



合肥大學
HEFEI UNIVERSITY



Datenbanken

27. Konzeptuelles Schema: Entitäten und Attribute

Thomas Weise (汤卫思)
tweise@hfuu.edu.cn

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

应用优化研究所
人工智能与大数据学院
合肥大学
中国安徽省合肥市

Databases



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

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



Outline



1. Einleitung
2. Entitäten und Attribute
3. Entity-Relationship Diagramme (ERDs)
4. ERDs mit yEd Zeichnen
5. Mehr Attribute
6. Zusammenfassung





Einleitung



Einleitung

- Beginnen wir nun mit dem konzeptuellen Modellieren von Applikationen.



Einleitung

- Beginnen wir nun mit dem konzeptuellen Modellieren von Applikationen.
- Durch die Anforderungsanalyse haben wir ein klares Verständnis über die Entitäten in unserem Szenario und die Beziehungen zwischen ihnen.



Einleitung



- Beginnen wir nun mit dem konzeptuellen Modellieren von Applikationen.
- Durch die Anforderungsanalyse haben wir ein klares Verständnis über die Entitäten in unserem Szenario und die Beziehungen zwischen ihnen.
- In unserem Beispiel haben wir die Entitäten der Plattform für der Management der Lehre einer Universität kennengelernt.

Einleitung



- Beginnen wir nun mit dem konzeptuellen Modellieren von Applikationen.
- Durch die Anforderungsanalyse haben wir ein klares Verständnis über die Entitäten in unserem Szenario und die Beziehungen zwischen ihnen.
- In unserem Beispiel haben wir die Entitäten der Plattform für der Management der Lehre einer Universität kennengelernt.
- Wir haben sie aber nur sehr informell diskutiert.

Einleitung



- Beginnen wir nun mit dem konzeptuellen Modellieren von Applikationen.
- Durch die Anforderungsanalyse haben wir ein klares Verständnis über die Entitäten in unserem Szenario und die Beziehungen zwischen ihnen.
- In unserem Beispiel haben wir die Entitäten der Plattform für der Management der Lehre einer Universität kennengelernt.
- Wir haben sie aber nur sehr informell diskutiert.
- Jetzt wollen wir die Informationen über die Entitäten in ein konsistentes Modell überführen.

Einleitung



- Beginnen wir nun mit dem konzeptuellen Modellieren von Applikationen.
- Durch die Anforderungsanalyse haben wir ein klares Verständnis über die Entitäten in unserem Szenario und die Beziehungen zwischen ihnen.
- In unserem Beispiel haben wir die Entitäten der Plattform für der Management der Lehre einer Universität kennengelernt.
- Wir haben sie aber nur sehr informell diskutiert.
- Jetzt wollen wir die Informationen über die Entitäten in ein konsistentes Modell überführen.
- Im konzeptuellen Designschrift sind diese Modelle unabhängig von konkreten Technologien.

Einleitung



- Beginnen wir nun mit dem konzeptuellen Modellieren von Applikationen.
- Durch die Anforderungsanalyse haben wir ein klares Verständnis über die Entitäten in unserem Szenario und die Beziehungen zwischen ihnen.
- In unserem Beispiel haben wir die Entitäten der Plattform für der Management der Lehre einer Universität kennengelernt.
- Wir haben sie aber nur sehr informell diskutiert.
- Jetzt wollen wir die Informationen über die Entitäten in ein konsistentes Modell überführen.
- Im konzeptuellen Designschritt sind diese Modelle unabhängig von konkreten Technologien.
- Dieser Schritt wird *entity relationship modeling* genannt^{3,42,76,77,99}.

Einleitung



- Beginnen wir nun mit dem konzeptuellen Modellieren von Applikationen.
- Durch die Anforderungsanalyse haben wir ein klares Verständnis über die Entitäten in unserem Szenario und die Beziehungen zwischen ihnen.
- In unserem Beispiel haben wir die Entitäten der Plattform für der Management der Lehre einer Universität kennengelernt.
- Wir haben sie aber nur sehr informell diskutiert.
- Jetzt wollen wir die Informationen über die Entitäten in ein konsistentes Modell überführen.
- Im konzeptuellen Designschrift sind diese Modelle unabhängig von konkreten Technologien.
- Dieser Schritt wird *entity relationship modeling* genannt^{3,42,76,77,99}.
- Er ist besonders bei großen Projekten sinnvoll.

Einleitung



- Beginnen wir nun mit dem konzeptuellen Modellieren von Applikationen.
- Durch die Anforderungsanalyse haben wir ein klares Verständnis über die Entitäten in unserem Szenario und die Beziehungen zwischen ihnen.
- In unserem Beispiel haben wir die Entitäten der Plattform für der Management der Lehre einer Universität kennengelernt.
- Wir haben sie aber nur sehr informell diskutiert.
- Jetzt wollen wir die Informationen über die Entitäten in ein konsistentes Modell überführen.
- Im konzeptuellen Designschrift sind diese Modelle unabhängig von konkreten Technologien.
- Dieser Schritt wird *entity relationship modeling* genannt^{3,42,76,77,99}.
- Er ist besonders bei großen Projekten sinnvoll.
- Bei kleinen Projekten, wie z. B. einer Datenbank für unsere eigenen Literaturreferenzen oder für unsere Musiksammlung kann man ihn überspringen und direkt zum logischen Design übergehen⁸³.

Einleitung



- Beginnen wir nun mit dem konzeptuellen Modellieren von Applikationen.
- Durch die Anforderungsanalyse haben wir ein klares Verständnis über die Entitäten in unserem Szenario und die Beziehungen zwischen ihnen.
- In unserem Beispiel haben wir die Entitäten der Plattform für das Management der Lehre einer Universität kennengelernt.
- Wir haben sie aber nur sehr informell diskutiert.
- Jetzt wollen wir die Informationen über die Entitäten in ein konsistentes Modell überführen.
- Im konzeptuellen Designschrift sind diese Modelle unabhängig von konkreten Technologien.
- Dieser Schritt wird *entity relationship modeling* genannt^{3,42,76,77,99}.
- Er ist besonders bei großen Projekten sinnvoll.
- Bei kleinen Projekten, wie z. B. einer Datenbank für unsere eigenen Literaturreferenzen oder für unsere Musiksammlung kann man ihn überspringen und direkt zum logischen Design übergehen⁸³.
- Unsere Lehrmanagementplattform ist aber ein großes Projekt und man muss daher mit dem konzeptuellen Design beginnen.



Entitäten und Attribute



Entitäten

- Die erste wichtige Komponente, die wir modellieren müssen, sind die Datenstrukturen, die wir in der Datenbank speichern wollen.



Entitäten



- Die erste wichtige Komponente, die wir modellieren müssen, sind die Datenstrukturen, die wir in der Datenbank speichern wollen.
- Es gibt dafür drei grundlegende Elemente für konzeptuelle Modelle: Entitäten, Attribute, und Entitätstypen.



Entitäten



- Die erste wichtige Komponente, die wir modellieren müssen, sind die Datenstrukturen, die wir in der Datenbank speichern wollen.
- Es gibt dafür drei grundlegende Elemente für konzeptuelle Modelle: Entitäten, Attribute, und Entitätstypen.

Definition: Entität

Eine *Entität* (EN: *entity*) ist ein Objekt oder Ding das für sich selbst in der Welt existiert.

Entitäten



- Die erste wichtige Komponente, die wir modellieren müssen, sind die Datenstrukturen, die wir in der Datenbank speichern wollen.
- Es gibt dafür drei grundlegende Elemente für konzeptuelle Modelle: Entitäten, Attribute, und Entitätstypen.

Definition: Entität

Eine *Entität* (EN: *entity*) ist ein Objekt oder Ding das für sich selbst in der Welt existiert. Sie kann von allen anderen Objekten unterschieden werden.

Entitäten



- Die erste wichtige Komponente, die wir modellieren müssen, sind die Datenstrukturen, die wir in der Datenbank speichern wollen.
- Es gibt dafür drei grundlegende Elemente für konzeptuelle Modelle: Entitäten, Attribute, und Entitätstypen.

Definition: Entität

Eine *Entität* (EN: *entity*) ist ein Objekt oder Ding das für sich selbst in der Welt existiert. Sie kann von allen anderen Objekten unterschieden werden.

- Z. B. ist der Student Herr Bibbo eine Entität.

Entitäten



- Die erste wichtige Komponente, die wir modellieren müssen, sind die Datenstrukturen, die wir in der Datenbank speichern wollen.
- Es gibt dafür drei grundlegende Elemente für konzeptuelle Modelle: Entitäten, Attribute, und Entitätstypen.

Definition: Entität

Eine *Entität* (EN: *entity*) ist ein Objekt oder Ding das für sich selbst in der Welt existiert. Sie kann von allen anderen Objekten unterschieden werden.

- Z. B. ist der Student Herr Bibbo eine Entität, das Module *Programming with Python*¹⁰¹ ist eine Entität.

Entitäten



- Die erste wichtige Komponente, die wir modellieren müssen, sind die Datenstrukturen, die wir in der Datenbank speichern wollen.
- Es gibt dafür drei grundlegende Elemente für konzeptuelle Modelle: Entitäten, Attribute, und Entitätstypen.

Definition: Entität

Eine *Entität* (EN: *entity*) ist ein Objekt oder Ding das für sich selbst in der Welt existiert. Sie kann von allen anderen Objekten unterschieden werden.

- Z. B. ist der Student Herr Bibbo eine Entität, das Module *Programming with Python*¹⁰¹ ist eine Entität, die Professorin Frau Bebb 教授 ist eine Entität.

Entitäten



- Die erste wichtige Komponente, die wir modellieren müssen, sind die Datenstrukturen, die wir in der Datenbank speichern wollen.
- Es gibt dafür drei grundlegende Elemente für konzeptuelle Modelle: Entitäten, Attribute, und Entitätstypen.

Definition: Entität

Eine *Entität* (EN: *entity*) ist ein Objekt oder Ding das für sich selbst in der Welt existiert. Sie kann von allen anderen Objekten unterschieden werden.

- Z. B. ist der Student Herr Bibbo eine Entität, das Module *Programming with Python*¹⁰¹ ist eine Entität, die Professorin Frau Bebbä 教授 ist eine Entität, und der Raum #36 202 ist auch eine Entität.

Entitäten



- Die erste wichtige Komponente, die wir modellieren müssen, sind die Datenstrukturen, die wir in der Datenbank speichern wollen.
- Es gibt dafür drei grundlegende Elemente für konzeptuelle Modelle: Entitäten, Attribute, und Entitätstypen.

Definition: Entität

Eine *Entität* (EN: *entity*) ist ein Objekt oder Ding das für sich selbst in der Welt existiert. Sie kann von allen anderen Objekten unterschieden werden.

- Z. B. ist der Student Herr Bibbo eine Entität, das Module *Programming with Python*¹⁰¹ ist eine Entität, die Professorin Frau Bebbba 教授 ist eine Entität, und der Raum #36 202 ist auch eine Entität.
- Entitäten können einfach in den Anforderungen oder Interviewnotizen entdeckt werden.

Entitäten



- Die erste wichtige Komponente, die wir modellieren müssen, sind die Datenstrukturen, die wir in der Datenbank speichern wollen.
- Es gibt dafür drei grundlegende Elemente für konzeptuelle Modelle: Entitäten, Attribute, und Entitätstypen.

Definition: Entität

Eine *Entität* (EN: *entity*) ist ein Objekt oder Ding das für sich selbst in der Welt existiert. Sie kann von allen anderen Objekten unterschieden werden.

- Z. B. ist der Student Herr Bibbo eine Entität, das Module *Programming with Python*¹⁰¹ ist eine Entität, die Professorin Frau Bebbba 教授 ist eine Entität, und der Raum #36 202 ist auch eine Entität.
- Entitäten können einfach in den Anforderungen oder Interviewnotizen entdeckt werden Sie entsprechen Eigennamen bzw. Hauptwörtern (EN: *proper nouns*)²¹, also Wörtern, die eine bestimmte Sache identifizieren³⁷.

Definition: Attribut

Ein *Attribut* (EN: *attribute*) a ist charakterisiert mit einem Namen und einer Domäne $\text{dom}(a)$ mit Werten die es annehmen kann.

Definition: Attribut

Ein *Attribut* (EN: *attribute*) a ist charakterisiert mit einem Namen und einer Domäne $\text{dom}(a)$ mit Werten die es annehmen kann.

- Attribute are sind Features und Charakteristiken von Entitäten.

Definition: Attribut

Ein *Attribut* (EN: *attribute*) a ist charakterisiert mit einem Namen und einer Domäne $\text{dom}(a)$ mit Werten die es annehmen kann.

- Attribute are sind Features und Charakteristiken von Entitäten.
- Die Entitäten in unserem Modell werden durch die Werte ihrer Attribute definiert.

Definition: Attribut

Ein *Attribut* (EN: *attribute*) a ist charakterisiert mit einem Namen und einer Domäne $\text{dom}(a)$ mit Werten die es annehmen kann.

- Attribute are sind Features und Charakteristiken von Entitäten.
- Die Entitäten in unserem Modell werden durch die Werte ihrer Attribute definiert.
- Eine Studentin könnte durch ihren Namen, ihre ID, ihre Studenten-ID, ihre Mobiltelefonnummer, ihre Heimataadresse, und durch ihr Geburtsdatum definiert werden.

Definition: Attribut

Ein *Attribut* (EN: *attribute*) a ist charakterisiert mit einem Namen und einer Domäne $\text{dom}(a)$ mit Werten die es annehmen kann.

- Attribute are sind Features und Charakteristiken von Entitäten.
- Die Entitäten in unserem Modell werden durch die Werte ihrer Attribute definiert.
- Eine Studentin könnte durch ihren Namen, ihre ID, ihre Studenten-ID, ihre Mobiltelefonnummer, ihre Heimataadresse, und durch ihr Geburtsdatum definiert werden.
- Ein Modul könnte durch seinen Titel, Syllabus, und Zusammenfassung definiert werden.

Definition: Attribut

Ein *Attribut* (EN: *attribute*) a ist charakterisiert mit einem Namen und einer Domäne $\text{dom}(a)$ mit Werten die es annehmen kann.

- Attribute are sind Features und Charakteristiken von Entitäten.
- Die Entitäten in unserem Modell werden durch die Werte ihrer Attribute definiert.
- Eine Studentin könnte durch ihren Namen, ihre ID, ihre Studenten-ID, ihre Mobiltelefonnummer, ihre Heimataadresse, und durch ihr Geburtsdatum definiert werden.
- Ein Modul könnte durch seinen Titel, Syllabus, und Zusammenfassung definiert werden.
- Neben solchen Eigenschaften können *Adjektive* in den Anforderungen oft als Attributwerte interpretiert werden²¹, z. B. rot, jung, erfolgreich, schwer, schnell. . .

Domäne



Definition: Domäne

Die *Domäne* (EN: *domain*) $\text{dom}(a)$ eines Attributs a ist die Menge der möglichen Werte, die es annehmen kann.

Domäne



Definition: Domäne

Die *Domäne* (EN: *domain*) $\text{dom}(a)$ eines Attributs a ist die Menge der möglichen Werte, die es annehmen kann.

- Wir können die Konzepte *Domäne* und *Datentyp* im Kontext von konzeptuellen Modellen unterscheiden⁸⁵.

Domäne



Definition: Domäne

Die *Domäne* (EN: *domain*) $\text{dom}(a)$ eines Attributs a ist die Menge der möglichen Werte, die es annehmen kann.

- Wir können die Konzepte *Domäne* und *Datentyp* im Kontext von konzeptuellen Modellen unterscheiden⁸⁵.
- Ein Datentyp ist ein mathematisches Konzept wohingegen eine Domäne ein logisches Konzept ist.

Domäne



Definition: Domäne

Die *Domäne* (EN: *domain*) $\text{dom}(a)$ eines Attributs a ist die Menge der möglichen Werte, die es annehmen kann.

- Wir können die Konzepte *Domäne* und *Datentyp* im Kontext von konzeptuellen Modellen unterscheiden⁸⁵.
- Ein Datentyp ist ein mathematisches Konzept wohingegen eine Domäne ein logisches Konzept ist.
- Z. B. sind `VARCHAR(100)`, `SMALLINT` und `REAL` alles Datentypen.

Domäne



Definition: Domäne

Die *Domäne* (EN: *domain*) $\text{dom}(a)$ eines Attributs a ist die Menge der möglichen Werte, die es annehmen kann.

- Wir können die Konzepte *Domäne* und *Datentyp* im Kontext von konzeptuellen Modellen unterscheiden⁸⁵.
- Ein Datentyp ist ein mathematisches Konzept wohingegen eine Domäne ein logisches Konzept ist.
- Z. B. sind `VARCHAR(100)`, `SMALLINT` und `REAL` alles Datentypen.
- Der Name eines Studenten ist hingegen ein logisches Konzept, auch wenn er durch einen Text-String dargestellt wird.

Domäne



Definition: Domäne

Die *Domäne* (EN: *domain*) $\text{dom}(a)$ eines Attributs a ist die Menge der möglichen Werte, die es annehmen kann.

- Wir können die Konzepte *Domäne* und *Datentyp* im Kontext von konzeptuellen Modellen unterscheiden⁸⁵.
- Ein Datentyp ist ein mathematisches Konzept wohingegen eine Domäne ein logisches Konzept ist.
- Z. B. sind `VARCHAR(100)`, `SMALLINT` und `REAL` alles Datentypen.
- Der Name eines Studenten ist hingegen ein logisches Konzept, auch wenn er durch einen Text-String dargestellt wird.
- Die Mobiltelefonnummern oder ID-Nummern können auch als Strings dargestellt werden, sind aber auch bestimmte Zeichen und bestimmte Längen beschränkt.

Domäne



Definition: Domäne

Die *Domäne* (EN: *domain*) $\text{dom}(a)$ eines Attributs a ist die Menge der möglichen Werte, die es annehmen kann.

- Wir können die Konzepte *Domäne* und *Datentyp* im Kontext von konzeptuellen Modellen unterscheiden⁸⁵.
- Ein Datentyp ist ein mathematisches Konzept wohingegen eine Domäne ein logisches Konzept ist.
- Z. B. sind `VARCHAR(100)`, `SMALLINT` und `REAL` alles Datentypen.
- Der Name eines Studenten ist hingegen ein logisches Konzept, auch wenn er durch einen Text-String dargestellt wird.
- Die Mobiltelefonnummern oder ID-Nummern können auch als Strings dargestellt werden, sind aber auch bestimmte Zeichen und bestimmte Längen beschränkt.
- Ein Geburtsdatum ist ein besonderes Datum, dessen Jahr aus einer bestimmten Menge möglicher Jahre kommt.

Domäne



Definition: Domäne

Die *Domäne* (EN: *domain*) $\text{dom}(a)$ eines Attributs a ist die Menge der möglichen Werte, die es annehmen kann.

- Wir können die Konzepte *Domäne* und *Datentyp* im Kontext von konzeptuellen Modellen unterscheiden⁸⁵.
- Ein Datentyp ist ein mathematisches Konzept wohingegen eine Domäne ein logisches Konzept ist.
- Der Name eines Studenten ist hingegen ein logisches Konzept, auch wenn er durch einen Text-String dargestellt wird.
- Die Mobiltelefonnummern oder ID-Nummern können auch als Strings dargestellt werden, sind aber auch bestimmte Zeichen und bestimmte Längen beschränkt.
- Ein Geburtsdatum ist ein besonderes Datum, dessen Jahr aus einer bestimmten Menge möglicher Jahre kommt.
- Das Ergebnis eine Prüfung kann eine Ganzzahl zwischen 0 und 100 sein.



Definition: Domäne

Die *Domäne* (EN: *domain*) $\text{dom}(a)$ eines Attributs a ist die Menge der möglichen Werte, die es annehmen kann.

- Wir können die Konzepte *Domäne* und *Datentyp* im Kontext von konzeptuellen Modellen unterscheiden⁸⁵.
- Ein Datentyp ist ein mathematisches Konzept wohingegen eine Domäne ein logisches Konzept ist.
- Die Mobiltelefonnummern oder ID-Nummern können auch als Strings dargestellt werden, sind aber auch bestimmte Zeichen und bestimmte Längen beschränkt.
- Ein Geburtsdatum ist ein besonderes Datum, dessen Jahr aus einer bestimmten Menge möglicher Jahre kommt.
- Das Ergebnis eine Prüfung kann eine Ganzzahl zwischen 0 und 100 sein.
- Die Domäne eines Attributes ist also eine Kombination aus Datentyp und einer Semantik, die die gültigen Werte einschränkt.

Definition: Entitätstyp

Die Menge aller Entitäten die die selben Attribute haben wird *Entitätstyp* (EN: *entity type*) genannt.



Definition: Entitätstyp

Die Menge aller Entitäten die die selben Attribute haben wird *Entitätstyp* (EN: *entity type*) genannt.

- Während also die Entität Herr Beppo ein einzelner Student ist, ist die Menge aller möglicher Studenten ein Entitätstyp.

Definition: Entitätstyp

Die Menge aller Entitäten die die selben Attribute haben wird *Entitätstyp* (EN: *entity type*) genannt.

- Während also die Entität Herr Beppo ein einzelner Student ist, ist die Menge aller möglicher Studenten ein Entitätstyp.
- Während das Modul *Programming with Python*¹⁰¹ eine einzelne Entität ist, ist die Menge aller möglicher Module ein Entitätstyp.

Definition: Entitätstyp

Die Menge aller Entitäten die die selben Attribute haben wird *Entitätstyp* (EN: *entity type*) genannt.

- Während also die Entität Herr Bebo ein einzelner Student ist, ist die Menge aller möglicher Studenten ein Entitätstyp.
- Während das Modul *Programming with Python*¹⁰¹ eine einzelne Entität ist, ist die Menge aller möglicher Module ein Entitätstyp.
- Die Professorin Frau Beba 教授 ist eine einzelne Entität, aber die Menge aller möglichen Professoren repräsentiert einen Entitätstyp.

Definition: Entitätstyp

Die Menge aller Entitäten die die selben Attribute haben wird *Entitätstyp* (EN: *entity type*) genannt.

- Während also die Entität Herr Beppo ein einzelner Student ist, ist die Menge aller möglicher Studenten ein Entitätstyp.
- Während das Modul *Programming with Python*¹⁰¹ eine einzelne Entität ist, ist die Menge aller möglicher Module ein Entitätstyp.
- Die Professorin Frau Beppo 教授 ist eine einzelne Entität, aber die Menge aller möglichen Professoren repräsentiert einen Entitätstyp.
- Entitätstypen sind Appellative oder Gattungsnamen (EN: *common nouns*) die für Gruppen von Dingen stehen^{21,37}.



Definition: Entitätstyp

Die Menge aller Entitäten die die selben Attribute haben wird *Entitätstyp* (EN: *entity type*) genannt.





Definition: Entitätstyp

Die Menge aller Entitäten die die selben Attribute haben wird *Entitätstyp* (EN: *entity type*) genannt.

- Beachten Sie den Plural *Attribute* in der Definition von Entitätstyp!

Gute Praxis



Definition: Entitätstyp

Die Menge aller Entitäten die die selben Attribute haben wird *Entitätstyp* (EN: *entity type*) genannt.

- Beachten Sie den Plural *Attribute* in der Definition von Entitätstyp!

Gute Praxis

Nur Dinge mit mehreren Attributen sollten Entitätstypen sein.

