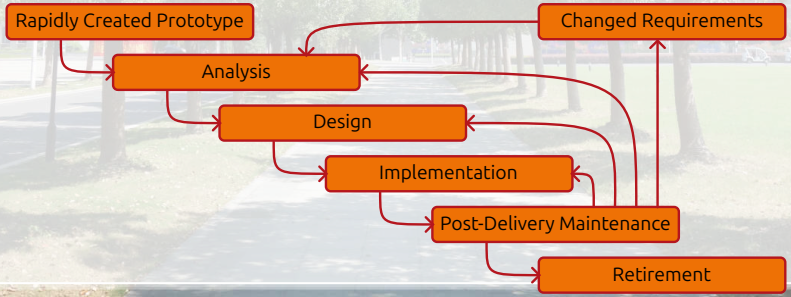
Rapid Application Development

- Rapid Application Development (RAD)[4,45] is similar to (rapid) prototyping.
- Die Benutzer können die (prototypische) Applikation ausprobieren, bevor sie ausgeliefert wird[29,46,57].
- Wenn die Stakeholders mit dem echten System herumspielen können, dann können sie wahrscheinlich besseres Feedback geben als wenn sie nur Spezifikationsdokumente zur Verfügung haben.
- Deshalb wird ein Prototyp so früh wie möglich entwickelt und den Benutzern gegeben.



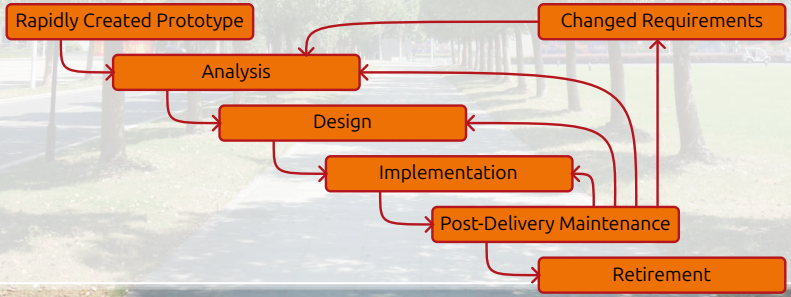
Rapid Application Development

- Die Benutzer können die (prototypische) Applikation ausprobieren, bevor sie ausgeliefert wird[29,46,57].
- Wenn die Stakeholders mit dem echten System herumspielen können, dann können sie wahrscheinlich besseres Feedback geben als wenn sie nur Spezifikationsdokumente zur Verfügung haben.
- Deshalb wird ein Prototyp so früh wie möglich entwickelt und den Benutzern gegeben.
- RAD-Projekte werden oft von kleinen, eng zusammenarbeitenden Teams über einen kurzen Zeitraum implementiert.



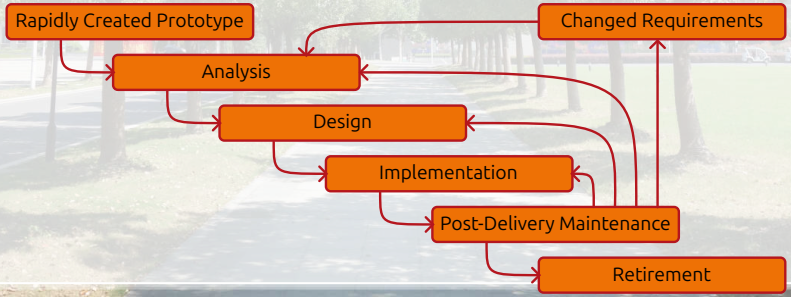
Rapid Application Development

- Wenn die Stakeholders mit dem echten System herumspielen können, dann können sie wahrscheinlich besseres Feedback geben als wenn sie nur Spezifikationsdokumente zur Verfügung haben.
- Deshalb wird ein Prototyp so früh wie möglich entwickelt und den Benutzern gegeben.
- RAD-Projekte werden oft von kleinen, eng zusammenarbeitenden Teams über einen kurzen Zeitraum implementiert.
- Die Projekte sind oft sehr interaktiv und von vergleichsweise geringer Komplexität.



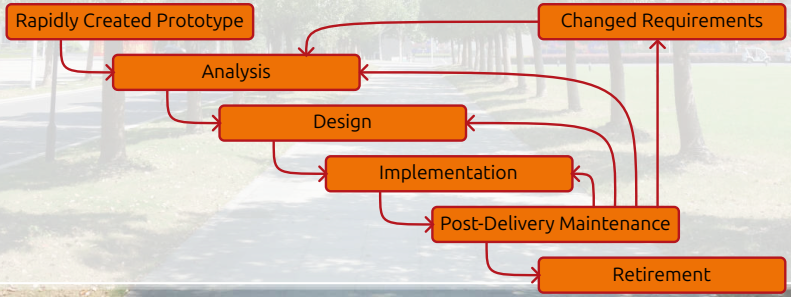
Rapid Application Development

- Deshalb wird ein Prototyp so früh wie möglich entwickelt und den Benutzern gegeben.
- RAD-Projekte werden oft von kleinen, eng zusammenarbeitenden Teams über einen kurzen Zeitraum implementiert.
- Die Projekte sind oft sehr interaktiv und von vergleichsweise geringer Komplexität.
- RAD führt daher soziale Komponenten und Team-Building in die Softwareentwicklung ein[4].



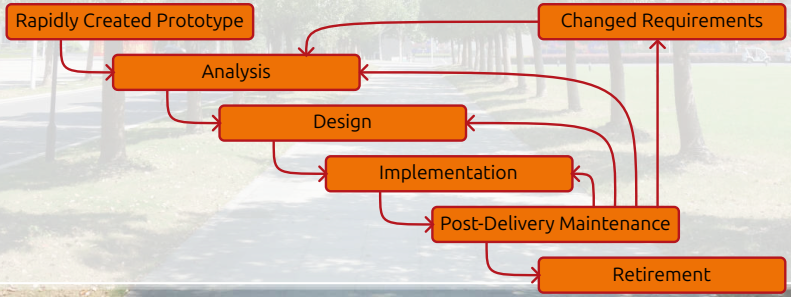
Rapid Application Development

- RAD-Projekte werden oft von kleinen, eng zusammenarbeitenden Teams über einen kurzen Zeitraum implementiert.
- Die Projekte sind oft sehr interaktiv und von vergleichsweise geringer Komplexität.
- RAD führt daher soziale Komponenten und Team-Building in die Softwareentwicklung ein[4].
- RAD-basierte Projekte tendieren dazu, seltener abgelehnt zu werden, wenn sie die Produktivphase erreichen.



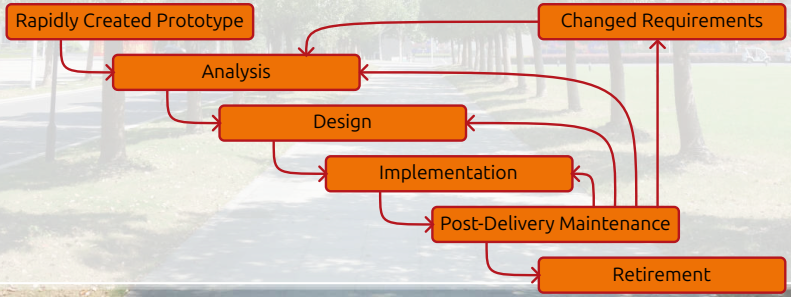
Rapid Application Development

- Die Projekte sind oft sehr interaktiv und von vergleichsweise geringer Komplexität.
- RAD führt daher soziale Komponenten und Team-Building in die Softwareentwicklung ein[4].
- RAD-basierte Projekte tendieren dazu, seltener abgelehnt zu werden, wenn sie die Produktivphase erreichen.
- Der Nachteil ist, dass so genannter *scope creep* häufiger auftritt.



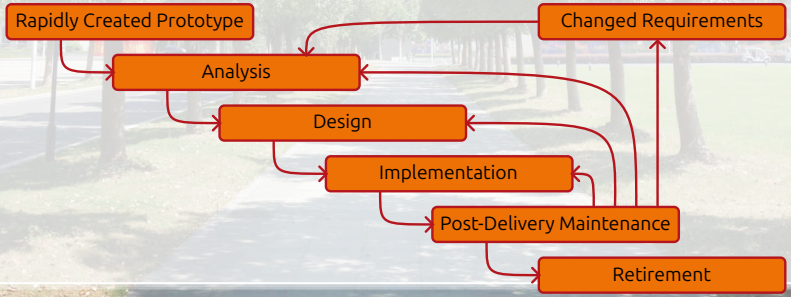
Rapid Application Development

- RAD führt daher soziale Komponenten und Team-Building in die Softwareentwicklung ein[4].
- RAD-basierte Projekte tendieren dazu, seltener abgelehnt zu werden, wenn sie die Produktivphase erreichen.
- Der Nachteil ist, dass so genannter *scope creep* häufiger auftritt:
- Wenn die Stakeholder ein Verändern der Applikation als einfach empfinden, dann fragen sie häufiger nach zusätzlichen Features.



Rapid Application Development

- RAD-basierte Projekte tendieren dazu, seltener abgelehnt zu werden, wenn sie die Produktivphase erreichen.
- Der Nachteil ist, dass so genannter *scope creep* häufiger auftritt:
- Wenn die Stakeholder ein Verändern der Applikation als einfach empfinden, dann fragen sie häufiger nach zusätzlichen Features.
- Das eigentliche Ziel ein voll funktionsfähiges System zu haben kann aus dem Fokus driften und das Budget bzw. der Zeitrahmen könnte überschritten werden.



Entwicklungsprozesse für Datenbanken

Datenbanken sind Anders

- Datenbanksysteme sind anders als normale Softwareanwendungen.

Datenbanken sind Anders

- Datenbanksysteme sind anders als normale Softwareanwendungen.
- Erstens sind sie sind oft die Grundlage für mehrere verschiedene Anwendungen.

Datenbanken sind Anders

- Datenbanksysteme sind anders als normale Softwareanwendungen.
- Erstens sind sie sind oft die Grundlage für mehrere verschiedene Anwendungen.
- Eine Studentenmanagementdatenbank einer Universität, z. B., könnte Studenden die Möglichkeit bieten, sich einzuloggen, sich für Module einzuschreiben, und ihre Noten zu sehen.

Datenbanken sind Anders

- Datenbanksysteme sind anders als normale Softwareanwendungen.
- Erstens sind sie sind oft die Grundlage für mehrere verschiedene Anwendungen.
- Eine Studentenmanagementdatenbank einer Universität, z. B., könnte Studenden die Möglichkeit bieten, sich einzuloggen, sich für Module einzuschreiben, und ihre Noten zu sehen.
- Es kann aber auch der Uni-Administration erlauben, neue Module zu erstellen und vielleicht sogar die Modulstruktur von Studiengängen zu managen.

Datenbanken sind Anders

- Datenbanksysteme sind anders als normale Softwareanwendungen.
- Erstens sind sie sind oft die Grundlage für mehrere verschiedene Anwendungen.
- Eine Studentenmanagementdatenbank einer Universität, z. B., könnte Studenden die Möglichkeit bieten, sich einzuloggen, sich für Module einzuschreiben, und ihre Noten zu sehen.
- Es kann aber auch der Uni-Administration erlauben, neue Module zu erstellen und vielleicht sogar die Modulstruktur von Studiengängen zu managen.
- Vielleicht übernimmt es auch noch die Raumplanung für Lehreinheiten.

Datenbanken sind Anders

- Datenbanksysteme sind anders als normale Softwareanwendungen.
- Erstens sind sie sind oft die Grundlage für mehrere verschiedene Anwendungen.
- Eine Studentenmanagementdatenbank einer Universität, z. B., könnte Studenden die Möglichkeit bieten, sich einzuloggen, sich für Module einzuschreiben, und ihre Noten zu sehen.
- Es kann aber auch der Uni-Administration erlauben, neue Module zu erstellen und vielleicht sogar die Modulstruktur von Studiengängen zu managen.
- Vielleicht übernimmt es auch noch die Raumplanung für Lehreinheiten.
- Und die Zeitplanung für das Semester, die Prüfungen, und die Stundenpläne der Studenten.

Datenbanken sind Anders

- Datenbanksysteme sind anders als normale Softwareanwendungen.
- Erstens sind sie sind oft die Grundlage für mehrere verschiedene Anwendungen.
- Eine Studentenmanagementdatenbank einer Universität, z. B., könnte Studenden die Möglichkeit bieten, sich einzuloggen, sich für Module einzuschreiben, und ihre Noten zu sehen.
- Es kann aber auch der Uni-Administration erlauben, neue Module zu erstellen und vielleicht sogar die Modulstruktur von Studiengängen zu managen.
- Vielleicht übernimmt es auch noch die Raumplanung für Lehreinheiten.
- Und die Zeitplanung für das Semester, die Prüfungen, und die Stundenpläne der Studenten.
- All das könnte über verschiedene Applikation mit verschiedenen Benutzerschnittstellen gehen.

Datenbanken sind Anders

- Datenbanksysteme sind anders als normale Softwareanwendungen.
- Erstens sind sie sind oft die Grundlage für mehrere verschiedene Anwendungen.
- Eine Studentenmanagementdatenbank einer Universität, z. B., könnte Studenden die Möglichkeit bieten, sich einzuloggen, sich für Module einzuschreiben, und ihre Noten zu sehen.
- Es kann aber auch der Uni-Administration erlauben, neue Module zu erstellen und vielleicht sogar die Modulstruktur von Studiengängen zu managen.
- Vielleicht übernimmt es auch noch die Raumplanung für Lehreinheiten.
- Und die Zeitplanung für das Semester, die Prüfungen, und die Stundenpläne der Studenten.
- All das könnte über verschiedene Applikation mit verschiedenen Benutzerschnittstellen gehen.
- Aber alle nutzen die selbe Datenbank als Backend.

Datenbanken sind Anders

- Datenbanksysteme sind anders als normale Softwareanwendungen.
- Erstens sind sie sind oft die Grundlage für mehrere verschiedene Anwendungen.
- Eine Studentenmanagementdatenbank einer Universität, z. B., könnte Studenden die Möglichkeit bieten, sich einzuloggen, sich für Module einzuschreiben, und ihre Noten zu sehen.
- Es kann aber auch der Uni-Administration erlauben, neue Module zu erstellen und vielleicht sogar die Modulstruktur von Studiengängen zu managen.
- Vielleicht übernimmt es auch noch die Raumplanung für Lehreinheiten.
- Und die Zeitplanung für das Semester, die Prüfungen, und die Stundenpläne der Studenten.
- All das könnte über verschiedene Applikation mit verschiedenen Benutzerschnittstellen gehen.
- Aber alle nutzen die selbe Datenbank als Backend.
- Zweitens haben wir bei der Datenbankentwicklung normalerweise eine Abfolge von drei Schemas.

Datenbanken sind Anders

- Datenbanksysteme sind anders als normale Softwareanwendungen.
- Erstens sind sie sind oft die Grundlage für mehrere verschiedene Anwendungen.
- Eine Studentenmanagementdatenbank einer Universität, z. B., könnte Studenden die Möglichkeit bieten, sich einzuloggen, sich für Module einzuschreiben, und ihre Noten zu sehen.
- Es kann aber auch der Uni-Administration erlauben, neue Module zu erstellen und vielleicht sogar die Modulstruktur von Studiengängen zu managen.
- Vielleicht übernimmt es auch noch die Raumplanung für Lehreinheiten.
- Und die Zeitplanung für das Semester, die Prüfungen, und die Stundenpläne der Studenten.
- All das könnte über verschiedene Applikation mit verschiedenen Benutzerschnittstellen gehen.
- Aber alle nutzen die selbe Datenbank als Backend.
- Zweitens haben wir bei der Datenbankentwicklung normalerweise eine Abfolge von drei Schemas.
- Am Anfang wird ein konzeptuelles Schema das die Entitäten und Prozesse der realen Welt modelliert.

Datenbanken sind Anders

- Eine Studentenmanagementdatenbank einer Universität, z. B., könnte Studenden die Möglichkeit bieten, sich einzuloggen, sich für Module einzuschreiben, und ihre Noten zu sehen.
- Es kann aber auch der Uni-Administration erlauben, neue Module zu erstellen und vielleicht sogar die Modulstruktur von Studiengängen zu managen.
- Vielleicht übernimmt es auch noch die Raumplanung für Lehreinheiten.
- Und die Zeitplanung für das Semester, die Prüfungen, und die Stundenpläne der Studenten.
- All das könnte über verschiedene Applikation mit verschiedenen Benutzerschnittstellen gehen.
- Aber alle nutzen die selbe Datenbank als Backend.
- Zweitens haben wir bei der Datenbankentwicklung normalerweise eine Abfolge von drei Schemas.
- Am Anfang wird ein konzeptuelles Schema das die Entitäten und Prozesse der realen Welt modelliert.
- Dieses wird dann auf ein logisches Schema basierend auf einer konkreten Technologie abgebildet.

Datenbanken sind Anders

- Es kann aber auch der Uni-Administration erlauben, neue Module zu erstellen und vielleicht sogar die Modulstruktur von Studiengängen zu managen.
- Vielleicht übernimmt es auch noch die Raumplanung für Lehreinheiten.
- Und die Zeitplanung für das Semester, die Prüfungen, und die Stundenpläne der Studenten.
- All das könnte über verschiedene Applikation mit verschiedenen Benutzerschnittstellen gehen.
- Aber alle nutzen die selbe Datenbank als Backend.
- Zweitens haben wir bei der Datenbankentwicklung normalerweise eine Abfolge von drei Schemas.
- Am Anfang wird ein konzeptuelles Schema das die Entitäten und Prozesse der realen Welt modelliert.
- Dieses wird dann auf ein logisches Schema basierend auf einer konkreten Technologie abgebildet.
- Am Ende kann ein physisches Schema stehen, dass mit zusätzlichen Konfigurationen die Performanz verbessert.

Datenbanken sind Anders

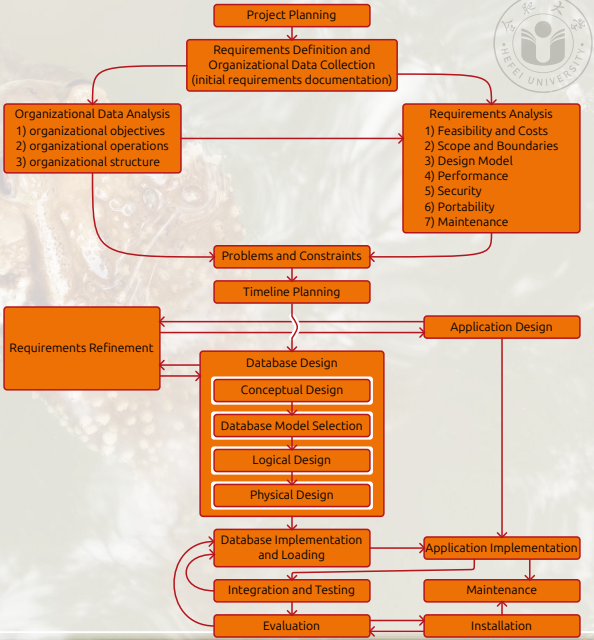
- Vielleicht übernimmt es auch noch die Raumplanung für Lehreinheiten.
- Und die Zeitplanung für das Semester, die Prüfungen, und die Stundenpläne der Studenten.
- All das könnte über verschiedene Applikation mit verschiedenen Benutzerschnittstellen gehen.
- Aber alle nutzen die selbe Datenbank als Backend.
- Zweitens haben wir bei der Datenbankentwicklung normalerweise eine Abfolge von drei Schemas.
- Am Anfang wird ein konzeptuelles Schema das die Entitäten und Prozesse der realen Welt modelliert.
- Dieses wird dann auf ein logisches Schema basierend auf einer konkreten Technologie abgebildet.
- Am Ende kann ein physisches Schema stehen, dass mit zusätzlichen Konfigurationen die Performanz verbessert.
- Drittens müssen wir immer beachten, dass Datenbanken sehr langlebige Artefakte sind, die über viele Jahre gewartet und verbessert werden müssen.

Datenbanken sind Anders

- Und die Zeitplanung für das Semester, die Prüfungen, und die Stundenpläne der Studenten.
- All das könnte über verschiedene Applikation mit verschiedenen Benutzerschnittstellen gehen.
- Aber alle nutzen die selbe Datenbank als Backend.
- Zweitens haben wir bei der Datenbankentwicklung normalerweise eine Abfolge von drei Schemas.
- Am Anfang wird ein konzeptuelles Schema das die Entitäten und Prozesse der realen Welt modelliert.
- Dieses wird dann auf ein logisches Schema basierend auf einer konkreten Technologie abgebildet.
- Am Ende kann ein physisches Schema stehen, dass mit zusätzlichen Konfigurationen die Performanz verbessert.
- Drittens müssen wir immer beachten, dass Datenbanken sehr langlebige Artefakte sind, die über viele Jahre gewartet und verbessert werden müssen.
- Aus diesem Grund wurden verschiedene Lebenszyklus-Strukturen für Datenbanken entwickelt.

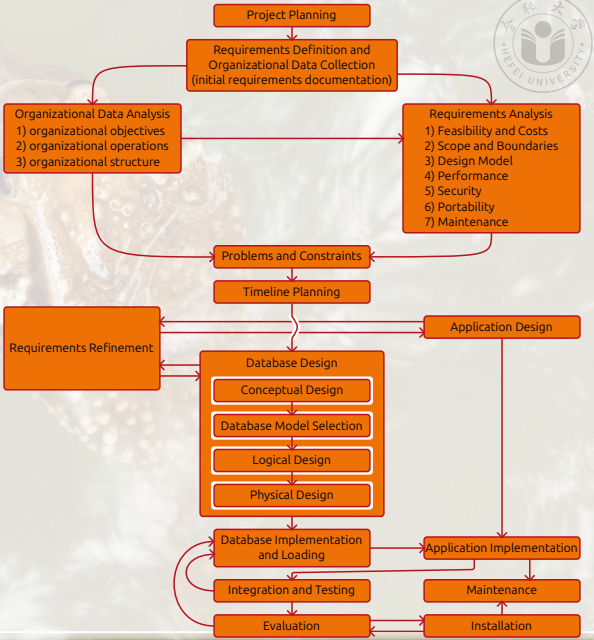
Beispiel Lebenszyklus

- Ich mag z. B. das Lebenszyklus-Modell von Gupta, Mata-Toledo und Monger[29].



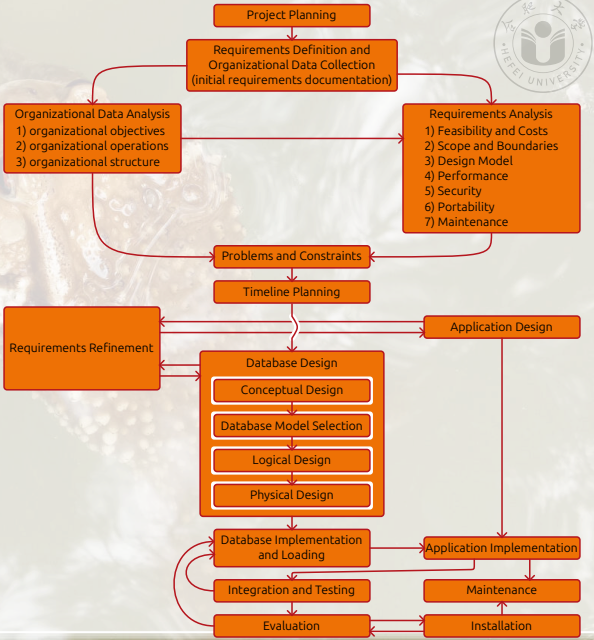
Beispiel Lebenszyklus

- Ich mag z. B. das Lebenszyklus-Modell von Gupta, Mata-Toledo und Monger[29].
- Wir finden darin viele der Aktivitäten, die ein guter DBA durchführen muss.



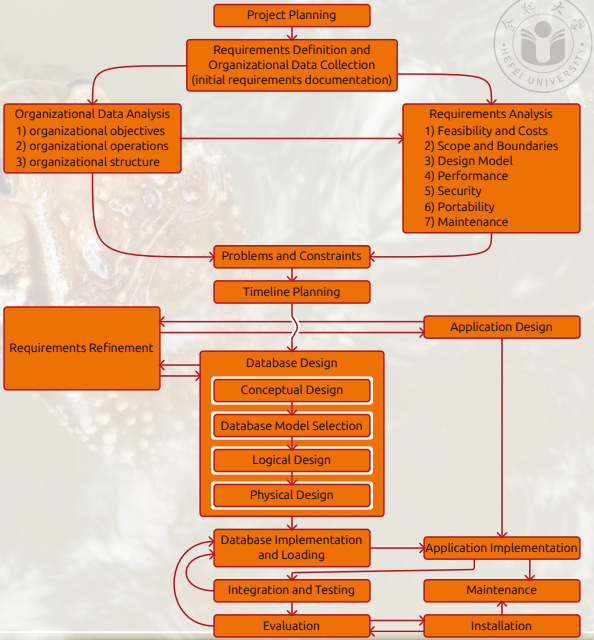
Beispiel Lebenszyklus

- Ich mag z. B. das Lebenszyklus-Modell von Gupta, Mata-Toledo und Monger[29].
- Wir finden darin viele der Aktivitäten, die ein guter DBA durchführen muss.
- Wir beginnen mit einer Planungsphase.



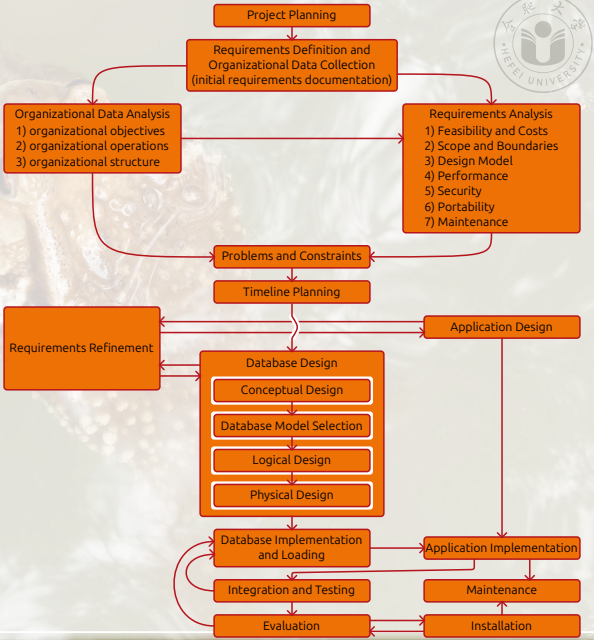
Beispiel Lebenszyklus

- Ich mag z. B. das Lebenszyklus-Modell von Gupta, Mata-Toledo und Monger[29].
- Wir finden darin viele der Aktivitäten, die ein guter DBA durchführen muss.
- Wir beginnen mit einer Planungsphase.
- Hier werden zwei Aktivitäten geplant.



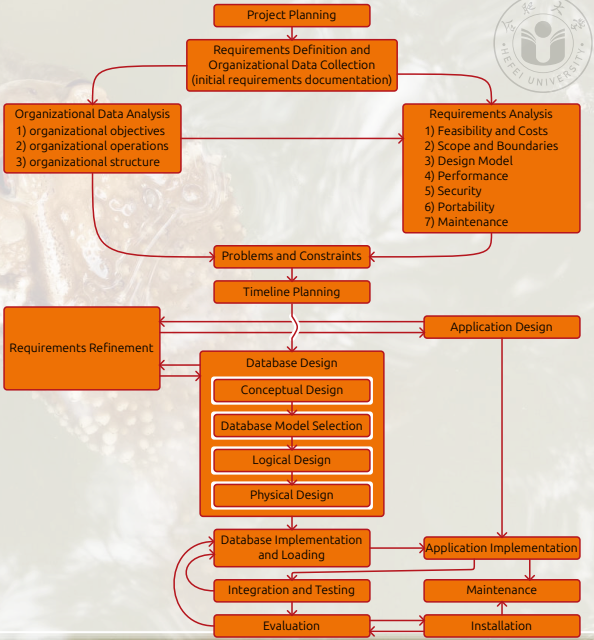
Beispiel Lebenszyklus

- Ich mag z. B. das Lebenszyklus-Modell von Gupta, Mata-Toledo und Monger[29].
- Wir finden darin viele der Aktivitäten, die ein guter DBA durchführen muss.
- Wir beginnen mit einer Planungsphase.
- Hier werden zwei Aktivitäten geplant: (1) das Sammeln von Informationen über die Prozesse der Organisation, die in der Datenbank gespiegelt werden sollen.



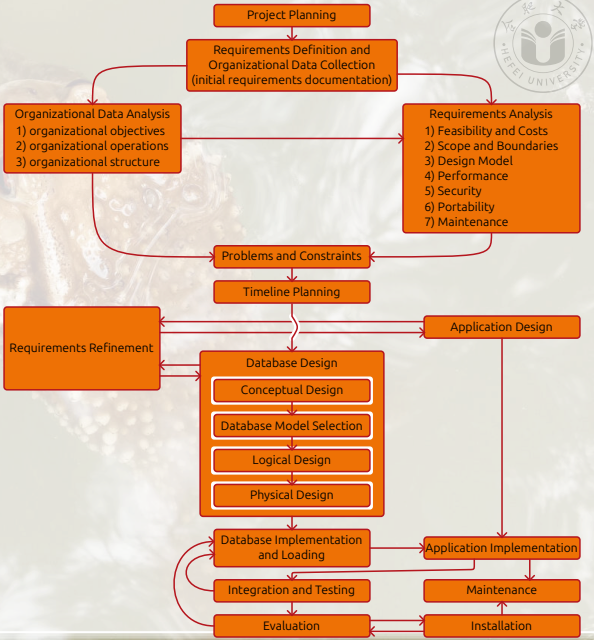
Beispiel Lebenszyklus

- Ich mag z. B. das Lebenszyklus-Modell von Gupta, Mata-Toledo und Monger[29].
- Wir finden darin viele der Aktivitäten, die ein guter DBA durchführen muss.
- Wir beginnen mit einer Planungsphase.
- Hier werden zwei Aktivitäten geplant: (1) das Sammeln von Informationen über die Prozesse der Organisation, die in der Datenbank gespiegelt werden sollen (2) die Anforderungsanalyse.



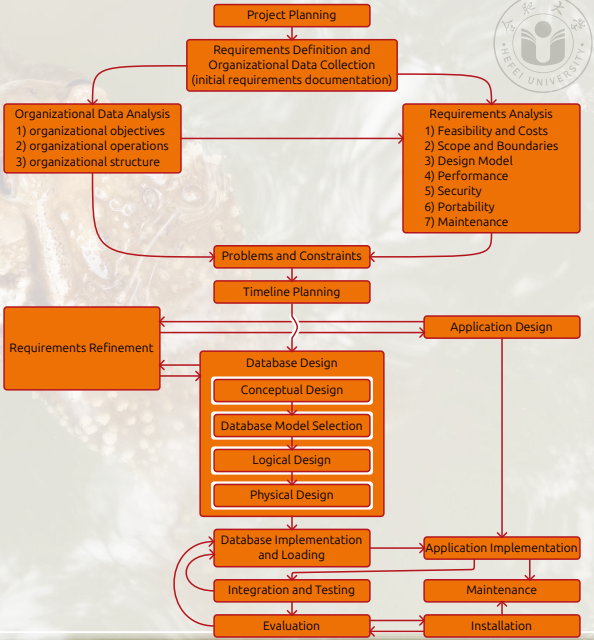
Beispiel Lebenszyklus

- Ich mag z. B. das Lebenszyklus-Modell von Gupta, Mata-Toledo und Monger[29].
- Wir finden darin viele der Aktivitäten, die ein guter DBA durchführen muss.
- Wir beginnen mit einer Planungsphase.
- Hier werden zwei Aktivitäten geplant: (1) das Sammeln von Informationen über die Prozesse der Organisation, die in der Datenbank gespiegelt werden sollen (2) die Anforderungsanalyse.
- Als Ergebnis hat man dann einen Plan.



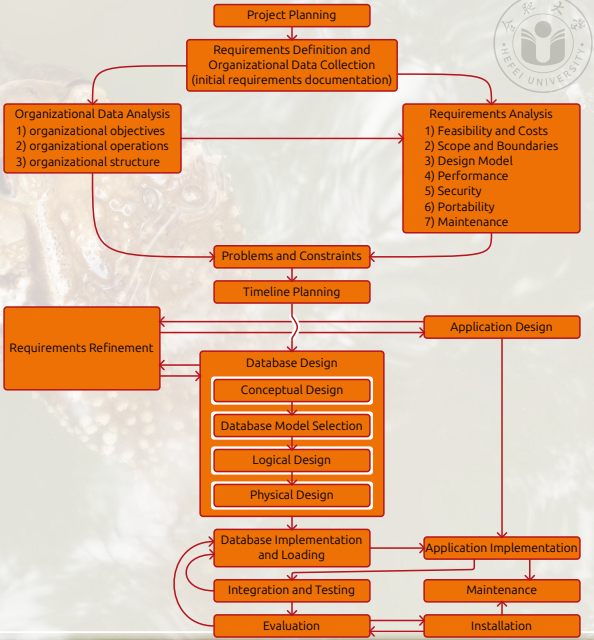
Beispiel Lebenszyklus

- Wir finden darin viele der Aktivitäten, die ein guter DBA durchführen muss.
- Wir beginnen mit einer Planungsphase.
- Hier werden zwei Aktivitäten geplant: (1) das Sammeln von Informationen über die Prozesse der Organisation, die in der Datenbank gespiegelt werden sollen (2) die Anforderungsanalyse.
- Als Ergebnis hat man dann einen Plan.
- Dann sammeln wir alle benötigten Daten und Informationen über die Organisation.



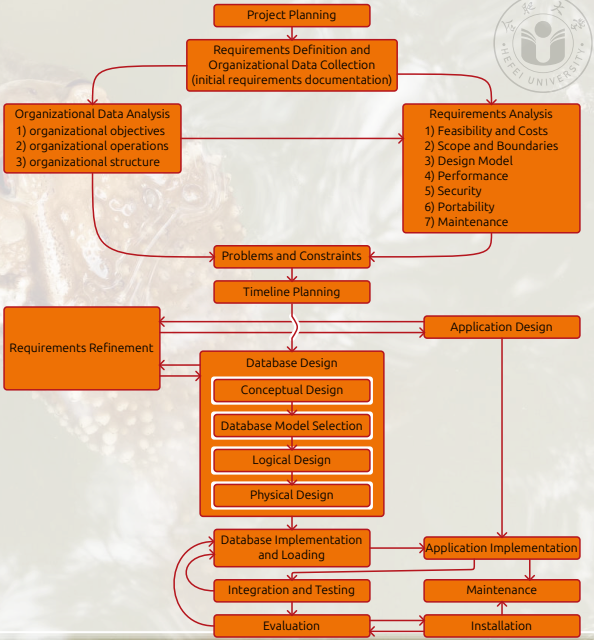
Beispiel Lebenszyklus

- Wir beginnen mit einer Planungsphase.
- Hier werden zwei Aktivitäten geplant: (1) das Sammeln von Informationen über die Prozesse der Organisation, die in der Datenbank gespiegelt werden sollen (2) die Anforderungsanalyse.
- Als Ergebnis hat man dann einen Plan.
- Dann sammeln wir alle benötigten Daten und Informationen über die Organisation.
- Die Entwickler und Designer arbeiten mit allen Stakeholders des Projekts auf allen ebenen zusammen, von den späteren Benutzern der Datenbank bis hin zum Management der Organisation.



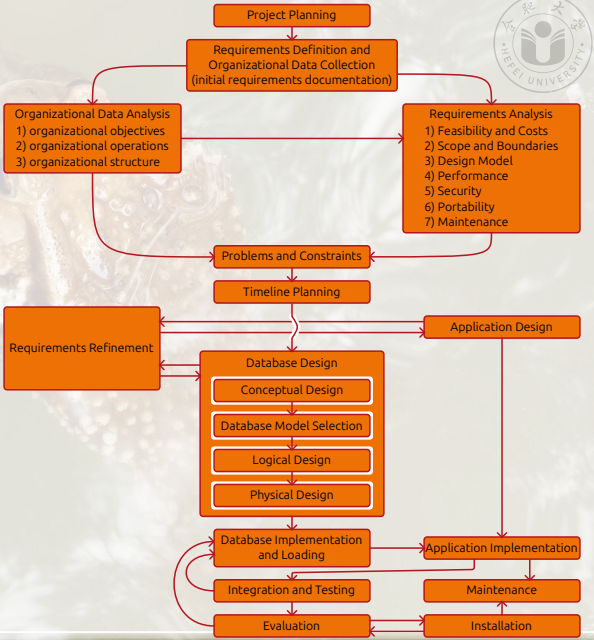
Beispiel Lebenszyklus

- Als Ergebnis hat man dann einen Plan.
- Dann sammeln wir alle benötigten Daten und Informationen über die Organisation.
- Die Entwickler und Designer arbeiten mit allen Stakeholders des Projekts auf allen ebenen zusammen, von den späteren Benutzern der Datenbank bis hin zum Management der Organisation.
- Die Dokumente und Prozesse in der Organisation werden erforscht, ebenso wie der Datenfluss durch die Organisation.



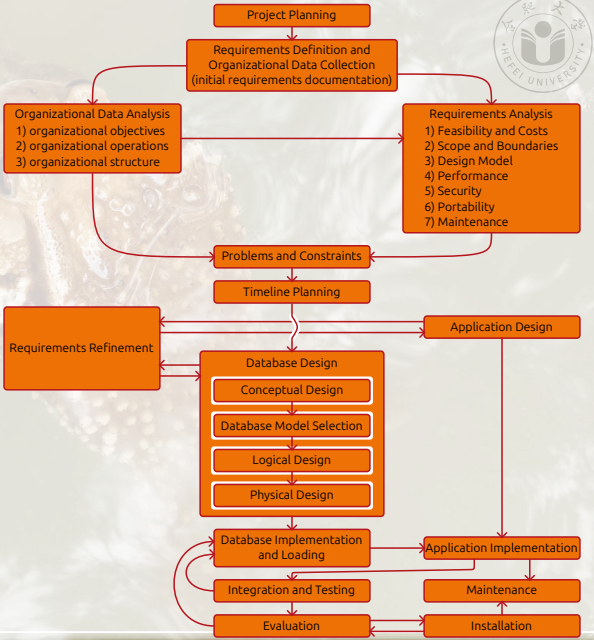
Beispiel Lebenszyklus

- Dann sammeln wir alle benötigten Daten und Informationen über die Organisation.
- Die Entwickler und Designer arbeiten mit allen Stakeholders des Projekts auf allen ebenen zusammen, von den späteren Benutzern der Datenbank bis hin zum Management der Organisation.
- Die Dokumente und Prozesse in der Organisation werden erforscht, ebenso wie der Datenfluss durch die Organisation.
- Wir prüfen welche Systeme bereits existieren, welche Eingabedaten sie brauchen, welche Ausgaben sie produzieren, wer sie benutzt und warum.



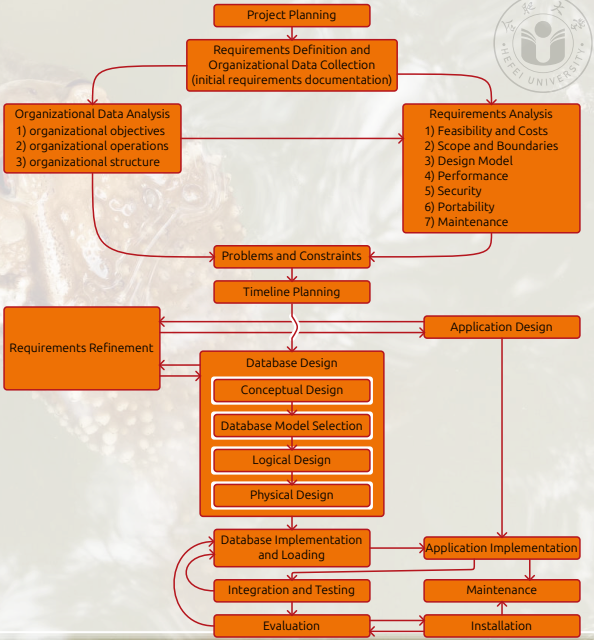
Beispiel Lebenszyklus

- Die Entwickler und Designer arbeiten mit allen Stakeholders des Projekts auf allen ebenen zusammen, von den späteren Benutzern der Datenbank bis hin zum Management der Organisation.
- Die Dokumente und Prozesse in der Organisation werden erforscht, ebenso wie der Datenfluss durch die Organisation.
- Wir prüfen welche Systeme bereits existieren, welche Eingabedaten sie brauchen, welche Ausgaben sie produzieren, wer sie benutzt und warum.
- Interviews und Fragebögen können ebenso benutzt werden, um mehr Daten zu sammeln.



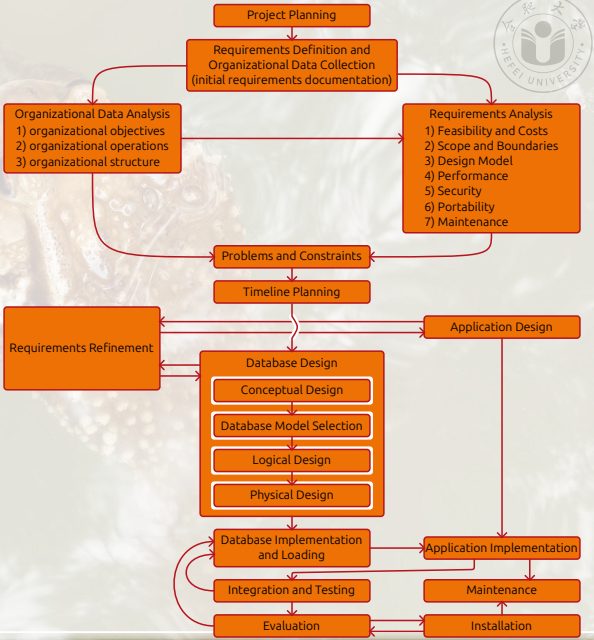
Beispiel Lebenszyklus

- Die Dokumente und Prozesse in der Organisation werden erforscht, ebenso wie der Datenfluss durch die Organisation.
- Wir prüfen welche Systeme bereits existieren, welche Eingabedaten sie brauchen, welche Ausgaben sie produzieren, wer sie benutzt und warum.
- Interviews und Fragebögen können ebenso benutzt werden, um mehr Daten zu sammeln.
- Die Aktivitäten von Personal auf allen Ebenen können auch direkt beobachtet und dokumentiert werden.



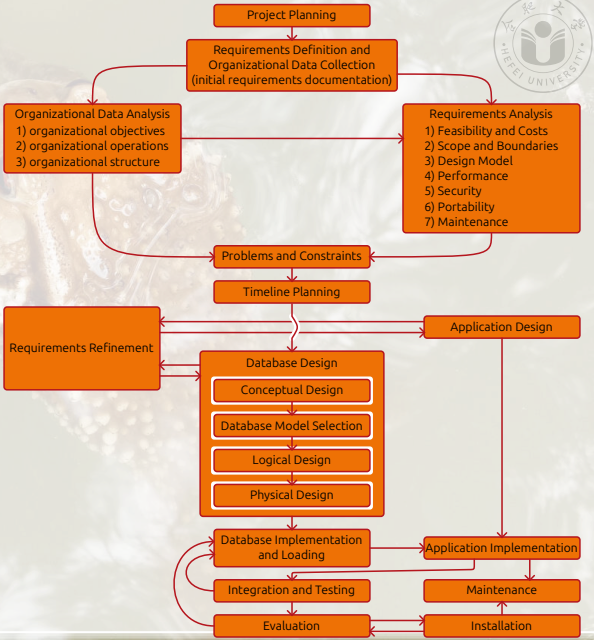
Beispiel Lebenszyklus

- Wir prüfen welche Systeme bereits existieren, welche Eingabedaten sie brauchen, welche Ausgaben sie produzieren, wer sie benutzt und warum.
- Interviews und Fragebögen können ebenso benutzt werden, um mehr Daten zu sammeln.
- Die Aktivitäten von Personal auf allen Ebenen können auch direkt beobachtet und dokumentiert werden.
- Die aktuellen Anforderungen und mögliche zukünftige Erweiterungen der Datenbank werden erforscht.



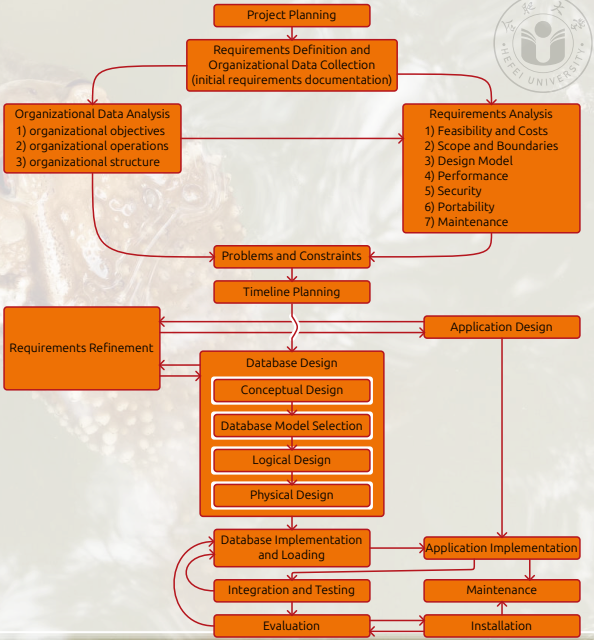
Beispiel Lebenszyklus

- Wir prüfen welche Systeme bereits existieren, welche Eingabedaten sie brauchen, welche Ausgaben sie produzieren, wer sie benutzt und warum.
- Interviews und Fragebögen können ebenso benutzt werden, um mehr Daten zu sammeln.
- Die Aktivitäten von Personal auf allen Ebenen können auch direkt beobachtet und dokumentiert werden.
- Die aktuellen Anforderungen und mögliche zukünftige Erweiterungen der Datenbank werden erforscht.
- Die Anforderungsspezifikation wird erarbeitet.



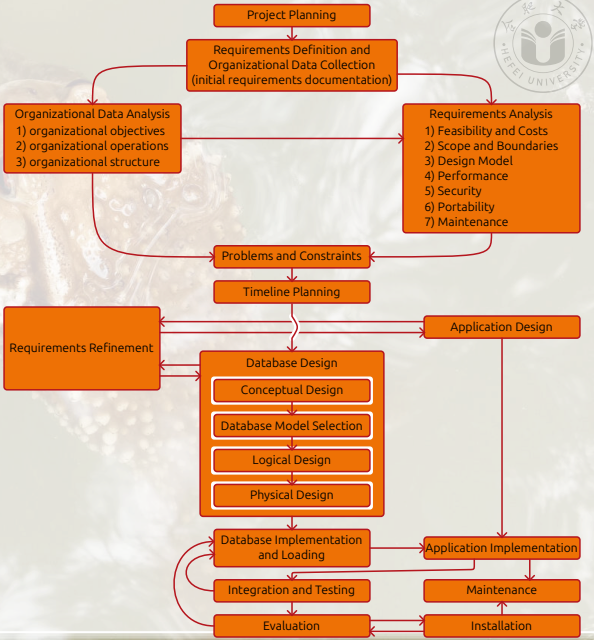
Beispiel Lebenszyklus

- Interviews und Fragebögen können ebenso benutzt werden, um mehr Daten zu sammeln.
- Die Aktivitäten von Personal auf allen Ebenen können auch direkt beobachtet und dokumentiert werden.
- Die aktuellen Anforderungen und mögliche zukünftige Erweiterungen der Datenbank werden erforscht.
- Die Anforderungsspezifikation wird erarbeitet.
- In der Anforderungsanalyse untersichen wir die gsammelten Daten um zu entscheiden, ob das Projekt durchführbar ist und um seine Kosten abzuschätzen.



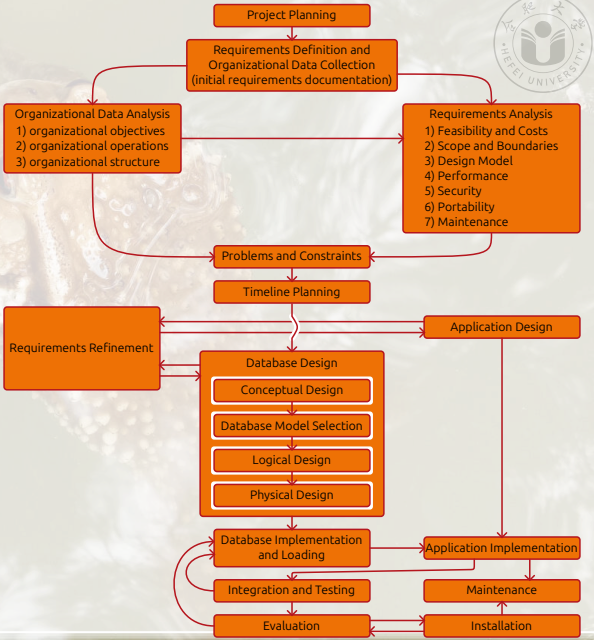
Beispiel Lebenszyklus

- Die Aktivitäten von Personal auf allen Ebenen können auch direkt beobachtet und dokumentiert werden.
- Die aktuellen Anforderungen und mögliche zukünftige Erweiterungen der Datenbank werden erforscht.
- Die Anforderungsspezifikation wird erarbeitet.
- In der Anforderungsanalyse untersichen wir die gsammelten Daten um zu entscheiden, ob das Projekt durchführbar ist und um seine Kosten abzuschätzen.
- Die Projektziele werden spezifiziert.