Gute Praxis



Definition: Entitätstyp

Die Menge aller Entitäten die die selben Attribute haben wird *Entitätstyp* (EN: *entity type*) genannt.

- Beachten Sie den Plural *Attribute* in der Definition von Entitätstyp!

Gute Praxis

Nur Dinge mit mehreren Attributen sollten Entitätstypen sein.

- Es ergibt ja wenig Sinn, so ein Konzept wie *Jahr* als Entitätstyp zu betrachten – es hat ja nicht mehrere Attribute.

Gute Praxis



- Beachten Sie den Plural *Attribute* in der Definition von Entitätstyp!

Gute Praxis

Nur Dinge mit mehreren Attributen sollten Entitätstypen sein.

- Es ergibt ja wenig Sinn, so ein Konzept wie *Jahr* als Entitätstyp zu betrachten – es hat ja nicht mehrere Attribute.

Gute Praxis

Jede Entität (und jeder Entitätstyp) soll genau ein Objekt (einen Objekttyp) aus der Realität modellieren (und nicht mehr als einen)⁷⁹.

Gute Praxis



- Beachten Sie den Plural *Attribute* in der Definition von Entitätstyp!

Gute Praxis

Nur Dinge mit mehreren Attributen sollten Entitätstypen sein.

- Es ergibt ja wenig Sinn, so ein Konzept wie *Jahr* als Entitätstyp zu betrachten – es hat ja nicht mehrere Attribute.

Gute Praxis

Jede Entität (und jeder Entitätstyp) soll genau ein Objekt (einen Objekttyp) aus der Realität modellieren (und nicht mehr als einen)⁷⁹.

- Wenn wir z. B. ein Autorennen modellieren, dann müssen Fahrer und Autos separate Entitätstypen sein, obwohl je ein Fahrer einem Auto zugeordnet ist.

Gute Praxis



- Es ergibt ja wenig Sinn, so ein Konzept wie *Jahr* als Entitätstyp zu betrachten – es hat ja nicht mehrere Attribute.

Gute Praxis

Jede Entität (und jeder Entitätstyp) soll genau ein Objekt (einen Objekttyp) aus der Realität modellieren (und nicht mehr als einen)⁷⁹.

- Wenn wir z. B. ein Autorennen modellieren, dann müssen Fahrer und Autos separate Entitätstypen sein, obwohl je ein Fahrer einem Auto zugeordnet ist.
- Wenn wir stattdessen den Namen des Fahrers und den Typ und die Eigenschaften der Autos in einen gemeinsamen Entitätstyp modellieren, dann werden wir wahrscheinlich Redundanz erzeugen.

Gute Praxis



- Es ergibt ja wenig Sinn, so ein Konzept wie *Jahr* als Entitätstyp zu betrachten – es hat ja nicht mehrere Attribute.

Gute Praxis

Jede Entität (und jeder Entitätstyp) soll genau ein Objekt (einen Objekttyp) aus der Realität modellieren (und nicht mehr als einen)⁷⁹.

- Wenn wir z. B. ein Autorennen modellieren, dann müssen Fahrer und Autos separate Entitätstypen sein, obwohl je ein Fahrer einem Auto zugeordnet ist.
- Wenn wir stattdessen den Namen des Fahrers und den Typ und die Eigenschaften der Autos in einen gemeinsamen Entitätstyp modellieren, dann werden wir wahrscheinlich Redundanz erzeugen.
- Wenn mehrere Autos des selben Typs am Rennen teilnehmen, dann speichern wir deren Informationen mehrmals, jeweils mit einem anderen Fahrer.

Entitätenmengen



Definition: Entitätenmenge

Eine *Entitätenmenge* (EN: *entity set*) ist eine Untermenge eines Entitätstyps.

中国安徽德国中心

合肥德国应用科学院

Entitätenmengen



Definition: Entitätenmenge

Eine *Entitätenmenge* (EN: *entity set*) ist eine Untermenge eines Entitätstyps. Sie ist eine Menge von einigen Entitäten eines Entitätstypes, die zu einer Zeit existieren.

中国安徽德国中心

合肥德国应用科技学院

Entitätenmengen



Definition: Entitätenmenge

Eine *Entitätenmenge* (EN: *entity set*) ist eine Untermenge eines Entitätstyps. Sie ist eine Menge von einigen Entitäten eines Entitätstypes, die zu einer Zeit existieren.

- Herr Bebbio ist eine einzelne Studenten-Entität, die Menge aller möglichen Studenten bildet einen Entitätstyp, und die Studenten Herr Bebbio, Herr Bibbo und Herr Bobbo zusammen sind eine Entitätenmenge.



Definition: Entitätenmenge

Eine *Entitätenmenge* (EN: *entity set*) ist eine Untermenge eines Entitätstyps. Sie ist eine Menge von einigen Entitäten eines Entitätstypes, die zu einer Zeit existieren.

- Herr Bebbio ist eine einzelne Studenten-Entität, die Menge aller möglichen Studenten bildet einen Entitätstyp, und die Studenten Herr Bebbio, Herr Bibbo und Herr Bobbo zusammen sind eine Entitätenmenge.
- Die Module *Programming with Python*¹⁰¹ und *Databases*¹⁰⁰ zusammen bilden eine Entitätenmenge, welche wiederum eine Untermenge des Entitätstyps für Module ist.



Definition: Entitätenmenge

Eine *Entitätenmenge* (EN: *entity set*) ist eine Untermenge eines Entitätstyps. Sie ist eine Menge von einigen Entitäten eines Entitätstypes, die zu einer Zeit existieren.

- Herr Bebbio ist eine einzelne Studenten-Entität, die Menge aller möglichen Studenten bildet einen Entitätstyp, und die Studenten Herr Bebbio, Herr Bibbo und Herr Bobbo zusammen sind eine Entitätenmenge.
- Die Module *Programming with Python*¹⁰¹ und *Databases*¹⁰⁰ zusammen bilden eine Entitätenmenge, welche wiederum eine Untermenge des Entitätstyps für Module ist.
- Der mathematische Terminus *Menge* ist hier wirklich korrekt.



Definition: Entitätenmenge

Eine *Entitätenmenge* (EN: *entity set*) ist eine Untermenge eines Entitätstyps. Sie ist eine Menge von einigen Entitäten eines Entitätstypes, die zu einer Zeit existieren.

- Herr Bebbio ist eine einzelne Studenten-Entität, die Menge aller möglichen Studenten bildet einen Entitätstyp, und die Studenten Herr Bebbio, Herr Bibbo und Herr Bobbo zusammen sind eine Entitätenmenge.
- Die Module *Programming with Python*¹⁰¹ und *Databases*¹⁰⁰ zusammen bilden eine Entitätenmenge, welche wiederum eine Untermenge des Entitätstyps für Module ist.
- Der mathematische Terminus *Menge* ist hier wirklich korrekt:
- Alle Entitäten haben eine einzigartige Identität und können daher von allen anderen Objekten unterschieden werden.



Definition: Entitätenmenge

Eine *Entitätenmenge* (EN: *entity set*) ist eine Untermenge eines Entitätstyps. Sie ist eine Menge von einigen Entitäten eines Entitätstypes, die zu einer Zeit existieren.

- Herr Beppo ist eine einzelne Studenten-Entität, die Menge aller möglichen Studenten bildet einen Entitätstyp, und die Studenten Herr Beppo, Herr Bibbo und Herr Bobbo zusammen sind eine Entitätenmenge.
- Die Module *Programming with Python*¹⁰¹ und *Databases*¹⁰⁰ zusammen bilden eine Entitätenmenge, welche wiederum eine Untermenge des Entitätstyps für Module ist.
- Der mathematische Terminus *Menge* ist hier wirklich korrekt
- Alle Entitäten haben eine einzigartige Identität und können daher von allen anderen Objekten unterschieden werden.
- Es gibt keine zwei identischen Herr Bibbos.



Definition: Entitätenmenge

Eine *Entitätenmenge* (EN: *entity set*) ist eine Untermenge eines Entitätstyps. Sie ist eine Menge von einigen Entitäten eines Entitätstypes, die zu einer Zeit existieren.

- Herr Bebbio ist eine einzelne Studenten-Entität, die Menge aller möglichen Studenten bildet einen Entitätstyp, und die Studenten Herr Bebbio, Herr Bibbo und Herr Bobbo zusammen sind eine Entitätenmenge.
- Die Module *Programming with Python*¹⁰¹ und *Databases*¹⁰⁰ zusammen bilden eine Entitätenmenge, welche wiederum eine Untermenge des Entitätstyps für Module ist.
- Der mathematische Terminus *Menge* ist hier wirklich korrekt
- Alle Entitäten haben eine einzigartige Identität und können daher von allen anderen Objekten unterschieden werden.
- Es gibt keine zwei identischen Herr Bibbos.
- Daher können Studenten in eine Menge gruppiert werden und auch Entitätstypen sind Mengen.

Interessante Aspekte 1: Fenster zur Welt

- Wenn wir uns das konzeptuelle Design von Datenbanken auf diese Art anschauen, dann bemerken wir ein paar interessante Aspekte.



Interessante Aspekte 1: Fenster zur Welt



- Wenn wir uns das konzeptuelle Design von Datenbanken auf diese Art anschauen, dann bemerken wir ein paar interessante Aspekte.
- Zum Beispiel modellieren wir immer nur ein kleines Fenster zur echten Welt.

Interessante Aspekte 1: Fenster zur Welt



- Wenn wir uns das konzeptuelle Design von Datenbanken auf diese Art anschauen, dann bemerken wir ein paar interessante Aspekte.
- Zum Beispiel modellieren wir immer nur ein kleines Fenster zur echten Welt.
- Wenn wir über Studenten, Module, und Studiengänge als Entitätstypen sprechen, dann nur im Kontext unserer spezifischen Applikation.

Interessante Aspekte 1: Fenster zur Welt



- Wenn wir uns das konzeptuelle Design von Datenbanken auf diese Art anschauen, dann bemerken wir ein paar interessante Aspekte.
- Zum Beispiel modellieren wir immer nur ein kleines Fenster zur echten Welt.
- Wenn wir über Studenten, Module, und Studiengänge als Entitätstypen sprechen, dann nur im Kontext unserer spezifischen Applikation.
- Natürlich gibt es in unserer realen, großen, weiten Welt Studenten auch an anderen Universitäten in anderen Ländern.

Interessante Aspekte 1: Fenster zur Welt



- Wenn wir uns das konzeptuelle Design von Datenbanken auf diese Art anschauen, dann bemerken wir ein paar interessante Aspekte.
- Zum Beispiel modellieren wir immer nur ein kleines Fenster zur echten Welt.
- Wenn wir über Studenten, Module, und Studiengänge als Entitätstypen sprechen, dann nur im Kontext unserer spezifischen Applikation.
- Natürlich gibt es in unserer realen, großen, weiten Welt Studenten auch an anderen Universitäten in anderen Ländern.
- Diese Studenten könnten andere Attribute haben als unsere.

Interessante Aspekte 1: Fenster zur Welt



- Wenn wir uns das konzeptuelle Design von Datenbanken auf diese Art anschauen, dann bemerken wir ein paar interessante Aspekte.
- Zum Beispiel modellieren wir immer nur ein kleines Fenster zur echten Welt.
- Wenn wir über Studenten, Module, und Studiengänge als Entitätstypen sprechen, dann nur im Kontext unserer spezifischen Applikation.
- Natürlich gibt es in unserer realen, großen, weiten Welt Studenten auch an anderen Universitäten in anderen Ländern.
- Diese Studenten könnten andere Attribute haben als unsere.
- Sie haben wahrscheinlich keinen chinesische Ausweis (中华人民共和国居民身份证).

Interessante Aspekte 1: Fenster zur Welt



- Wenn wir uns das konzeptuelle Design von Datenbanken auf diese Art anschauen, dann bemerken wir ein paar interessante Aspekte.
- Zum Beispiel modellieren wir immer nur ein kleines Fenster zur echten Welt.
- Wenn wir über Studenten, Module, und Studiengänge als Entitätstypen sprechen, dann nur im Kontext unserer spezifischen Applikation.
- Natürlich gibt es in unserer realen, großen, weiten Welt Studenten auch an anderen Universitäten in anderen Ländern.
- Diese Studenten könnten andere Attribute haben als unsere.
- Sie haben wahrscheinlich keinen chinesische Ausweis (中华人民共和国居民身份证).
- Sie spielen aber in unserem Modell von unserem kleinen Teil der Welt keine Rolle.

Interessante Aspekte 1: Fenster zur Welt



- Wenn wir uns das konzeptuelle Design von Datenbanken auf diese Art anschauen, dann bemerken wir ein paar interessante Aspekte.
- Zum Beispiel modellieren wir immer nur ein kleines Fenster zur echten Welt.
- Wenn wir über Studenten, Module, und Studiengänge als Entitätstypen sprechen, dann nur im Kontext unserer spezifischen Applikation.
- Natürlich gibt es in unserer realen, großen, weiten Welt Studenten auch an anderen Universitäten in anderen Ländern.
- Diese Studenten könnten andere Attribute haben als unsere.
- Sie haben wahrscheinlich keinen chinesische Ausweis (中华人民共和国居民身份证).
- Sie spielen aber in unserem Modell von unserem kleinen Teil der Welt keine Rolle.
- Auch modellieren wir nur die Aspekte von Studenten, die für unsere Anwendung relevant sind.

Interessante Aspekte 1: Fenster zur Welt



- Wenn wir uns das konzeptuelle Design von Datenbanken auf diese Art anschauen, dann bemerken wir ein paar interessante Aspekte.
- Zum Beispiel modellieren wir immer nur ein kleines Fenster zur echten Welt.
- Wenn wir über Studenten, Module, und Studiengänge als Entitätstypen sprechen, dann nur im Kontext unserer spezifischen Applikation.
- Natürlich gibt es in unserer realen, großen, weiten Welt Studenten auch an anderen Universitäten in anderen Ländern.
- Diese Studenten könnten andere Attribute haben als unsere.
- Sie haben wahrscheinlich keinen chinesische Ausweis (中华人民共和国居民身份证).
- Sie spielen aber in unserem Modell von unserem kleinen Teil der Welt keine Rolle.
- Auch modellieren wir nur die Aspekte von Studenten, die für unsere Anwendung relevant sind.
- Wir modellieren weder deren Hobbies, Lieblingslied, Schuhgröße, Lieblingsessen noch deren Haarfarbe.

Interessante Aspekte 1: Fenster zur Welt



- Wenn wir uns das konzeptuelle Design von Datenbanken auf diese Art anschauen, dann bemerken wir ein paar interessante Aspekte.
- Zum Beispiel modellieren wir immer nur ein kleines Fenster zur echten Welt.
- Wenn wir über Studenten, Module, und Studiengänge als Entitätstypen sprechen, dann nur im Kontext unserer spezifischen Applikation.
- Natürlich gibt es in unserer realen, großen, weiten Welt Studenten auch an anderen Universitäten in anderen Ländern.
- Diese Studenten könnten andere Attribute haben als unsere.
- Sie haben wahrscheinlich keinen chinesische Ausweis (中华人民共和国居民身份证).
- Sie spielen aber in unserem Modell von unserem kleinen Teil der Welt keine Rolle.
- Auch modellieren wir nur die Aspekte von Studenten, die für unsere Anwendung relevant sind.
- Wir modellieren weder deren Hobbies, Lieblingslied, Schuhgröße, Lieblingsessen noch deren Haarfarbe.
- Diese spielen nämlich auch keine Rolle in unserer Anwendung.

Interessante Aspekte 1: Fenster zur Welt



- Zum Beispiel modellieren wir immer nur ein kleines Fenster zur echten Welt.
- Wenn wir über Studenten, Module, und Studiengänge als Entitätstypen sprechen, dann nur im Kontext unserer spezifischen Applikation.
- Natürlich gibt es in unserer realen, großen, weiten Welt Studenten auch an anderen Universitäten in anderen Ländern.
- Diese Studenten könnten andere Attribute haben als unsere.
- Sie haben wahrscheinlich keinen chinesische Ausweis (中华人民共和国居民身份证).
- Sie spielen aber in unserem Modell von unserem kleinen Teil der Welt keine Rolle.
- Auch modellieren wir nur die Aspekte von Studenten, die für unsere Anwendung relevant sind.
- Wir modellieren weder deren Hobbies, Lieblingslied, Schuhgröße, Lieblingsessen noch deren Haarfarbe.
- Diese spielen nämlich auch keine Rolle in unserer Anwendung.
- Der Entitätentyp „Student“ umfasst also nicht das komplette reale Konzept von Studenten.

Interessante Aspekte 1: Fenster zur Welt



- Wenn wir über Studenten, Module, und Studiengänge als Entitätstypen sprechen, dann nur im Kontext unserer spezifischen Applikation.
- Natürlich gibt es in unserer realen, großen, weiten Welt Studenten auch an anderen Universitäten in anderen Ländern.
- Diese Studenten könnten andere Attribute haben als unsere.
- Sie haben wahrscheinlich keinen chinesische Ausweis (中华人民共和国居民身份证).
- Sie spielen aber in unserem Modell von unserem kleinen Teil der Welt keine Rolle.
- Auch modellieren wir nur die Aspekte von Studenten, die für unsere Anwendung relevant sind.
- Wir modellieren weder deren Hobbies, Lieblingslied, Schuhgröße, Lieblingsessen noch deren Haarfarbe.
- Diese spielen nämlich auch keine Rolle in unserer Anwendung.
- Der Entitätentyp „Student“ umfasst also nicht das komplette reale Konzept von Studenten.
- Er ist nur eine simplifizierte Abbildung dieses Konzepts in unsere Anwendung.

Interessante Aspekte 2: OOP



- Wenn Sie schon einen Kurs über Programmieren (wie z. B. ¹⁰¹) gehabt haben, dann wird sich diese Art zu modellieren etwas wie Objekt-orientierte Programmierung (OOP) anfühlen.

Interessante Aspekte 2: OOP



- Wenn Sie schon einen Kurs über Programmieren (wie z. B. ¹⁰¹) gehabt haben, dann wird sich diese Art zu modellieren etwas wie Objekt-orientierte Programmierung (OOP) anfühlen.
- Wir könnten uns Entitätstypen wie Klassen vorstellen und Entitäten wie deren Instanzen.

Interessante Aspekte 2: OOP



- Wenn Sie schon einen Kurs über Programmieren (wie z. B. ¹⁰¹) gehabt haben, dann wird sich diese Art zu modellieren etwas wie Objekt-orientierte Programmierung (OOP) anfühlen.
- Wir könnten uns Entitätstypen wie Klassen vorstellen und Entitäten wie deren Instanzen.
- Attribute wären dann, naja, Attribute.

Interessante Aspekte 2: OOP



- Wenn Sie schon einen Kurs über Programmieren (wie z. B. ¹⁰¹) gehabt haben, dann wird sich diese Art zu modellieren etwas wie Objekt-orientierte Programmierung (OOP) anfühlen.
- Wir könnten uns Entitätstypen wie Klassen vorstellen und Entitäten wie deren Instanzen.
- Attribute wären dann, naja, Attribute.
- Ich denke, Datenbank-Theoretiker würden diese Art zu denken vielleicht nicht leiden, aber ich denke, dass sie nicht falsch ist.

Interessante Aspekte 2: OOP



- Wenn Sie schon einen Kurs über Programmieren (wie z. B. ¹⁰¹) gehabt haben, dann wird sich diese Art zu modellieren etwas wie Objekt-orientierte Programmierung (OOP) anfühlen.
- Wir könnten uns Entitätstypen wie Klassen vorstellen und Entitäten wie deren Instanzen.
- Attribute wären dann, naja, Attribute.
- Ich denke, Datenbank-Theoretiker würden diese Art zu denken vielleicht nicht leiden, aber ich denke, dass sie nicht falsch ist.
- Das ist keine schlechte Analogie.

Interessante Aspekte 2: OOP



- Wenn Sie schon einen Kurs über Programmieren (wie z. B. ¹⁰¹) gehabt haben, dann wird sich diese Art zu modellieren etwas wie Objekt-orientierte Programmierung (OOP) anfühlen.
- Wir könnten uns Entitätstypen wie Klassen vorstellen und Entitäten wie deren Instanzen.
- Attribute wären dann, naja, Attribute.
- Ich denke, Datenbank-Theoretiker würden diese Art zu denken vielleicht nicht leiden, aber ich denke, dass sie nicht falsch ist.
- Das ist keine schlechte Analogie.
- Wenn wir dann später zu Beziehungen zwischen Entitäten kommen, passt sie vielleicht aber nicht mehr so gut.



Entity-Relationship Diagramme (ERDs)



- Wir wissen, dass Entitätstypen und ihre Attribute im Grunde Datenstrukturen beim Programmieren entsprechen.

- Wir wissen, dass Entitätstypen und ihre Attribute im Grunde Datenstrukturen beim Programmieren entsprechen.
- Sie sind eine der wichtigen Komponenten von konzeptuellen Modellen.

- Wir wissen, dass Entitätstypen und ihre Attribute im Grunde Datenstrukturen beim Programmieren entsprechen.
- Sie sind eine der wichtigen Komponenten von konzeptuellen Modellen.
- Aber wie schreiben wir sie auf?

- Wir wissen, dass Entitätstypen und ihre Attribute im Grunde Datenstrukturen beim Programmieren entsprechen.
- Sie sind eine der wichtigen Komponenten von konzeptuellen Modellen.
- Aber wie schreiben wir sie auf?
- Wie können wir sie spezifizieren?

Entity-Relationship Diagramme (ERDs)

- Es gibt eine grafische Methode zum Spezifizieren von Entitätstypen und ihren Beziehungen.



Entity-Relationship Diagramme (ERDs)



- Es gibt eine grafische Methode zum Spezifizieren von Entitätstypen und ihren Beziehungen: Entity-Relationship Diagramme (ERDs)^{3,22-24,52,102}.

Entity-Relationship Diagramme (ERDs)



- Es gibt eine grafische Methode zum Spezifizieren von Entitätstypen und ihren Beziehungen: Entity-Relationship Diagramme (ERDs)^{3,22-24,52,102}.
- Es gibt eine Vielzahl von Notationen, die für ERDs genutzt werden kann.

Entity-Relationship Diagramme (ERDs)



- Es gibt eine grafische Methode zum Spezifizieren von Entitätstypen und ihren Beziehungen: Entity-Relationship Diagramme (ERDs)^{3,22-24,52,102}.
- Es gibt eine Vielzahl von Notationen, die für ERDs genutzt werden kann:
 - die ursprüngliche Notationen von Bachman² und Chen^{23,24} werden noch immer verwendet.

Entity-Relationship Diagramme (ERDs)



- Es gibt eine grafische Methode zum Spezifizieren von Entitätstypen und ihren Beziehungen: Entity-Relationship Diagramme (ERDs)^{3,22-24,52,102}.
- Es gibt eine Vielzahl von Notationen, die für ERDs genutzt werden kann:
 - die ursprüngliche Notationen von Bachman² und Chen^{23,24} werden noch immer verwendet,
 - die Krähenfußnotation (EN: *Crow's Foot*) wird sehr häufig eingesetzt^{19,38}.