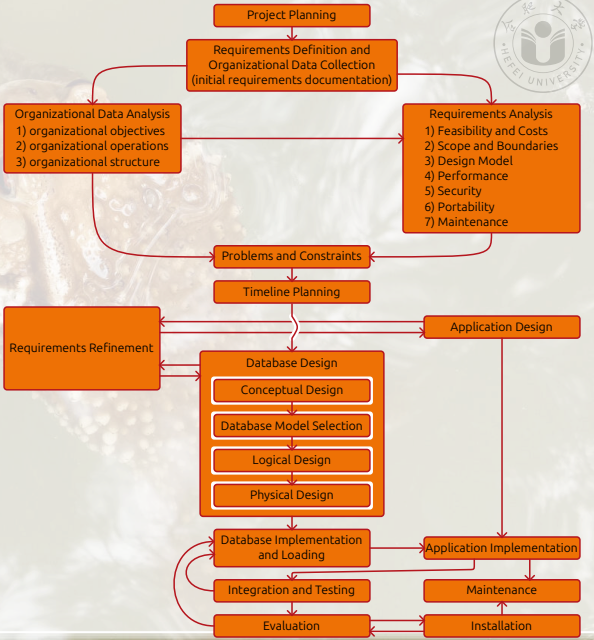
Beispiel Lebenszyklus

- Die aktuellen Anforderungen und mögliche zukünftige Erweiterungen der Datenbank werden erforscht.
- Die Anforderungsspezifikation wird erarbeitet.
- In der Anforderungsanalyse untersichen wir die gsammelten Daten um zu entscheiden, ob das Projekt durchführbar ist und um seine Kosten abzuschätzen.
- Die Projektziele werden spezifiziert.
- Die Grenzen des Projekts (Budget, Equipment, verwendbare Software) werden definiert ebenso wie die Anforderungen an Performanz, Sicherheit, und Wartbarkeit, ...



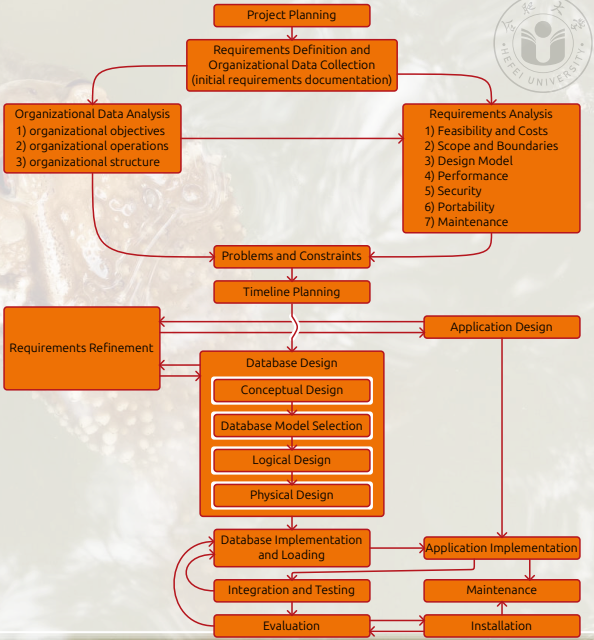
Beispiel Lebenszyklus

- Die Anforderungsspezifikation wird erarbeitet.
- In der Anforderungsanalyse untersichen wir die gsammelten Daten um zu entscheiden, ob das Projekt durchführbar ist und um seine Kosten abzuschätzen.
- Die Projektziele werden spezifiziert.
- Die Grenzen des Projekts (Budget, Equipment, verwendbare Software) werden definiert ebenso wie die Anforderungen an Performanz, Sicherheit, und Wartbarkeit, ...
- All das ist wichtig für den Erfolg des Projekts.



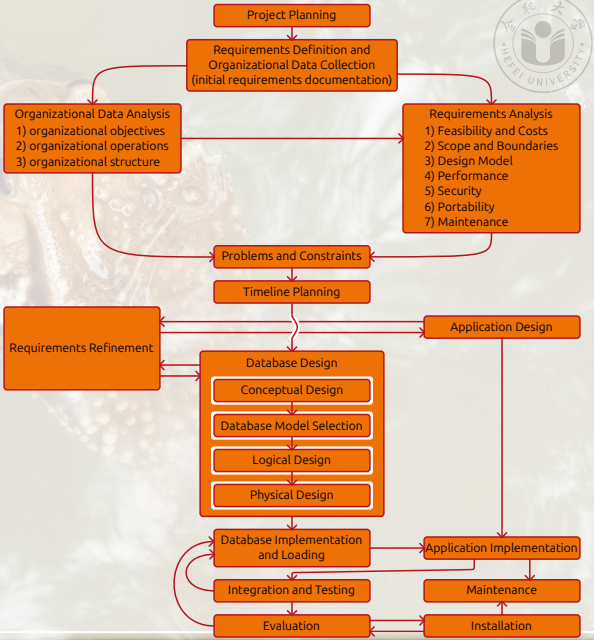
Beispiel Lebenszyklus

- In der Anforderungsanalyse untersichen wir die gsammelten Daten um zu entscheiden, ob das Projekt durchführbar ist und um seine Kosten abzuschätzen.
- Die Projektziele werden spezifiziert.
- Die Grenzen des Projekts (Budget, Equipment, verwendbare Software) werden definiert ebenso wie die Anforderungen an Performanz, Sicherheit, und Wartbarkeit, ...
- All das ist wichtig für den Erfolg des Projekts.
- Als Ergebnis können wir mögliche Probleme, die später auftreten könnten, bereits jetzt identifizieren.



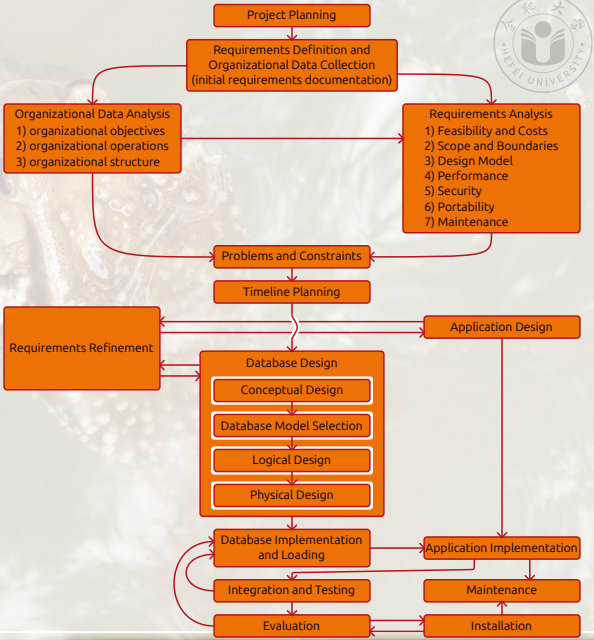
Beispiel Lebenszyklus

- Die Projektziele werden spezifiziert.
- Die Grenzen des Projekts (Budget, Equipment, verwendbare Software) werden definiert ebenso wie die Anforderungen an Performanz, Sicherheit, und Wartbarkeit, ...
- All das ist wichtig für den Erfolg des Projekts.
- Als Ergebnis können wir mögliche Probleme, die später auftreten könnten, bereits jetzt identifizieren.
- Nachdem die Anforderungen gesammelt und analysiert sind, können wir einen Zeitplan für den Rest des Projektes festlegen.



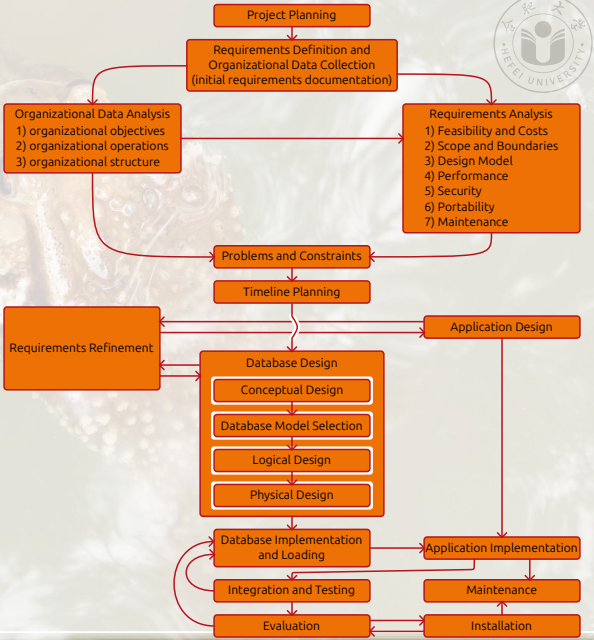
Beispiel Lebenszyklus

- All das ist wichtig für den Erfolg des Projekts.
- Als Ergebnis können wir mögliche Probleme, die später auftreten könnten, bereits jetzt identifizieren.
- Nachdem die Anforderungen gesammelt und analysiert sind, können wir einen Zeitplan für den Rest des Projektes festlegen.
- Datenbanken und die Applikationen, die mit den Daten arbeiten, können nicht völlig von einander getrennt werden.



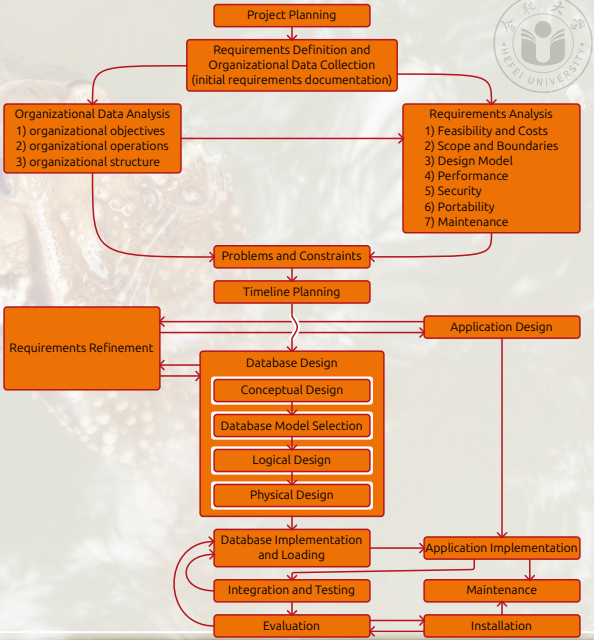
Beispiel Lebenszyklus

- All das ist wichtig für den Erfolg des Projekts.
- Als Ergebnis können wir mögliche Probleme, die später auftreten könnten, bereits jetzt identifizieren.
- Nachdem die Anforderungen gesammelt und analysiert sind, können wir einen Zeitplan für den Rest des Projektes festlegen.
- Datenbanken und die Applikationen, die mit den Daten arbeiten, können nicht völlig von einander getrennt werden.
- Die Datenbank und die Applikationen werden daher parallel in zwei Zweigen des Projekts entwickelt.



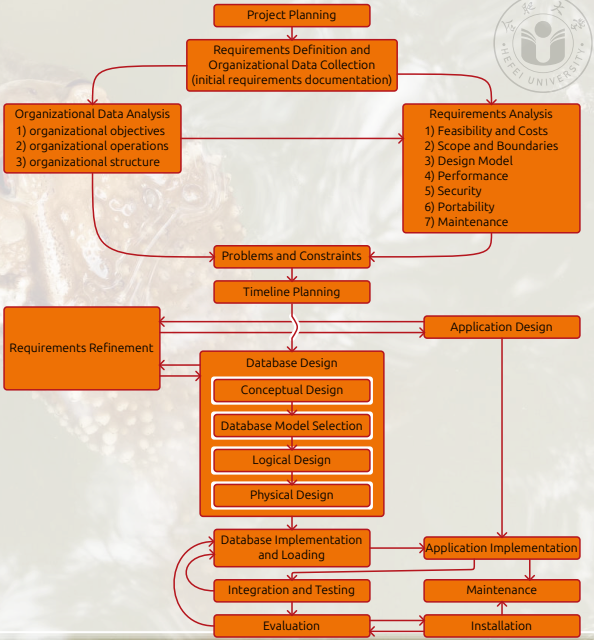
Beispiel Lebenszyklus

- Als Ergebnis können wir mögliche Probleme, die später auftreten könnten, bereits jetzt identifizieren.
- Nachdem die Anforderungen gesammelt und analysiert sind, können wir einen Zeitplan für den Rest des Projektes festlegen.
- Datenbanken und die Applikationen, die mit den Daten arbeiten, können nicht völlig von einander getrennt werden.
- Die Datenbank und die Applikationen werden daher parallel in zwei Zweigen des Projekts entwickelt.
- Beide updaten sich gegenseitig.



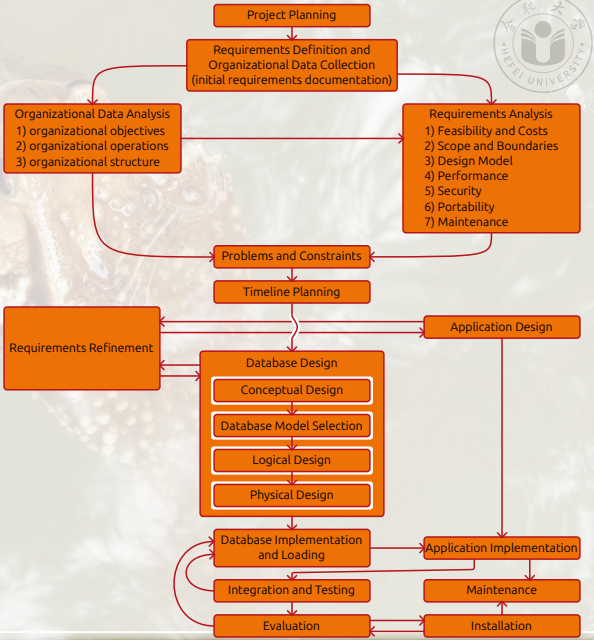
Beispiel Lebenszyklus

- Nachdem die Anforderungen gesammelt und analysiert sind, können wir einen Zeitplan für den Rest des Projektes festlegen.
- Datenbanken und die Applikationen, die mit den Daten arbeiten, können nicht völlig von einander getrennt werden.
- Die Datenbank und die Applikationen werden daher parallel in zwei Zweigen des Projekts entwickelt.
- Beide updaten sich gegenseitig.
- Der interne Datenbankdesign-Block entspricht dann wieder der allgemeinen Abfolge der drei Schemas[23].



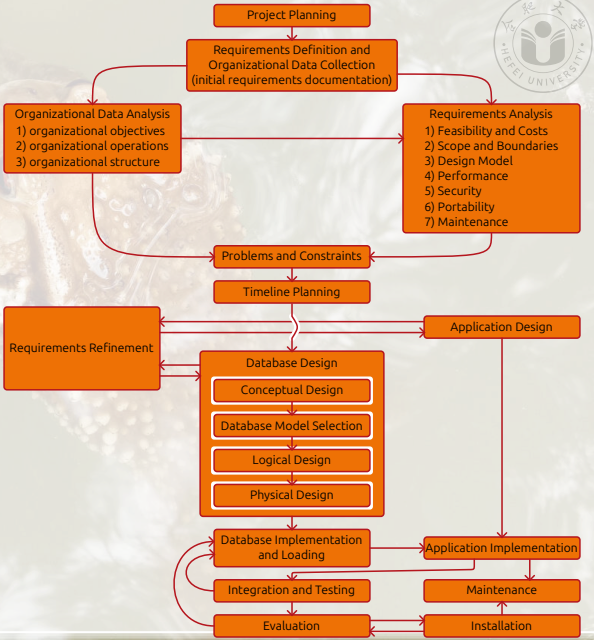
Beispiel Lebenszyklus

- Datenbanken und die Applikationen, die mit den Daten arbeiten, können nicht völlig von einander getrennt werden.
- Die Datenbank und die Applikationen werden daher parallel in zwei Zweigen des Projekts entwickelt.
- Beide updaten sich gegenseitig.
- Der interne Datenbankdesign-Block entspricht dann wieder der allgemeinen Abfolge der drei Schemas[23].
- Zuerst wird das konzeptuelle Schema entworfen, also die „High-Level“-Sicht auf die Datenbank, welche völlig technologieunabhängig ist.



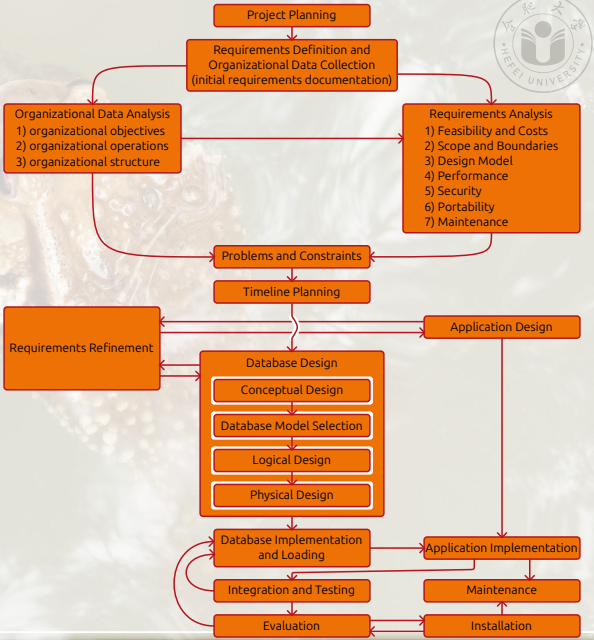
Beispiel Lebenszyklus

- Die Datenbank und die Applikationen werden daher parallel in zwei Zweigen des Projekts entwickelt.
- Beide updaten sich gegenseitig.
- Der interne Datenbankdesign-Block entspricht dann wieder der allgemeinen Abfolge der drei Schemas[23].
- Zuerst wird das konzeptuelle Schema entworfen, also die „High-Level“-Sicht auf die Datenbank, welche völlig technologieunabhängig ist.
- Wir wollen die Datenbank auf einer abstrakten Ebene modellieren.



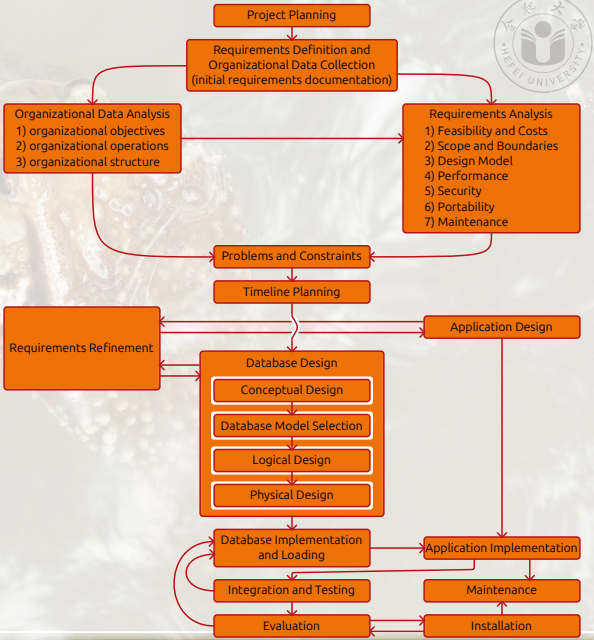
Beispiel Lebenszyklus

- Beide updaten sich gegenseitig.
- Der interne Datenbankdesign-Block entspricht dann wieder der allgemeinen Abfolge der drei Schemas[23].
- Zuerst wird das konzeptuelle Schema entworfen, also die „High-Level“-Sicht auf die Datenbank, welche völlig technologieunabhängig ist.
- Wir wollen die Datenbank auf einer abstrakten Ebene modellieren.
- Dieses Modell basiert auf der Weltsicht und den Prozessen der Stakeholders.



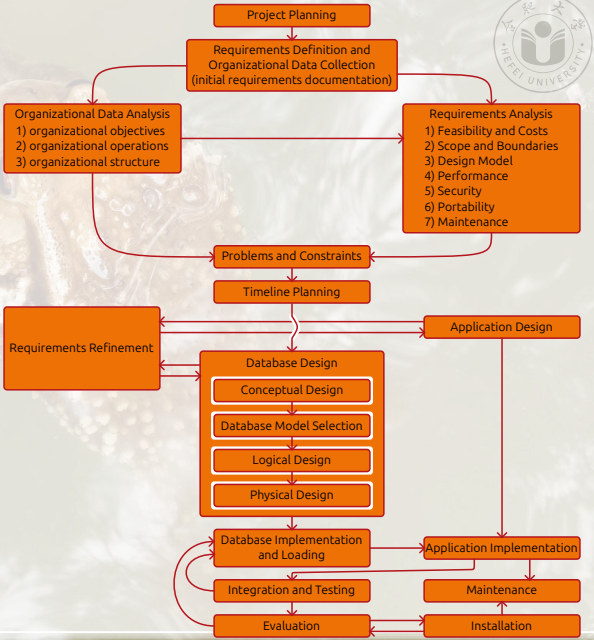
Beispiel Lebenszyklus

- Der interne Datenbankdesign-Block entspricht dann wieder der allgemeinen Abfolge der drei Schemas[23].
- Zuerst wird das konzeptuelle Schema entworfen, also die „High-Level“-Sicht auf die Datenbank, welche völlig technologieunabhängig ist.
- Wir wollen die Datenbank auf einer abstrakten Ebene modellieren.
- Dieses Modell basiert auf der Weltsicht und den Prozessen der Stakeholders.
- Dieses Design kann als ERD modelliert werden[1,83].



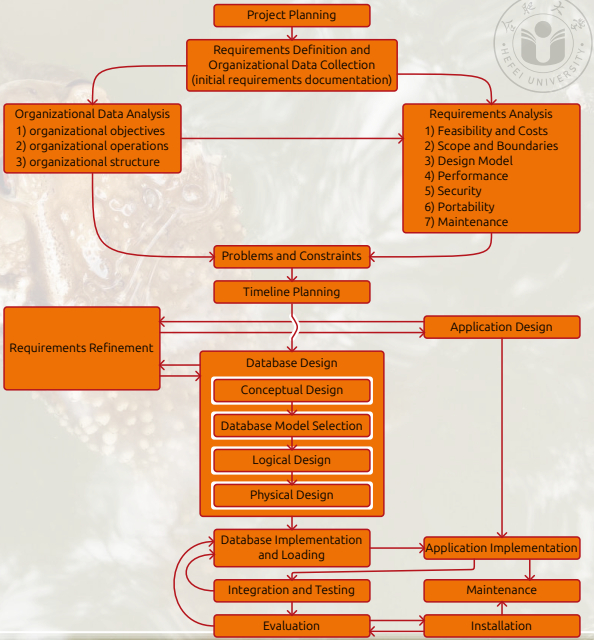
Beispiel Lebenszyklus

- Zuerst wird das konzeptuelle Schema entworfen, also die „High-Level“-Sicht auf die Datenbank, welche völlig technologieunabhängig ist.
- Wir wollen die Datenbank auf einer abstrakten Ebene modellieren.
- Dieses Modell basiert auf der Weltsicht und den Prozessen der Stakeholders.
- Dieses Design kann als ERD modelliert werden[1,83].
- Normalerweise malt man nicht nur *ein* riesiges ERD.



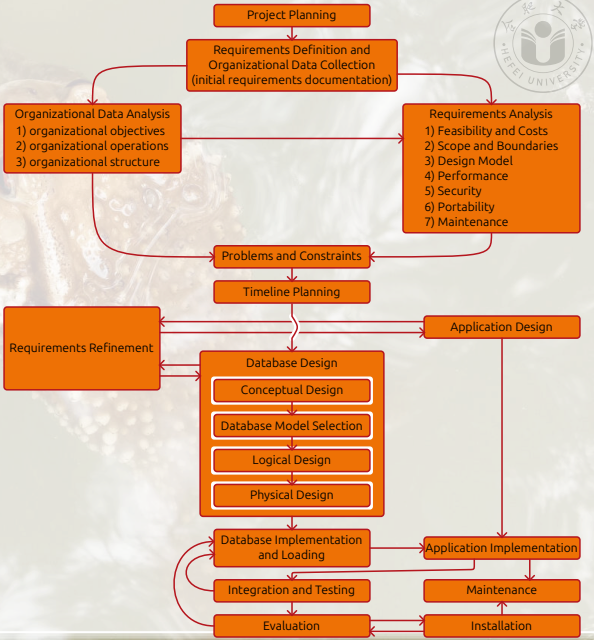
Beispiel Lebenszyklus

- Wir wollen die Datenbank auf einer abstrakten Ebene modellieren.
- Dieses Modell basiert auf der Weltsicht und den Prozessen der Stakeholders.
- Dieses Design kann als ERD modelliert werden[1,83].
- Normalerweise malt man nicht nur *ein* riesiges ERD.
- Stattdessen werden oft mehrere kleinere ERDs gemalt, die jeweils nicht mehr als zehn Entitätstypen beinhalten[83].



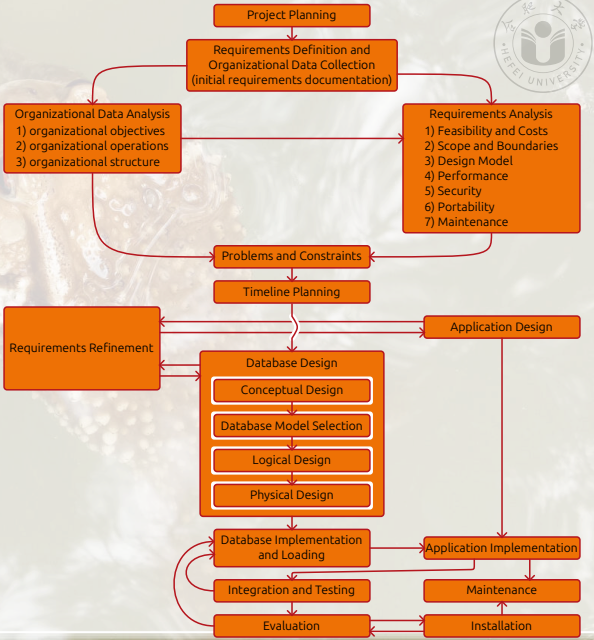
Beispiel Lebenszyklus

- Dieses Modell basiert auf der Weltsicht und den Prozessen der Stakeholders.
- Dieses Design kann als ERD modelliert werden[1,83].
- Normalerweise malt man nicht nur *ein* riesiges ERD.
- Stattdessen werden oft mehrere kleinere ERDs gemalt, die jeweils nicht mehr als zehn Entitätstypen beinhalten[83].
- Dann wird das Datenmodell gewählt.



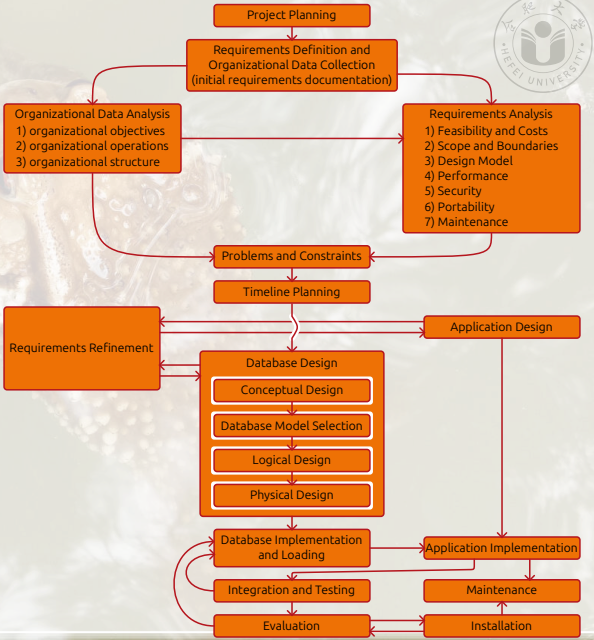
Beispiel Lebenszyklus

- Dieses Design kann als ERD modelliert werden[1,83].
- Normalerweise malt man nicht nur *ein* riesiges ERD.
- Stattdessen werden oft mehrere kleinere ERDs gemalt, die jeweils nicht mehr als zehn Entitätstypen beinhalten[83].
- Dann wird das Datenmodell gewählt.
- Wir fokussieren uns hier auf relationale Datenbanken, wo die Daten in Tabellen gespeichert werden.



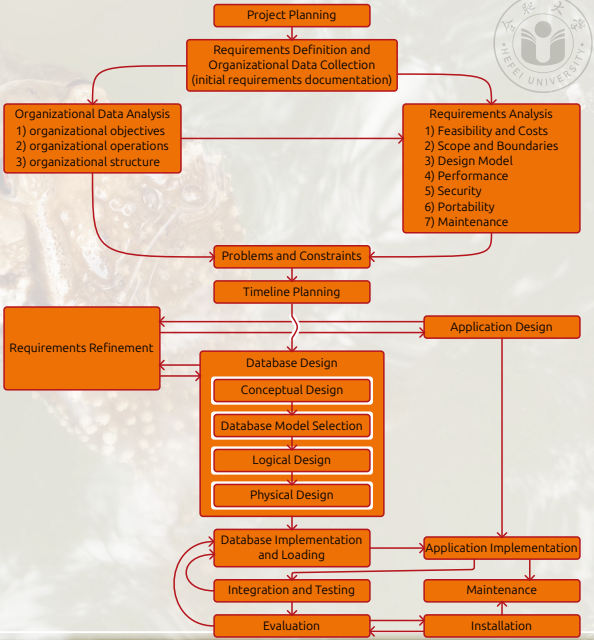
Beispiel Lebenszyklus

- Normalerweise malt man nicht nur *ein* riesiges ERD.
- Stattdessen werden oft mehrere kleinere ERDs gemalt, die jeweils nicht mehr als zehn Entitätstypen beinhalten[83].
- Dann wird das Datenmodell gewählt.
- Wir fokussieren uns hier auf relationale Datenbanken, wo die Daten in Tabellen gespeichert werden.
- Das muss aber nicht immer die richtige Wahl sein.



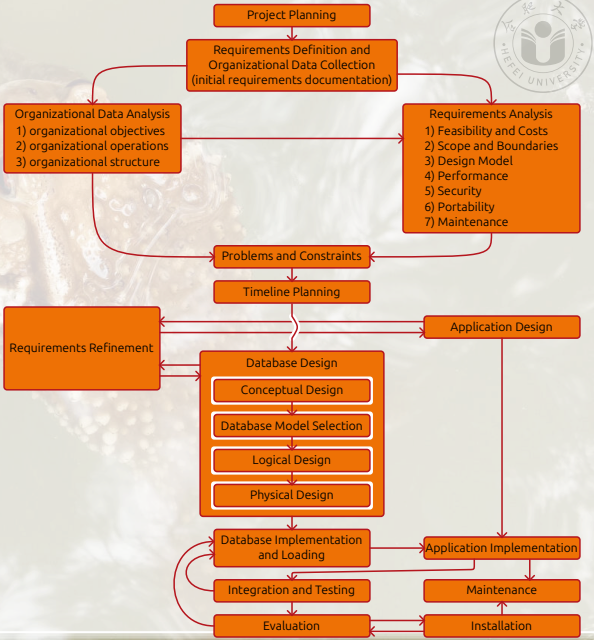
Beispiel Lebenszyklus

- Stattdessen werden oft mehrere kleinere ERDs gemalt, die jeweils nicht mehr als zehn Entitätstypen beinhalten[83].
- Dann wird das Datenmodell gewählt.
- Wir fokussieren uns hier auf relationale Datenbanken, wo die Daten in Tabellen gespeichert werden.
- Das muss aber nicht immer die richtige Wahl sein.
- Manchmal arbeiten wir vielleicht mit Bilddaten oder Videos, oder wir haben es mit Simulationen und Experimenten zu tun.



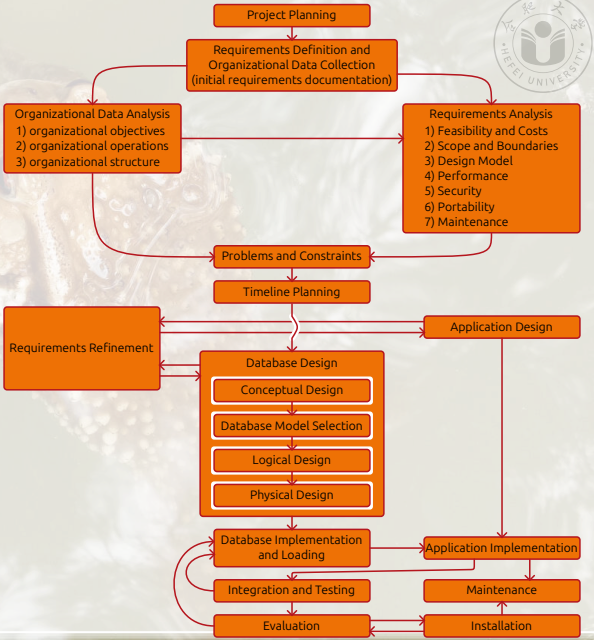
Beispiel Lebenszyklus

- Dann wird das Datenmodell gewählt.
- Wir fokussieren uns hier auf relationale Datenbanken, wo die Daten in Tabellen gespeichert werden.
- Das muss aber nicht immer die richtige Wahl sein.
- Manchmal arbeiten wir vielleicht mit Bilddaten oder Videos, oder wir haben es mit Simulationen und Experimenten zu tun.
- Dann wollen wir vielleicht andere Arten von Datenbanken verwenden.



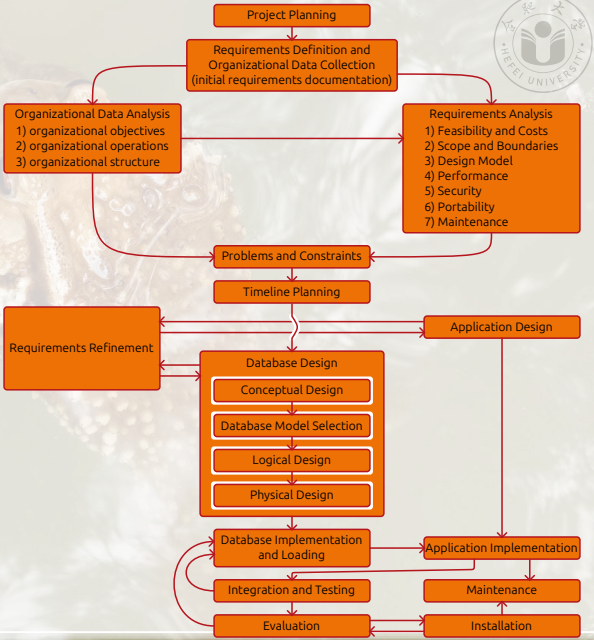
Beispiel Lebenszyklus

- Wir fokussieren uns hier auf relationale Datenbanken, wo die Daten in Tabellen gespeichert werden.
- Das muss aber nicht immer die richtige Wahl sein.
- Manchmal arbeiten wir vielleicht mit Bilddaten oder Videos, oder wir haben es mit Simulationen und Experimenten zu tun.
- Dann wollen wir vielleicht andere Arten von Datenbanken verwenden.
- Lassen Sie uns trotzdem annehmen, dass wir uns für eine relationales Datenbank entschieden haben.



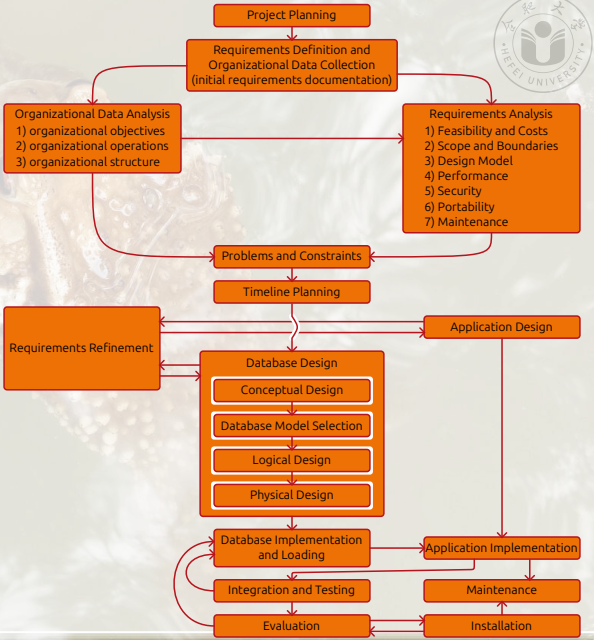
Beispiel Lebenszyklus

- Das muss aber nicht immer die richtige Wahl sein.
- Manchmal arbeiten wir vielleicht mit Bilddaten oder Videos, oder wir haben es mit Simulationen und Experimenten zu tun.
- Dann wollen wir vielleicht andere Arten von Datenbanken verwenden.
- Lassen Sie uns trotzdem annehmen, dass wir uns für eine relationales Datenbank entschieden haben.
- Nun entscheiden wir uns auch gleich für ein DBMS.



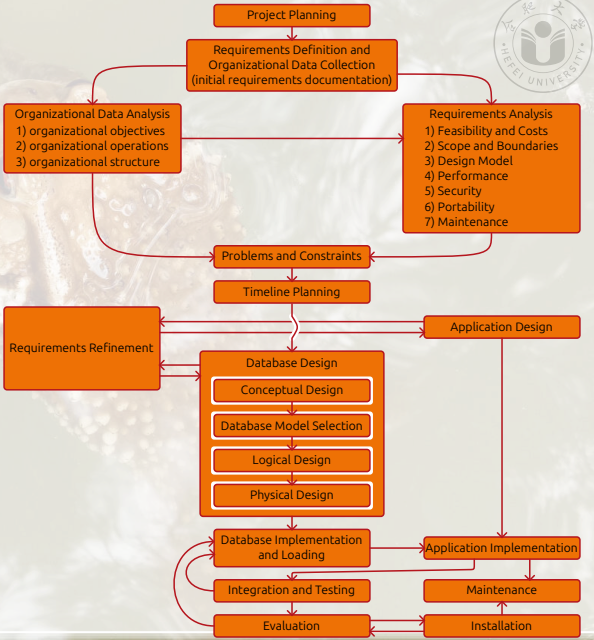
Beispiel Lebenszyklus

- Manchmal arbeiten wir vielleicht mit Bilddaten oder Videos, oder wir haben es mit Simulationen und Experimenten zu tun.
- Dann wollen wir vielleicht andere Arten von Datenbanken verwenden.
- Lassen Sie uns trotzdem annehmen, dass wir uns für eine relationales Datenbank entschieden haben.
- Nun entscheiden wir uns auch gleich für ein DBMS.
- Die Entscheidung wird getroffen basierend auf Kosten, Wrtung, Lizensen, und Trainingsmöglichkeiten.



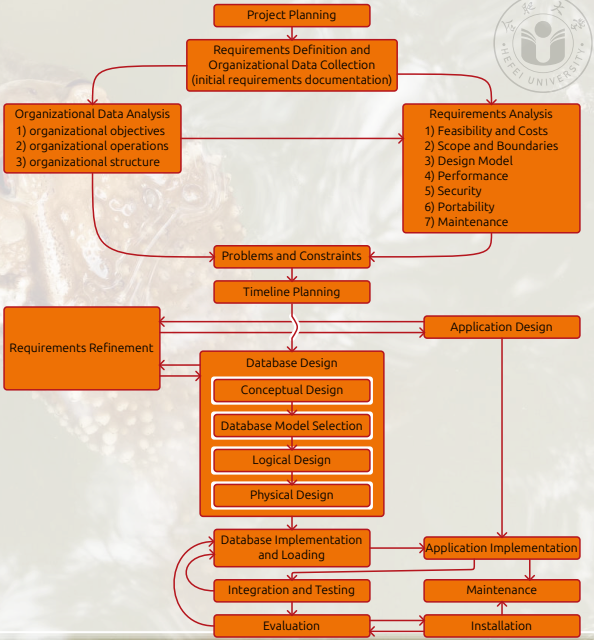
Beispiel Lebenszyklus

- Dann wollen wir vielleicht andere Arten von Datenbanken verwenden.
- Lassen Sie uns trotzdem annehmen, dass wir uns für eine relationales Datenbank entschieden haben.
- Nun entscheiden wir uns auch gleich für ein DBMS.
- Die Entscheidung wird getroffen basierend auf Kosten, Wrtung, Lizensen, und Trainingsmöglichkeiten.
- In unserem Kurs setzen wir auf PostgreSQL, weil es ein kostenloses und sehr weit entwickeltes SQL DBMS ist.



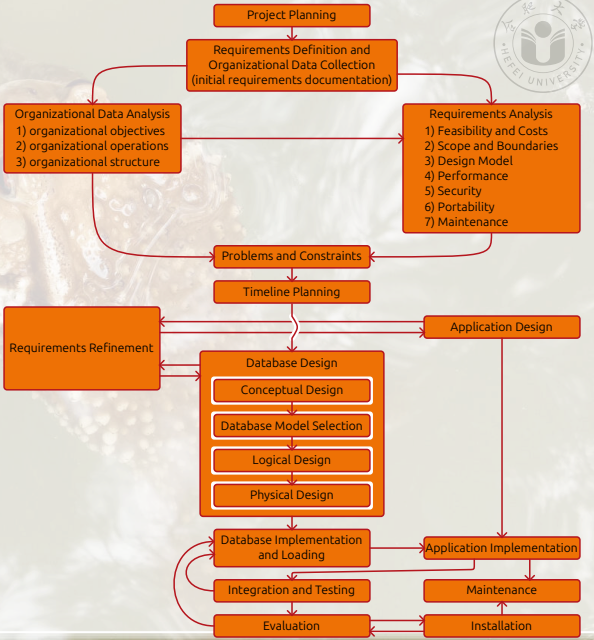
Beispiel Lebenszyklus

- Lassen Sie uns trotzdem annehmen, dass wir uns für eine relationales Datenbank entschieden haben.
- Nun entscheiden wir uns auch gleich für ein DBMS.
- Die Entscheidung wird getroffen basierend auf Kosten, Wrtung, Lizensen, und Trainingsmöglichkeiten.
- In unserem Kurs setzen wir auf PostgreSQL, weil es ein kostenloses und sehr weit entwickeltes SQL DBMS ist.
- Dann wird das konzeptuelle Schmea in ein logisches Schema transformiert.



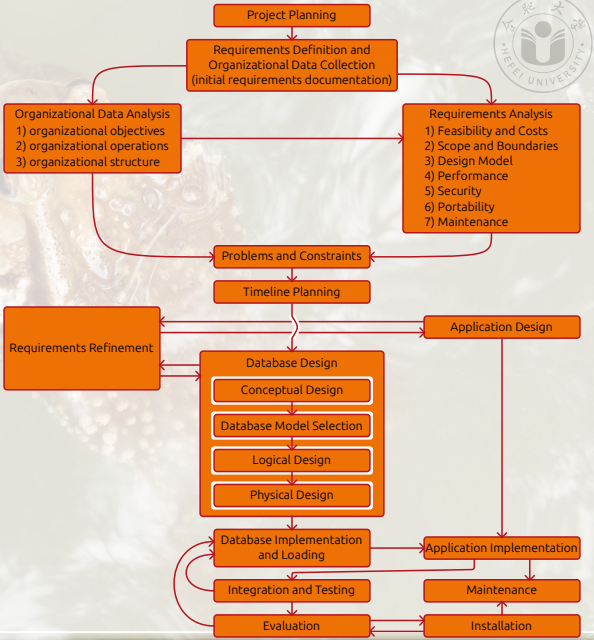
Beispiel Lebenszyklus

- Nun entscheiden wir uns auch gleich für ein DBMS.
- Die Entscheidung wird getroffen basierend auf Kosten, Wrtung, Lizensen, und Trainingsmöglichkeiten.
- In unserem Kurs setzen wir auf PostgreSQL, weil es ein kostenloses und sehr weit entwickeltes SQL DBMS ist.
- Dann wird das konzeptuelle Schmea in ein logisches Schema transformiert.
- Dabei werden ggf. alle Entitätstypen aus den ERDs des konzeptuellen Models in Tabellen und die Beziehungen zwischen ihnen abgebildet[58,60].



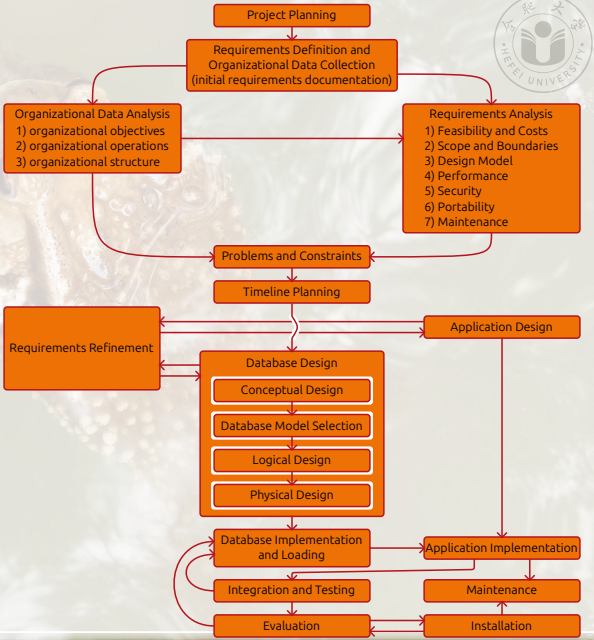
Beispiel Lebenszyklus

- Die Entscheidung wird getroffen basierend auf Kosten, Wrtung, Lizensen, und Trainingsmöglichkeiten.
- In unserem Kurs setzen wir auf PostgreSQL, weil es ein kostenloses und sehr weit entwickeltes SQL DBMS ist.
- Dann wird das konzeptuelle Schmea in ein logisches Schema transformiert.
- Dabei werden ggf. alle Entitätstypen aus den ERDs des konzeptuellen Models in Tabellen und die Beziehungen zwischen ihnen abgebildet[58,60].
- Hierbei führen wir Anpassungen durch, um die Effizent zu verbessern.



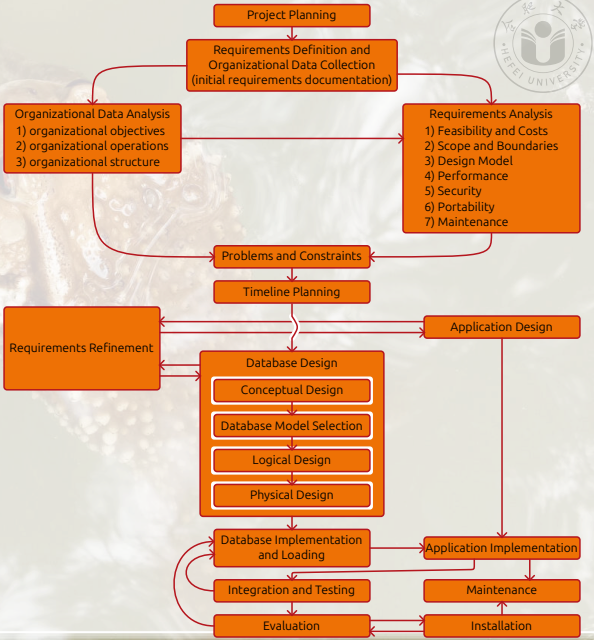
Beispiel Lebenszyklus

- Dann wird das konzeptuelle Schmea in ein logisches Schema transformiert.
- Dabei werden ggf. alle Entitätstypen aus den ERDs des konzeptuellen Models in Tabellen und die Beziehungen zwischen ihnen abgebildet[58,60].
- Hierbei führen wir Anpassungen durch, um die Effizent zu verbessern.
- Z. B. teilen wir wir dabei manchmal einen Entitätstypen in mehrere Tabellen auf oder kombinieren mehrere Entitätstypen in eine Tabelle, manchmal benutzen wir auch andere Primärschlüssel, usw.



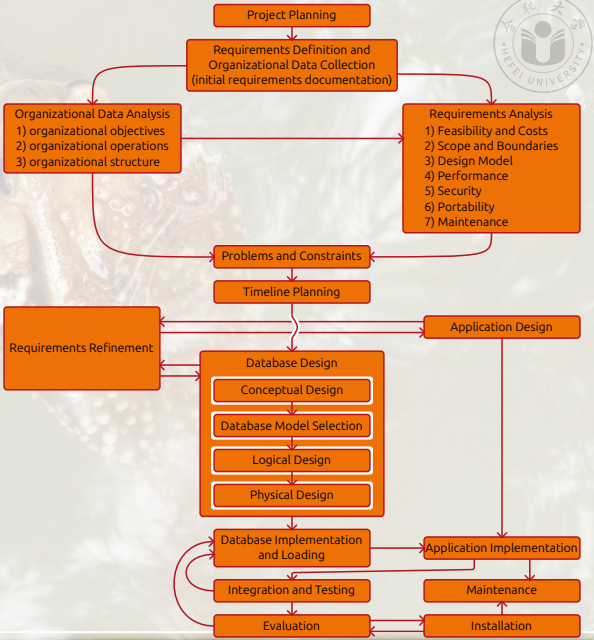
Beispiel Lebenszyklus

- Dabei werden ggf. alle Entitätstypen aus den ERDs des konzeptuellen Models in Tabellen und die Beziehungen zwischen ihnen abgebildet[58,60].
- Hierbei führen wir Anpassungen durch, um die Effizent zu verbessern.
- Z. B. teilen wir wir dabei manchmal einen Entitätstypen in mehrere Tabellen auf oder kombinieren mehrere Entitätstypen in eine Tabelle, manchmal benutzen wir auch andere Primärschlüssel, usw.
- Wir bilden also das konzeptuelle Modell auf die interne Struktur ab, die uns das DBMS anbietet.