Entity-Relationship Diagramme (ERDs)



- Es gibt eine grafische Methode zum Spezifizieren von Entitätstypen und ihren Beziehungen: Entity-Relationship Diagramme (ERDs)^{3,22-24,52,102}.
- Es gibt eine Vielzahl von Notationen, die für ERDs genutzt werden kann:
 - die ursprüngliche Notationen von Bachman² und Chen^{23,24} werden noch immer verwendet,
 - die Krähenfußnotation (EN: *Crow's Foot*) wird sehr häufig eingesetzt^{19,38},
 - es gibt auch die umfassende und standardisierte IDEF1X Syntax^{14,51}.

Entity-Relationship Diagramme (ERDs)



- Es gibt eine grafische Methode zum Spezifizieren von Entitätstypen und ihren Beziehungen: Entity-Relationship Diagramme (ERDs)^{3,22-24,52,102}.
- Es gibt eine Vielzahl von Notationen, die für ERDs genutzt werden kann:
 - die ursprüngliche Notationen von Bachman² und Chen^{23,24} werden noch immer verwendet,
 - die Krähenfußnotation (EN: *Crow's Foot*) wird sehr häufig eingesetzt^{19,38},
 - es gibt auch die umfassende und standardisierte IDEF1X Syntax^{14,51} und
 - auch UML^{9,68,97} wird verwendet.

Entity-Relationship Diagramme (ERDs)



- Es gibt eine grafische Methode zum Spezifizieren von Entitätstypen und ihren Beziehungen: Entity-Relationship Diagramme (ERDs)^{3,22-24,52,102}.
- Es gibt eine Vielzahl von Notationen, die für ERDs genutzt werden kann:
 - die ursprüngliche Notationen von Bachman² und Chen^{23,24} werden noch immer verwendet,
 - die Krähenfußnotation (EN: *Crow's Foot*) wird sehr häufig eingesetzt^{19,38},
 - es gibt auch die umfassende und standardisierte IDEF1X Syntax^{14,51} und
 - auch UML^{9,68,97} wird verwendet.
- Es gibt viele Varianten von Diagrammen, die als ERD verwendet werden können.

Entity-Relationship Diagramme (ERDs)



- Es gibt eine grafische Methode zum Spezifizieren von Entitätstypen und ihren Beziehungen: Entity-Relationship Diagramme (ERDs)^{3,22-24,52,102}.
- Es gibt eine Vielzahl von Notationen, die für ERDs genutzt werden kann.:
 - die ursprüngliche Notationen von Bachman² und Chen^{23,24} werden noch immer verwendet,
 - die Krähenfußnotation (EN: *Crow's Foot*) wird sehr häufig eingesetzt^{19,38},
 - es gibt auch die umfassende und standardisierte IDEF1X Syntax^{14,51} und
 - auch UML^{9,68,97} wird verwendet.
- Es gibt viele Varianten von Diagrammen, die als ERD verwendet werden können.
- Shamshin listet neun verschiedene Varianten⁸³.

Entity-Relationship Diagramme (ERDs)



- Es gibt eine grafische Methode zum Spezifizieren von Entitätstypen und ihren Beziehungen: Entity-Relationship Diagramme (ERDs)^{3,22-24,52,102}.
- Es gibt eine Vielzahl von Notationen, die für ERDs genutzt werden kann.:
 - die ursprüngliche Notationen von Bachman² und Chen^{23,24} werden noch immer verwendet,
 - die Krähenfußnotation (EN: *Crow's Foot*) wird sehr häufig eingesetzt^{19,38},
 - es gibt auch die umfassende und standardisierte IDEF1X Syntax^{14,51} und
 - auch UML^{9,68,97} wird verwendet.
- Es gibt viele Varianten von Diagrammen, die als ERD verwendet werden können.
- Shamshin listet neun verschiedene Varianten⁸³.
- Schweppe und Scholz listet zwei grundlegende Methoden und gibt auch verschiedene Varianten von UML-basierten Ansätzen an⁷⁷.

Entity-Relationship Diagramme (ERDs)



- Es gibt eine grafische Methode zum Spezifizieren von Entitätstypen und ihren Beziehungen: Entity-Relationship Diagramme (ERDs)^{3,22-24,52,102}.
- Es gibt eine Vielzahl von Notationen, die für ERDs genutzt werden kann.:
 - die ursprüngliche Notationen von Bachman² und Chen^{23,24} werden noch immer verwendet,
 - die Krähenfußnotation (EN: *Crow's Foot*) wird sehr häufig eingesetzt^{19,38},
 - es gibt auch die umfassende und standardisierte IDEF1X Syntax^{14,51} und
 - auch UML^{9,68,97} wird verwendet.
- Es gibt viele Varianten von Diagrammen, die als ERD verwendet werden können.
- Shamshin listet neun verschiedene Varianten⁸³.
- Schweppe und Scholz listet zwei grundlegende Methoden und gibt auch verschiedene Varianten von UML-basierten Ansätzen an⁷⁷.
- Die Notationen, die von Vandenberg verwendet werden, sind wieder leicht anders⁹⁹.

Entity-Relationship Diagramme (ERDs)



- Es gibt eine grafische Methode zum Spezifizieren von Entitätstypen und ihren Beziehungen: Entity-Relationship Diagramme (ERDs)^{3,22-24,52,102}.
- Es gibt eine Vielzahl von Notationen, die für ERDs genutzt werden kann.:
 - die ursprüngliche Notationen von Bachman² und Chen^{23,24} werden noch immer verwendet,
 - die Krähenfußnotation (EN: *Crow's Foot*) wird sehr häufig eingesetzt^{19,38},
 - es gibt auch die umfassende und standardisierte IDEF1X Syntax^{14,51} und
 - auch UML^{9,68,97} wird verwendet.
- Es gibt viele Varianten von Diagrammen, die als ERD verwendet werden können.
- Shamshin listet neun verschiedene Varianten⁸³.
- Schweppe und Scholz listet zwei grundlegende Methoden und gibt auch verschiedene Varianten von UML-basierten Ansätzen an⁷⁷.
- Die Notationen, die von Vandenberg verwendet werden, sind wieder leicht anders⁹⁹.
- ERDs werden also wahrscheinlich leicht anders aussehen, je nach dem, wer sie gemalt hat und welche Werkzeuge diese Person verwendet hat.

Entity-Relationship Diagramme (ERDs)



- Es gibt eine grafische Methode zum Spezifizieren von Entitätstypen und ihren Beziehungen: Entity-Relationship Diagramme (ERDs)^{3,22-24,52,102}.
- Es gibt eine Vielzahl von Notationen, die für ERDs genutzt werden kann.:
 - die ursprüngliche Notationen von Bachman² und Chen^{23,24} werden noch immer verwendet,
 - die Krähenfußnotation (EN: *Crow's Foot*) wird sehr häufig eingesetzt^{19,38},
 - es gibt auch die umfassende und standardisierte IDEF1X Syntax^{14,51} und
 - auch UML^{9,68,97} wird verwendet.
- Es gibt viele Varianten von Diagrammen, die als ERD verwendet werden können.
- Shamshin listet neun verschiedene Varianten⁸³.
- Schweppe und Scholz listet zwei grundlegende Methoden und gibt auch verschiedene Varianten von UML-basierten Ansätzen an⁷⁷.
- Die Notationen, die von Vandenberg verwendet werden, sind wieder leicht anders⁹⁹.
- ERDs werden also wahrscheinlich leicht anders aussehen, je nach dem, wer sie gemalt hat und welche Werkzeuge diese Person verwendet hat.
- Sie sind aber leicht zu verstehen, also sind diese Unterschiede nicht so wichtig.

ERDs Zeichnen

- Wir werden uns im Folgenden verschiedene ERDs anschauen.



ERDs Zeichnen

- Wir werden uns im Folgenden verschiedene ERDs anschauen.
- Unser Ziel ist aber immer, *anwendbares* Wissen zu lernen.



ERDs Zeichnen

- Wir werden uns im Folgenden verschiedene ERDs anschauen.
- Unser Ziel ist aber immer, *anwendbares* Wissen zu lernen.
- Wir werden also nicht nur ERDs anschauen, wir werden sie auch zeichnen.



ERDs Zeichnen

- Wir werden uns im Folgenden verschiedene ERDs anschauen.
- Unser Ziel ist aber immer, *anwendbares* Wissen zu lernen.
- Wir werden also nicht nur ERDs anschauen, wir werden sie auch zeichnen.
- Das ist sehr einfach.



ERDs Zeichnen



- Wir werden uns im Folgenden verschiedene ERDs anschauen.
- Unser Ziel ist aber immer, *anwendbares* Wissen zu lernen.
- Wir werden also nicht nur ERDs anschauen, wir werden sie auch zeichnen.
- Das ist sehr einfach.
- Wenn Sie die grundlegende Syntax verstanden haben, dann können Sie diese Diagramme mit beliebiger Zeichensoftware malen.

ERDs Zeichnen



- Wir werden uns im Folgenden verschiedene ERDs anschauen.
- Unser Ziel ist aber immer, *anwendbares* Wissen zu lernen.
- Wir werden also nicht nur ERDs anschauen, wir werden sie auch zeichnen.
- Das ist sehr einfach.
- Wenn Sie die grundlegende Syntax verstanden haben, dann können Sie diese Diagramme mit beliebiger Zeichensoftware malen:
 - Sie könnten Grafikprogramme wie LibreOffice Draw oder Inkscape^{53,74} verwenden.

ERDs Zeichnen



- Wir werden uns im Folgenden verschiedene ERDs anschauen.
- Unser Ziel ist aber immer, *anwendbares* Wissen zu lernen.
- Wir werden also nicht nur ERDs anschauen, wir werden sie auch zeichnen.
- Das ist sehr einfach.
- Wenn Sie die grundlegende Syntax verstanden haben, dann können Sie diese Diagramme mit beliebiger Zeichensoftware malen:
 - Sie könnten Grafikprogramme wie LibreOffice Draw oder Inkscape^{53,74} verwenden. Dann müssen Sie aber alle Formen selber einzeln malen, was zu Inkonsistenzen führt und generell sehr aufwändig ist.

ERDs Zeichnen



- Wir werden uns im Folgenden verschiedene ERDs anschauen.
- Unser Ziel ist aber immer, *anwendbares* Wissen zu lernen.
- Wir werden also nicht nur ERDs anschauen, wir werden sie auch zeichnen.
- Das ist sehr einfach.
- Wenn Sie die grundlegende Syntax verstanden haben, dann können Sie diese Diagramme mit beliebiger Zeichensoftware malen:
 - Sie könnten Grafikprogramme wie LibreOffice Draw oder Inkscape^{53,74} verwenden. Dann müssen Sie aber alle Formen selber einzeln malen, was zu Inkonsistenzen führt und generell sehr aufwändig ist.
 - Dann gibt es Werkzeuge wie PgModeler¹ oder MySQL Workbench⁶², die viel besser Unterstützung für ERDs bieten.

ERDs Zeichnen



- Wir werden uns im Folgenden verschiedene ERDs anschauen.
- Unser Ziel ist aber immer, *anwendbares* Wissen zu lernen.
- Wir werden also nicht nur ERDs anschauen, wir werden sie auch zeichnen.
- Das ist sehr einfach.
- Wenn Sie die grundlegende Syntax verstanden haben, dann können Sie diese Diagramme mit beliebiger Zeichensoftware malen:
 - Sie könnten Grafikprogramme wie LibreOffice Draw oder Inkscape^{53,74} verwenden. Dann müssen Sie aber alle Formen selber einzeln malen, was zu Inkonsistenzen führt und generell sehr aufwändig ist.
 - Dann gibt es Werkzeuge wie PgModeler¹ oder MySQL Workbench⁶², die viel besser Unterstützung für ERDs bieten. Diese Werkzeuge sind aber an bestimmte Technologien gebunden (hier: PostgreSQL und MySQL).

ERDs Zeichnen



- Wir werden uns im Folgenden verschiedene ERDs anschauen.
- Unser Ziel ist aber immer, *anwendbares* Wissen zu lernen.
- Wir werden also nicht nur ERDs anschauen, wir werden sie auch zeichnen.
- Das ist sehr einfach.
- Wenn Sie die grundlegende Syntax verstanden haben, dann können Sie diese Diagramme mit beliebiger Zeichensoftware malen:
 - Sie könnten Grafikprogramme wie LibreOffice Draw oder Inkscape^{53,74} verwenden. Dann müssen Sie aber alle Formen selber einzeln malen, was zu Inkonsistenzen führt und generell sehr aufwändig ist.
 - Dann gibt es Werkzeuge wie PgModeler¹ oder MySQL Workbench⁶², die viel besser Unterstützung für ERDs bieten. Diese Werkzeuge sind aber an bestimmte Technologien gebunden (hier: PostgreSQL und MySQL). Sie sind daher für die Arbeit auf der konzeptuellen Ebene, die eben gerade technologieunabhängig sein soll, ungeeignet.

ERDs Zeichnen



- Wir werden uns im Folgenden verschiedene ERDs anschauen.
- Unser Ziel ist aber immer, *anwendbares* Wissen zu lernen.
- Wir werden also nicht nur ERDs anschauen, wir werden sie auch zeichnen.
- Das ist sehr einfach.
- Wenn Sie die grundlegende Syntax verstanden haben, dann können Sie diese Diagramme mit beliebiger Zeichensoftware malen:
 - Sie könnten Grafikprogramme wie LibreOffice Draw oder Inkscape^{53,74} verwenden. Dann müssen Sie aber alle Formen selber einzeln malen, was zu Inkonsistenzen führt und generell sehr aufwändig ist.
 - Dann gibt es Werkzeuge wie PgModeler¹ oder MySQL Workbench⁶², die viel besser Unterstützung für ERDs bieten. Diese Werkzeuge sind aber an bestimmte Technologien gebunden (hier: PostgreSQL und MySQL). Sie sind daher für die Arbeit auf der konzeptuellen Ebene, die eben gerade technologieunabhängig sein soll, ungeeignet.
 - Es gibt viele Werkzeuge, die wir ausprobieren könnten ... Brumm listed über 70 in [15].

ERDs Zeichnen



- Wir werden uns im Folgenden verschiedene ERDs anschauen.
- Unser Ziel ist aber immer, *anwendbares* Wissen zu lernen.
- Wir werden also nicht nur ERDs anschauen, wir werden sie auch zeichnen.
- Das ist sehr einfach.
- Wenn Sie die grundlegende Syntax verstanden haben, dann können Sie diese Diagramme mit beliebiger Zeichensoftware malen:
 - Sie könnten Grafikprogramme wie LibreOffice Draw oder Inkscape^{53,74} verwenden. Dann müssen Sie aber alle Formen selber einzeln malen, was zu Inkonsistenzen führt und generell sehr aufwändig ist.
 - Dann gibt es Werkzeuge wie PgModeler¹ oder MySQL Workbench⁶², die viel besser Unterstützung für ERDs bieten. Diese Werkzeuge sind aber an bestimmte Technologien gebunden (hier: PostgreSQL und MySQL). Sie sind daher für die Arbeit auf der konzeptuellen Ebene, die eben gerade technologieunabhängig sein soll, ungeeignet.
 - Es gibt viele Werkzeuge, die wir ausprobieren könnten ... Brumm listed über 70 in [15].
 - Ich habe mich für yEd^{80,106} entschieden.

ERDs Zeichnen



- Wir werden uns im Folgenden verschiedene ERDs anschauen.
- Unser Ziel ist aber immer, *anwendbares* Wissen zu lernen.
- Wir werden also nicht nur ERDs anschauen, wir werden sie auch zeichnen.
- Das ist sehr einfach.
- Wenn Sie die grundlegende Syntax verstanden haben, dann können Sie diese Diagramme mit beliebiger Zeichensoftware malen:
 - Sie könnten Grafikprogramme wie LibreOffice Draw oder Inkscape^{53,74} verwenden. Dann müssen Sie aber alle Formen selber einzeln malen, was zu Inkonsistenzen führt und generell sehr aufwändig ist.
 - Dann gibt es Werkzeuge wie PgModeler¹ oder MySQL Workbench⁶², die viel besser Unterstützung für ERDs bieten. Diese Werkzeuge sind aber an bestimmte Technologien gebunden (hier: PostgreSQL und MySQL). Sie sind daher für die Arbeit auf der konzeptuellen Ebene, die eben gerade technologieunabhängig sein soll, ungeeignet.
 - Es gibt viele Werkzeuge, die wir ausprobieren könnten ... Brumm listed über 70 in [15].
 - Ich habe mich für yEd^{80,106} entschieden. Das ist ein Graphen-Editor, der das Zeichnen und Arrangieren von ERDs unterstützt und unabhängig von einem Datenmodell ist.

ERDs Zeichnen



- Wir werden uns im Folgenden verschiedene ERDs anschauen.
- Unser Ziel ist aber immer, *anwendbares* Wissen zu lernen.
- Wir werden also nicht nur ERDs anschauen, wir werden sie auch zeichnen.
- Das ist sehr einfach.
- Wenn Sie die grundlegende Syntax verstanden haben, dann können Sie diese Diagramme mit beliebiger Zeichensoftware malen:
 - Sie könnten Grafikprogramme wie LibreOffice Draw oder Inkscape^{53,74} verwenden. Dann müssen Sie aber alle Formen selber einzeln malen, was zu Inkonsistenzen führt und generell sehr aufwändig ist.
 - Dann gibt es Werkzeuge wie PgModeler¹ oder MySQL Workbench⁶², die viel besser Unterstützung für ERDs bieten. Diese Werkzeuge sind aber an bestimmte Technologien gebunden (hier: PostgreSQL und MySQL). Sie sind daher für die Arbeit auf der konzeptuellen Ebene, die eben gerade technologieunabhängig sein soll, ungeeignet.
 - Es gibt viele Werkzeuge, die wir ausprobieren könnten ... Brumm listed über 70 in [15].
 - Ich habe mich für yEd^{80,106} entschieden. Das ist ein Graphen-Editor, der das Zeichnen und Arrangieren von ERDs unterstützt und unabhängig von einem Datenmodell ist.
- Wir probieren diesen Editor an einem Beispiel aus, dann können Sie ihn selbst verwenden.



ERDs mit yEd Zeichnen



- Kurze Notiz: Im Kontext von ERDs werden Entitätstypen manchmal Entitäten genannt.

Vorbereitung



- Kurze Notiz: Im Kontext von ERDs werden Entitätstypen manchmal Entitäten genannt.
- Egal, lassen wir uns davon nicht verwirren.

Vorbereitung



- Kurze Notiz: Im Kontext von ERDs werden Entitätstypen manchmal Entitäten genannt.
- Egal, lassen wir uns davon nicht verwirren.
- Wir wollen nun den Entitätstyp *Student* modellieren.



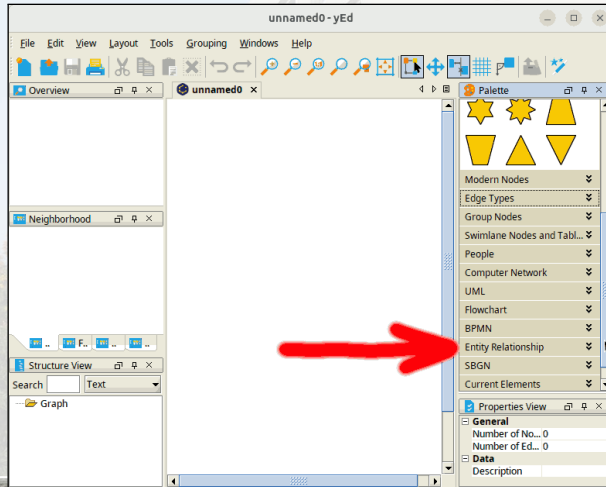
- Kurze Notiz: Im Kontext von ERDs werden Entitätstypen manchmal Entitäten genannt.
- Egal, lassen wir uns davon nicht verwirren.
- Wir wollen nun den Entitätstyp *Student* modellieren.
- Von der Anforderungsanalyse wissen wir, dass jeder Student einen Namen hat, einen von der Regierung ausgestellten Ausweis (ID, 中国公民身份号码¹¹⁰), und auch einen Studentenausweis, eine Mobiltelefonnummer, eine Adresse, und ein Geburtsdatum (EN: *date of birth, DOB*).



- Kurze Notiz: Im Kontext von ERDs werden Entitätstypen manchmal Entitäten genannt.
- Egal, lassen wir uns davon nicht verwirren.
- Wir wollen nun den Entitätstyp *Student* modellieren.
- Von der Anforderungsanalyse wissen wir, dass jeder Student einen Namen hat, einen von der Regierung ausgestellten Ausweis (ID, 中国公民身份号码¹¹⁰), und auch einen Studentenausweis, eine Mobiltelefonnummer, eine Adresse, und ein Geburtsdatum (EN: *date of birth*, *DOB*).
- Das wollen wir nun modellieren.

Student ERD mit yEd Zeichnen

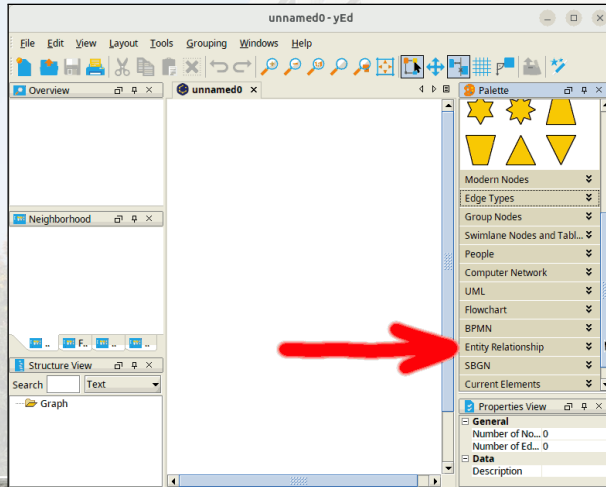
- Wir öffnen yEd.



Student ERD mit yEd Zeichnen



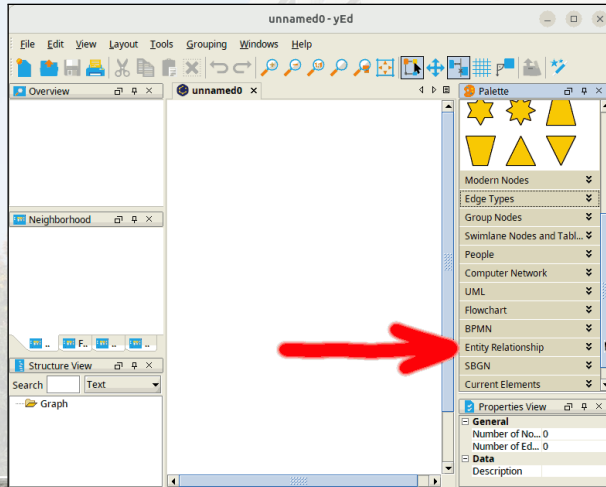
- Wir öffnen yEd.
- Wir scrollen die **Palette** rechts runter, bis wir **Entity Relationship** finden.