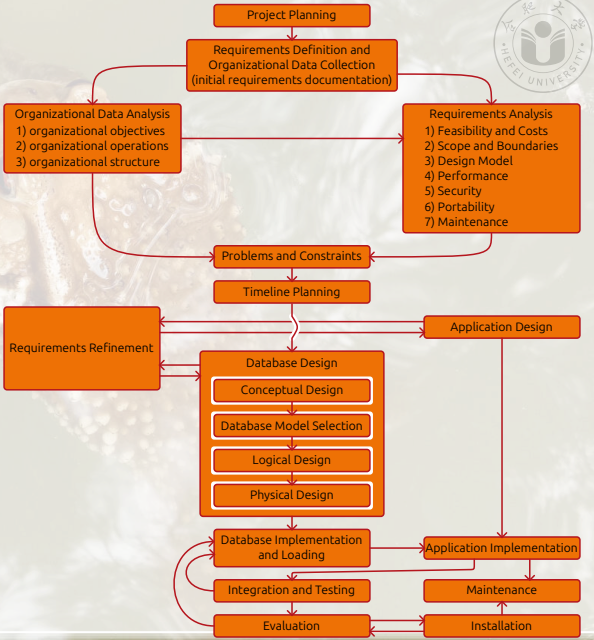
Beispiel Lebenszyklus

- Hierbei führen wir Anpassungen durch, um die Effizent zu verbessern.
- Z. B. teilen wir wir dabei manchmal einen Entitätstypen in mehrere Tabellen auf oder kombinieren mehrere Entitätstypen in eine Tabelle, manchmal benutzen wir auch andere Primärschlüssel, usw.
- Wir bilden also das konzeptuelle Modell auf die interne Struktur ab, die uns das DBMS anbietet.
- Wir haben nun ein unoptimiertes Design unserer Datenbank.



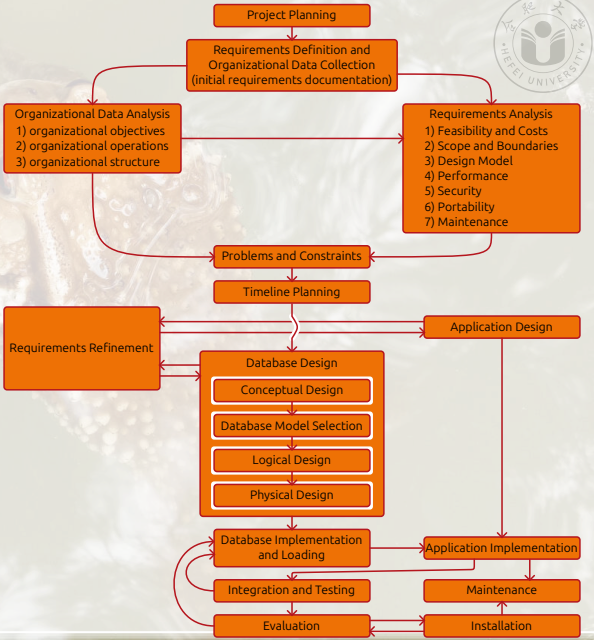
Beispiel Lebenszyklus

- Hierbei führen wir Anpassungen durch, um die Effizent zu verbessern.
- Z. B. teilen wir wir dabei manchmal einen Entitätstypen in mehrere Tabellen auf oder kombinieren mehrere Entitätstypen in eine Tabelle, manchmal benutzen wir auch andere Primärschlüssel, usw.
- Wir bilden also das konzeptuelle Modell auf die interne Struktur ab, die uns das DBMS anbietet.
- Wir haben nun ein unoptimiertes Design unserer Datenbank.
- Das logische Schema wird dann in ein physisches Schema übersetzt.



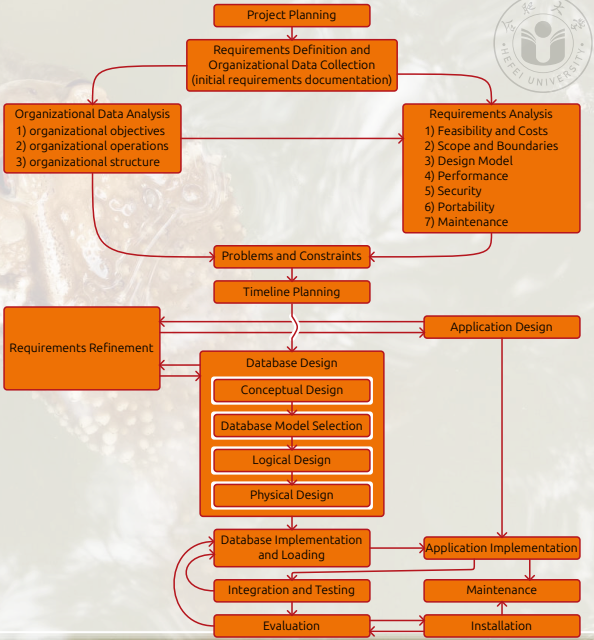
Beispiel Lebenszyklus

- Z. B. teilen wir wir dabei manchmal einen Entitätstypen in mehrere Tabellen auf oder kombinieren mehrere Entitätstypen in eine Tabelle, manchmal benutzen wir auch andere Primärschlüssel, usw.
- Wir bilden also das konzeptuelle Modell auf die interne Struktur ab, die uns das DBMS anbietet.
- Wir haben nun ein unoptimiertes Design unserer Datenbank.
- Das logische Schema wird dann in ein physisches Schema übersetzt.
- Dieser Schritt fokussiert sich auf die internen Aspekte der Datenbank, z. B., Zugriffspfade, Indexe, Partitionen, usw.



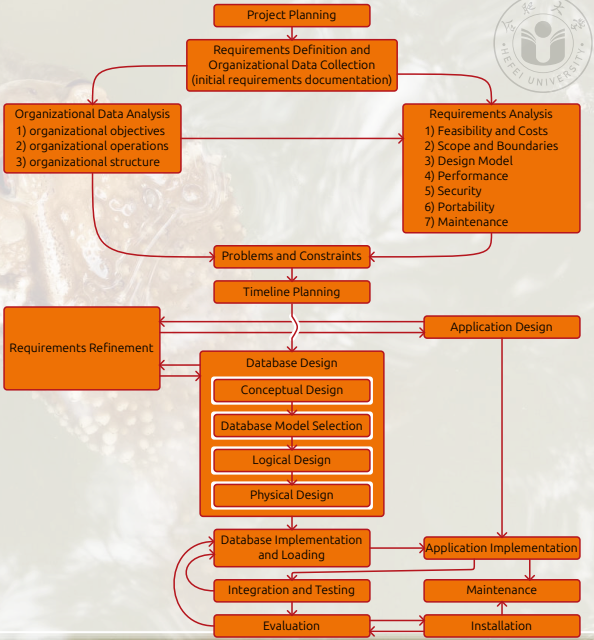
Beispiel Lebenszyklus

- Wir bilden also das konzeptuelle Modell auf die interne Struktur ab, die uns das DBMS anbietet.
- Wir haben nun ein unoptimiertes Design unserer Datenbank.
- Das logische Schema wird dann in ein physisches Schema übersetzt.
- Dieser Schritt fokussiert sich auf die internen Aspekte der Datenbank, z. B., Zugriffspfade, Indexe, Partitionen, usw.
- Danach haben wir ein komplettes Datenbankdesign, optimiert für die geplanten Operationen.



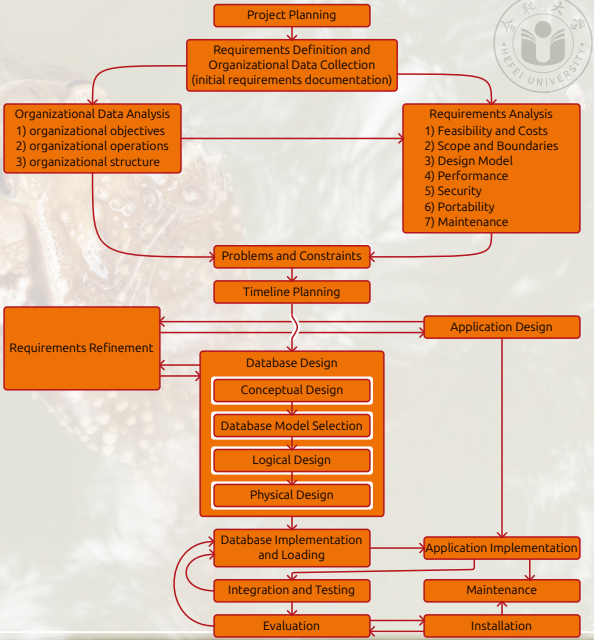
Beispiel Lebenszyklus

- Wir haben nun ein unoptimiertes Design unserer Datenbank.
- Das logische Schema wird dann in ein physisches Schema übersetzt.
- Dieser Schritt fokussiert sich auf die internen Aspekte der Datenbank, z. B., Zugriffspfade, Indexe, Partitionen, usw.
- Danach haben wir ein komplettes Datenbankdesign, optimiert für die geplanten Operationen.
- Die Unterteilung in konzeptuelles, logisches, und physisches Schema wird nicht immer so durchgezogen.



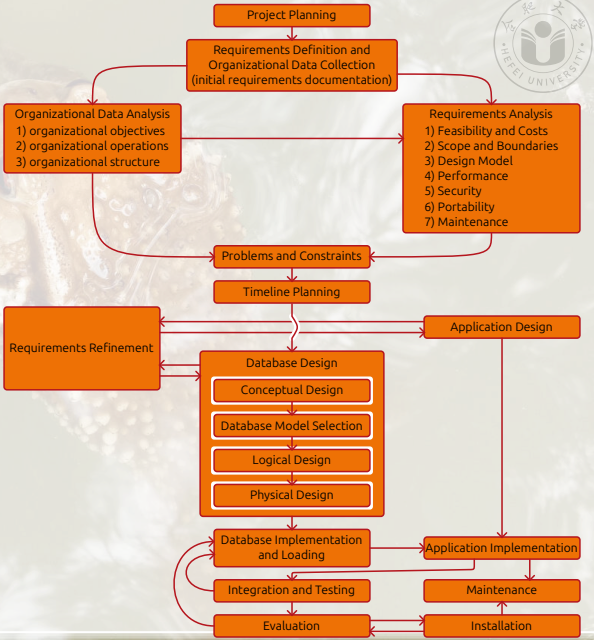
Beispiel Lebenszyklus

- Das logische Schema wird dann in ein physisches Schema übersetzt.
- Dieser Schritt fokussiert sich auf die internen Aspekte der Datenbank, z. B., Zugriffspfade, Indexe, Partitionen, usw.
- Danach haben wir ein komplettes Datenbankdesign, optimiert für die geplanten Operationen.
- Die Unterteilung in konzeptuelles, logisches, und physisches Schema wird nicht immer so durchgezogen.
- Während dem Datenbankentwurf muss immer mit den Stakeholders des Projekts gesprochen werden.



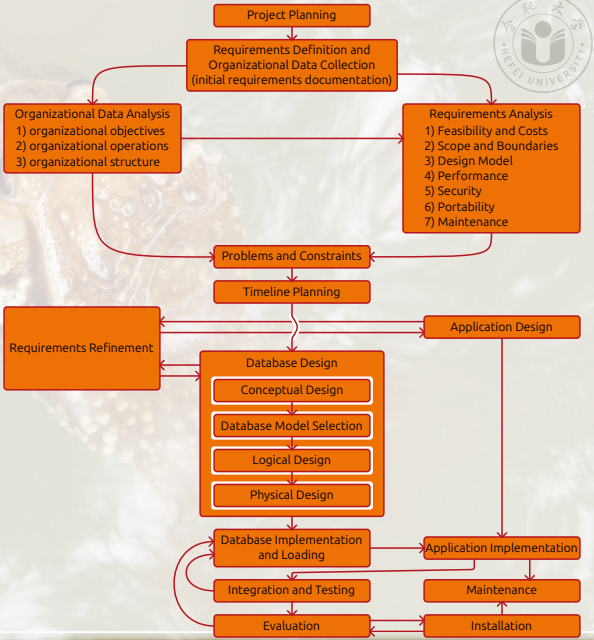
Beispiel Lebenszyklus

- Dieser Schritt fokussiert sich auf die internen Aspekte der Datenbank, z. B., Zugriffspfade, Indexe, Partitionen, usw.
- Danach haben wir ein komplettes Datenbankdesign, optimiert für die geplanten Operationen.
- Die Unterteilung in konzeptuelles, logisches, und physisches Schema wird nicht immer so durchgezogen.
- Während dem Datenbankentwurf muss immer mit den Stakeholders des Projekts gesprochen werden.
- Manchmal ändern sich dadurch die Anforderungen, was dann wieder ordentlich dokumentiert werden muss.



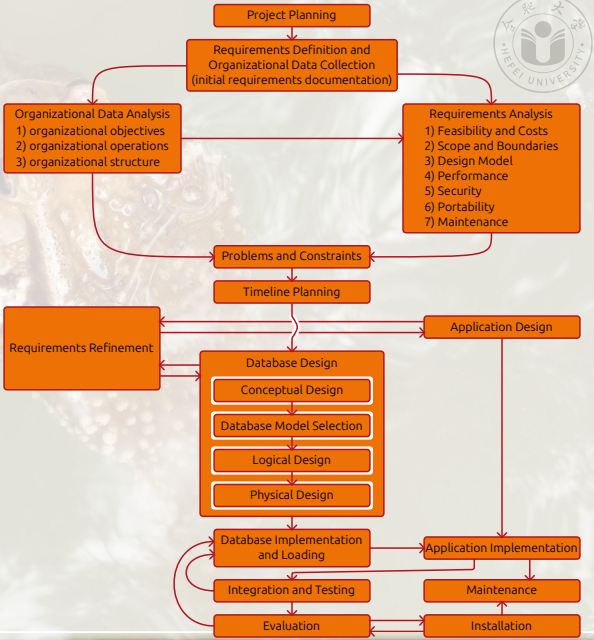
Beispiel Lebenszyklus

- Danach haben wir ein komplettes Datenbankdesign, optimiert für die geplanten Operationen.
- Die Unterteilung in konzeptuelles, logisches, und physisches Schema wird nicht immer so durchgezogen.
- Während dem Datenbankentwurf muss immer mit den Stakeholders des Projekts gesprochen werden.
- Manchmal ändern sich dadurch die Anforderungen, was dann wieder ordentlich dokumentiert werden muss.
- Währen die Datenbank entwickelt wird, wird gleichzeitig die Applikation für die Benutzer ebenfalls entworfen.



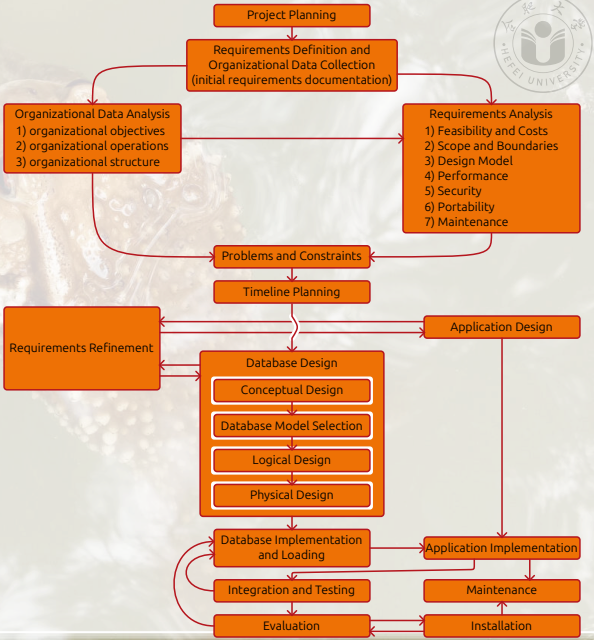
Beispiel Lebenszyklus

- Die Unterteilung in konzeptuelles, logisches, und physisches Schema wird nicht immer so durchgezogen.
- Während dem Datenbankentwurf muss immer mit den Stakeholders des Projekts gesprochen werden.
- Manchmal ändern sich dadurch die Anforderungen, was dann wieder ordentlich dokumentiert werden muss.
- Währen die Datenbank entwickelt wird, wird gleichzeitig die Applikation für die Benutzer ebenfalls entworfen.
- Auch hierbei können notwendige Änderungen durch Diskussionen mit den Benutzern entdeckt werden.



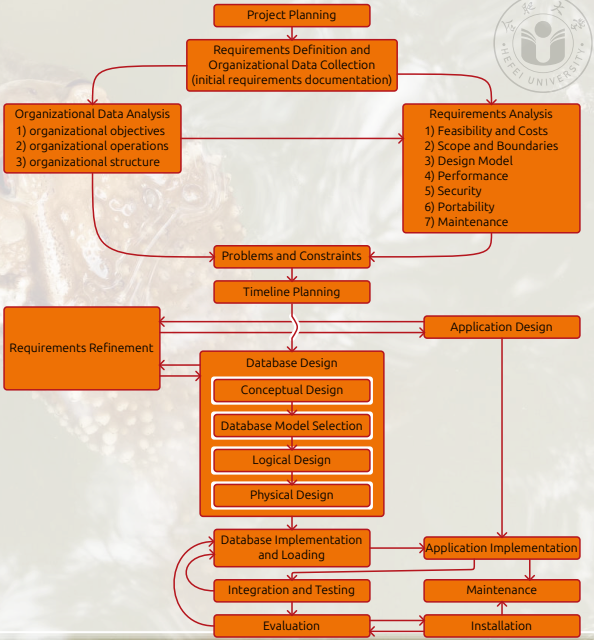
Beispiel Lebenszyklus

- Während dem Datenbankentwurf muss immer mit den Stakeholders des Projekts gesprochen werden.
- Manchmal ändern sich dadurch die Anforderungen, was dann wieder ordentlich dokumentiert werden muss.
- Währen die Datenbank entwickelt wird, wird gleichzeitig die Applikation für die Benutzer ebenfalls entworfen.
- Auch hierbei können notwendige Änderungen durch Diskussionen mit den Benutzern entdeckt werden.
- Nach dem Entwurf wird die Datenbank auf Basis des Designdokuments entwickelt.



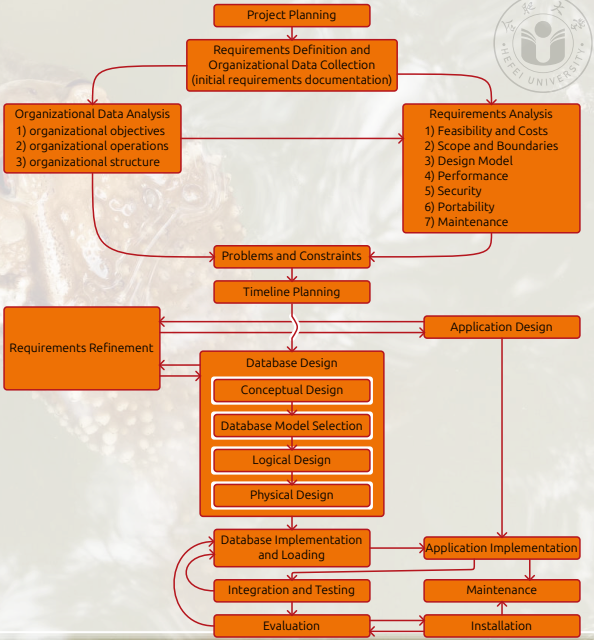
Beispiel Lebenszyklus

- Manchmal ändern sich dadurch die Anforderungen, was dann wieder ordentlich dokumentiert werden muss.
- Währen die Datenbank entwickelt wird, wird gleichzeitig die Applikation für die Benutzer ebenfalls entworfen.
- Auch hierbei können notwendige Änderungen durch Diskussionen mit den Benutzern entdeckt werden.
- Nach dem Entwurf wird die Datenbank auf Basis des Designdokuments entwickelt.
- Tabellen werden erstellt und mit Daten gefüllt, Einschränkungen und Anfragen werden implementiert.



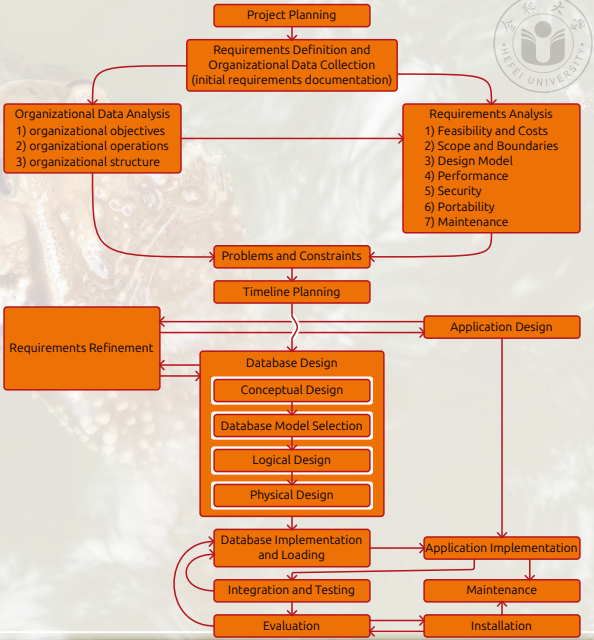
Beispiel Lebenszyklus

- Währen die Datenbank entwickelt wird, wird gleichzeitig die Applikation für die Benutzer ebenfalls entworfen.
- Auch hierbei können notwendige Änderungen durch Diskussionen mit den Benutzern entdeckt werden.
- Nach dem Entwurf wird die Datenbank auf Basis des Designdokuments entwickelt.
- Tabellen werden erstellt und mit Daten gefüllt, Einschränkungen und Anfragen werden implementiert.
- Nun da die Datenbank existiert kann auch die eigentliche Anwendung (auf Basis der Entwürfe) entwickelt werden.



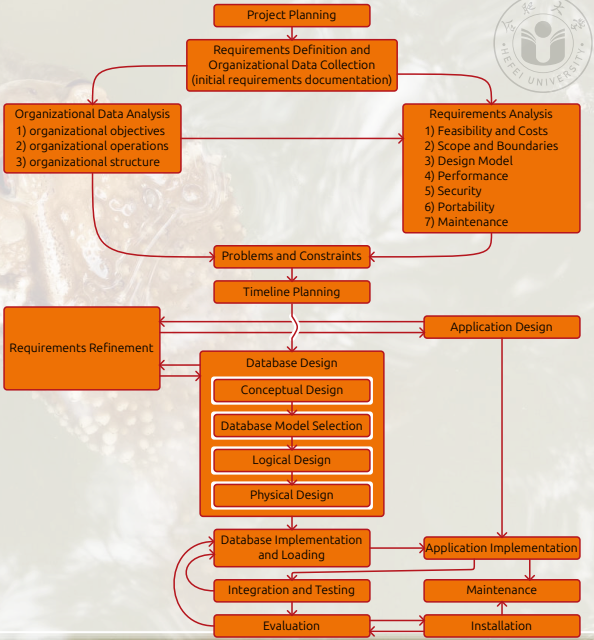
Beispiel Lebenszyklus

- Auch hierbei können notwendige Änderungen durch Diskussionen mit den Benutzern entdeckt werden.
- Nach dem Entwurf wird die Datenbank auf Basis des Designdokuments entwickelt.
- Tabellen werden erstellt und mit Daten gefüllt, Einschränkungen und Anfragen werden implementiert.
- Nun da die Datenbank existiert kann auch die eigentliche Anwendung (auf Basis der Entwürfe) entwickelt werden.
- Sie wird dann mit der Datenbank integriert.



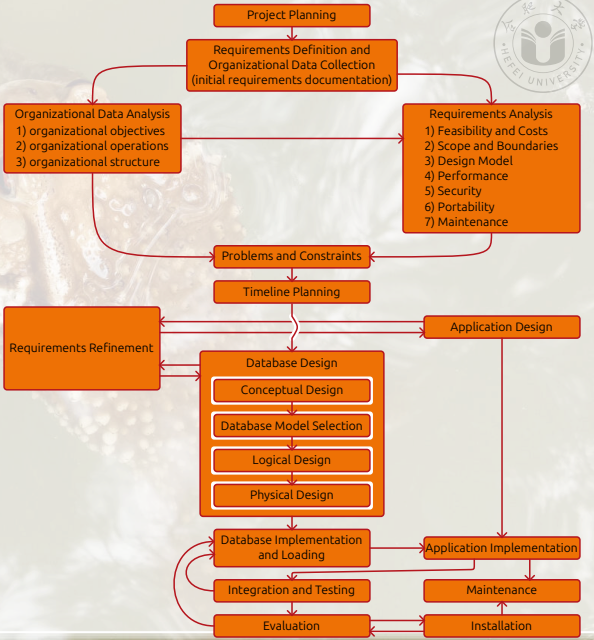
Beispiel Lebenszyklus

- Auch hierbei können notwendige Änderungen durch Diskussionen mit den Benutzern entdeckt werden.
- Nach dem Entwurf wird die Datenbank auf Basis des Designdokuments entwickelt.
- Tabellen werden erstellt und mit Daten gefüllt, Einschränkungen und Anfragen werden implementiert.
- Nun da die Datenbank existiert kann auch die eigentliche Anwendung (auf Basis der Entwürfe) entwickelt werden.
- Sie wird dann mit der Datenbank integriert.
- Das System wird getestet.