Student ERD mit yEd Zeichnen



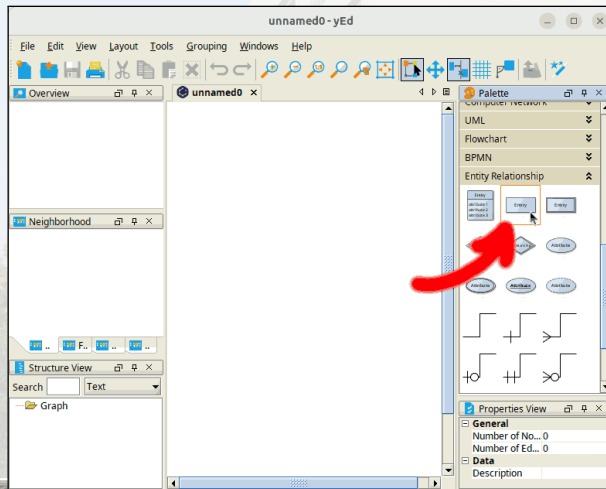
- Wir scrollen die **Palette** rechts runter, bis wir **Entity Relationship** finden.
- Wir klicken auf **Entity Relationship**.



Student ERD mit yEd Zeichnen

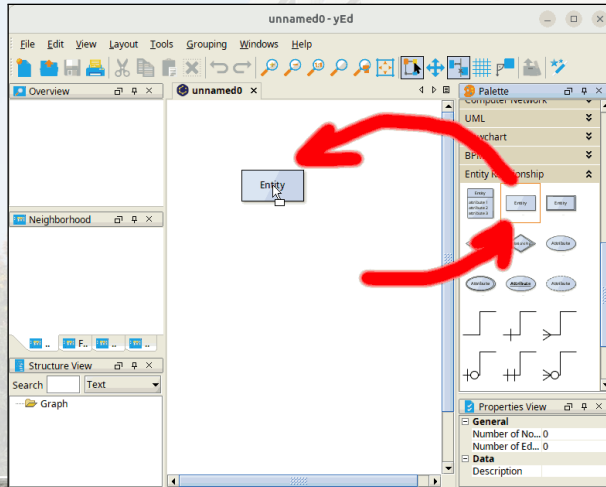


- Wir klicken nun auf der Entity-Symbol und ziehen es in das leere Dokument.



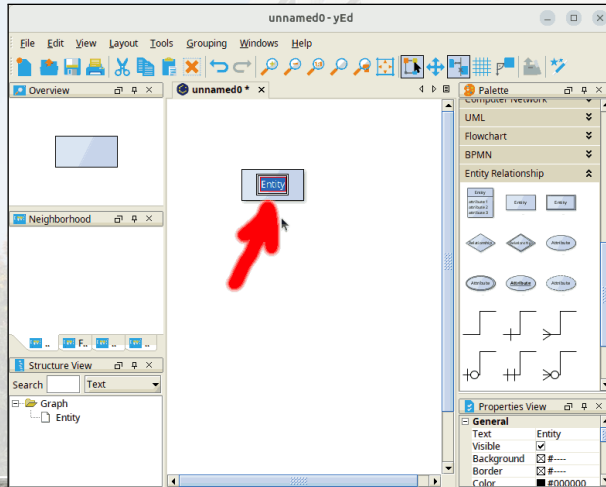
Student ERD mit yEd Zeichnen

- Wir haben das Entity-Symbol in das Dokument gezogen.



Student ERD mit yEd Zeichnen

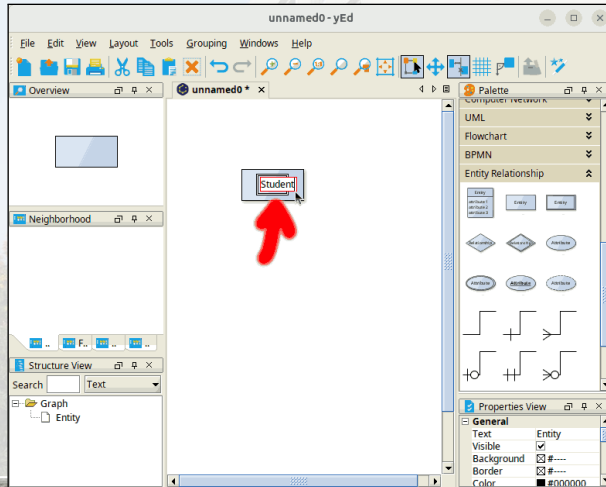
- Wir doppel-klicken in das neue Entity-Symbol im Dokument um seinen Namen zu bearbeiten.



Student ERD mit yEd Zeichnen

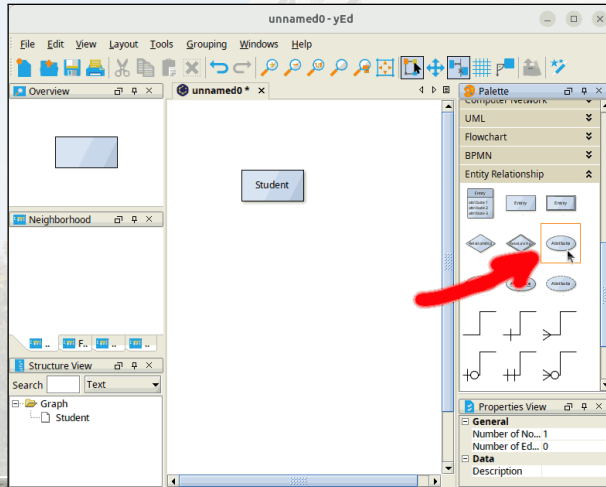


- Wir ändern den Name des Entitätstyps in „Student“ und drücken .



Student ERD mit yEd Zeichnen

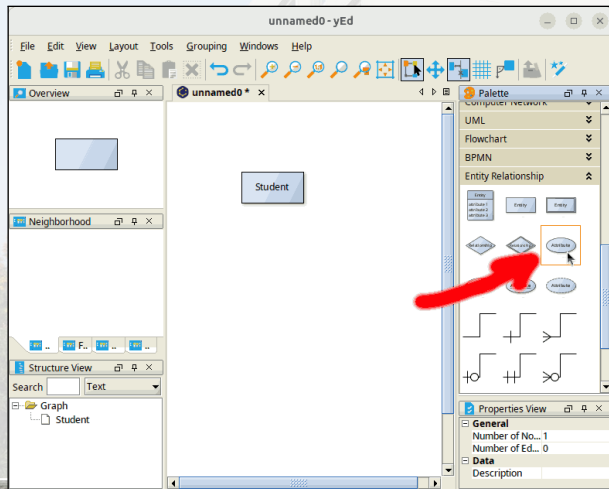
- Der Name hat sich geändert.



Student ERD mit yEd Zeichnen

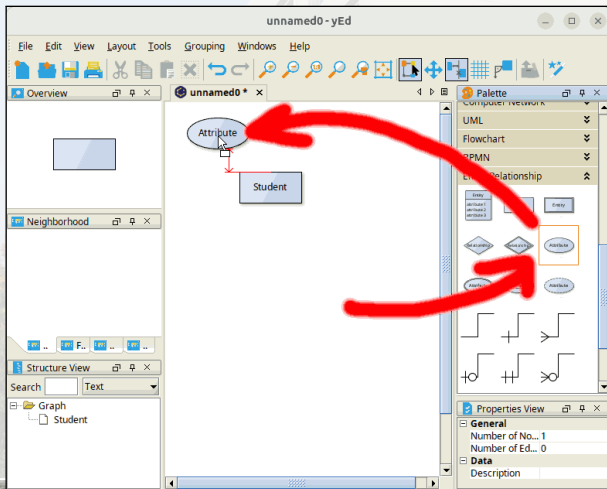


- Der Name hat sich geändert.
- Wir klicken nun auf das **Attribute**-Symbol in der Palette.



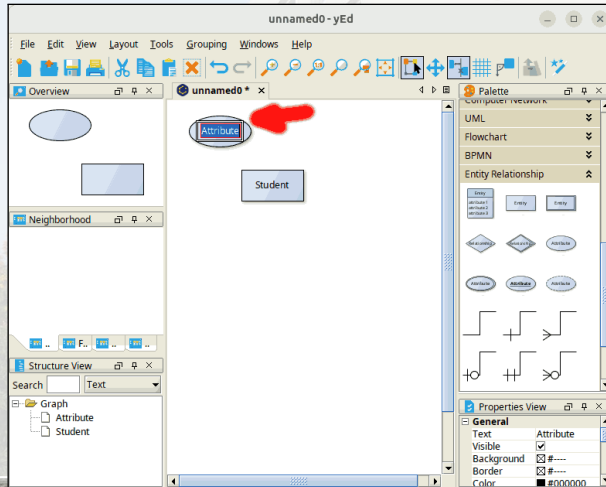
Student ERD mit yEd Zeichnen

- Wir ziehen das Attributsymbol in unser Dokument.



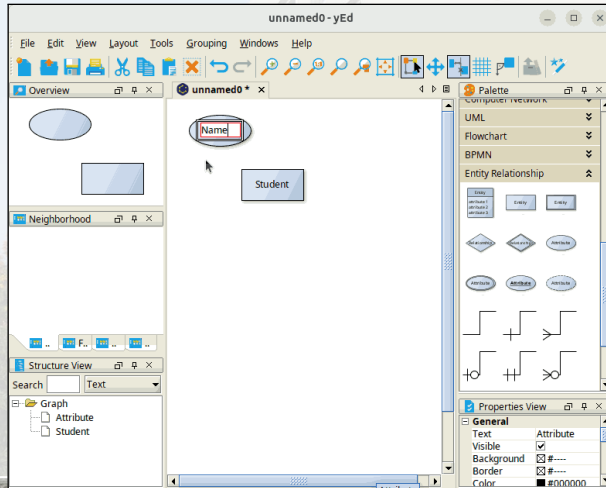
Student ERD mit yEd Zeichnen

- Wir doppel-klicken in es um seinen Namen zu ändern.



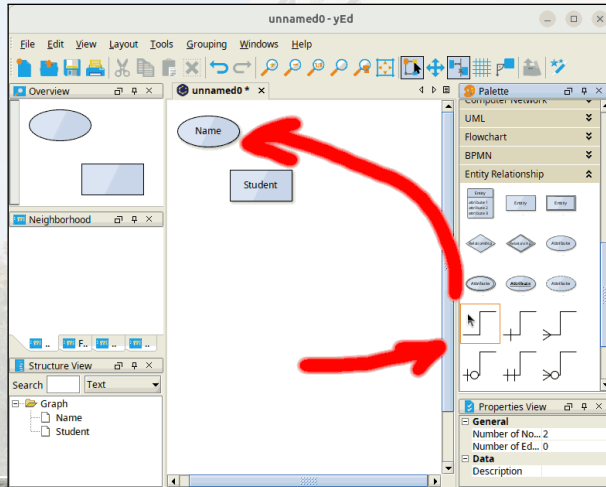
Student ERD mit yEd Zeichnen

- Wir ändern den Name zu, nun ja, „Name“.



Student ERD mit yEd Zeichnen

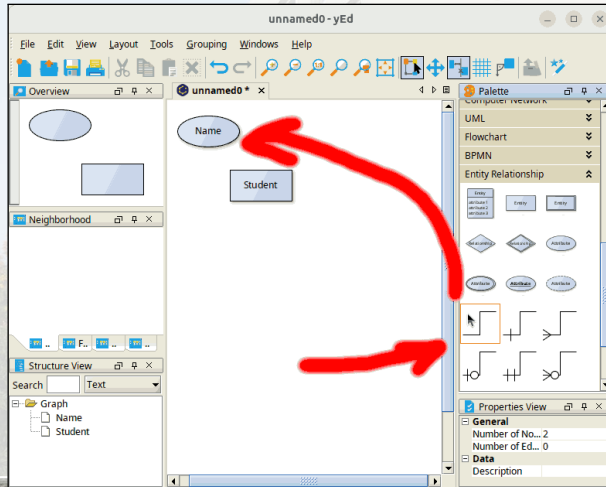
- Wir haben den Attributname geändert.



Student ERD mit yEd Zeichnen

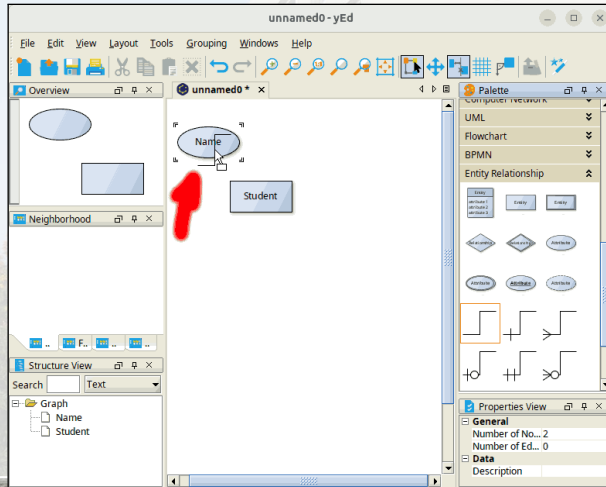


- Wir haben den Attributname geändert.
- Jetzt klicken wir auf das Verbindungs-Symbol in der Palette und ziehen es auf das Attribut.



Student ERD mit yEd Zeichnen

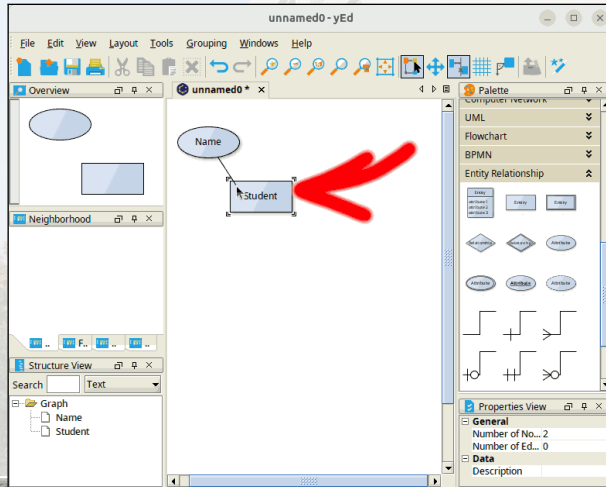
- Wir lassen das Verbindungssymbol auf dem Attribut *Name* fallen.



Student ERD mit yEd Zeichnen

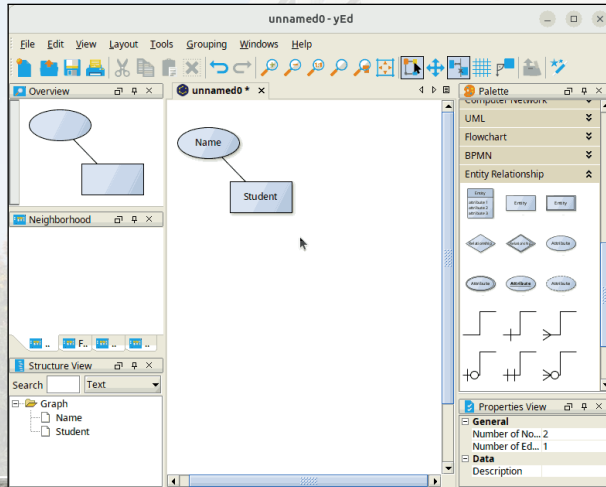


- Dann klicken wir auf den Entitätstyp und verbinden das Attribut *Name* mit dem Entitätstyp *Student*.



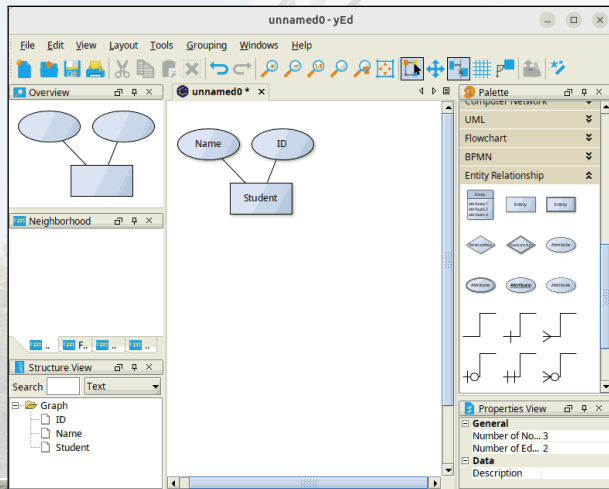
Student ERD mit yEd Zeichnen

- Das Attribut *Name* ist nun mit dem Entitätstyp *Student* verbunden.



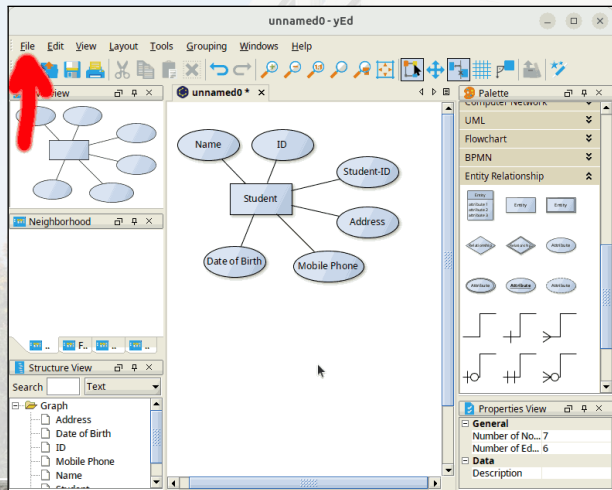
Student ERD mit yEd Zeichnen

- Wir fügen das Attribut *ID* auf genau die gleiche Art hinzu.



Student ERD mit yEd Zeichnen

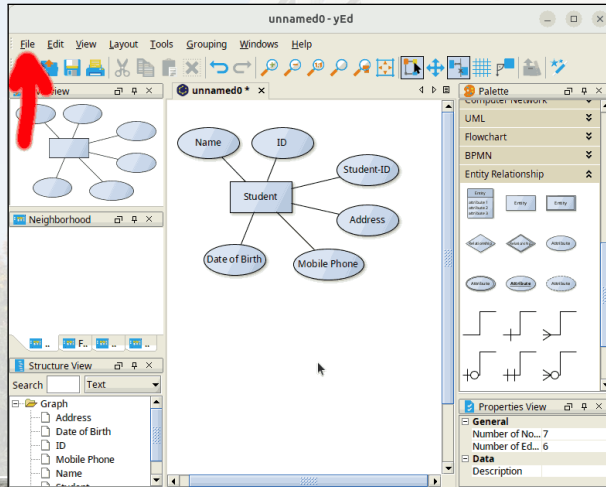
- Wir fügen noch mehrere weitere Attribute hinzu.



Student ERD mit yEd Zeichnen

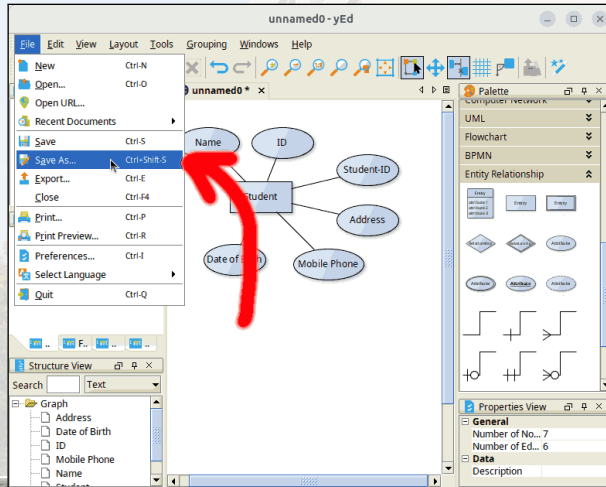


- Wir fügen noch mehrere weitere Attribute hinzu.
- Danach klicken wir auf das Menü **File**.



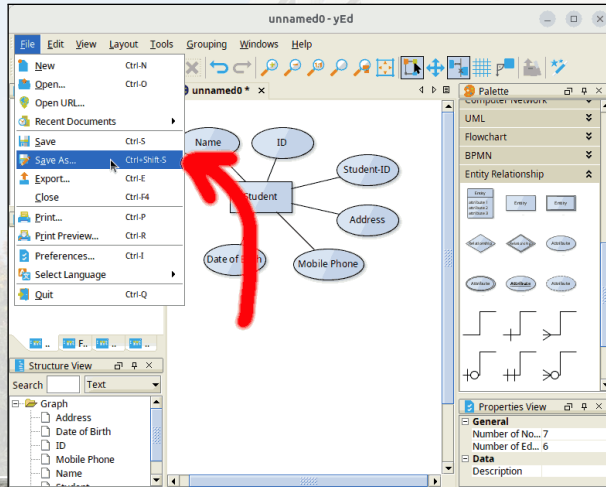
Student ERD mit yEd Zeichnen

- Wir wollen das Dokument nun speichern.



Student ERD mit yEd Zeichnen

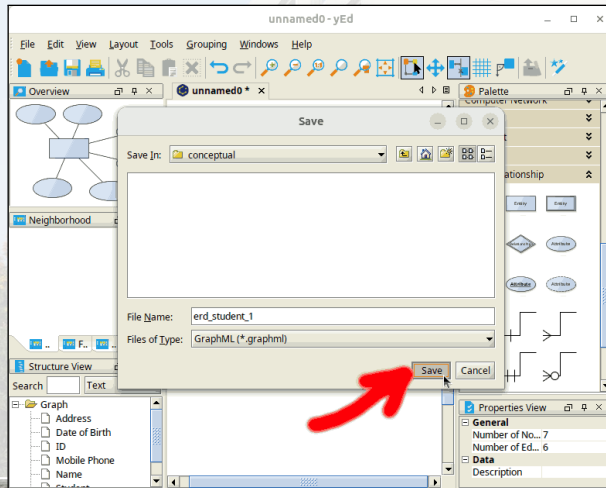
- Wir wollen das Dokument nun speichern.
- Wir klicken auf **Save As**.



Student ERD mit yEd Zeichnen



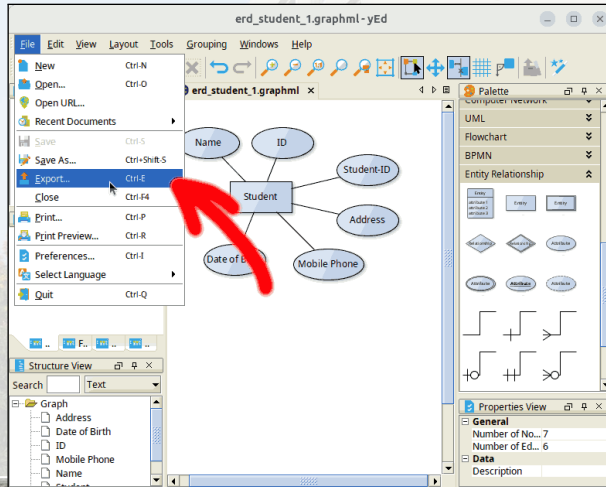
- Wir speichern das Diagramm unter dem Name `erd_student_1` im `graphml`-Format, in dem wir diesen Namen eingeben und dann auf `Save` klicken.



Student ERD mit yEd Zeichnen

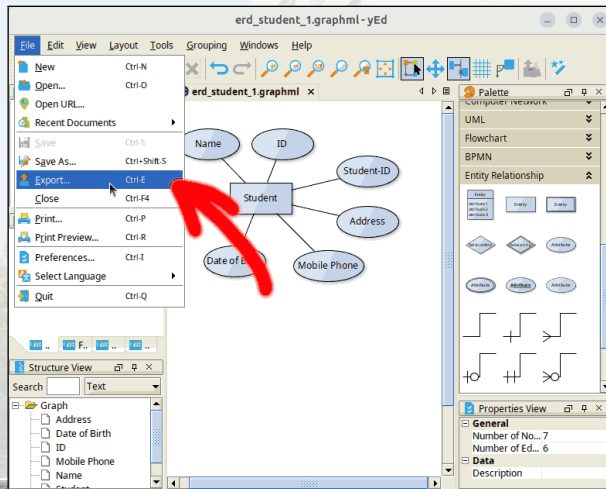


- Wir können das Diagramm auch in ein Format exportieren, dass wir in anderen Dokumenten verwenden können.



Student ERD mit yEd Zeichnen

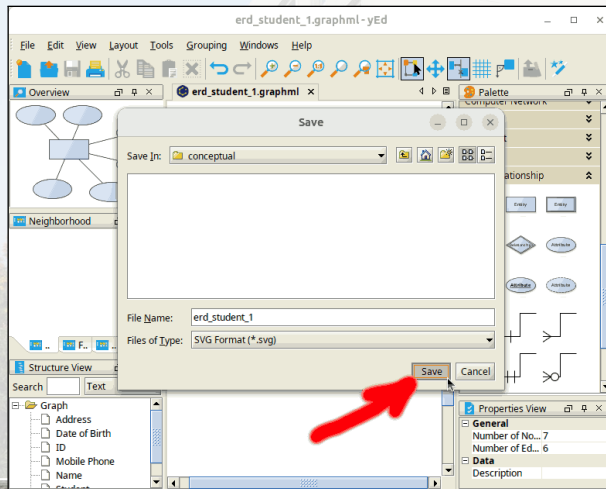
- Dafür klicken wir auf **Export** im Menü **File**.



Student ERD mit yEd Zeichnen

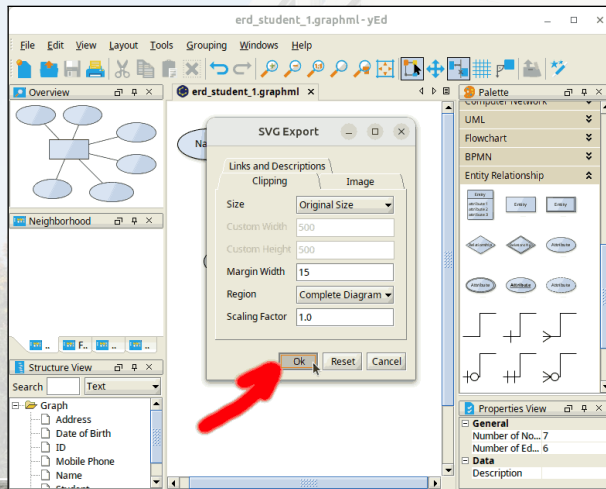


- Wir wählen das SVG-Format und klicken **Save**.



Student ERD mit yEd Zeichnen

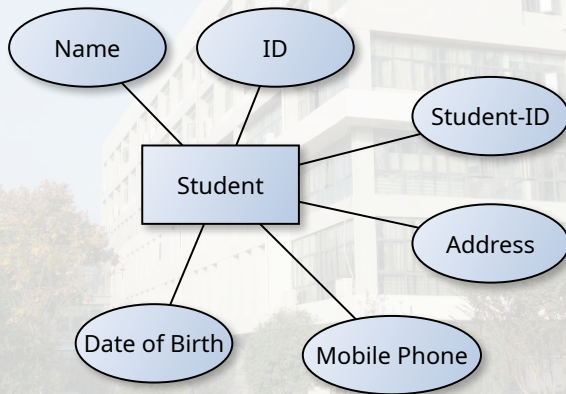
- Wir lassen alle Einstellungen so wie sie sind und klicken **OK**.



Student ERD mit yEd Zeichnen



- Das Ergebnis sieht genau so aus.





Mehr Attribute



Feedback



- Wir nehmen also unser ERD und diskutieren es mit unseren Partnern in der Universität.

Feedback



- Wir nehmen also unser ERD und diskutieren es mit unseren Partnern in der Universität.
- Wir wollen prüfen, ob das Model sinnvoll ist.

Feedback



- Wir nehmen also unser ERD und diskutieren es mit unseren Partnern in der Universität.
- Wir wollen prüfen, ob das Model sinnvoll ist.
- Dabei entdecken wir ein paar Probleme. . .

Zusammengesetzte Attribute

- Namen sind komplizierter als wir dachten.



Zusammengesetzte Attribute



- Namen sind komplizierter als wir dachten.
- Studenten haben einen Vorname, einen Familienname, und eine Anrede.



Zusammengesetzte Attribute



- Namen sind komplizierter als wir dachten.
- Studenten haben einen Vorname, einen Familienname, und eine Anrede.
- Zum Beispieler ist der Name von „Herr Bebbo“ eigentlich „Herr Fred Bebbo“.



Zusammengesetzte Attribute



- Namen sind komplizierter als wir dachten.
- Studenten haben einen Vorname, einen Familienname, und eine Anrede.
- Zum Beispiels ist der Name von „Herr Bebbo“ eigentlich „Herr Fred Bebbo“, wobei „Bebbo“ der Familienname und „Fred“ der Vorname sind.
- Um dem Rechnung zu tragen, sollte das *Name*-Attribut kein einzelnes Attribut sein.



Zusammengesetzte Attribute



- Namen sind komplizierter als wir dachten.
- Studenten haben einen Vorname, einen Familienname, und eine Anrede.
- Zum Beispiele ist der Name von „Herr Bebbo“ eigentlich „Herr Fred Bebbo“, wobei „Bebbo“ der Familienname und „Fred“ der Vorname sind.
- Um dem Rechnung zu tragen, sollte das *Name*-Attribut kein einzelnes Attribut sein.
- Wir könnten es in ein *zusammengesetztes Attribut* umbauen, dass aus einem Vorname (EN: *given name*), Familienname (EN: *family name*) und einer Anrede (EN: *salutation*) besteht.

Zusammengesetzte Attribute



- Namen sind komplizierter als wir dachten.
- Studenten haben einen Vorname, einen Familienname, und eine Anrede.
- Zum Beispiele ist der Name von „Herr Bebbo“ eigentlich „Herr Fred Bebbo“, wobei „Bebbo“ der Familienname und „Fred“ der Vorname sind.
- Um dem Rechnung zu tragen, sollte das *Name*-Attribut kein einzelnes Attribut sein.
- Wir könnten es in ein *zusammengesetztes Attribut* umbauen, dass aus einem Vorname (EN: *given name*), Familienname (EN: *family name*) und einer Anrede (EN: *salutation*) besteht.
- Das System kann diese dann vernünftig zusammensetzen, wenn Studenten angesprochen oder Dokumente ausgestellt werden sollen.

Zusammengesetzte Attribute



- Namen sind komplizierter als wir dachten.
- Studenten haben einen Vorname, einen Familienname, und eine Anrede.
- Zum Beispiele ist der Name von „Herr Bebbo“ eigentlich „Herr Fred Bebbo“, wobei „Bebbo“ der Familienname und „Fred“ der Vorname sind.
- Um dem Rechnung zu tragen, sollte das *Name*-Attribut kein einzelnes Attribut sein.
- Wir könnten es in ein *zusammengesetztes Attribut* umbauen, dass aus einem Vorname (EN: *given name*), Familienname (EN: *family name*) und einer Anrede (EN: *salutation*) besteht.
- Das System kann diese dann vernünftig zusammensetzen, wenn Studenten angesprochen oder Dokumente ausgestellt werden sollen.

Definition: Zusammengesetztes Attribut

Ein *zusammengesetztes Attribut* (EN: *composite attribute*) ist ein Attribut, das aus mehreren Teilen besteht, die eigene Namen und Domänen haben.

Zusammengesetzte Attribute



- Um dem Rechnung zu tragen, sollte das *Name*-Attribut kein einzelnes Attribut sein.
- Wir könnten es in ein *zusammengesetztes Attribut* umbauen, dass aus einem Vorname (EN: *given name*), Familienname (EN: *family name*) und einer Anrede (EN: *salutation*) besteht.
- Das System kann diese dann vernünftig zusammensetzen, wenn Studenten angesprochen oder Dokumente ausgestellt werden sollen.

Definition: Zusammengesetztes Attribut

Ein *zusammengesetztes Attribut* (EN: *composite attribute*) ist ein Attribut, das aus mehreren Teilen besteht, die eigene Namen und Domänen haben.

Definition: Einfaches Attribut

Ein *einfaches Attribut* (EN: *simple attribute*) ist ein Attribut das nur einen einzelnen Namen und eine Domäne hat.

Zusammengesetzte Attribute



- Um dem Rechnung zu tragen, sollte das *Name*-Attribut kein einzelnes Attribut sein.
- Wir könnten es in ein *zusammengesetztes Attribut* umbauen, dass aus einem Vorname (EN: *given name*), Familienname (EN: *family name*) und einer Anrede (EN: *salutation*) besteht.
- Das System kann diese dann vernünftig zusammensetzen, wenn Studenten angesprochen oder Dokumente ausgestellt werden sollen.

Definition: Zusammengesetztes Attribut

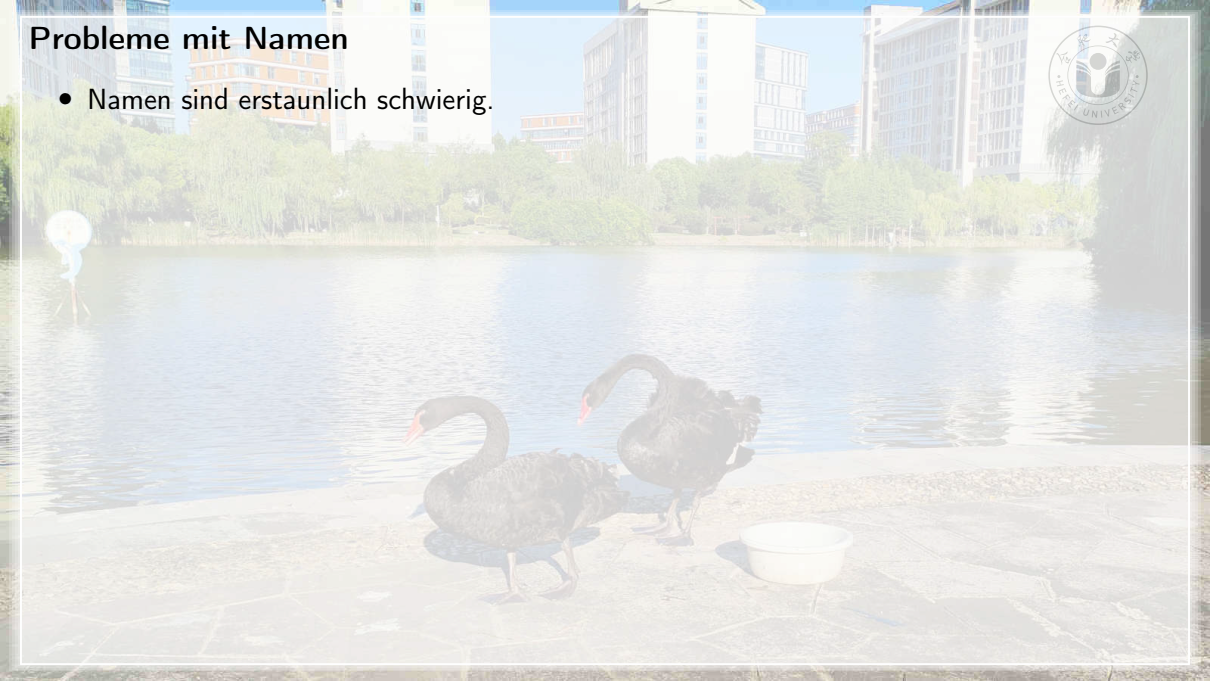
Ein *zusammengesetztes Attribut* (EN: *composite attribute*) ist ein Attribut, das aus mehreren Teilen besteht, die eigene Namen und Domänen haben.

Definition: Einfaches Attribut

Ein *einfaches Attribut* (EN: *simple attribute*) ist ein Attribut das nur einen einzelnen Namen und eine Domäne hat. Einfache Attribute haben keine Komponenten.

Probleme mit Namen

- Namen sind erstaunlich schwierig.



Probleme mit Namen

- Namen sind erstaunlich schwierig.
- In westlichen Kulturen wird der volle Name oft zusammengesetzt, in dem wir erst den Vorname und dann den Familiennamen schreiben.



Probleme mit Namen

- Namen sind erstaunlich schwierig.
- In westlichen Kulturen wird der volle Name oft zusammengesetzt, in dem wir erst den Vorname und dann den Familiennamen schreiben.
- Deshalb ist der volle Name von Herrn Bebbo nämlich Fred Bebbo.



Probleme mit Namen



- Namen sind erstaunlich schwierig.
- In westlichen Kulturen wird der volle Name oft zusammengesetzt, in dem wir erst den Vorname und dann den Familiennamen schreiben.
- Deshalb ist der volle Name von Herrn Bebbo nämlich Fred Bebbo.
- In östlichen Kulturen ist es oft andersherum.



Probleme mit Namen



- Namen sind erstaunlich schwierig.
- In westlichen Kulturen wird der volle Name oft zusammengesetzt, in dem wir erst den Vorname und dann den Familiennamen schreiben.
- Deshalb ist der volle Name von Herrn Bebbo nämlich Fred Bebbo.
- In östlichen Kulturen ist es oft andersherum:
- Der Name des berühmten chinesischen Mathematikers 刘徽 wird als LIU Hui im lateinischen Alphabet transkribiert^{66,91,107}.



Probleme mit Namen



- Namen sind erstaunlich schwierig.
- In westlichen Kulturen wird der volle Name oft zusammengesetzt, in dem wir erst den Vorname und dann den Familiennamen schreiben.
- Deshalb ist der volle Name von Herrn Bebbo nämlich Fred Bebbo.
- In östlichen Kulturen ist es oft andersherum:
- Der Name des berühmten chinesischen Mathematikers 刘徽 wird als LIU Hui im lateinischen Alphabet transkribiert^{66,91,107}.
- Dabei ist LIU (刘) der Familienname und kommt zuerst und der Vorname Hui (徽) kommt danach.

Probleme mit Namen



- Namen sind erstaunlich schwierig.
- In westlichen Kulturen wird der volle Name oft zusammengesetzt, in dem wir erst den Vorname und dann den Familiennamen schreiben.
- Deshalb ist der volle Name von Herrn Bebbo nämlich Fred Bebbo.
- In östlichen Kulturen ist es oft andersherum:
- Der Name des berühmten chinesischen Mathematikers 刘徽 wird als LIU Hui im lateinischen Alphabet transkribiert^{66,91,107}.
- Dabei ist LIU (刘) der Familienname und kommt zuerst und der Vorname Hui (徽) kommt danach.
- Abhängig vom kulturellen Hintergrund könnte das System also den Namen korrekt zusammensetzen.

Probleme mit Namen



- Namen sind erstaunlich schwierig.
- In westlichen Kulturen wird der volle Name oft zusammengesetzt, in dem wir erst den Vorname und dann den Familiennamen schreiben.
- Deshalb ist der volle Name von Herrn Bebbo nämlich Fred Bebbo.
- In östlichen Kulturen ist es oft andersherum:
- Der Name des berühmten chinesischen Mathematikers 刘徽 wird als LIU Hui im lateinischen Alphabet transkribiert^{66,91,107}.
- Dabei ist LIU (刘) der Familienname und kommt zuerst und der Vorname Hui (徽) kommt danach.
- Abhängig vom kulturellen Hintergrund könnte das System also den Namen korrekt zusammensetzen.
- Das klingt aber nach einem grässlichen Problem und nach extrem fehleranfälligen Code.

Probleme mit Namen



- Namen sind erstaunlich schwierig.
- In westlichen Kulturen wird der volle Name oft zusammengesetzt, in dem wir erst den Vorname und dann den Familiennamen schreiben.
- Deshalb ist der volle Name von Herrn Bebbo nämlich Fred Bebbo.
- In östlichen Kulturen ist es oft andersherum:
- Der Name des berühmten chinesischen Mathematikers 刘徽 wird als LIU Hui im lateinischen Alphabet transkribiert^{66,91,107}.
- Dabei ist LIU (刘) der Familienname und kommt zuerst und der Vorname Hui (徽) kommt danach.
- Abhängig vom kulturellen Hintergrund könnte das System also den Namen korrekt zusammensetzen.
- Das klingt aber nach einem grässlichen Problem und nach extrem fehleranfälliger Kode.
- Wir entscheiden uns also dagegen, den Namen aus Vor- und Nachname zusammenzusetzen.

Probleme mit Namen



- Namen sind erstaunlich schwierig.
- In westlichen Kulturen wird der volle Name oft zusammengesetzt, in dem wir erst den Vorname und dann den Familiennamen schreiben.
- Deshalb ist der volle Name von Herrn Bebbo nämlich Fred Bebbo.
- In östlichen Kulturen ist es oft andersherum:
- Der Name des berühmten chinesischen Mathematikers 刘徽 wird als LIU Hui im lateinischen Alphabet transkribiert^{66,91,107}.
- Dabei ist LIU (刘) der Familienname und kommt zuerst und der Vorname Hui (徽) kommt danach.
- Abhängig vom kulturellen Hintergrund könnte das System also den Namen korrekt zusammensetzen.
- Das klingt aber nach einem grässlichen Problem und nach extrem fehleranfälliger Code.
- Wir entscheiden uns also dagegen, den Namen aus Vor- und Nachname zusammenzusetzen.
- Stattdessen benutzen wir zwei andere Komponenten des Namensattributes.

Probleme mit Namen



- In westlichen Kulturen wird der volle Name oft zusammengesetzt, in dem wir erst den Vorname und dann den Familiennamen schreiben.
- Deshalb ist der volle Name von Herrn Bebbo nämlich Fred Bebbo.
- In östlichen Kulturen ist es oft andersherum:
- Der Name des berühmten chinesischen Mathematikers 刘徽 wird als LIU Hui im lateinischen Alphabet transkribiert^{66,91,107}.
- Dabei ist LIU (刘) der Familienname und kommt zuerst und der Vorname Hui (徽) kommt danach.
- Abhängig vom kulturellen Hintergrund könnte das System also den Namen korrekt zusammensetzen.
- Das klingt aber nach einem grässlichen Problem und nach extrem fehleranfälligem Kode.
- Wir entscheiden uns also dagegen, den Namen aus Vor- und Nachname zusammenzusetzen.
- Stattdessen benutzen wir zwei andere Komponenten des Namensattributes:
 1. Der volle Name (EN: *Full Name*) ist der komplette offizielle Name der Person, genauso geschrieben wie im Ausweis.

Probleme mit Namen



- Deshalb ist der volle Name von Herrn Bebbo nämlich Fred Bebbo.
- In östlichen Kulturen ist es oft andersherum:
- Der Name des berühmten chinesischen Mathematikers 刘徽 wird als LIU Hui im lateinischen Alphabet transkribiert^{66,91,107}.
- Dabei ist LIU (刘) der Familienname und kommt zuerst und der Vorname Hui (徽) kommt danach.
- Abhängig vom kulturellen Hintergrund könnte das System also den Namen korrekt zusammensetzen.
- Das klingt aber nach einem grässlichen Problem und nach extrem fehleranfälligem Kode.
- Wir entscheiden uns also dagegen, den Namen aus Vor- und Nachname zusammenzusetzen.
- Stattdessen benutzen wir zwei andere Komponenten des Namensattributes:
 1. Der volle Name (EN: *Full Name*) ist der komplette offizielle Name der Person, genauso geschrieben wie im Ausweis. Das wäre so etwas wie 刘徽 für eine chinesische Person und Fred Bebbo für den Westerler Herr Bebbo.

Probleme mit Namen



- In östlichen Kulturen ist es oft andersherum:
- Der Name des berühmten chinesischen Mathematikers 刘徽 wird als LIU Hui im lateinischen Alphabet transkribiert^{66,91,107}.
- Dabei ist LIU (刘) der Familienname und kommt zuerst und der Vorname Hui (徽) kommt danach.
- Abhängig vom kulturellen Hintergrund könnte das System also den Namen korrekt zusammensetzen.
- Das klingt aber nach einem grässlichen Problem und nach extrem fehleranfälligem Kode.
- Wir entscheiden uns also dagegen, den Namen aus Vor- und Nachname zusammenzusetzen.
- Stattdessen benutzen wir zwei andere Komponenten des Namensattributes:
 1. Der volle Name (EN: *Full Name*) ist der komplette offizielle Name der Person, genauso geschrieben wie im Ausweis. Das wäre so etwas wie 刘徽 für eine chinesische Person und Fred Bebbö für den Westerler Herr Bebbö.
 2. Als zweite Komponente speichern wir die volle Anrede (EN: *Salutation*).

Probleme mit Namen



- Der Name des berühmten chinesischen Mathematikers 刘徽 wird als LIU Hui im lateinischen Alphabet transkribiert ^{66,91,107}.
- Dabei ist LIU (刘) der Familienname und kommt zuerst und der Vorname Hui (徽) kommt danach.
- Abhängig vom kulturellen Hintergrund könnte das System also den Namen korrekt zusammensetzen.
- Das klingt aber nach einem grässlichen Problem und nach extrem fehleranfälligem Kode.
- Wir entscheiden uns also dagegen, den Namen aus Vor- und Nachname zusammenzusetzen.
- Stattdessen benutzen wir zwei andere Komponenten des Namensattributes:
 1. Der volle Name (EN: *Full Name*) ist der komplette offizielle Name der Person, genauso geschrieben wie im Ausweis. Das wäre so etwas wie 刘徽 für eine chinesische Person und Fred Bebbo für den Westerler Herr Bebbo.
 2. Als zweite Komponente speichern wir die volle Anrede (EN: *Salutation*). Für 刘徽 könnte das 刘教授 sein und für Fred Bebbo vielleicht Herr Bebbo.

Probleme mit Namen



- Dabei ist LIU (刘) der Familienname und kommt zuerst und der Vorname Hui (徽) kommt danach.
- Abhängig vom kulturellen Hintergrund könnte das System also den Namen korrekt zusammensetzen.
- Das klingt aber nach einem grässlichen Problem und nach extrem fehleranfälligem Kode.
- Wir entscheiden uns also dagegen, den Namen aus Vor- und Nachname zusammenzusetzen.
- Stattdessen benutzen wir zwei andere Komponenten des Namensattributes:
 1. Der volle Name (EN: *Full Name*) ist der komplette offizielle Name der Person, genauso geschrieben wie im Ausweis. Das wäre so etwas wie 刘徽 für eine chinesische Person und Fred Bebbo für den Westerler Herr Bebbo.
 2. Als zweite Komponente speichern wir die volle Anrede (EN: *Salutation*). Für 刘徽 könnte das 刘教授 sein und für Fred Bebbo vielleicht Herr Bebbo.
- Vielleicht erlauben wir den Studenten später, die *Salutation* selbst auszuwählen.

Probleme mit Namen



- Abhängig vom kulturellen Hintergrund könnte das System also den Namen korrekt zusammensetzen.
- Das klingt aber nach einem grässlichen Problem und nach extrem fehleranfälligem Kode.
- Wir entscheiden uns also dagegen, den Namen aus Vor- und Nachname zusammenzusetzen.
- Stattdessen benutzen wir zwei andere Komponenten des Namensattributes:
 1. Der volle Name (EN: *Full Name*) ist der komplette offizielle Name der Person, genauso geschrieben wie im Ausweis. Das wäre so etwas wie 刘徽 für eine chinesische Person und Fred Bebbo für den Westerler Herr Bebbo.
 2. Als zweite Komponente speichern wir die volle Anrede (EN: *Salutation*). Für 刘徽 könnte das 刘教授 sein und für Fred Bebbo vielleicht Herr Bebbo.
- Vielleicht erlauben wir den Studenten später, die *Salutation* selbst auszuwählen.
- Der *Full Name* wird vom Administrator genauso eingegeben, wie er auf dem Ausweis steht.