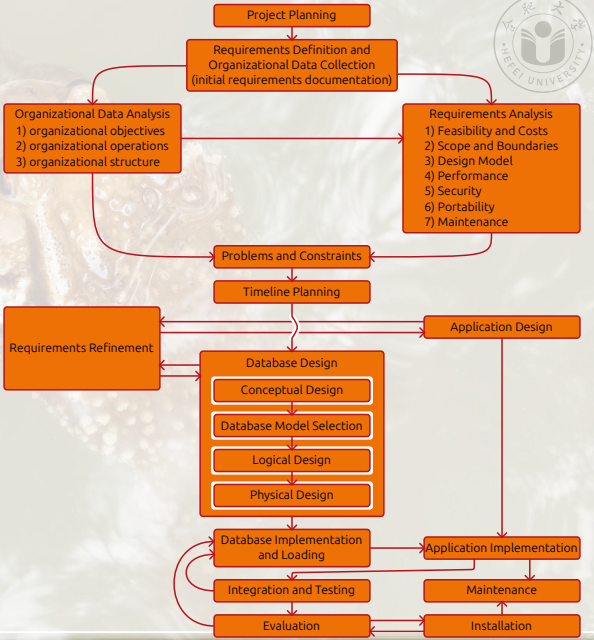
Beispiel Lebenszyklus

- Nach dem Entwurf wird die Datenbank auf Basis des Designdokuments entwickelt.
- Tabellen werden erstellt und mit Daten gefüllt, Einschränkungen und Anfragen werden implementiert.
- Nun da die Datenbank existiert kann auch die eigentliche Anwendung (auf Basis der Entwürfe) entwickelt werden.
- Sie wird dann mit der Datenbank integriert.
- Das System wird getestet.
- Nun wird das System installiert.



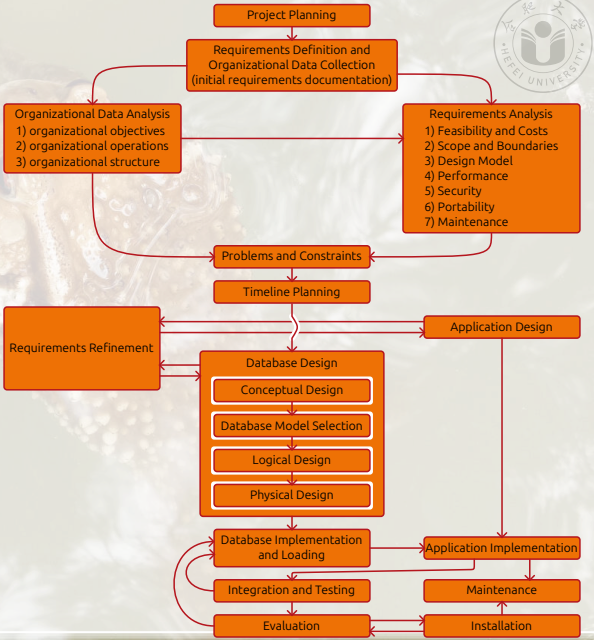
Beispiel Lebenszyklus

- Nach dem Entwurf wird die Datenbank auf Basis des Designdokuments entwickelt.
- Tabellen werden erstellt und mit Daten gefüllt, Einschränkungen und Anfragen werden implementiert.
- Nun da die Datenbank existiert kann auch die eigentliche Anwendung (auf Basis der Entwürfe) entwickelt werden.
- Sie wird dann mit der Datenbank integriert.
- Das System wird getestet.
- Nun wird das System installiert.
- Die Benutzer können es ausprobieren.



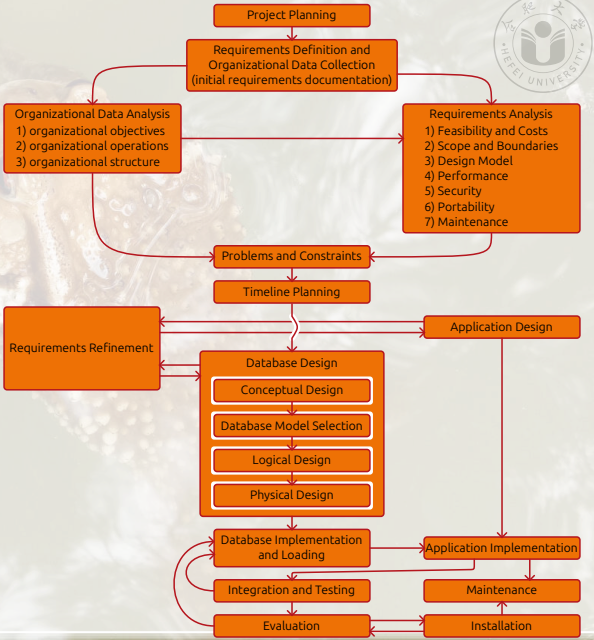
Beispiel Lebenszyklus

- Tabellen werden erstellt und mit Daten gefüllt, Einschränkungen und Anfragen werden implementiert.
- Nun da die Datenbank existiert kann auch die eigentliche Anwendung (auf Basis der Entwürfe) entwickelt werden.
- Sie wird dann mit der Datenbank integriert.
- Das System wird getestet.
- Nun wird das System installiert.
- Die Benutzer können es ausprobieren.
- Die Stakeholder bewerten die Anwendungen im Hinblick auf Funktionalität und Performanz.



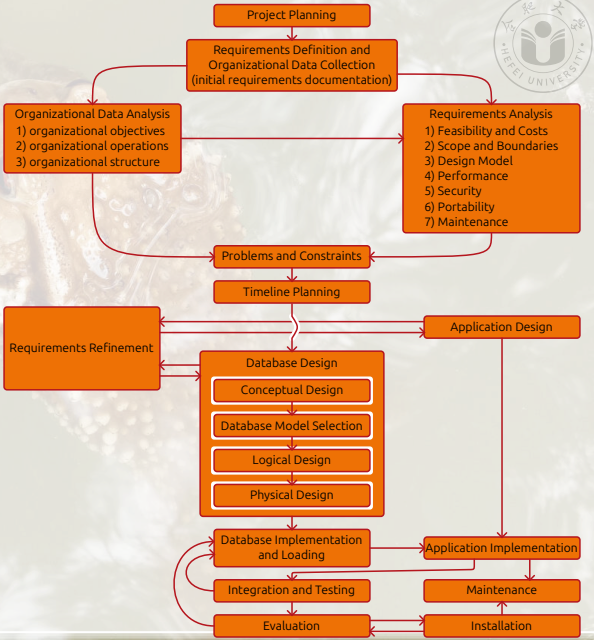
Beispiel Lebenszyklus

- Nun da die Datenbank existiert kann auch die eigentliche Anwendung (auf Basis der Entwürfe) entwickelt werden.
- Sie wird dann mit der Datenbank integriert.
- Das System wird getestet.
- Nun wird das System installiert.
- Die Benutzer können es ausprobieren.
- Die Stakeholder bewerten die Anwendungen im Hinblick auf Funktionalität und Performanz.
- Nachdem das System akzeptiert wurde, betritt es die Produktiv-Phase und wird von nun an regelmäßig gewartet.



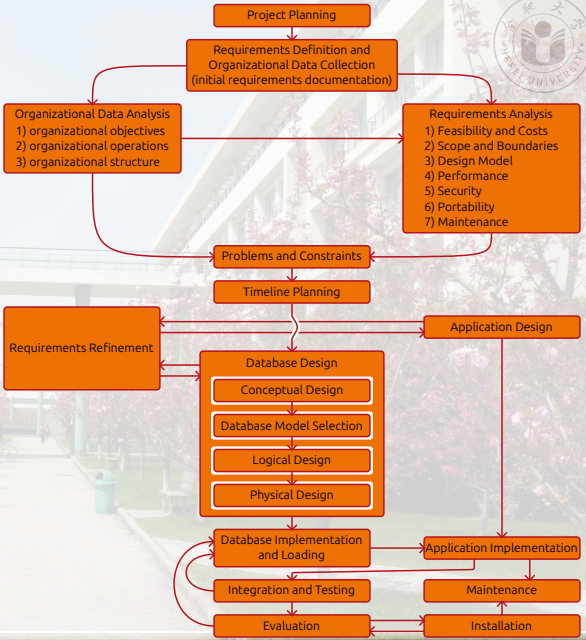
Beispiel Lebenszyklus

- Sie wird dann mit der Datenbank integriert.
- Das System wird getestet.
- Nun wird das System installiert.
- Die Benutzer können es ausprobieren.
- Die Stakeholder bewerten die Anwendungen im Hinblick auf Funktionalität und Performanz.
- Nachdem das System akzeptiert wurde, betritt es die Produktiv-Phase und wird von nun an regelmäßig gewartet.
- Das System wird kontinuierlich upgedated, verbessert und erweitert, bis es irgendwann sein Lebensende erreicht.



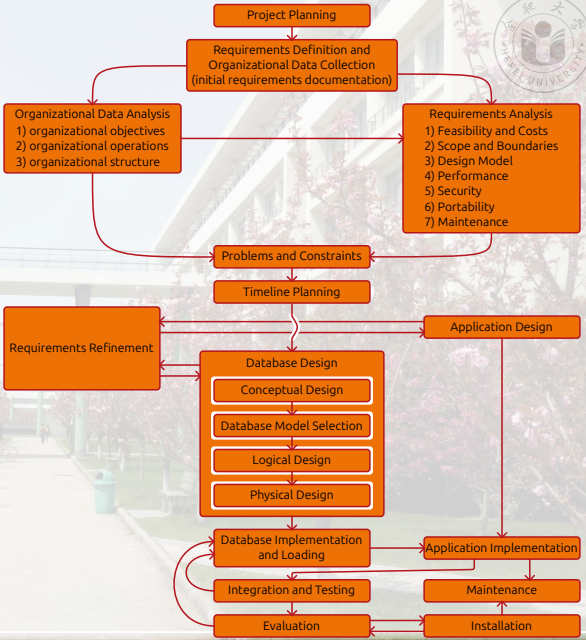
Kern des Designs

- Die Methode von Gupta, Mata-Toledo und Monger[29] ist ein umfassender Ansatz für das Ko-Design von Datenbank und Software in großen Projekten.



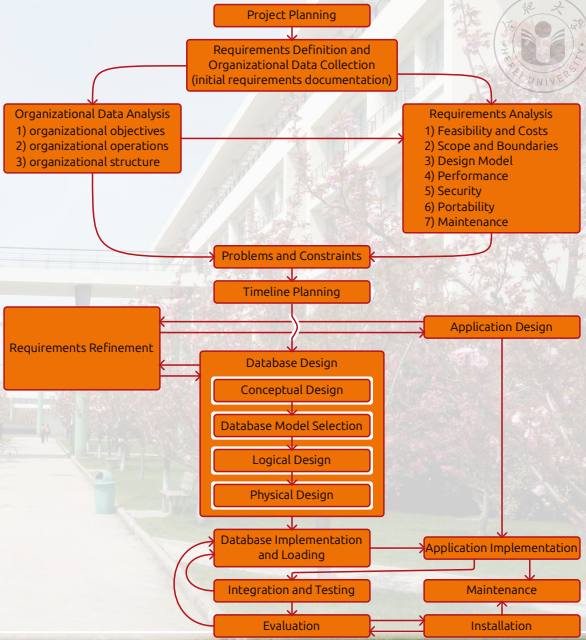
Kern des Designs

- Die Methode von Gupta, Mata-Toledo und Monger[29] ist ein umfassender Ansatz für das Ko-Design von Datenbank und Software in großen Projekten.
- Das Hauptziel ist es, Datanbankprojekte besser vorhersehbar zu machen.



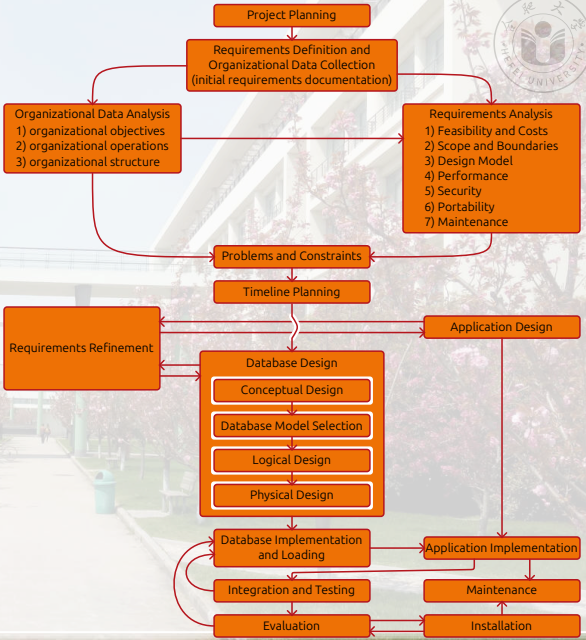
Kern des Designs

- Die Methode von Gupta, Mata-Toledo und Monger[29] ist ein umfassender Ansatz für das Ko-Design von Datenbank und Software in großen Projekten.
- Das Hauptziel ist es, Datanbankprojekte besser vorhersehbar zu machen.
- Die meisten Diskussionen über Datenbankdesign fükussieren sich eher auf den Kern des Entwicklungsprozesses, der hier *Database Design* genannt wird.



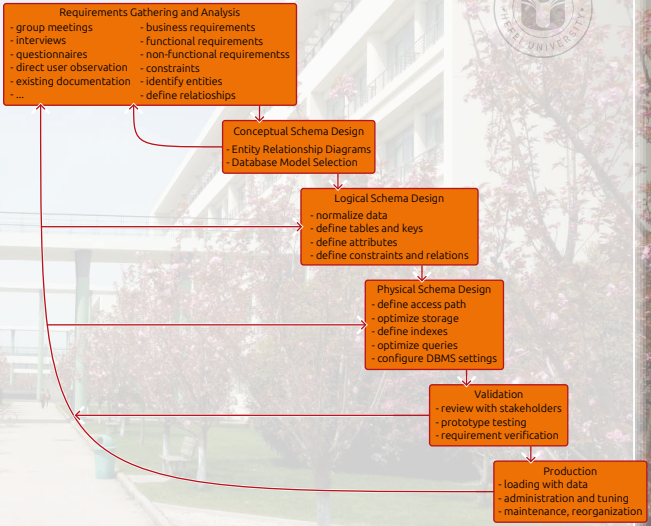
Kern des Designs

- Die Methode von Gupta, Mata-Toledo und Monger[29] ist ein umfassender Ansatz für das Ko-Design von Datenbank und Software in großen Projekten.
- Das Hauptziel ist es, Datanbankprojekte besser vorhersehbar zu machen.
- Die meisten Diskussionen über Datenbankdesign fükussieren sich eher auf den Kern des Entwicklungsprozesses, der hier *Database Design* genannt wird.
- Sie kombinieren auch viele der Rahmentätigkeiten wie Installation und Wartung in eine Aktivität.



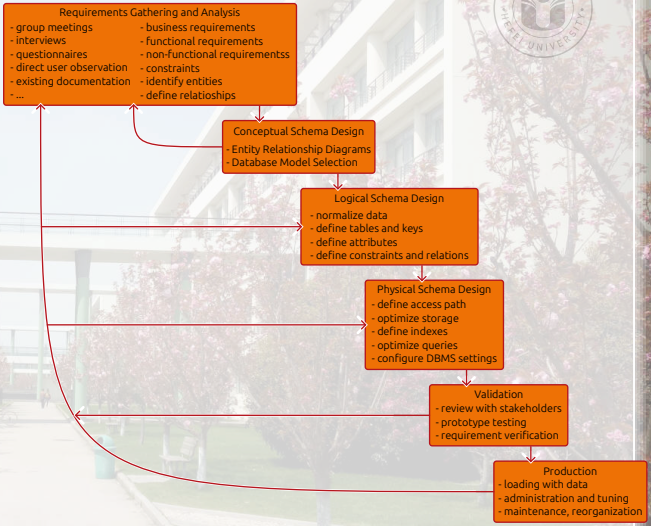
Kern des Designs

- Das Hauptziel ist es, Datanbankprojekte besser vorhersehbar zu machen.
- Die meisten Diskussionen über Datenbankdesign fükussieren sich eher auf den Kern des Entwicklungsprozesses, der hier *Database Design* genannt wird.
- Sie kombinieren auch viele der Rahmentätigkeiten wie Installation und Wartung in eine Aktivität.
- Wenn wir den Designprozess für Datenbanken weiter erforschen und ausprobieren, werden auch wir weniger Schritte verwenden.



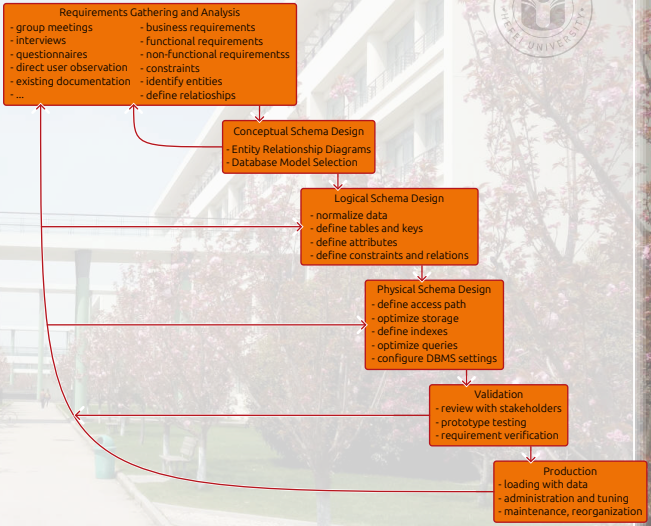
Kern des Designs

- Das Hauptziel ist es, Datanbankprojekte besser vorhersehbar zu machen.
- Die meisten Diskussionen über Datenbankdesign fükussieren sich eher auf den Kern des Entwicklungsprozesses, der hier *Database Design* genannt wird.
- Sie kombinieren auch viele der Rahmentätigkeiten wie Installation und Wartung in eine Aktivität.
- Wenn wir den Designprozess für Datenbanken weiter erforschen und ausprobieren, werden auch wir weniger Schritte verwenden.
- Unser vereinfachtes Modell des Datanbanklebenszyklus kombiniert Ideen verschiedener Quellen[21,58,60].



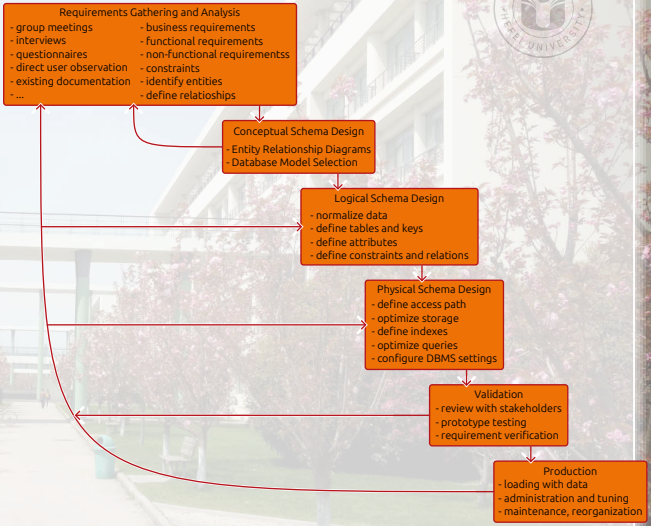
Kern des Designs

- Die meisten Diskussionen über Datenbankdesign fükussieren sich eher auf den Kern des Entwicklungsprozesses, der hier *Database Design* genannt wird.
- Sie kombinieren auch viele der Rahmentätigkeiten wie Installation und Wartung in eine Aktivität.
- Wenn wir den Designprozess für Datenbanken weiter erforschen und ausprobieren, werden auch wir weniger Schritte verwenden.
- Unser vereinfachtes Modell des Datanbanklebenszyklus kombiniert Ideen verschiedener Quellen[21,58,60].
- Wir beginnen mit dem Sammeln und Analysieren von Anforderungen.



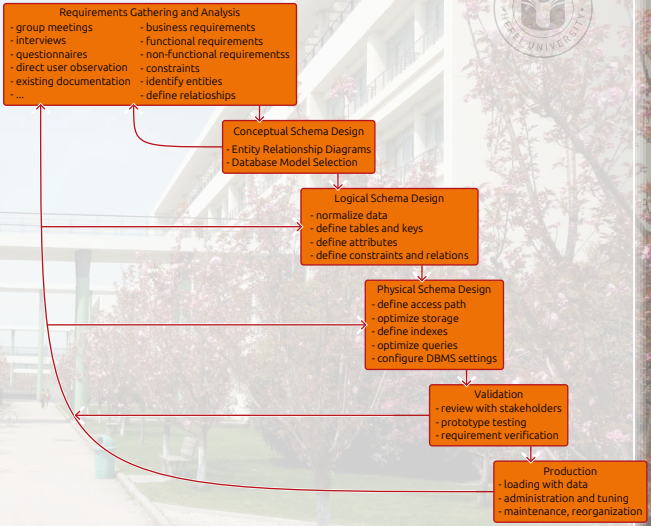
Kern des Designs

- Sie kombinieren auch viele der Rahmentätigkeiten wie Installation und Wartung in eine Aktivität.
- Wenn wir den Designprozess für Datenbanken weiter erforschen und ausprobieren, werden auch wir weniger Schritte verwenden.
- Unser vereinfachtes Modell des Datanbanklebenszyklus kombiniert Ideen verschiedener Quellen[21,58,60].
- Wir beginnen mit dem Sammeln und Analysieren von Anforderungen.
- Dann werden die drei Schemas (konzeptuell, logisch, und physisch) eines nach dem anderen entworfen[23].



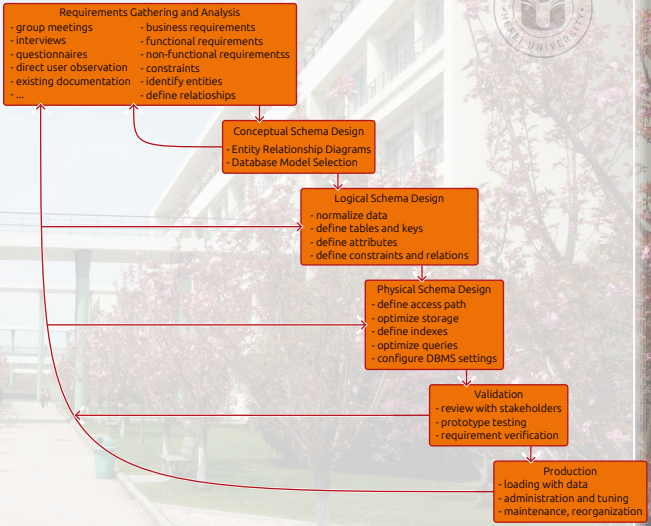
Kern des Designs

- Wenn wir den Designprozess für Datenbanken weiter erforschen und ausprobieren, werden auch wir weniger Schritte verwenden.
- Unser vereinfachtes Modell des Datanbanklebenszyklus kombiniert Ideen verschiedener Quellen[21,58,60].
- Wir beginnen mit dem Sammeln und Analysieren von Anforderungen.
- Dann werden die drei Schemas (konzeptuell, logisch, und physisch) eines nach dem anderen entworfen[23].
- Wir beginnen mit dem konzeptuellen Model, wobei Entitäten und ihre Beziehungen mit ERDs illustriert werden.



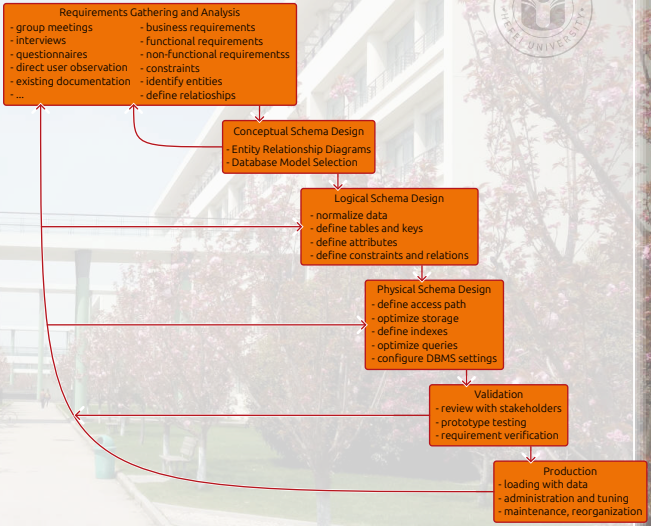
Kern des Designs

- Unser vereinfachtes Modell des Datanbanklebenszyklus kombiniert Ideen verschiedener Quellen[21,58,60].
- Wir beginnen mit dem Sammeln und Analysieren von Anforderungen.
- Dann werden die drei Schemas (konzeptuell, logisch, und physisch) eines nach dem anderen entworfen[23].
- Wir beginnen mit dem konzeptuellen Model, wobei Entitäten und ihre Beziehungen mit ERDs illustriert werden.
- Dieses Modell wird mit den Stakeholders diskutiert.



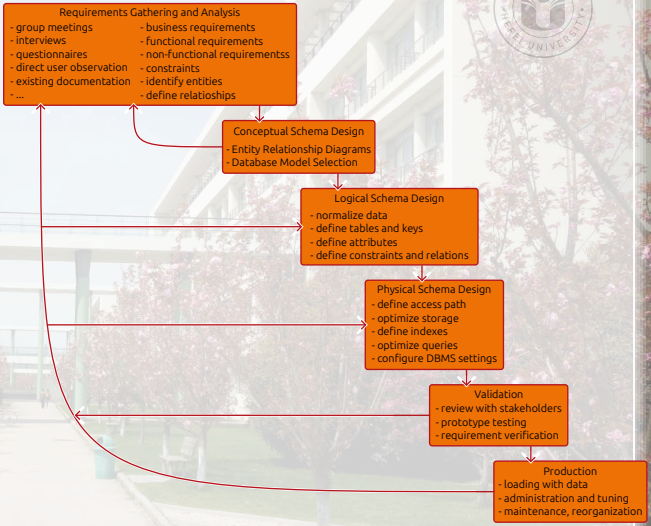
Kern des Designs

- Wir beginnen mit dem Sammeln und Analysieren von Anforderungen.
- Dann werden die drei Schemas (konzeptuell, logisch, und physisch) eines nach dem anderen entworfen[23].
- Wir beginnen mit dem konzeptuellen Model, wobei Entitäten und ihre Beziehungen mit ERDs illustriert werden.
- Dieses Modell wird mit den Stakeholders diskutiert.
- Werden dabei Inkonsistenzen gefunden, dann werden die Anforderungen angepasst.



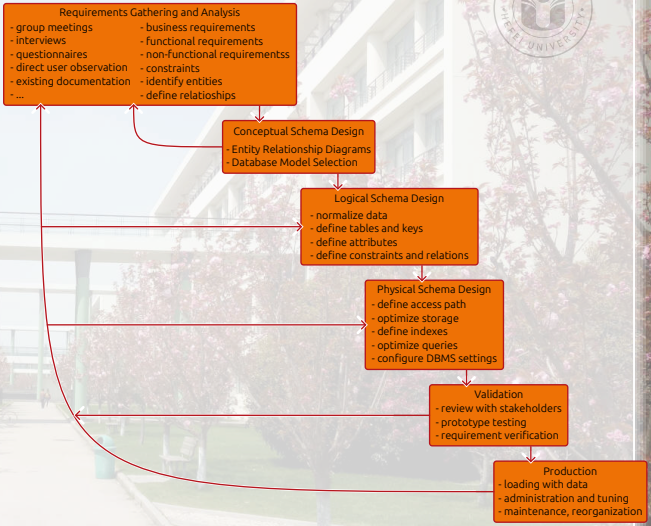
Kern des Designs

- Dann werden die drei Schemas (konzeptuell, logisch, und physisch) eines nach dem anderen entworfen[23].
- Wir beginnen mit dem konzeptuellen Model, wobei Entitäten und ihre Beziehungen mit ERDs illustriert werden.
- Dieses Modell wird mit den Stakeholders diskutiert.
- Werden dabei Inkonsistenzen gefunden, dann werden die Anforderungen angepasst.
- Am Ende dieses Schrittes wählen wir ein Datenmodell.



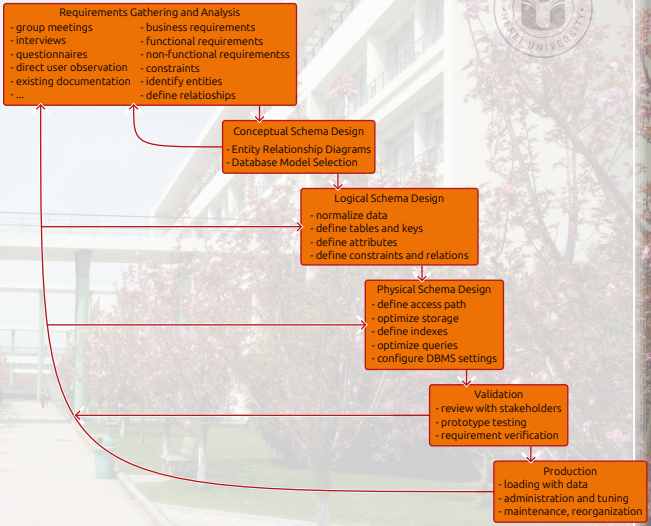
Kern des Designs

- Wir beginnen mit dem konzeptuellen Model, wobei Entitäten und ihre Beziehungen mit ERDs illustriert werden.
- Dieses Modell wird mit den Stakeholders diskutiert.
- Werden dabei Inkonsistenzen gefunden, dann werden die Anforderungen angepasst.
- Am Ende dieses Schrittes wählen wir ein Datenmodell.
- Die Entitäten des konzeptuellen Modells werden dann in das logische Modell übersetzt.



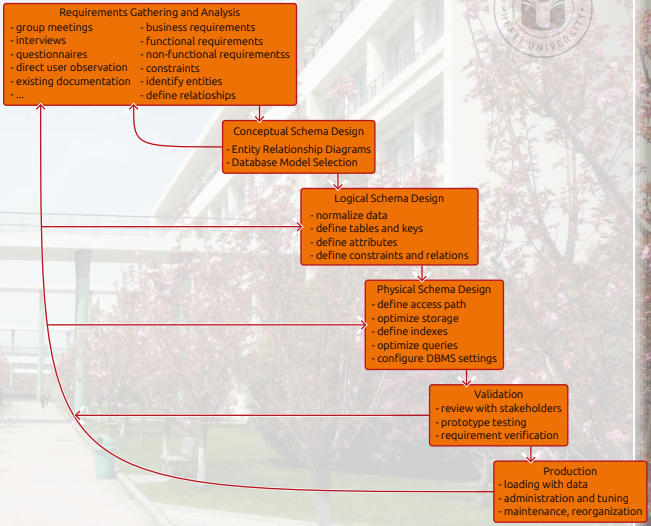
Kern des Designs

- Dieses Modell wird mit den Stakeholders diskutiert.
- Werden dabei Inkonsistenzen gefunden, dann werden die Anforderungen angepasst.
- Am Ende dieses Schrittes wählen wir ein Datenmodell.
- Die Entitäten des konzeptuellen Modells werden dann in das logische Modell übersetzt.
- Im Kontext eines relationalen Datenmodells werden dabei Tabellen erstellt, Schlüssel und Attribute definiert, und die Daten werden normalisiert.



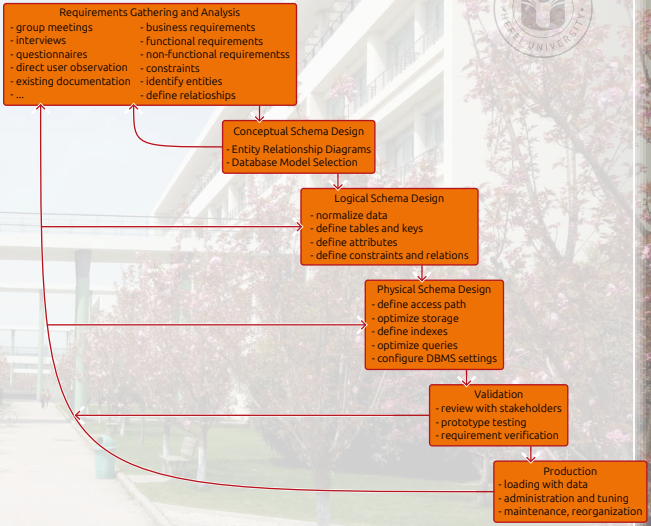
Kern des Designs

- Dieses Modell wird mit den Stakeholders diskutiert.
- Werden dabei Inkonsistenzen gefunden, dann werden die Anforderungen angepasst.
- Am Ende dieses Schrittes wählen wir ein Datenmodell.
- Die Entitäten des konzeptuellen Modells werden dann in das logische Modell übersetzt.
- Im Kontext eines relationalen Datenmodells werden dabei Tabellen erstellt, Schlüssel und Attribute definiert, und die Daten werden normalisiert.



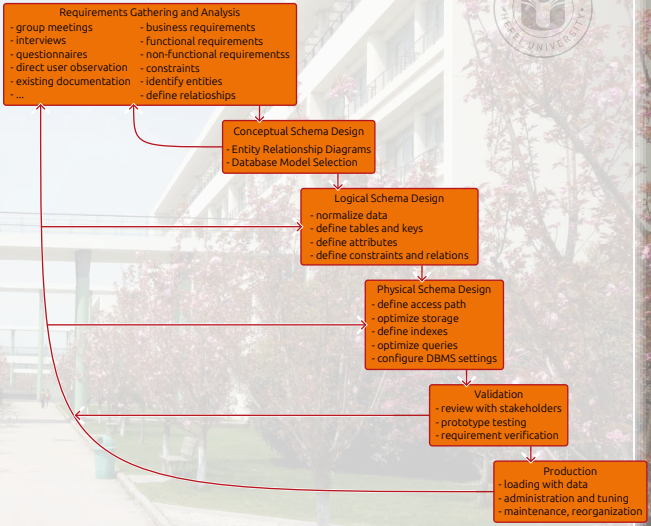
Kern des Designs

- Werden dabei Inkonsistenzen gefunden, dann werden die Anforderungen angepasst.
- Am Ende dieses Schrittes wählen wir ein Datenmodell.
- Die Entitäten des konzeptuellen Modells werden dann in das logische Modell übersetzt.
- Im Kontext eines relationalen Datenmodells werden dabei Tabellen erstellt, Schlüssel und Attribute definiert, und die Daten werden normalisiert.
- Das Modell kann in Form von SQL definiert werden.



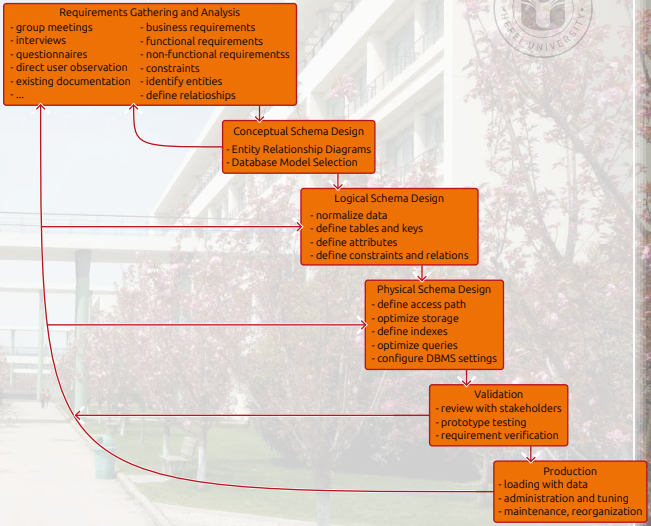
Kern des Designs

- Am Ende dieses Schrittes wählen wir ein Datenmodell.
- Die Entitäten des konzeptuellen Modells werden dann in das logische Modell übersetzt.
- Im Kontext eines relationalen Datenmodells werden dabei Tabellen erstellt, Schlüssel und Attribute definiert, und die Daten werden normalisiert.
- Das Modell kann in Form von SQL definiert werden.
- Das logische Modell kann dann auf ein physisches Modell übersetzt werden.



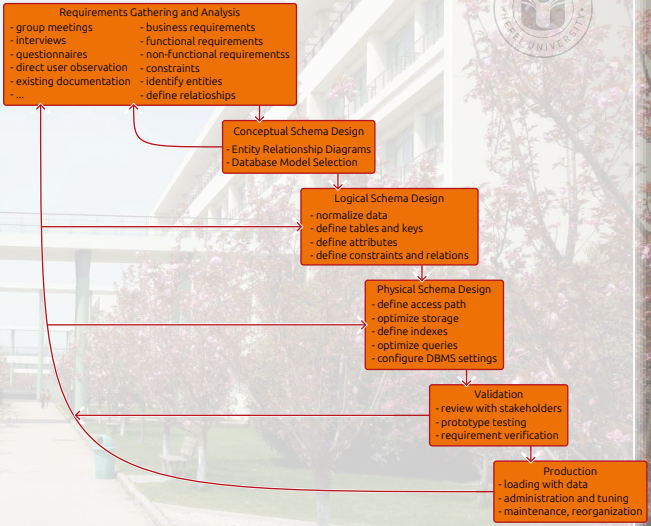
Kern des Designs

- Die Entitäten des konzeptuellen Modells werden dann in das logische Modell übersetzt.
- Im Kontext eines relationalen Datenmodells werden dabei Tabellen erstellt, Schlüssel und Attribute definiert, und die Daten werden normalisiert.
- Das Modell kann in Form von SQL definiert werden.
- Das logische Modell kann dann auf ein physisches Modell übersetzt werden.
- Das logische Modell beschreibt, wie auf die Daten zugegriffen wird.



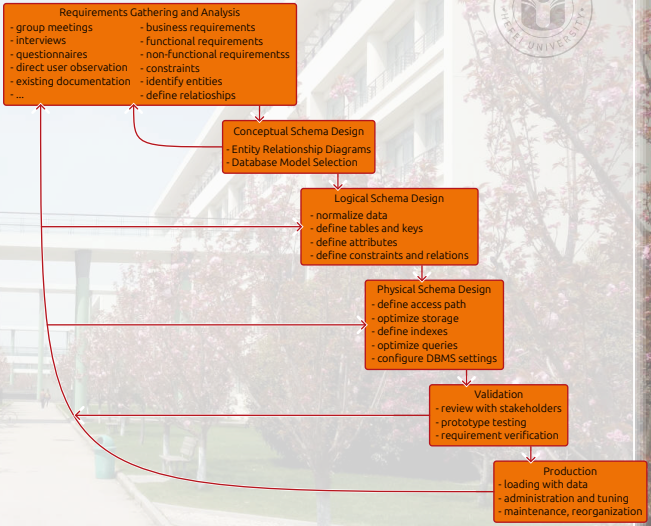
Kern des Designs

- Im Kontext eines relationalen Datenmodells werden dabei Tabellen erstellt, Schlüssel und Attribute definiert, und die Daten werden normalisiert.
- Das Modell kann in Form von SQL definiert werden.
- Das logische Modell kann dann auf ein physisches Modell übersetzt werden.
- Das logische Modell beschreibt, wie auf die Daten zugegriffen wird.
- Das physische Modell beschreibt, wie dieser Zugriff effizient gemacht werden kann.



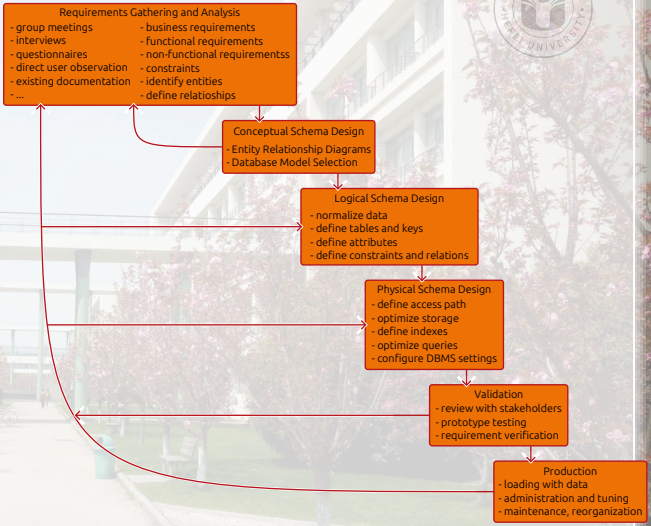
Kern des Designs

- Das Modell kann in Form von SQL definiert werden.
- Das logische Modell kann dann auf ein physisches Modell übersetzt werden.
- Das logische Modell beschreibt, wie auf die Daten zugegriffen wird.
- Das physische Modell beschreibt, wie dieser Zugriff effizient gemacht werden kann.
- Nun kann ein Prototyp entwickelt und von den Stakeholders evaluiert werden.



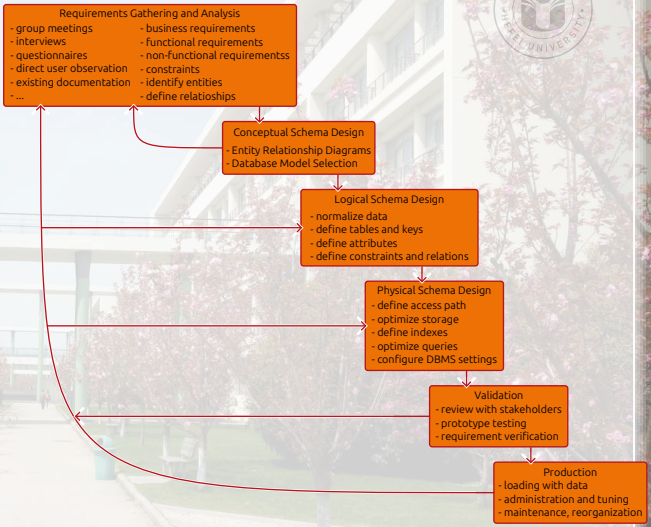
Kern des Designs

- Das Modell kann in Form von SQL definiert werden.
- Das logische Modell kann dann auf ein physisches Modell übersetzt werden.
- Das logische Modell beschreibt, wie auf die Daten zugegriffen wird.
- Das physische Modell beschreibt, wie dieser Zugriff effizient gemacht werden kann.
- Nun kann ein Prototyp entwickelt und von den Stakeholders evaluiert werden.
- Die Datenbank tritt in den Produktivbetrieb ein.



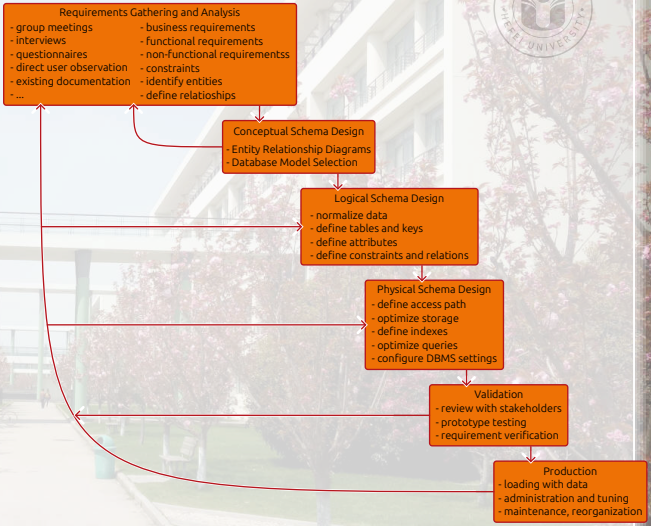
Kern des Designs

- Das logische Modell kann dann auf ein physisches Modell übersetzt werden.
- Das logische Modell beschreibt, wie auf die Daten zugegriffen wird.
- Das physische Modell beschreibt, wie dieser Zugriff effizient gemacht werden kann.
- Nun kann ein Prototyp entwickelt und von den Stakeholders evaluiert werden.
- Die Datenbank tritt in den Produktivbetrieb ein.
- Von nun an geht es um Wartung, Backups, Feinjustierungen, sowie das Hinzufügen von Features und das Anpassen an neue Situationen.



Kern des Designs

- Das physische Modell beschreibt, wie dieser Zugriff effizient gemacht werden kann.
- Nun kann ein Prototyp entwickelt und von den Stakeholders evaluiert werden.
- Die Datenbank tritt in den Produktivbetrieb ein.
- Von nun an geht es um Wartung, Backups, Feinjustierungen, sowie das Hinzufügen von Features und das Anpassen an neue Situationen.
- Die letzten beiden Phasen können Feedback in frühere Phasen geben, also zu änderungen am logischen und physischen Modell führen.



Zusammenfassung: Datenbanken

- Datenbanken sind langlebige und wichtige Assets einer Organisation.

Zusammenfassung: Datenbanken

- Datenbanken sind langlebige und wichtige Assets einer Organisation.
- Sie liegen irgendwo in der Grauzonen zwischen Dokumenten und Software.

Zusammenfassung: Datenbanken

- Datenbanken sind langlebige und wichtige Assets einer Organisation.
- Sie liegen irgendwo in der Grauzonen zwischen Dokumenten und Software.
- Sie bilden die Basis für viele der wichtigsten Applikationen einer Organisation.

Zusammenfassung: Datenbanken

- Datenbanken sind langlebige und wichtige Assets einer Organisation.
- Sie liegen irgendwo in der Grauzonen zwischen Dokumenten und Software.
- Sie bilden die Basis für viele der wichtigsten Applikationen einer Organisation.
- Über hr Design und ihre Performanz haben sie Einfluss auf viele der wichtigstigen Prozesse einer Organisation.

Zusammenfassung: Datenbanken

- Datenbanken sind langlebige und wichtige Assets einer Organisation.
- Sie liegen irgendwo in der Grauzonen zwischen Dokumenten und Software.
- Sie bilden die Basis für viele der wichtigsten Applikationen einer Organisation.
- Über hr Design und ihre Performanz haben sie Einfluss auf viele der wichtigstigen Prozesse einer Organisation.
- Und sie tun das über eine sehr lange Zeit.

Zusammenfassung: Datenbanken entwickeln

- Wenn wir eine Datenbank entwickeln wollen, dann können wir nicht einfach irgendwie planlos loslegen.

Zusammenfassung: Datenbanken entwickeln

- Wenn wir eine Datenbank entwickeln wollen, dann können wir nicht einfach irgendwie planlos loslegen.
- Stattdessen sollten wir einem der bekannten Entwicklungsprozesse folgen.

Zusammenfassung: Datenbanken entwickeln

- Wenn wir eine Datenbank entwickeln wollen, dann können wir nicht einfach irgendwie planlos loslegen.
- Stattdessen sollten wir einem der bekannten Entwicklungsprozesse folgen.
- Welchem wir folgen ist wahrscheinlich nicht so wichtig.

Zusammenfassung: Datenbanken entwickeln

- Wenn wir eine Datenbank entwickeln wollen, dann können wir nicht einfach irgendwie planlos loslegen.
- Stattdessen sollten wir einem der bekannten Entwicklungsprozesse folgen.
- Welchem wir folgen ist wahrscheinlich nicht so wichtig.
- Wichtig ist, dass alle eine Struktur in unsere Arbeit bringen.

Zusammenfassung: Datenbanken entwickeln

- Wenn wir eine Datenbank entwickeln wollen, dann können wir nicht einfach irgendwie planlos loslegen.
- Stattdessen sollten wir einem der bekannten Entwicklungsprozesse folgen.
- Welchem wir folgen ist wahrscheinlich nicht so wichtig.
- Wichtig ist, dass alle eine Struktur in unsere Arbeit bringen.
- Sie können unsere Arbeit anleiten.

Zusammenfassung: Datenbanken entwickeln

- Wenn wir eine Datenbank entwickeln wollen, dann können wir nicht einfach irgendwie planlos loslegen.
- Stattdessen sollten wir einem der bekannten Entwicklungsprozesse folgen.
- Welchem wir folgen ist wahrscheinlich nicht so wichtig.
- Wichtig ist, dass alle eine Struktur in unsere Arbeit bringen.
- Sie können unsere Arbeit anleiten.
- Sie machen die Projekte planbar.

Zusammenfassung: Datenbanken entwickeln

- Wenn wir eine Datenbank entwickeln wollen, dann können wir nicht einfach irgendwie planlos loslegen.
- Stattdessen sollten wir einem der bekannten Entwicklungsprozesse folgen.
- Welchem wir folgen ist wahrscheinlich nicht so wichtig.
- Wichtig ist, dass alle eine Struktur in unsere Arbeit bringen.
- Sie können unsere Arbeit anleiten.
- Sie machen die Projekte planbar.
- Und vielleicht das Wichtigste: Sie verringern Risiken.

Zusammenfassung: Datenbanken entwickeln

- Wenn wir eine Datenbank entwickeln wollen, dann können wir nicht einfach irgendwie planlos loslegen.
- Stattdessen sollten wir einem der bekannten Entwicklungsprozesse folgen.
- Welchem wir folgen ist wahrscheinlich nicht so wichtig.
- Wichtig ist, dass alle eine Struktur in unsere Arbeit bringen.
- Sie können unsere Arbeit anleiten.
- Sie machen die Projekte planbar.
- Und vielleicht das Wichtigste: Sie verringern Risiken.
- Und Risiken gibt es viele.

Zusammenfassung: Datenbanken entwickeln

- Wenn wir eine Datenbank entwickeln wollen, dann können wir nicht einfach irgendwie planlos loslegen.
- Stattdessen sollten wir einem der bekannten Entwicklungsprozesse folgen.
- Welchem wir folgen ist wahrscheinlich nicht so wichtig.
- Wichtig ist, dass alle eine Struktur in unsere Arbeit bringen.
- Sie können unsere Arbeit anleiten.
- Sie machen die Projekte planbar.
- Und vielleicht das Wichtigste: Sie verringern Risiken.
- Und Risiken gibt es viele.
- Vielleicht gehen uns Zeit oder Geld aus, vielleicht können wir die Benutzer nicht zufrieden stellen...