Probleme mit Namen



- Das klingt aber nach einem grässlichen Problem und nach extrem fehleranfälligem Kode.
- Wir entscheiden uns also dagegen, den Namen aus Vor- und Nachname zusammenzusetzen.
- Stattdessen benutzen wir zwei andere Komponenten des Namensattributes:
 1. Der volle Name (EN: *Full Name*) ist der komplette offizielle Name der Person, genauso geschrieben wie im Ausweis. Das wäre so etwas wie 刘徽 für eine chinesische Person und Fred Bebbo für den Westerler Herr Bebbo.
 2. Als zweite Komponente speichern wir die volle Anrede (EN: *Salutation*). Für 刘徽 könnte das 刘教授 sein und für Fred Bebbo vielleicht Herr Bebbo.
- Vielleicht erlauben wir den Studenten später, die *Salutation* selbst auszuwählen.
- Der *Full Name* wird vom Administrator genauso eingegeben, wie er auf dem Ausweis steht.
- Unser System würde die *Salutation* verwenden, wenn es Nachrichten an Herrn Bebbo schickt.

Probleme mit Namen



- Wir entscheiden uns also dagegen, den Namen aus Vor- und Nachname zusammenzusetzen.
- Stattdessen benutzen wir zwei andere Komponenten des Namensattributes:
 1. Der volle Name (*EN: Full Name*) ist der komplette offizielle Name der Person, genauso geschrieben wie im Ausweis. Das wäre so etwas wie 刘徽 für eine chinesische Person und Fred Bebbo für den Westerler Herr Bebbo.
 2. Als zweite Komponente speichern wir die volle Anrede (*EN: Salutation*). Für 刘徽 könnte das 刘教授 sein und für Fred Bebbo vielleicht Herr Bebbo.
- Vielleicht erlauben wir den Studenten später, die *Salutation* selbst auszuwählen.
- Der *Full Name* wird vom Administrator genauso eingegeben, wie er auf dem Ausweis steht.
- Unser System würde die *Salutation* verwenden, wenn es Nachrichten an Herrn Bebbo schickt.
- Auf Zertifikaten wird stattdessen der *Full Name* verwendet.

Probleme mit Namen

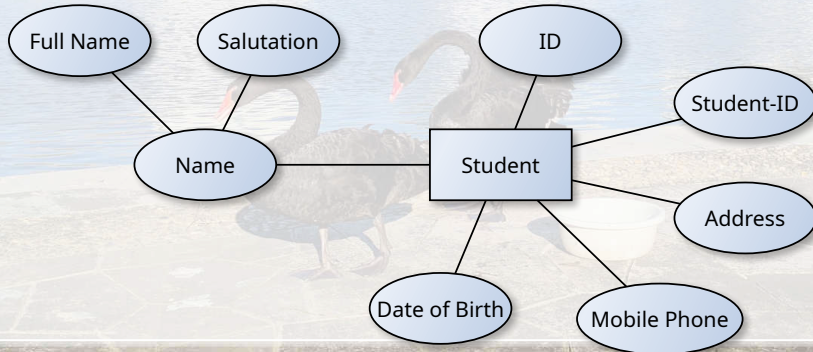


- Stattdessen benutzen wir zwei andere Komponenten des Namensattributes:
 1. Der volle Name (EN: *Full Name*) ist der komplette offizielle Name der Person, genauso geschrieben wie im Ausweis. Das wäre so etwas wie 刘徽 für eine chinesische Person und Fred Bebbio für den Westerler Herr Bebbio.
 2. Als zweite Komponente speichern wir die volle Anrede (EN: *Salutation*). Für 刘徽 könnte das 刘教授 sein und für Fred Bebbio vielleicht Herr Bebbio.
- Vielleicht erlauben wir den Studenten später, die *Salutation* selbst auszuwählen.
- Der *Full Name* wird vom Administrator genauso eingegeben, wie er auf dem Ausweis steht.
- Unser System würde die *Salutation* verwenden, wenn es Nachrichten an Herrn Bebbio schickt.
- Auf Zertifikaten wird stattdessen der *Full Name* verwendet.
- Ein Zusammengesetztes Attribut wird in ERDs als ein Attribut gemalt, dass mit anderen Attributen verbunden ist.

Probleme mit Namen



- Unser System würde die *Salutation* verwenden, wenn es Nachrichten an Herrn Bebboschickt.
- Auf Zertifikaten wird stattdessen der *Full Name* verwendet.
- Ein Zusammengesetztes Attribut wird in ERDs als ein Attribut gemalt, dass mit anderen Attributen verbunden ist.



Mehrwertige und Optionale Attribute



Definition: Mehrwertiges Attribut

Eine Entität kann mehrere Werte für eine mehrwertiges (EN: *multivalued*) Attribut haben.



Mehrwertige und Optionale Attribute



Definition: Mehrwertiges Attribut

Eine Entität kann mehrere Werte für ein mehrwertiges (EN: *multivalued*) Attribut haben.

Definition: Einwertiges Attribut

Eine Entität hat (höchstens) einen Wert für ein einwertiges (EN: *single-valued*) Attribut.

Mehrwertige und Optionale Attribute



Definition: Mehrwertiges Attribut

Eine Entität kann mehrere Werte für ein mehrwertiges (EN: *multivalued*) Attribut haben.

Definition: Einwertiges Attribut

Eine Entität hat (höchstens) einen Wert für ein einwertiges (EN: *single-valued*) Attribut.

Definition: Optionales Attribut

Eine Entität kann einen Wert für ein optionales (EN: *optional*) Attribut haben, oder auch nicht.

Mehrwertige und Optionale Attribute



Definition: Mehrwertiges Attribut

Eine Entität kann mehrere Werte für ein mehrwertiges (EN: *multivalued*) Attribut haben.

Definition: Einwertiges Attribut

Eine Entität hat (höchstens) einen Wert für ein einwertiges (EN: *single-valued*) Attribut.

Definition: Optionales Attribut

Eine Entität kann einen Wert für ein optionales (EN: *optional*) Attribut haben, oder auch nicht. Wenn ein Attribut keinen Wert hat, dann wird das oft durch **NULL** dargestellt.

Mehrwertige und Optionale Attribute



Definition: Mehrwertiges Attribut

Eine Entität kann mehrere Werte für ein mehrwertiges (EN: *multivalued*) Attribut haben.

Definition: Einwertiges Attribut

Eine Entität hat (höchstens) einen Wert für ein einwertiges (EN: *single-valued*) Attribut.

Definition: Optionales Attribut

Eine Entität kann einen Wert für ein optionales (EN: *optional*) Attribut haben, oder auch nicht. Wenn ein Attribut keinen Wert hat, dann wird das oft durch **NULL** dargestellt.

- Ein optionales einwertiges Attribut hat entweder einen Wert oder keinen Wert.

Mehrwertige und Optionale Attribute



Definition: Einwertiges Attribut

Eine Entität hat (höchstens) einen Wert für ein einwertiges (EN: *single-valued*) Attribut.

Definition: Optionales Attribut

Eine Entität kann einen Wert für ein optionales (EN: *optional*) Attribut haben, oder auch nicht. Wenn ein Attribut keinen Wert hat, dann wird das oft durch **NULL** dargestellt.

- Ein optionales einwertiges Attribut hat entweder einen Wert oder keinen Wert.
- Ein nicht-optionales einwertiges Attribut hat immer genau einen Wert.

Mehrwertige und Optionale Attribute



Definition: Einwertiges Attribut

Eine Entität hat (höchstens) einen Wert für ein einwertiges (EN: *single-valued*) Attribut.

Definition: Optionales Attribut

Eine Entität kann einen Wert für ein optionales (EN: *optional*) Attribut haben, oder auch nicht. Wenn ein Attribut keinen Wert hat, dann wird das oft durch **NULL** dargestellt.

- Ein optionales einwertiges Attribut hat entweder einen Wert oder keinen Wert.
- Ein nicht-optionales einwertiges Attribut hat immer genau einen Wert.
- Ein optionales mehrwertiges Attribut hat entweder keinen Wert, einen Wert, oder mehrere Werte.

Mehrwertige und Optionale Attribute



Definition: Einwertiges Attribut

Eine Entität hat (höchstens) einen Wert für ein einwertiges (EN: *single-valued*) Attribut.

Definition: Optionales Attribut

Eine Entität kann einen Wert für ein optionales (EN: *optional*) Attribut haben, oder auch nicht. Wenn ein Attribut keinen Wert hat, dann wird das oft durch **NULL** dargestellt.

- Ein optionales einwertiges Attribut hat entweder einen Wert oder keinen Wert.
- Ein nicht-optionales einwertiges Attribut hat immer genau einen Wert.
- Ein optionales mehrwertiges Attribut hat entweder keinen Wert, einen Wert, oder mehrere Werte.
- Ein nicht-optionales mehrwertiges Attribut hat entweder einen Wert oder mehrere Werte.

Mehrwertige und Optionale Attribute



Definition: Einwertiges Attribut

Eine Entität hat (höchstens) einen Wert für ein einwertiges (EN: *single-valued*) Attribut.

Definition: Optionales Attribut

Eine Entität kann einen Wert für ein optionales (EN: *optional*) Attribut haben, oder auch nicht. Wenn ein Attribut keinen Wert hat, dann wird das oft durch **NULL** dargestellt.

- Ein optionales einwertiges Attribut hat entweder einen Wert oder keinen Wert.
- Ein nicht-optionales einwertiges Attribut hat immer genau einen Wert.
- Ein optionales mehrwertiges Attribut hat entweder keinen Wert, einen Wert, oder mehrere Werte.
- Ein nicht-optionales mehrwertiges Attribut hat entweder einen Wert oder mehrere Werte.
- Ein mehrwertiges Attribut wird in ERDs durch zwei ineinanderliegende Ellipsen dargestellt.



Definition: Optionales Attribut

Eine Entität kann einen Wert für ein optionales (EN: *optional*) Attribut haben, oder auch nicht. Wenn ein Attribut keinen Wert hat, dann wird das oft durch **NULL** dargestellt.

- Ein optionales einwertiges Attribut hat entweder einen Wert oder keinen Wert.
- Ein nicht-optionales einwertiges Attribut hat immer genau einen Wert.
- Ein optionales mehrwertiges Attribut hat entweder keinen Wert, einen Wert, oder mehrere Werte.
- Ein nicht-optionales mehrwertiges Attribut hat entweder einen Wert oder mehrere Werte.
- Ein mehrwertiges Attribut wird in ERDs durch zwei ineinanderliegende Ellipsen dargestellt.
- Optionale Attribute werden durch kleine Kreise am Ende der Linie die sie mit den Entitätstypen verbinden dargestellt.

Mobiltelefonnummern



- Wir stellen fest, dass ein Student mehrere Mobiltelefonnummern haben kann.

Mobiltelefonnummern



- Wir stellen fest, dass ein Student mehrere Mobiltelefonnummern haben kann.
- Das kann mit einem mehrwertigen Attribut modelliert werden.

Mobiltelefonnummern



- Wir stellen fest, dass ein Student mehrere Mobiltelefonnummern haben kann.
- Das kann mit einem mehrwertigen Attribut modelliert werden.
- Bei genauerer Betrachtung stellen wir fest, dass das Mobiltelefonnummer-Attribut am besten als optionales mehrwertiges Attribut modelliert werden sollten.

Mobiltelefonnummern



- Wir stellen fest, dass ein Student mehrere Mobiltelefonnummern haben kann.
- Das kann mit einem mehrwertigen Attribut modelliert werden.
- Bei genauerer Betrachtung stellen wir fest, dass das Mobiltelefonnummer-Attribut am besten als optionales mehrwertiges Attribut modelliert werden sollten.
- Es ist heutzutage zwar eher unwahrscheinlich, dass ein Student keine Mobiltelefonnummer hat, aber es könnte sein.

Mobiltelefonnummern



- Wir stellen fest, dass ein Student mehrere Mobiltelefonnummern haben kann.
- Das kann mit einem mehrwertigen Attribut modelliert werden.
- Bei genauerer Betrachtung stellen wir fest, dass das Mobiltelefonnummer-Attribut am besten als optionales mehrwertiges Attribut modelliert werden sollten.
- Es ist heutzutage zwar eher unwahrscheinlich, dass ein Student keine Mobiltelefonnummer hat, aber es könnte sein.
- Vielleicht trifft das ja auf ausländische Austauschstudenten (留学生) zu, die gerade erst nach China gekommen sind.

Ausweisnummern



- Und wo wir gerade bei dem Thema Austauschstudenten sind...

Ausweisnummern



- Und wo wir gerade bei dem Thema Austauschstudenten sind...
- Sie haben wahrscheinlich keine chinesische Ausweisnummer (中国公民身份号码¹¹⁰).

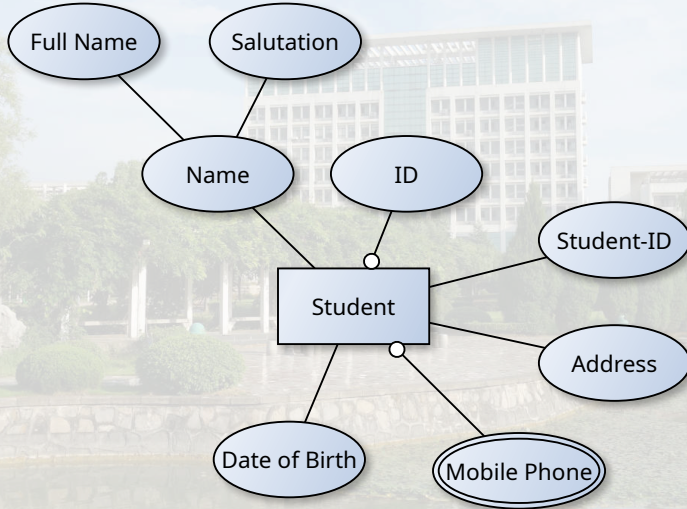
Ausweisnummern



- Und wo wir gerade bei dem Thema Austauschstudenten sind...
- Sie haben wahrscheinlich keine chinesische Ausweisnummer (中国公民身份号码¹¹⁰).
- Deshalb sollte auch ID ein optionales Attribut sein.

Neues ERD

Wir bekommen das folgende neue ERD.



Adressen



- Wir bemerken auch, dass *Address* wahrscheinlich auch ein zusammengesetztes Attribut sein sollte.

Adressen



- Wir bemerken auch, dass *Address* wahrscheinlich auch ein zusammengesetztes Attribut sein sollte.
- Eine Adresse ist ja nicht nur eine Zeile Text.

Adressen



- Wir bemerken auch, dass *Address* wahrscheinlich auch ein zusammengesetztes Attribut sein sollte.
- Eine Adresse ist ja nicht nur eine Zeile Text.
- Sie besteht aus Land, Provinz, Stadt, Distrikt, Postleitzahl, Straße und Hausnummer.

Adressen



- Wir bemerken auch, dass *Address* wahrscheinlich auch ein zusammengesetztes Attribut sein sollte.
- Eine Adresse ist ja nicht nur eine Zeile Text.
- Sie besteht aus Land, Provinz, Stadt, Distrikt, Postleitzahl, Straße und Hausnummer.
- Das zu modellieren wäre umständlich.

Adressen



- Wir bemerken auch, dass *Address* wahrscheinlich auch ein zusammengesetztes Attribut sein sollte.
- Eine Adresse ist ja nicht nur eine Zeile Text.
- Sie besteht aus Land, Provinz, Stadt, Distrikt, Postleitzahl, Straße und Hausnummer.
- Das zu modellieren wäre umständlich.
- Wir entscheiden uns dafür, dass alles nach der Postleitzahl (EN: *postal code*) einfach eine Zeichenkette mit Namen *street address* werden soll.

Adressen



- Wir bemerken auch, dass *Address* wahrscheinlich auch ein zusammengesetztes Attribut sein sollte.
- Eine Adresse ist ja nicht nur eine Zeile Text.
- Sie besteht aus Land, Provinz, Stadt, Distrikt, Postleitzahl, Straße und Hausnummer.
- Das zu modellieren wäre umständlich.
- Wir entscheiden uns dafür, dass alles nach der Postleitzahl (EN: *postal code*) einfach eine Zeichenkette mit Namen *street address* werden soll.
- Ein automatisches System kann Informationen die feiner auflöst als eine Postleitzahl wahrscheinlich sowieso nicht sinnvoll verarbeiten.

Adressen

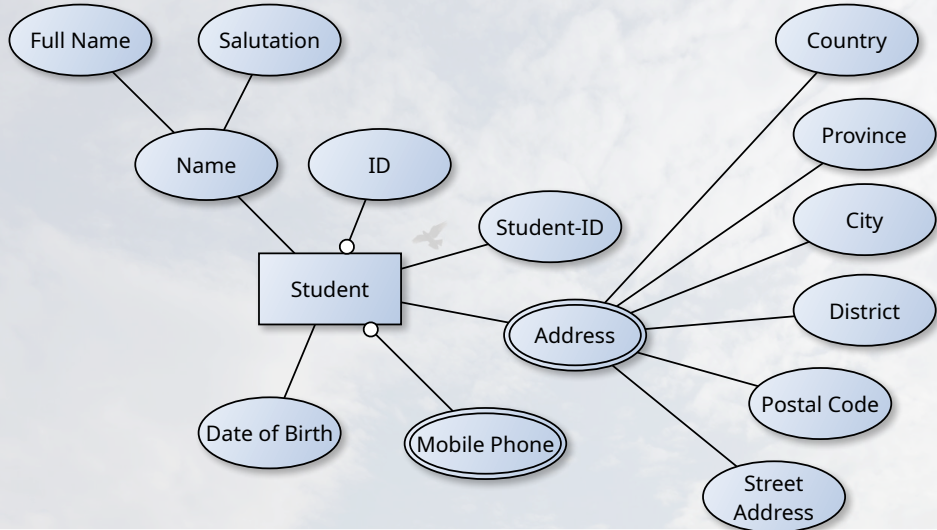


- Wir bemerken auch, dass *Address* wahrscheinlich auch ein zusammengesetztes Attribut sein sollte.
- Eine Adresse ist ja nicht nur eine Zeile Text.
- Sie besteht aus Land, Provinz, Stadt, Distrikt, Postleitzahl, Straße und Hausnummer.
- Das zu modellieren wäre umständlich.
- Wir entscheiden uns dafür, dass alles nach der Postleitzahl (EN: *postal code*) einfach eine Zeichenkette mit Namen *street address* werden soll.
- Ein automatisches System kann Informationen die feiner auflöst als eine Postleitzahl wahrscheinlich sowieso nicht sinnvoll verarbeiten.
- Während wir diese Adressverarbeitung mit unseren Partnern diskutieren, stellen wir fest, dass Studenten auch mehrere Adressen haben können.

Adressen



- Wir bekommen also ein neues ERD.



Abgeleitete Attribute



Definition: Abgeleitetes Attribut

Ein abgeleitetes Attribut (EN: *derived attribute*) wird aus den Werten anderer Attribute berechnet.

Abgeleitete Attribute



Definition: Abgeleitetes Attribut

Ein abgeleitetes Attribut (EN: *derived attribute*) wird aus den Werten anderer Attribute berechnet.

- Abgeleitete Attribute müssen nicht in der Datenbank gespeichert werden.

Abgeleitete Attribute



Definition: Abgeleitetes Attribut

Ein abgeleitetes Attribut (EN: *derived attribute*) wird aus den Werten anderer Attribute berechnet.

- Abgeleitete Attribute müssen nicht in der Datenbank gespeichert werden.
- Ein typisches Beispiel ist das Alter einer Person.

Abgeleitete Attribute



Definition: Abgeleitetes Attribut

Ein abgeleitetes Attribut (EN: *derived attribute*) wird aus den Werten anderer Attribute berechnet.

- Abgeleitete Attribute müssen nicht in der Datenbank gespeichert werden.
- Ein typisches Beispiel ist das Alter einer Person.
- Wenn wir den Geburtstag (DOB) und das aktuelle Datum kennen, dann kann das Alter berechnet werden.

Abgeleitete Attribute



Definition: Abgeleitetes Attribut

Ein abgeleitetes Attribut (EN: *derived attribute*) wird aus den Werten anderer Attribute berechnet.

- Abgeleitete Attribute müssen nicht in der Datenbank gespeichert werden.
- Ein typisches Beispiel ist das Alter einer Person.
- Wenn wir den Geburtstag (DOB) und das aktuelle Datum kennen, dann kann das Alter berechnet werden.
- Das geht sehr schnell.

Abgeleitete Attribute



Definition: Abgeleitetes Attribut

Ein abgeleitetes Attribut (EN: *derived attribute*) wird aus den Werten anderer Attribute berechnet.

- Abgeleitete Attribute müssen nicht in der Datenbank gespeichert werden.
- Ein typisches Beispiel ist das Alter einer Person.
- Wenn wir den Geburtstag (DOB) und das aktuelle Datum kennen, dann kann das Alter berechnet werden.
- Das geht sehr schnell.
- Es gibt keinen Grund, das Alter (EN: *age*) einer Person in der Datenbank zu speichern.

Abgeleitete Attribute



Definition: Abgeleitetes Attribut

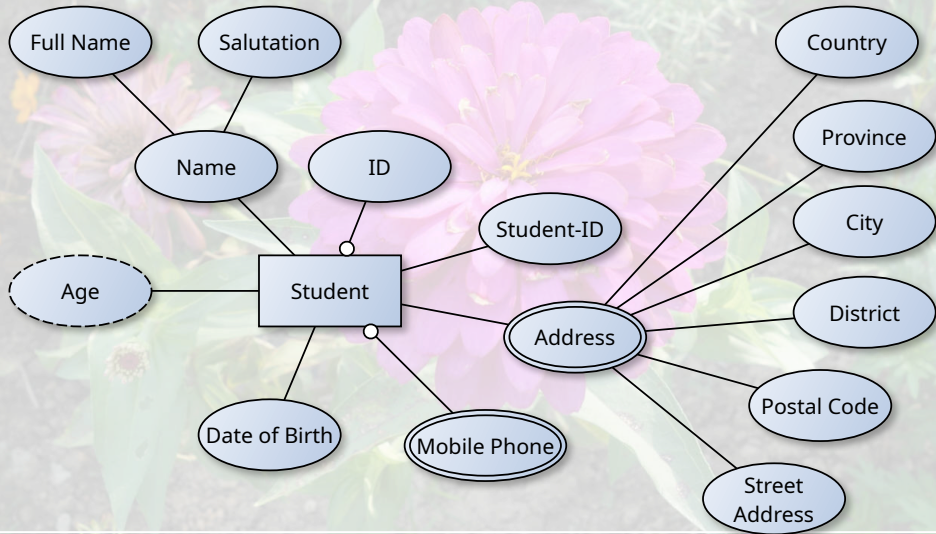
Ein abgeleitetes Attribut (EN: *derived attribute*) wird aus den Werten anderer Attribute berechnet.