Zusammenfassung: Datenbanken entwickeln

- Wenn wir eine Datenbank entwickeln wollen, dann können wir nicht einfach irgendwie planlos loslegen.
- Stattdessen sollten wir einem der bekannten Entwicklungsprozesse folgen.
- Welchem wir folgen ist wahrscheinlich nicht so wichtig.
- Wichtig ist, dass alle eine Struktur in unsere Arbeit bringen.
- Sie können unsere Arbeit anleiten.
- Sie machen die Projekte planbar.
- Und vielleicht das Wichtigste: Sie verringern Risiken.
- Und Risiken gibt es viele.
- Vielleicht gehen uns Zeit oder Geld aus, vielleicht können wir die Benutzer nicht zufrieden stellen...
- Um solche Risiken zu mildern, folgen wir eben diesen ingenieursmäßigen Ansätzen.

Zusammenfassung: Entwurfsprozesse

- Die meisten Datenbank-Entwurfsprozesse arbeiten sich im Kern iterativ von einem konzeptuellen über ein logsches hin zu einem physischen Schema durch.

Zusammenfassung: Entwurfsprozesse

- Die meisten Datenbank-Entwurfsprozesse arbeiten sich im Kern iterativ von einem konzeptuellen über ein logsches hin zu einem physischen Schema durch.
- Vorher erfolgen normalerweise das Sammeln und Analysieren von Anforderungen.

Zusammenfassung: Entwurfsprozesse

- Die meisten Datenbank-Entwurfsprozesse arbeiten sich im Kern iterativ von einem konzeptuellen über ein logsches hin zu einem physischen Schema durch.
- Vorher erfolgen normalerweise das Sammeln und Analysieren von Anforderungen.
- Während dem Design wird oft enger Kontakt mir den Stakeholdern empfohlen.

Zusammenfassung: Entwurfsprozesse

- Die meisten Datenbank-Entwurfsprozesse arbeiten sich im Kern iterativ von einem konzeptuellen über ein logsches hin zu einem physischen Schema durch.
- Vorher erfolgen normalerweise das Sammeln und Analysieren von Anforderungen.
- Während dem Design wird oft enger Kontakt mir den Stakeholdern empfohlen.
- Dabei können uns Prototypen helfen, Feedback zu sammeln und Missverständnisse zu entdeken.

Zusammenfassung: Entwurfsprozesse

- Die meisten Datenbank-Entwurfsprozesse arbeiten sich im Kern iterativ von einem konzeptuellen über ein logsches hin zu einem physischen Schema durch.
- Vorher erfolgen normalerweise das Sammeln und Analysieren von Anforderungen.
- Während dem Design wird oft enger Kontakt mir den Stakeholdern empfohlen.
- Dabei können uns Prototypen helfen, Feedback zu sammeln und Missverständnisse zu entdeken.
- Damit können wir dann, falls notwendig, Design-Dokumente und Schemas überarbeiten.

Zusammenfassung: Entwurfsprozesse

- Die meisten Datenbank-Entwurfsprozesse arbeiten sich im Kern iterativ von einem konzeptuellen über ein logsches hin zu einem physischen Schema durch.
- Vorher erfolgen normalerweise das Sammeln und Analysieren von Anforderungen.
- Während dem Design wird oft enger Kontakt mir den Stakeholdern empfohlen.
- Dabei können uns Prototypen helfen, Feedback zu sammeln und Missverständnisse zu entdeken.
- Damit können wir dann, falls notwendig, Design-Dokumente und Schemas überarbeiten.
- Zum Schluss, nachdem die Applikation im Produktiveinsatz angekommen ist, erfolgen periodische Tätigkeiten wie Wartung, Updates, und Backup.

Zusammenfassung: Entwurfsprozesse

- Die meisten Datenbank-Entwurfsprozesse arbeiten sich im Kern iterativ von einem konzeptuellen über ein logsches hin zu einem physischen Schema durch.
- Vorher erfolgen normalerweise das Sammeln und Analysieren von Anforderungen.
- Während dem Design wird oft enger Kontakt mir den Stakeholdern empfohlen.
- Dabei können uns Prototypen helfen, Feedback zu sammeln und Missverständnisse zu entdeken.
- Damit können wir dann, falls notwendig, Design-Dokumente und Schemas überarbeiten.
- Zum Schluss, nachdem die Applikation im Produktiveinsatz angekommen ist, erfolgen periodische Tätigkeiten wie Wartung, Updates, und Backup.
- In den folgenden Einheiten werden wir uns beispielhaft durch so einen Entwicklungsprozess durcharbeiten.

谢谢您们！

Thank you!

Vielen Dank!

References I

[1] Richard Barker. *Case*Method: Entity Relationship Modelling (Oracle)*. 1. Aufl. Redwood City, CA, USA: Addison Wesley Longman Publishing Co., Inc., Jan. 1990. ISBN: 978-0-201-41696-1 (siehe S. 194–226, 314).

[2] 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. 315).

[3] Ben Beitler. *Hands-On Microsoft Access 2019*. Birmingham, England, UK: Packt Publishing Ltd, März 2020. ISBN: 978-1-83898-747-3 (siehe S. 314, 315).

[4] Paul Beynon-Davies, Chris Carne, Hugh Mackay und Doug Tudhope. "Rapid Application Development (RAD): An Empirical Review". *European Journal of Information Systems (EJIS)* 8(3), Sep. 1999. London, England, UK: Taylor and Francis Ltd. ISSN: 0960-085X. doi:10.1057/palgrave.ejis.3000325. URL: https://www.researchgate.net/publication/31978101 (besucht am 2025-10-07) (siehe S. 168–177, 316).

[5] Elizabeth Bjarnason, Franz Lang und Alexander Mjöberg. "An Empirically based Model of Software Prototyping: A Mapping Study and a Multi-Case Study". *Empirical Software Engineering: An International Journal* 28(5):115, Aug. 2023. London, England, UK: Springer Nature Limited. ISSN: 1382-3256. doi:10.1007/S10664-023-10331-W. URL: https://www.researchgate.net/publication/373517644 (besucht am 2025-10-07) (siehe S. 147–154).

[6] Bernard Obeng Boateng. *Data Modeling with Microsoft Excel*. Birmingham, England, UK: Packt Publishing Ltd, Nov. 2023. ISBN: 978-1-80324-028-2 (siehe S. 315).

[7] Barry W. Boehm. "A Spiral Model of Software Development and Enhancement". In: *International Workshop on the Software Process and Software Environments*. 27.–29. März 1985, Coto de Caza, Trabuco Canyon, CA, USA. New York, NY, USA: Association for Computing Machinery (ACM), 1985, S. 22–42. ISBN: 978-0-89791-202-0. doi:10.1145/12944.12948. Also contained in[55] (siehe S. 156–159).

[8] Barry W. Boehm. "A Spiral Model of Software Development and Enhancement". *Computer* 21(5):61–72, Mai 1988. Piscataway, NJ, USA: Institute of Electrical and Electronics Engineers (IEEE). ISSN: 0018-9162. doi:10.1109/2.59 (siehe S. 156–159).

[9] Ed Bott. *Windows 11 Inside Out*. Hoboken, NJ, USA: Microsoft Press, Pearson Education, Inc., Feb. 2023. ISBN: 978-0-13-769132-6 (siehe S. 315).

References II

[10] Ben Brumm. "A Guide to the Entity Relationship Diagram (ERD)". In: *Database Star*. Armadale, VIC, Australia: Elevated Online Services PTY Ltd., 30. Juli 2019–23. Dez. 2023. URL: https://www.databasestar.com/entity-relationship-diagram (besucht am 2025-03-29) (siehe S. 314).

[11] 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. 316).

[12] Peter Pin-Shan Chen. "Entity-Relationship Modeling: Historical Events, Future Trends, and Lessons Learned". In: *Software Pioneers: Contributions to Software Engineering*. Hrsg. von Manfred Broy und Ernst Denert. Berlin/Heidelberg, Germany: Springer-Verlag GmbH Germany, Feb. 2002, S. 296–310. doi:10.1007/978-3-642-59412-0_17. URL: http://bit.csc.lsu.edu/%7Echen/pdf/Chen_Pioneers.pdf (besucht am 2025-03-06) (siehe S. 314).

[13] Peter Pin-Shan Chen. "The Entity-Relationship Model – Toward a Unified View of Data". *ACM Transactions on Database Systems (TODS)* 1(1):9–36, März 1976. New York, NY, USA: Association for Computing Machinery (ACM). ISSN: 0362-5915. doi:10.1145/320434.320440 (siehe S. 305, 314).

[14] Peter Pin-Shan Chen. "The Entity-Relationship Model: Toward a Unified View of Data". In: *1st International Conference on Very Large Data Bases (VLDB'1975)*. 22.–24. Sep. 1975, Framingham, MA, USA. Hrsg. von Douglas S. Kerr. New York, NY, USA: Association for Computing Machinery (ACM), 1975, S. 173. ISBN: 978-1-4503-3920-9. doi:10.1145/1282480.1282492. See[13] for a more comprehensive introduction. (Siehe S. 314).

[15] 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. 314, 315).

[16] 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. 316).

[17] 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. 316).

References III

[18] *Database Language SQL*. Techn. Ber. ANSI X3.135-1986. Washington, D.C., USA: American National Standards Institute (ANSI), 1986 (siehe S. 316).

[19] 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. 316).

[20] *Database Language SQL*. International Standard ISO 9075-1987. Geneva, Switzerland: International Organization for Standardization (ISO), 1987 (siehe S. 316).

[21] Adam „djeada“ Djellouli. "Database Requirements Analysis". In: *Database Notes*. Berlin, Germany, 11. März 2025. URL: https://adamdjellouli.com/articles/databases_notes/02_database_design/01_requirements_analysis (besucht am 2025-03-27) (siehe S. 258–267).

[22] 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. 315).

[23] Ramez Elmasri und Shamkant Navathe. *Fundamentals of Database Systems*. 7. Aufl. Hoboken, NJ, USA: Pearson Education, Inc., Juni 2015. ISBN: 978-0-13-397077-7 (siehe S. 25–31, 194–222, 258–269).

[24] Steve Fanning, Vasudev Narayanan, „flywire“, Olivier Hallot, Jean Hollis Weber, Jenna Sargent, Pulkit Krishna, Dan Lewis, Peter Schofield, Jochen Schiffers, Robert Großkopf, Jost Lange, Martin Fox, Hazel Russman, Steve Schwettman, Alain Romedenne, Andrew Pitonyak, Jean-Pierre Ledure, Drew Jensen 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. 314).

[25] 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. 316).

References IV

[26] 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 (IFIPAICT). 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. 314, 315).

[27] Aakanksha Gaur, Gloria Lotha, Tara Ramanathan, Erik Gregersen, Emily Rodriguez, Anthony Lin, Parul Jain und William L. Hosch. *List of Windows Versions*. Hrsg. von The Editors of Encyclopaedia Britannica. Chicago, IL, USA: Encyclopædia Britannica, Inc., 15. Jan. 2009–19. Feb. 2025. URL: https://www.britannica.com/technology/list-of-Windows-versions (besucht am 2024-12-14) (siehe S. 77–93).

[28] 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. 315).

[29] Pranshu Gupta, Ramon A. Mata-Toledo und Morgan D. Monger. "Database Development Life Cycle". *Journal of Information Systems and Operations Management (JISOM)* 5(1):8–17, Mai 2011. Bucharest (București), Romania: Romanian-American University (RAU), Scientific Research Department. ISSN: 1843-4711. URL: http://www.rebe.rau.ro/RePEc/rau/jisomg/SP11/JISOM-SP11-A1.pdf (besucht am 2025-03-20) (siehe S. 77–88, 115–126, 128–132, 147–172, 194–200, 258–261).

[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. 316).

[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. 316).

[32] Michael Hausenblas. *Learning Modern Linux*. Sebastopol, CA, USA: O'Reilly Media, Inc., Apr. 2022. ISBN: 978-1-0981-0894-6 (siehe S. 315).

[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. 316).

References V

[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. 315).

[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. 316).

[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 (besucht am 2025-01-08). Consists of several parts, see https://modern-sql.com/standard for information where to obtain them. (Siehe S. 316).

[37] Joseph Ingeno. *Software Architect's Handbook*. Birmingham, England, UK: Packt Publishing Ltd, Aug. 2018. ISBN: 978-1-78862-406-0 (siehe S. 77–89, 128–132, 316).

[38] *International Workshop on the Software Process and Software Environments*. 27.–29. März 1985, Coto de Caza, Trabuco Canyon, CA, USA. New York, NY, USA: Association for Computing Machinery (ACM), 1985. ISBN: 978-0-89791-202-0. Also contained in[55] (siehe S. 310).

[39] Shannon Kempe und Paul Williams. *A Short History of the ER Diagram and Information Modeling*. Studio City, CA, USA: Dataversity Digital LLC, 25. Sep. 2012. URL: https://www.dataversity.net/a-short-history-of-the-er-diagram-and-information-modeling (besucht am 2025-03-06) (siehe S. 314).

[40] 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. 315).

[41] 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. 316).

[42] *LibreOffice – The Document Foundation*. Berlin, Germany: The Document Foundation, 2024. URL: https://www.libreoffice.org (besucht am 2024-12-12) (siehe S. 314, 315).

References VI

[43] Luqi. “Software Evolution Through Rapid Prototyping”. *Computer* 22(5):13–25, Mai 1989. Piscataway, NJ, USA: Institute of Electrical and Electronics Engineers (IEEE). ISSN: 0018-9162. doi:10.1109/2.27953 (siehe S. 147–155).

[44] Mark Lutz. *Learning Python*. 6. Aufl. Sebastopol, CA, USA: O'Reilly Media, Inc., März 2025. ISBN: 978-1-0981-7130-8 (siehe S. 316).

[45] James Martin. *Rapid Application Development*. Indianapolis, IN, USA: Macmillan Publishing Co., Inc., Jan. 1991. ISBN: 978-0-02-376775-3 (siehe S. 168–171).

[46] Steve McConnel. *Rapid Development: Taming Wild Software Schedules*. Hoboken, NJ, USA: Microsoft Press, Pearson Education, Inc., Juli 1996. ISBN: 978-1-55615-900-8 (siehe S. 168–172, 316).

[47] 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. 315).

[48] 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. 316).

[49] *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. 315).

[50] Catherine Nelson. *Software Engineering for Data Scientists*. Sebastopol, CA, USA: O'Reilly Media, Inc., Apr. 2024. ISBN: 978-1-0981-3620-8 (siehe S. 77–89, 128–132, 316).

[51] 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. 316).

[52] Shari Lawrence Pfleeger und Joanne M. Atlee. *Software Engineering: Theory and Practice*. Hoboken, NJ, USA: Pearson Education, Inc., Feb. 2009. ISBN: 978-0-13-606169-4 (siehe S. 147–159).

[53] *PostgreSQL Essentials: Leveling Up Your Data Work*. Sebastopol, CA, USA: O'Reilly Media, Inc., März 2024 (siehe S. 316).

References VII

[54] Roger S. Pressman und Bruce R. Maxim. *Software Engineering: A Practitioner's Approach (SEPA)*. 9. Aufl. New York, NY, USA: McGraw-Hill, 2020. ISBN: 978-1-259-87297-6 (siehe S. 128–132).

[55] "Proceedings of the *International Workshop on the Software Process and Software Environments* , 1985[38]". *ACM SIGSOFT Software Engineering Notes* 11(4), Juni 1985. New York, NY, USA: Association for Computing Machinery (ACM). ISSN: 0163-5948. URL: https://dl.acm.org/action/showFmPdf?doi=10.1145%2F12944 (besucht am 2025-10-07) (siehe S. 304, 308).

[56] Winston Walker Royce. "Managing the Development of Large Software Systems". In: *Papers Presented at the Western Electronic Show and Convention (IEEE WESCON'1970)*. 25.–28. Aug. 1970, Los Angeles, CA, USA. Piscataway, NJ, USA: Institute of Electrical and Electronics Engineers (IEEE), 1970, S. 328–388. ISSN: 1095-791X. URL: https://blog.jbrains.ca/assets/articles/royce1970.pdf (besucht am 2025-10-06) (siehe S. 128–144).

[57] Stephen R. Schach. *Object-Oriented Software Engineering*. New York, NY, USA: McGraw-Hill, Sep. 2007. ISBN: 978-0-07-352333-0 (siehe S. 156–172, 316).

[58] Heinz Schweppe und Manuel Scholz. "Conceptual Database Design: Requirement Analysis and Modeling Languages". In: *Einführung in die Datenbanksysteme. Datenbanken für die Bioinformatik*. Berlin, Germany: Freie Universität Berlin, Apr.–Okt. 2005. Kap. 2.1/2.2. URL: https://www.inf.fu-berlin.de/lehre/SS05/19517-V/FolienEtc/dbs05-02-ConceptualModeling1-2.pdf (besucht am 2025-03-24) (siehe S. 194–238, 258–267).

[59] Heinz Schweppe und Manuel Scholz. *Einführung in die Datenbanksysteme. Datenbanken für die Bioinformatik*. Berlin, Germany: Freie Universität Berlin, Apr.–Okt. 2005. URL: https://www.inf.fu-berlin.de/lehre/SS05/19517-V (besucht am 2025-01-08).

[60] 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. 77–91, 194–238, 258–267).

[61] 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. 314, 315).

References VIII

[62] Yuriy Shamshin. "Conceptual Database Model. Entity Relationship Diagram (ERD)". In: *Databases*. Riga, Latvia: ISMA University of Applied Sciences, Mai 2024. Kap. 04. URL: https://dbs.academy.lv/lection/dbs_LS04EN_erd.pdf (besucht am 2025-03-29) (siehe S. 314).

[63] 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. 315).

[64] Bryan Sills, Brian Gardner, Kristin Marsicano und Chris Stewart. *Android Programming: The Big Nerd Ranch Guide*. 5. Aufl. Reading, MA, USA: Addison-Wesley Professional, Mai 2022. ISBN: 978-0-13-764579-4 (siehe S. 314).

[65] Anna Skoulikari. *Learning Git*. Sebastopol, CA, USA: O'Reilly Media, Inc., Mai 2023. ISBN: 978-1-0981-3391-7 (siehe S. 314).

[66] Ioannis Skoulis, Panos Vassiliadis und Apostolos V. Zarras. "Open-Source Databases: Within, Outside, or Beyond Lehman's Laws of Software Evolution?" In: *26th International Conference on Advanced Information Systems Engineering (CAiSE'2014)*. 16.–20. Juni 2014, Thessaloniki, Central Macedonia, Greece. Hrsg. von Matthias Jarke, John Mylopoulos, Christoph Quix, Colette Rolland, Yannis Manolopoulos, Haralambos Mouratidis und Jennifer Horkoff. Bd. 8484 der Reihe Lecture Notes in Computer Science (LNCS). Cham, Switzerland: Springer, 2014, S. 379–393. ISSN: 0302-9743. ISBN: 978-3-319-07880-9. doi:10.1007/978-3-319-07881-6_26 (siehe S. 77–87).

[67] Drew Smith. *Modern Apple Platform Administration – macOS and iOS Essentials (2025)*. Birmingham, England, UK: Packt Publishing Ltd, Feb. 2025. ISBN: 978-1-80580-309-6 (siehe S. 315).

[68] 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. 316).

[69] "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. 316).

[70] 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. 312, 316).

References IX

[71] 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. Burgthann, Bayern, Germany: Markt+Technik Verlag GmbH, Feb. 1998. ISBN: 978-3-8272-2020-2. Translation of[70] (siehe S. 316).

[72] Allen Taylor. *Introducing SQL and Relational Databases*. New York, NY, USA: Apress Media, LLC, Sep. 2018. ISBN: 978-1-4842-3841-7 (siehe S. 316).

[73] 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. 316).

[74] *The Document Foundation Wiki: ReleasePlan*. Berlin, Germany: The Document Foundation, 21. Jan. 2011–22. Nov. 2024. URL: https://wiki.documentfoundation.org/ReleasePlan (besucht am 2025-03-21) (siehe S. 77–93).

[75] *The Ubuntu Lifecycle and Release Cadence*. London, England, UK: Canonical Ltd., Okt. 2024. URL: https://ubuntu.com/about/release-cycle (besucht am 2025-03-21) (siehe S. 77–93, 315).

[76] 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. 315).

[77] 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. 314, 316).

[78] 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. 314, 315).

[79] Sander van Vugt. *Linux Fundamentals*. 2. Aufl. Hoboken, NJ, USA: Pearson IT Certification, Juni 2022. ISBN: 978-0-13-792931-3 (siehe S. 315).

[80] 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. 314, 316).

References X

[81] 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. 316).

[82] John R. Weitzel und Larry Kerschberg. "Developing Knowledge-Based Systems: Reorganizing the System Development Life Cycle". *Communications of the ACM (CACM)* 32(4):482–488, Apr. 1989. New York, NY, USA: Association for Computing Machinery (ACM). ISSN: 0001-0782. doi:10.1145/63334.63340 (siehe S. 115–126).

[83] Matthew West. *Developing High Quality Data Models*. Version: 2.0, Issue: 2.1. London, England, UK: Shell International Limited und European Process Industries STEP Technical Liaison Executive (EPISTLE); Burlington, MA, USA/San Mateo, CA, USA: Morgan Kaufmann Publishers, 8. Dez. 1995–Dez. 2010. ISBN: 978-0-12-375107-2. URL: https://www.researchgate.net/publication/286610894 (besucht am 2025-03-24). Edited by Julian Fowler (siehe S. 194–228, 314).

[84] *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. 316).

[85] 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. 314).

[86] 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). 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. 315).

Glossary (in English) I

Android	is a common operating system for mobile phones[64].
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*[80].
DBA	A *database administrator* is the person or group responsible for the effective use of database technology in an organization or enterprise.
DBMS	A *database management system* is the software layer located between the user or application and the DB. The DBMS allows the user/application to create, read, write, update, delete, and otherwise manipulate the data in the DB[85].
DBS	A *database system* is the combination of a DB and a the corresponding database management system (DBMS), i.e., basically, an installation of a DBMS on a computer together with one or multiple DBs. DBS = DB + DBMS.
ERD	Entity relationship diagrams show the relationships between objects, e.g., between the tables in a DB and how they reference each other[1,10,12–14,39,62,83]
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[65,77]. Learn more at `https://git-scm.com`.
IT	information technology
LibreOffice	is on open source office suite[26,42,61] which is a good and free alternative to Microsoft Office. It offers software such as LibreOffice Writer, LibreOffice Calc, and LibreOffice Base. See[80] 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[24,61]. It is part of LibreOffice[26,42,61] and has functionality that is comparable to Microsoft Access[3,15,78].
LibreOffice Calc	is a spreadsheet software that allows you to arrange and perform calculations with data in a tabular grid. It is a free and open source spread sheet software[42,61], i.e., an alternative to Microsoft Excel. It is part of LibreOffice[26,42,61].

Glossary (in English) II

LibreOffice Writer is a free and open source text writing program[86] and part of LibreOffice[26,42,61]. It is a good alternative to Microsoft Word.

Linux is the leading open source operating system, i.e., a free alternative for Microsoft Windows[2,32,63,76,79]. 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.

LTS Many types of software for which new versions appear regularly offer so-called *long-term support* (LTS) versions. Such versions are guaranteed to be supported for a longer-than-usual time. The Ubuntu Operating System (OS) releases a new long-term support (LTS) version every two years[75], for example.

macOS or Mac OS is the operating system that powers Apple Mac(intosh) computers[67]. Learn more at https://www.apple.com/macos.

Microsoft Access is a DBMS that can work on DBs stored in single, stand-alone files but also connect to other popular relational databases[3,15,47,78]. 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[6,28,40]. It is part of Microsoft Office. A free alternative to this commercial software is LibreOffice Calc[42,61].

Microsoft Office is a commercial suite of office software, including Microsoft Excel, Microsoft Word, and Microsoft Access[40]. LibreOffice is a free and open source alternative.

Microsoft Windows is a commercial proprietary operating system[9]. 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[22,49,86] and part of Microsoft Office. A free alternative to this commercial software is the LibreOffice Writer.

OS Operating System, the system that runs your computer, see, e.g., Linux, Microsoft Windows, macOS, and Android.

OSS Open source software, i.e., software that can freely be used, whose source code is made availabe in the internet, and which is usually developed cooperatively over the internet as well[34]. Typical examples are Python, Linux, Git, and PostgreSQL.

Glossary (in English) III

PostgreSQL An open source object-relational DBMS[25,51,53,73]. See https://postgresql.org for more information.

Python The Python programming language[35,41,44,81], i.e., what you will learn about in our book[81]. Learn more at https://python.org.

RAD Rapid Application Development[4,46,57]

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[17,30,31,68,72,80,84].

SDLC Software Development Life Cycle[37,50]

SQL The *Structured Query Language* is basically a programming language for querying and manipulating relational databases[11,18–20,36,48,69–72]. It is understood by many DBMSes. You find the Structured Query Language (SQL) commands supported by PostgreSQL in the reference[69].

Ubuntu is a variant of the open source operating system Linux[16,33]. 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 code[77]. 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.