- Abgeleitete Attribute müssen nicht in der Datenbank gespeichert werden.
- Ein typisches Beispiel ist das Alter einer Person.
- Wenn wir den Geburtstag (DOB) und das aktuelle Datum kennen, dann kann das Alter berechnet werden.
- Das geht sehr schnell.
- Es gibt keinen Grund, das Alter (EN: *age*) einer Person in der Datenbank zu speichern.
- Abgeleitete Attribute werden in ERDs mit Ellipsen mit gestrichelten Linien dargestellt.

Abgeleitete Attribute



- Abgeleitete Attribute werden in ERDs mit Ellipsen mit gestrichelten Linien dargestellt.





Zusammenfassung



Zusammenfassung: Entitäten



- Wir haben nun sehr viel über Entitäten und Attribute gelernt.

Zusammenfassung: Entitäten



- Wir haben nun sehr viel über Entitäten und Attribute gelernt.
- Eine Entität modelliert ein Objekt aus der realen Welt, wie z. B. eine Person, ein Ding, einen Ort, ein Ereignis, oder ein Konzept.

Zusammenfassung: Entitäten



- Wir haben nun sehr viel über Entitäten und Attribute gelernt.
- Eine Entität modelliert ein Objekt aus der realen Welt, wie z. B. eine Person, ein Ding, einen Ort, ein Ereignis, oder ein Konzept.
- Eine Entität existiert nicht einfach so.

Zusammenfassung: Entitäten



- Wir haben nun sehr viel über Entitäten und Attribute gelernt.
- Eine Entität modelliert ein Objekt aus der realen Welt, wie z. B. eine Person, ein Ding, einen Ort, ein Ereignis, oder ein Konzept.
- Eine Entität existiert nicht einfach so.
- Sie ist ein Tupel aus ihren Attributwerten.

Zusammenfassung: Entitäten



- Wir haben nun sehr viel über Entitäten und Attribute gelernt.
- Eine Entität modelliert ein Objekt aus der realen Welt, wie z. B. eine Person, ein Ding, einen Ort, ein Ereignis, oder ein Konzept.
- Eine Entität existiert nicht einfach so.
- Sie ist ein Tupel aus ihren Attributwerten.
- Während Entitäten einzelne Objekte beschreiben, beschreiben Entitätstypen Gruppen von Objekten, wie Personen, Dinge, Orte, eine Klasse von Ereignissen, usw.

Zusammenfassung: Attribute

- Attribute können ebenfalls verschiedene Eigenschaften haben.



Zusammenfassung: Attribute

- Attribute können ebenfalls verschiedene Eigenschaften haben.
- Sie können einfach sein, also einen Wert haben, der nicht weiter unterteilt werden kann.



Zusammenfassung: Attribute

- Attribute können ebenfalls verschiedene Eigenschaften haben.
- Sie können einfach sein, also einen Wert haben, der nicht weiter unterteilt werden kann, wie eine Telefonnummer.



Zusammenfassung: Attribute



- Attribute können ebenfalls verschiedene Eigenschaften haben.
- Sie können einfach sein, also einen Wert haben, der nicht weiter unterteilt werden kann, wie eine Telefonnummer.
- Sie können zusammengesetzt sein, also Teilen bestehen.

Zusammenfassung: Attribute



- Attribute können ebenfalls verschiedene Eigenschaften haben.
- Sie können einfach sein, also einen Wert haben, der nicht weiter unterteilt werden kann, wie eine Telefonnummer.
- Sie können zusammengesetzt sein, also Teilen bestehen, wie Adressen, die aus Land, Provinz, Stadt, usw. bestehen.

Zusammenfassung: Attribute



- Attribute können ebenfalls verschiedene Eigenschaften haben.
- Sie können einfach sein, also einen Wert haben, der nicht weiter unterteilt werden kann, wie eine Telefonnummer.
- Sie können zusammengesetzt sein, also Teilen bestehen, wie Adressen, die aus Land, Provinz, Stadt, usw. bestehen.
- Es ist nicht immer klar, wie ein Attribut modelliert werden soll.

Zusammenfassung: Attribute



- Attribute können ebenfalls verschiedene Eigenschaften haben.
- Sie können einfach sein, also einen Wert haben, der nicht weiter unterteilt werden kann, wie eine Telefonnummer.
- Sie können zusammengesetzt sein, also Teilen bestehen, wie Adressen, die aus Land, Provinz, Stadt, usw. bestehen.
- Es ist nicht immer klar, wie ein Attribut modelliert werden soll.
- Zum Beispiel haben unsere Studenten ein Attribut Date of Birth (DOB).

Zusammenfassung: Attribute



- Attribute können ebenfalls verschiedene Eigenschaften haben.
- Sie können einfach sein, also einen Wert haben, der nicht weiter unterteilt werden kann, wie eine Telefonnummer.
- Sie können zusammengesetzt sein, also Teilen bestehen, wie Adressen, die aus Land, Provinz, Stadt, usw. bestehen.
- Es ist nicht immer klar, wie ein Attribut modelliert werden soll.
- Zum Beispiel haben unsere Studenten ein Attribut Date of Birth (DOB).
- Technisch haben wir bereits gelernt, dass ein Datum ein atomarer Datentyp in SQL ist.

Zusammenfassung: Attribute



- Attribute können ebenfalls verschiedene Eigenschaften haben.
- Sie können einfach sein, also einen Wert haben, der nicht weiter unterteilt werden kann, wie eine Telefonnummer.
- Sie können zusammengesetzt sein, also Teilen bestehen, wie Adressen, die aus Land, Provinz, Stadt, usw. bestehen.
- Es ist nicht immer klar, wie ein Attribut modelliert werden soll.
- Zum Beispiel haben unsere Studenten ein Attribut Date of Birth (DOB).
- Technisch haben wir bereits gelernt, dass ein Datum ein atomarer Datentyp in SQL ist.
- Wir können DOB also durchaus als einfaches Attribut modellieren.

Zusammenfassung: Attribute



- Attribute können ebenfalls verschiedene Eigenschaften haben.
- Sie können einfach sein, also einen Wert haben, der nicht weiter unterteilt werden kann, wie eine Telefonnummer.
- Sie können zusammengesetzt sein, also Teilen bestehen, wie Adressen, die aus Land, Provinz, Stadt, usw. bestehen.
- Es ist nicht immer klar, wie ein Attribut modelliert werden soll.
- Zum Beispiel haben unsere Studenten ein Attribut Date of Birth (DOB).
- Technisch haben wir bereits gelernt, dass ein Datum ein atomarer Datentyp in SQL ist.
- Wir können DOB also durchaus als einfaches Attribut modellieren.
- Wir könnten es aber auch als zusammengesetztes Attribut aus Jahr, Monat, und Tag modellieren.

Zusammenfassung: Attribute



- Attribute können ebenfalls verschiedene Eigenschaften haben.
- Sie können einfach sein, also einen Wert haben, der nicht weiter unterteilt werden kann, wie eine Telefonnummer.
- Sie können zusammengesetzt sein, also Teilen bestehen, wie Adressen, die aus Land, Provinz, Stadt, usw. bestehen.
- Es ist nicht immer klar, wie ein Attribut modelliert werden soll.
- Zum Beispiel haben unsere Studenten ein Attribut Date of Birth (DOB).
- Technisch haben wir bereits gelernt, dass ein Datum ein atomarer Datentyp in SQL ist.
- Wir können DOB also durchaus als einfaches Attribut modellieren.
- Wir könnten es aber auch als zusammengesetztes Attribut aus Jahr, Monat, und Tag modellieren.
- Das würde allerdings kein vernünftig denkender Mensch machen, denn dann muss man ja auch Dinge wie Zeitzonen und verschiedene Kalender beachten...

Zusammenfassung: Attribute



- Attribute können ebenfalls verschiedene Eigenschaften haben.
- Sie können einfach sein, also einen Wert haben, der nicht weiter unterteilt werden kann, wie eine Telefonnummer.
- Sie können zusammengesetzt sein, also Teilen bestehen, wie Adressen, die aus Land, Provinz, Stadt, usw. bestehen.
- Es ist nicht immer klar, wie ein Attribut modelliert werden soll.
- Zum Beispiel haben unsere Studenten ein Attribut Date of Birth (DOB).
- Technisch haben wir bereits gelernt, dass ein Datum ein atomarer Datentyp in SQL ist.
- Wir können DOB also durchaus als einfaches Attribut modellieren.
- Wir könnten es aber auch als zusammengesetztes Attribut aus Jahr, Monat, und Tag modellieren.
- Das würde allerdings kein vernünftig denkender Mensch machen, denn dann muss man ja auch Dinge wie Zeitzonen und verschiedene Kalender beachten...
- Eine Adresse kann ein einfacher Textstring oder ein zusammengesetztes Attribut sein.

Zusammenfassung: Attribute



- Sie können einfach sein, also einen Wert haben, der nicht weiter unterteilt werden kann, wie eine Telefonnummer.
- Sie können zusammengesetzt sein, also Teilen bestehen, wie Adressen, die aus Land, Provinz, Stadt, usw. bestehen.
- Es ist nicht immer klar, wie ein Attribut modelliert werden soll.
- Zum Beispiel haben unsere Studenten ein Attribut Date of Birth (DOB).
- Technisch haben wir bereits gelernt, dass ein Datum ein atomarer Datentyp in SQL ist.
- Wir können DOB also durchaus als einfaches Attribut modellieren.
- Wir könnten es aber auch als zusammengesetztes Attribut aus Jahr, Monat, und Tag modellieren.
- Das würde allerdings kein vernünftig denkender Mensch machen, denn dann muss man ja auch Dinge wie Zeitzonen und verschiedene Kalender beachten...
- Eine Adresse kann ein einfach Textstring oder ein zusammengesetztes Attribut sein.
- Die Entscheidung, wie wir Attribute modellieren, hängt oft von den Daten ab, die wir brauchen.

Zusammenfassung: Attribute



- Sie können zusammengesetzt sein, also Teilen bestehen, wie Adressen, die aus Land, Provinz, Stadt, usw. bestehen.
- Es ist nicht immer klar, wie ein Attribut modelliert werden soll.
- Zum Beispiel haben unsere Studenten ein Attribut Date of Birth (DOB).
- Technisch haben wir bereits gelernt, dass ein Datum ein atomarer Datentyp in SQL ist.
- Wir können DOB also durchaus als einfaches Attribut modellieren.
- Wir könnten es aber auch als zusammengesetztes Attribut aus Jahr, Monat, und Tag modellieren.
- Das würde allerdings kein vernünftig denkender Mensch machen, denn dann muss man ja auch Dinge wie Zeitzonen und verschiedene Kalender beachten...
- Eine Adresse kann ein einfach Textstring oder ein zusammengesetztes Attribut sein.
- Die Entscheidung, wie wir Attribute modellieren, hängt oft von den Daten ab, die wir brauchen.
- Wenn wir das Geburtsdatum als einfaches Attribut vom Datentyp „Datum“ speichern, dann kann wir sehr einfach das Geburtsjahr oder das Alter einer Person berechnen.

Zusammenfassung: Attribute



- Es ist nicht immer klar, wie ein Attribut modelliert werden soll.
- Zum Beispiel haben unsere Studenten ein Attribut Date of Birth (DOB).
- Technisch haben wir bereits gelernt, dass ein Datum ein atomarer Datentyp in SQL ist.
- Wir können DOB also durchaus als einfaches Attribut modellieren.
- Wir könnten es aber auch als zusammengesetztes Attribut aus Jahr, Monat, und Tag modellieren.
- Das würde allerdings kein vernünftig denkender Mensch machen, denn dann muss man ja auch Dinge wie Zeitzonen und verschiedene Kalender beachten...
- Eine Adresse kann ein einfach Textstring oder ein zusammengesetztes Attribut sein.
- Die Entscheidung, wie wir Attribute modellieren, hängt oft von den Daten ab, die wir brauchen.
- Wenn wir das Geburtsdatum als einfaches Attribut vom Datentyp „Datum“ speichern, dann kann wir sehr einfach das Geburtsjahr oder das Alter einer Person berechnen.
- Speichern wir die Adresse jedoch als einfaches Attribut, dann wird es eben sehr schwer, das Land, die Provinz, oder die Stadt automatisch zu extrahieren.

Zusammenfassung: Attribute



- Zum Beispiel haben unsere Studenten ein Attribut Date of Birth (DOB).
- Technisch haben wir bereits gelernt, dass ein Datum ein atomarer Datentyp in SQL ist.
- Wir können DOB also durchaus als einfaches Attribut modellieren.
- Wir könnten es aber auch als zusammengesetztes Attribut aus Jahr, Monat, und Tag modellieren.
- Das würde allerdings kein vernünftig denkender Mensch machen, denn dann muss man ja auch Dinge wie Zeitzonen und verschiedene Kalender beachten...
- Eine Adresse kann ein einfach Textstring oder ein zusammengesetztes Attribut sein.
- Die Entscheidung, wie wir Attribute modellieren, hängt oft von den Daten ab, die wir brauchen.
- Wenn wir das Geburtsdatum als einfaches Attribut vom Datentyp „Datum“ speichern, dann kann wir sehr einfach das Geburtsjahr oder das Alter einer Person berechnen.
- Speichern wir die Adresse jedoch als einfaches Attribut, dann wird es eben sehr schwer, das Land, die Provinz, oder die Stadt automatisch zu extrahieren.
- Attribute können zusätzlich noch einwertig oder mehrwertig sein, optional oder nicht-optional.

Zusammenfassung



- Wir können nun die Daten, die wir in unsere Datenbank speichern, schon ziemlich gut modellieren.

Zusammenfassung



- Wir können nun die Daten, die wir in unsere Datenbank speichern, schon ziemlich gut modellieren.
- In einer Datenbank geht es aber nicht nur um Daten.

Zusammenfassung



- Wir können nun die Daten, die wir in unsere Datenbank speichern, schon ziemlich gut modellieren.
- In einer Datenbank geht es aber nicht nur um Daten.
- Es geht auch um das Einschränken der Daten und die Beziehungen zwischen den Daten.

Zusammenfassung



- Wir können nun die Daten, die wir in unsere Datenbank speichern, schon ziemlich gut modellieren.
- In einer Datenbank geht es aber nicht nur um Daten.
- Es geht auch um das Einschränken der Daten und die Beziehungen zwischen den Daten.
- Und als nächstes lernen wir eine bestimmte Einschränkung kennen...



谢谢你们！
Thank you!
Vielen Dank!



References I



- [1] Raphael „rkhaotix“ Araújo e Silva. *pgModeler – PostgreSQL Database Modeler*. Palmas, Tocantins, Brazil, 2006–2025. URL: <https://pgmodeler.io> (besucht am 2025-04-12) (siehe S. 98–111, 256).
- [2] Charles William „Charlie“ Bachman. “Data Structure Diagrams”. *DATA BASE – ACM SIGMIS Database: The DATABASE for Advances in Information Systems* 1(2):4–10, Sommer 1969. New York, NY, USA: Association for Computing Machinery (ACM). ISSN: 0095-0033. doi:10.1145/1017466.1017467 (siehe S. 85–97).
- [3] 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. 5–14, 85–97, 254).
- [4] Daniel J. Barrett. *Efficient Linux at the Command Line*. Sebastopol, CA, USA: O'Reilly Media, Inc., Feb. 2022. ISBN: 978-1-0981-1340-7 (siehe S. 255, 257).
- [5] Ben Beitler. *Hands-On Microsoft Access 2019*. Birmingham, England, UK: Packt Publishing Ltd, März 2020. ISBN: 978-1-83898-747-3 (siehe S. 255).
- [6] Tim Berners-Lee. *Re: Qualifiers on Hypertext links...* Geneva, Switzerland: World Wide Web project, European Organization for Nuclear Research (CERN) und Newsgroups: alt.hypertext, 6. Aug. 1991. URL: <https://www.w3.org/People/Berners-Lee/1991/08/art-6484.txt> (besucht am 2025-02-05) (siehe S. 257).
- [7] Alex Berson. *Client/Server Architecture*. 2. Aufl. Computer Communications Series. New York, NY, USA: McGraw-Hill, 29. März 1996. ISBN: 978-0-07-005664-0 (siehe S. 254).
- [8] Bernard Obeng Boateng. *Data Modeling with Microsoft Excel*. Birmingham, England, UK: Packt Publishing Ltd, Nov. 2023. ISBN: 978-1-80324-028-2 (siehe S. 255).
- [9] Grady Booch, James Rumbaugh und Ivar Jacobson. *The Unified Modeling Language Reference Manual*. 1. Aufl. Reading, MA, USA: Addison-Wesley Professional, Jan. 1999. ISBN: 978-0-201-57168-4 (siehe S. 85–97, 257).
- [10] Silvia Botros und Jeremy Tinley. *High Performance MySQL*. 4. Aufl. Sebastopol, CA, USA: O'Reilly Media, Inc., Nov. 2021. ISBN: 978-1-4920-8051-0 (siehe S. 256).

References II



- [11] Ed Bott. *Windows 11 Inside Out*. Hoboken, NJ, USA: Microsoft Press, Pearson Education, Inc., Feb. 2023. ISBN: 978-0-13-769132-6 (siehe S. 256).
- [12] Ron Brash und Ganesh Naik. *Bash Cookbook*. Birmingham, England, UK: Packt Publishing Ltd, Juli 2018. ISBN: 978-1-78862-936-2 (siehe S. 254).
- [13] Tim Bray, Jean Paoli, C. M. Sperberg-McQueen und Eve Maler, Hrsg. *Extensible Markup Language (XML) 1.0 (Fifth Edition)*. W3C Recommendation. Wakefield, MA, USA: World Wide Web Consortium (W3C), 26. Nov. 2008–7. Feb. 2013. URL: <http://www.w3.org/TR/2008/REC-xml-20081126> (besucht am 2024-12-15) (siehe S. 258).
- [14] Ronald H. Brown, Mary L. Good, Arati Prabhakar und James H. Burrows. *Integration Definition for Information Modeling (IDEF1X)*. Federal Information Processing Standards Publication (FIPS PUB) 184. Gaithersburg, MD, USA: U.S. Department of Commerce, National Institute of Standards and Technology (NIST), 21. Dez. 1993. URL: <https://nvlpubs.nist.gov/nistpubs/Legacy/FIPS/fipspub184.pdf> (besucht am 2025-03-29). Software Standard (siehe S. 85–97, 255).
- [15] Ben Brumm. "79 Data Modeling Tools Compared". In: *Database Star*. Armadale, VIC, Australia: Elevated Online Services PTY Ltd., 20. Okt. 2018–26. März 2023. URL: <https://www.databasestar.com/data-modeling-tools> (besucht am 2025-04-05) (siehe S. 98–111).
- [16] 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. 254).
- [17] Ben Brumm. *Database Star*. Armadale, VIC, Australia: Elevated Online Services PTY Ltd., Dez. 2024. URL: <https://www.databasestar.com> (besucht am 2025-03-29).
- [18] Jason Cannon. *High Availability for the LAMP Stack*. Shelter Island, NY, USA: Manning Publications, Juni 2022 (siehe S. 255, 257).
- [19] John Vincent Carlis und Joseph D. Maguire. *Mastering Data Modeling: A User Driven Approach*. Reading, MA, USA: Addison-Wesley Professional, Nov. 2000. ISBN: 978-0-201-70045-9 (siehe S. 85–97).

References III



- [20] 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. 257).
- [21] Peter Pin-Shan Chen. "English, Chinese and ER Diagrams". *Data & Knowledge Engineering (DKE)* 23(1):5–16, Juni 1997. Amsterdam, The Netherlands: Elsevier B.V. ISSN: 0169-023X. doi:10.1016/S0169-023X(97)00017-7. URL: https://www.csc.lsu.edu/~chen/pdf/ER_C.pdf (besucht am 2025-04-06) (siehe S. 16–31, 41–45).
- [22] 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. 85–97, 254).
- [23] 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. 85–97, 243, 254).
- [24] 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. See²³ for a more comprehensive introduction. (Siehe S. 85–97, 254).
- [25] 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. 255).
- [26] 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. 257).
- [27] 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. 256).

References IV



- [28] Timothy W. Cole und Myung-Ja K. Han. *XML for Catalogers and Metadata Librarians (Third Millennium Cataloging)*. 1. Aufl. Dublin, OH, USA: Libraries Unlimited, 23. Mai 2013. ISBN: **978-1-59884-519-8** (siehe S. **258**).
- [29] “**CSV** – CSV File Reading and Writing”. In: *Python 3 Documentation. The Python Standard Library*. Beaverton, OR, USA: Python Software Foundation (PSF), 2001–2025. URL: <https://docs.python.org/3/library/csv.html> (besucht am 2024-11-14) (siehe S. **254**).
- [30] Erik Dahlström, Patrick Dengler, Anthony Grasso, Chris Lilley, Cameron McCormack, Doug Schepers, Jonathan Watt, Jon Ferraiolo, Jun Fujisawa und Dean Jackson, Hrsg. *Scalable Vector Graphics (SVG) 1.1 (Second Edition)*. W3C Recommendation. Wakefield, MA, USA: World Wide Web Consortium (W3C), 16. Aug. 2011. URL: <http://www.w3.org/TR/2011/REC-SVG11-20110816> (besucht am 2024-12-17) (siehe S. **257**).
- [31] *Database Language SQL*. Techn. Ber. ANSI X3.135-1986. Washington, D.C., USA: American National Standards Institute (ANSI), 1986 (siehe S. **257**).
- [32] 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. **257**).
- [33] *Database Language SQL*. International Standard ISO 9075-1987. Geneva, Switzerland: International Organization for Standardization (ISO), 1987 (siehe S. **257**).
- [34] Paul Deitel, Harvey Deitel und Abbey Deitel. *Internet & World Wide Web: How to Program*. 5. Aufl. Hoboken, NJ, USA: Pearson Education, Inc., Nov. 2011. ISBN: **978-0-13-299045-5** (siehe S. **257**).
- [35] 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](https://doi.org/10.1145/3351470) (siehe S. **256**).
- [36] Russell J.T. Dyer. *Learning MySQL and MariaDB*. Sebastopol, CA, USA: O'Reilly Media, Inc., März 2015. ISBN: **978-1-4493-6290-4** (siehe S. **256**).

References V



- [37] .“Common and Proper Nouns: What’s the difference? Learn which ones get capitals.”. In: *Merriam-Webster: America’s Most Trusted Dictionary*. Hrsg. von Editors of Merriam-Webster. 6. Apr.–12. Mai 2023. URL: <https://www.merriam-webster.com/grammar/common-and-proper-nouns-whats-the-difference> (besucht am 2025-03-29) (siehe S. 16–25, 41–45).
- [38] Gordon C. Everest. “Basic Data Structure Models Explained with a Common Example”. In: *Fifth Texas Conference on Computing Systems (Computing Systems’1976)*. 18.–19. Okt. 1976, Austin, TX, USA. Long Beach, CA, USA: IEEE Computer Society Publications Office, 1976, S. 39–46. ISSN: 0730-8310. URL: <https://www.researchgate.net/publication/291448084> (besucht am 2025-04-04) (siehe S. 85–97).
- [39] Steve Fanning, Vasudev Narayanan, „flywire“, Olivier Hallot, Jean Hollis Weber, Jenna Sargent, Pulkit Krishna, Dan Lewis, Peter Schofield, Jochen Schiffrers, Robert Großkopf, Jost Lange, Martin Fox, Hazel Russman, Steve Schwettman, Alain Romedenne, Andrew Pitonyak, Jean-Pierre Ledure, Drew Jensen and Randolph Gam. *Base Guide 7.3. Revision 1. Based on LibreOffice 7.3 Community*. Berlin, Germany: The Document Foundation, Aug. 2022. URL: <https://books.libreoffice.org/en/BG73/BG73-BaseGuide.pdf> (besucht am 2025-01-13) (siehe S. 255).
- [40] 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. 256).
- [41] 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. 255).
- [42] Todd J. Green. “Conceptual Modeling using the Entity-Relationship Model”. In: *ECS 165A Winter 2011 – Introduction to Database Systems*. Hrsg. von Todd J. Green. Davis, CA, USA: University of California, Davis, Winter 2011. Kap. 2. URL: <https://web.cs.ucdavis.edu/~green/courses/ecs165a-w11/2-er.pdf> (besucht am 2025-03-27) (siehe S. 5–14).
- [43] 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. 255).

References VI



- [44] 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. 256).
- [45] 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. 256).
- [46] Michael Hausenblas. *Learning Modern Linux*. Sebastopol, CA, USA: O'Reilly Media, Inc., Apr. 2022. ISBN: **978-1-0981-0894-6** (siehe S. 255).
- [47] Christian Heimes. "[defusedxml](https://pypi.org/project/defusedxml) 0.7.1: XML Bomb Protection for Python stdlib Modules". In: 8. März 2021. URL: <https://pypi.org/project/defusedxml> (besucht am 2024-12-15) (siehe S. 258).
- [48] 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. 255, 257).
- [49] 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](https://doi.org/10.1007/978-3-031-35122-8) (siehe S. 256).
- [50] *Information Technology – Database Languages – SQL – Part 1: Framework (SQL/Framework), Part 1*. International Standard ISO/IEC 9075-1:2023(E), Sixth Edition, (ANSI X3.135). Geneva, Switzerland: International Organization for Standardization (ISO) und International Electrotechnical Commission (IEC), Juni 2023. URL: [https://standards.iso.org/ittf/PubliclyAvailableStandards/ISO_IEC_9075-1_2023_ed_6_-_id_76583_Publication_PDF_\(en\).zip](https://standards.iso.org/ittf/PubliclyAvailableStandards/ISO_IEC_9075-1_2023_ed_6_-_id_76583_Publication_PDF_(en).zip) (besucht am 2025-01-08). Consists of several parts, see <https://modern-sql.com/standard> for information where to obtain them. (Siehe S. 257).
- [51] *Information Technology -- Modeling Languages -- Part 2: Syntax and Semantics for IDEF1X₉₇ (IDEF_{object})*. ISO/IEC/IEEE International Standard 31320-2-2012(E). Geneva, Switzerland: International Organization for Standardization (ISO), International Electrotechnical Commission (IEC) und New York, NY, USA: Institute of Electrical and Electronics Engineers (IEEE), 15. Sep.–30. Okt. 2012. doi:[10.1109/IEEESTD.2012.6357338](https://doi.org/10.1109/IEEESTD.2012.6357338) (siehe S. 85–97, 255).

References VII



- [52] 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. 85–97, 254).
- [53] Dmitry Kirsanov. *The Book of Inkscape*. 2. Aufl. San Francisco, CA, USA, Nov. 2021. ISBN: 978-1-7185-0175-1 (siehe S. 98–111, 255).
- [54] Katie Kodes. *Intro to XML, JSON, & YAML*. London, England, UK: Payhip, 2019–4. Sep. 2020 (siehe S. 258).
- [55] Jay LaCroix. *Mastering Ubuntu Server*. 4. Aufl. Birmingham, England, UK: Packt Publishing Ltd, Sep. 2022. ISBN: 978-1-80323-424-3 (siehe S. 257).
- [56] 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. 255, 256).
- [57] 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. 256).
- [58] *LibreOffice – The Document Foundation*. Berlin, Germany: The Document Foundation, 2024. URL: <https://www.libreoffice.org> (besucht am 2024-12-12) (siehe S. 255).
- [59] Gloria Lotha, Aakanksha Gaur, Erik Gregersen, Swati Chopra und William L. Hosch. "Client-Server Architecture". In: *Encyclopaedia Britannica*. Hrsg. von The Editors of Encyclopaedia Britannica. Chicago, IL, USA: Encyclopædia Britannica, Inc., 3. Jan. 2025. URL: <https://www.britannica.com/technology/client-server-architecture> (besucht am 2025-01-20) (siehe S. 254).
- [60] Mark Lutz. *Learning Python*. 6. Aufl. Sebastopol, CA, USA: O'Reilly Media, Inc., März 2025. ISBN: 978-1-0981-7130-8 (siehe S. 256).
- [61] 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. 255).
- [62] Michael McLaughlin. *MySQL Workbench: Data Modeling & Development*. Oracle Press. New York, NY, USA: McGraw-Hill, 9. Apr. 2013. ISBN: 978-0-07-179188-5 (siehe S. 98–111, 256).

References VIII



- [63] 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. **257**).
- [64] *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. **256**).
- [65] Cameron Newham und Bill Rosenblatt. *Learning the Bash Shell – Unix Shell Programming: Covers Bash 3.0*. 3. Aufl. Sebastopol, CA, USA: O'Reilly Media, Inc., 2005. ISBN: **978-0-596-00965-6** (siehe S. **254**).
- [66] John J. O'Connor und Edmund F. Robertson. *Liu Hui*. St Andrews, Scotland, UK: University of St Andrews, School of Mathematics and Statistics, Dez. 2003. URL: https://mathshistory.st-andrews.ac.uk/Biographies/Liu_Hui (besucht am 2024-08-10) (siehe S. **159–172**).
- [67] 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. **256**).
- [68] *OMG® Unified Modeling Language® (OMG UML®). Version 2.5.1*. OMG Document formal/2017-12-05. Milford, MA, USA: Object Management Group, Inc. (OMG), Dez. 2017. URL: <https://www.omg.org/spec/UML/2.5.1/PDF> (besucht am 2025-03-30) (siehe S. **85–97, 257**).
- [69] Robert Orfali, Dan Harkey und Jeri Edwards. *Client/Server Survival Guide*. 3. Aufl. Chichester, West Sussex, England, UK: John Wiley and Sons Ltd., 25. Jan. 1999. ISBN: **978-0-471-31615-2** (siehe S. **254**).
- [70] *PostgreSQL Essentials: Leveling Up Your Data Work*. Sebastopol, CA, USA: O'Reilly Media, Inc., März 2024 (siehe S. **256**).
- [71] Abhishek Ratan, Eric Chou, Pradeeban Kathiravelu und Dr. M.O. Faruque Sarker. *Python Network Programming*. Birmingham, England, UK: Packt Publishing Ltd, Jan. 2019. ISBN: **978-1-78883-546-6** (siehe S. **254**).
- [72] Mike Reichardt, Michael Gundall und Hans D. Schotten. "Benchmarking the Operation Times of NoSQL and MySQL Databases for Python Clients". In: *47th Annual Conference of the IEEE Industrial Electronics Society (IECON'2021)*. 13.–15. Okt. 2021, Toronto, ON, Canada. Piscataway, NJ, USA: Institute of Electrical and Electronics Engineers (IEEE), 2021, S. 1–8. ISSN: **2577-1647**. ISBN: **978-1-6654-3554-3**. doi:[10.1109/IECON48115.2021.9589382](https://doi.org/10.1109/IECON48115.2021.9589382) (siehe S. **256**).

References IX



- [73] Mark Richards und Neal Ford. *Fundamentals of Software Architecture: An Engineering Approach*. Sebastopol, CA, USA: O'Reilly Media, Inc., Jan. 2020. ISBN: **978-1-4920-4345-4** (siehe S. **254**).
- [74] Christopher Rogers. *Design Made Easy with Inkscape 1.3. A practical guide to your journey from beginner to pro-level vector illustration*. Birmingham, England, UK: Packt Publishing Ltd, Apr. 2023. ISBN: **978-1-80107-877-1** (siehe S. **98–111, 255**).
- [75] Stephen R. Schach. *Object-Oriented Software Engineering*. New York, NY, USA: McGraw-Hill, Sep. 2007. ISBN: **978-0-07-352333-0** (siehe S. **256**).
- [76] Heinz Schweppe und Manuel Scholz. "Conceptual Database Design: Integrity Constraints and Modeling Patterns". In: *Einführung in die Datenbanksysteme. Datenbanken für die Bioinformatik*. Berlin, Germany: Freie Universität Berlin, Apr.–Okt. 2005. Kap. 2.3/2.4. URL: <https://www.inf.fu-berlin.de/lehre/SS05/19517-V/FolienEtc/dbs05-03-ConceptualModeling2-2.pdf> (besucht am 2025-03-27) (siehe S. **5–14**).
- [77] 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. **5–14, 85–97**).
- [78] 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).
- [79] Heinz Schweppe und Manuel Scholz. "Normalization: Quality of Relational Designs". In: *Einführung in die Datenbanksysteme. Datenbanken für die Bioinformatik*. Berlin, Germany: Freie Universität Berlin, Apr.–Okt. 2005. Kap. 5. URL: <https://www.inf.fu-berlin.de/lehre/SS05/19517-V/FolienEtc/dbs05-07-FA-1-2.pdf> (besucht am 2025-05-06) (siehe S. **46–53**).
- [80] Matthias Sedlmeier und Martin Gogolla. "Model Driven ActiveRecord with yEd". In: *25th International Conference on Information Modelling and Knowledge Bases XXVII (EJC'2015)*. 8.–12. Juni 2015, Maribor, Štajerska, Podravska, Slovenia. Hrsg. von Tatjana Welzer, Hannu Jaakkola, Bernhard Thalheim, Yasushi Kiyoki und Naofumi Yoshida. Bd. 280 der Reihe Frontiers in Artificial Intelligence and Applications. Amsterdam, The Netherlands: IOS Press BV, 2015, S. 65–76. ISSN: **0922-6389**. ISBN: **978-1-61499-610-1**. doi:[10.3233/978-1-61499-611-8-65](https://doi.org/10.3233/978-1-61499-611-8-65) (siehe S. **98–111, 258**).

References X



- [81] 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. **255**).
- [82] Yakov Shafranovich. *Common Format and MIME Type for Comma-Separated Values (CSV) Files*. Request for Comments (RFC) 4180. Wilmington, DE, USA: Internet Engineering Task Force (IETF), Okt. 2005. URL: <https://www.ietf.org/rfc/rfc4180.txt> (besucht am 2025-02-05) (siehe S. **254**).
- [83] 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. **5–14, 85–97, 254**).
- [84] Yuriy Shamshin. *Databases*. Riga, Latvia: ISMA University of Applied Sciences, Mai 2024. URL: <https://dbs.academy.lv> (besucht am 2025-01-11).
- [85] Yuriy Shamshin. "Logical Data Models. Relation Model. Relation Algebra". In: *Databases*. Riga, Latvia: ISMA University of Applied Sciences, Mai 2024. Kap. 05. URL: https://dbs.academy.lv/lection/dbs_LS05EN_rm.pdf (besucht am 2025-04-10) (siehe S. **32–40**).
- [86] 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. **255**).
- [87] 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](https://doi.org/10.1145/361020.361025) (siehe S. **256**).
- [88] "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. **257**).
- [89] 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. **251, 257**).

References XI



- [90] 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 ⁸⁹ (siehe S. 257).
- [91] Philip D. Straffin Jr. "Liu Hui and the First Golden Age of Chinese Mathematics". *Mathematics Magazine* 71(3):163–181, Juni 1998. London, England, UK: Taylor and Francis Ltd. ISSN: 0025-570X. doi:10.2307/2691200. URL: <https://www.researchgate.net/publication/237334342> (besucht am 2024-08-10) (siehe S. 159–172).
- [92] Allen Taylor. *Introducing SQL and Relational Databases*. New York, NY, USA: Apress Media, LLC, Sep. 2018. ISBN: 978-1-4842-3841-7 (siehe S. 256, 257).
- [93] 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. 256).
- [94] 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. 255).
- [95] 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. 255).
- [96] *UML Distilled: A Brief Guide to the Standard Object Modeling Language*. 3. Aufl. The Addison-Wesley Object Technology Series. Reading, MA, USA: Addison-Wesley Professional, Sep. 2003. ISBN: 978-0-321-19368-1 (siehe S. 257).
- [97] *UML Notation Guide*. Version 1.1. Santa Clara, CA, USA: Rational Software Corporation, Redmond, WA, USA: Microsoft Corporation, Palo Alto, CA, USA: Hewlett-Packard Company, Redwood Shores, CA, USA: Oracle Corporation, Dallas, TX, USA: Sterling Software, Ottawa, ON, Canada: MCI Systemhouse Corporation, Blue Bell, PA, USA: Unisys Corporation, Blue Bell, PA, USA: ICON Computing, Santa Clara, CA, USA: IntelliCorp, Burlington, MA, USA: i-Logix, Armonk, NY, USA: International Business Machines Corporation (IBM), Kanata, ON, Canada: ObjecTime Limited, Chicago, IL, USA: Platinum Technology Inc., Boston, MA, USA: Ptech Inc., Orlando, FL, USA: Taskon A/S, Paoli, PA, USA: Reich Technologies und Paris, Île-de-France, France: Softeam, 1. Sep. 1997. URL: <https://web.cse.msu.edu/~cse870/Materials/uml-notation-guide-9-97.pdf> (besucht am 2025-03-30) (siehe S. 85–97, 257).

References XII



- [98] Sander van Vugt. *Linux Fundamentals*. 2. Aufl. Hoboken, NJ, USA: Pearson IT Certification, Juni 2022. ISBN: 978-0-13-792931-3 (siehe S. 255).
- [99] Scott L. Vandenberg. "Conceptual Design using the Entity-Relationship Model". In: *CSE 594: Database Management Systems*. Seattle, WA, USA: University of Washington, Herbst 1999. URL: <https://courses.cs.washington.edu/courses/csep544/99au/lectures/class2.pdf> (besucht am 2025-03-29) (siehe S. 5–14, 85–97).
- [100] 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. 54–61, 254–256).
- [101] 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. 16–25, 41–45, 54–61, 74–79, 256).
- [102] 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. 85–97, 254).
- [103] *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. 256).
- [104] Ulf Michael „Monty“ Widenius, David Axmark und Uppsala, Sweden: MySQL AB. *MySQL Reference Manual – Documentation from the Source*. Sebastopol, CA, USA: O'Reilly Media, Inc., 9. Juli 2002. ISBN: 978-0-596-00265-7 (siehe S. 256).
- [105] 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. 254).
- [106] *yEd Graph Editor Manual*. Tübingen, Baden-Württemberg, Germany: yWorks GmbH, 2011–2025. URL: <https://yed.yworks.com/support/manual/index.html> (besucht am 2025-03-31) (siehe S. 98–111, 258).

References XIII



- [107] Wenqi Ying (应雯棋), Hrsg. *Commemoration of Ancient Chinese Mathematical Master Liu Hui for his Timeless Influence on Mathematics and Civilizational Exchange*. Bd. 48 (Special Issue) der Reihe CAST Newsletter. China, Beijing (中国北京市): 中国科学技术协会 (China Association for Science and Technology, CAST), Nov. 2024. URL: https://english.cast.org.cn/cms_files/filemanager/1941250207/attach/202412/8f23655a82364d19ad7874eb37b23035.pdf (besucht am 2025-08-24). Proofreader: Yumeng Wei (魏雨萌), Designer: Shan Zhang (张珊) (siehe S. 159–172).
- [108] 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. 255, 256).
- [109] Giorgio Zarrelli. *Mastering Bash*. Birmingham, England, UK: Packt Publishing Ltd, Juni 2017. ISBN: 978-1-78439-687-9 (siehe S. 254).
- [110] 公民身份号码 (Citizen Identification Number). 中华人民共和国国家标准 (National Standard of the People's Republic of China, GB) GB11643-1999. China, Beijing (中国北京市): 中华人民共和国国家质量监督检验检疫总局 (General Administration of Quality Supervision, Inspection and Quarantine of the People's Republic of China), 中国国家标准化管理委员会 (Standardization Administration of the People's Republic of China, SAC) und 中国标准出版社 (Standards Press of China), 19. Jan.–3. Nov. 1999. URL: <https://openstd.samr.gov.cn/bzgk/gb/newGbInfo?hcno=080D6FBF2BB468F9007657F26D60013E> (besucht am 2024-07-26) (siehe S. 113–117, 194–196).

Glossary (in English) I



Bash is a the shell used under Ubuntu Linux, i.e., the program that „runs“ in the terminal and interprets your commands, allowing you to start and interact with other programs^{12,65,109}. Learn more at <https://www.gnu.org/software/bash>.

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

client-server architecture is a system design where a central server receives requests from one or multiple clients^{7,59,69,71,73}. These requests and responses are usually sent over network connections. A typical example for such a system is the World Wide Web (WWW), where web servers host websites and make them available to web browsers, the clients. Another typical example is the structure of database (DB) software, where a central server, the DBMS, offers access to the DB to the different clients. Here, the client can be some terminal software shipping with the DBMS, such as `psql`, or the different applications that access the DBs.

CSV *Comma-Separated Values* is a very common and simple text format for exchanging tabular or matrix data⁸². Each row in the text file represents one row in the table or matrix. The elements in the row are separated by a fixed delimiter, usually a comma („,"), sometimes a semicolon („;"). Python offers some out-of-the-box CSV support in the `csv` module²⁹.

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*¹⁰⁰.

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 DB¹⁰⁵.

DOB Date of Birth

ERD Entity relationship diagrams show the relationships between objects, e.g., between the tables in a DB and how they reference each other^{3,16,22–24,52,83,102}

Glossary (in English) II



- IDEF1X** The Integration Definition for Information Modeling (IDEF1X) is a standardized syntax for entity relationship diagrams (ERDs). It was originally developed by the United States Air Force for data modeling, focusing on entities, relationships, and key structures^{14,51}.
- Inkscape** is a free and open source vector graphics editor, which primarily works with the Scalable Vector Graphics (SVG) format^{53,74}. This is the tool that I would recommend for professional graphic design. Learn more at <https://inkscape.org>.
- LAMP Stack** A system setup for web applications: Linux, Apache (a web server), MySQL, and the server-side scripting language PHP^{18,48}.
- LibreOffice** is on open source office suite^{41,58,81} which is a good and free alternative to Microsoft Office. It offers software such as LibreOffice Writer, LibreOffice Calc, and LibreOffice Base. See¹⁰⁰ 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^{39,81}. It is part of LibreOffice^{41,58,81} and has functionality that is comparable to Microsoft Access^{5,25,95}.
- 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^{58,81}, i.e., an alternative to Microsoft Excel. It is part of LibreOffice^{41,58,81}.
- LibreOffice Writer** is a free and open source text writing program¹⁰⁸ and part of LibreOffice^{41,58,81}. It is a good alternative to Microsoft Word.
- Linux** is the leading open source operating system, i.e., a free alternative for Microsoft Windows^{4,46,86,94,98}. 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.
- Microsoft Access** is a DBMS that can work on DBs stored in single, stand-alone files but also connect to other popular relational databases^{5,25,61,95}. 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^{8,43,56}. It is part of Microsoft Office. A free alternative to this commercial software is LibreOffice Calc^{58,81}.

Glossary (in English) III



- Microsoft Office is a commercial suite of office software, including Microsoft Excel, Microsoft Word, and Microsoft Access⁵⁶. LibreOffice is a free and open source alternative.
- Microsoft Windows is a commercial proprietary operating system¹¹. It is widely spread, but we recommend using a Linux variant such as Ubuntu for software development and for our course. Learn more at <https://www.microsoft.com/windows>.
- Microsoft Word is one of the leading text writing programs^{35,64,108} and part of Microsoft Office. A free alternative to this commercial software is the LibreOffice Writer.
- MySQL An open source relational database management system^{10,36,72,93,104}. MySQL is famous for its use in the LAMP Stack. See <https://www.mysql.com> for more information.
- MySQL Workbench is a visual tool for DB designers that offers tools ranging from graphical modeling to performance analysis⁶². Learn more at <https://www.mysql.com/products/workbench>.
- OOP Object-Oriented Programming⁷⁵
- PgModeler the PostgreSQL DB modeler is a tool that allows for graphical modeling of logical schemas for DBs using an ERD-like notation¹. Learn more at <https://pgmodeler.io>.
- PostgreSQL An open source object-relational DBMS^{40,67,70,93}. See <https://postgresql.org> for more information.
- psql is the client program used to access the PostgreSQL DBMS server.
- Python The Python programming language^{49,57,60,101}, i.e., what you will learn about in our book¹⁰¹. Learn more at <https://python.org>.
- relational database A relational DB is a database that organizes data into rows (tuples, records) and columns (attributes), which collectively form tables (relations) where the data points are related to each other^{27,44,45,87,92,100,103}.

Glossary (in English) IV



server In a client-server architecture, the server is a process that fulfills the requests of the clients. It usually waits for incoming communication carrying the requests from the clients. For each request, it takes the necessary actions, performs the required computations, and then sends a response with the result of the request. Typical examples for servers are web servers¹⁸ in the internet as well as DBMSes. It is also common to refer to the computer running the server processes as server as well, i.e., to call it the „server computer“⁵⁵.

SQL The *Structured Query Language* is basically a programming language for querying and manipulating relational databases^{20,31–33,50,63,88–90,92}. It is understood by many DBMSes. You find the Structured Query Language (SQL) commands supported by PostgreSQL in the reference⁸⁸.

SVG The Scalable Vector Graphics (SVG) format is an Extensible Markup Language (XML)-based format for vector graphics³⁰. Vector graphics are composed of geometric shapes like lines, rectangles, circles, and text. As opposed to raster / pixel graphics, they can be scaled seamlessly and without artifacts. They are stored losslessly.

terminal A terminal is a text-based window where you can enter commands and execute them^{4,26}. Knowing what a terminal is and how to use it is very essential in any programming- or system administration-related task. If you want to open a terminal under Microsoft Windows, you can Druck auf  + , dann Schreiben von `cmd`, dann Druck auf . Under Ubuntu Linux,  +  +  opens a terminal, which then runs a Bash shell inside.

Ubuntu is a variant of the open source operating system Linux^{26,48}. We recommend that you use this operating system to follow this class, for software development, and for research. Learn more at <https://ubuntu.com>. If you are in China, you can download it from <https://mirrors.ustc.edu.cn/ubuntu-releases>.

UML The Unified Modeling Language (UML) is a graphical language for visualizing, specifying, constructing, and documenting the artifacts of distributed object systems^{9,68,96,97}

WWW World Wide Web^{6,34}

Glossary (in English) V



XML The *Extensible Markup Language* is a text-based language for storing and transporting of data^{13,28,54}. It allows you to define elements in the form `<myElement myAttr="x">...text...</myElement>`. Different from comma-separated values (CSV), elements in XML can be hierarchically nested, like `<a><b><c>test</c></b><b>bla</b></a>`, and thus easily represent tree structures. XML is one of most-used data interchange formats. To process XML in Python, use the `defusedxml` library⁴⁷, as it protects against several security issues.

yEd is a graph editor for high-quality graph-based diagrams^{80,106}, suitable to draw, e.g., technology-independent ERDs, control flow charts, or Unified Modeling Language (UML) class diagrams. An online version of the editor is available at <https://www.yworks.com/yed-live>. Learn more at <https://www.yworks.com/products/yed>.

dom(*a*) the domain of an attribute *a*.