合肥大學
HEFEI UNIVERSITY

Datenbanken

31. Konzeptuelles Schema: Beziehungskardinalität

Thomas Weise (汤卫思)
tweise@hfuu.edu.cn

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

应用优化研究所
人工智能与大数据学院
合肥大学
中国安徽省合肥市

Version: 2025-11-19

Databases

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

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

Outline

Einleitung

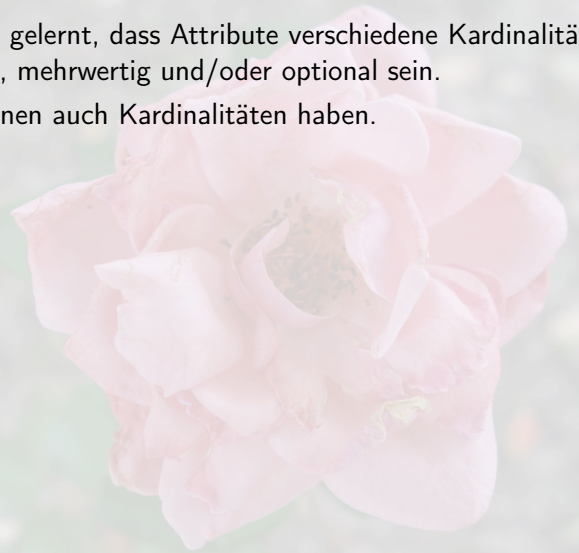
Einleitung

- Wir haben schon gelernt, dass Attribute verschiedene Kardinalitäten haben können.

Einleitung

- Wir haben schon gelernt, dass Attribute verschiedene Kardinalitäten haben können: Sie können einwertig, mehrwertig und/oder optional sein.

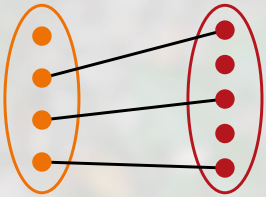
Einleitung



- Wir haben schon gelernt, dass Attribute verschiedene Kardinalitäten haben können: Sie können einwertig, mehrwertig und/oder optional sein.
- Beziehungen können auch Kardinalitäten haben.

Einleitung

- Wir haben schon gelernt, dass Attribute verschiedene Kardinalitäten haben können: Sie können einwertig, mehrwertig und/oder optional sein.
- Beziehungen können auch Kardinalitäten haben: wir unterscheiden oft eins-zu-eins Beziehungen.

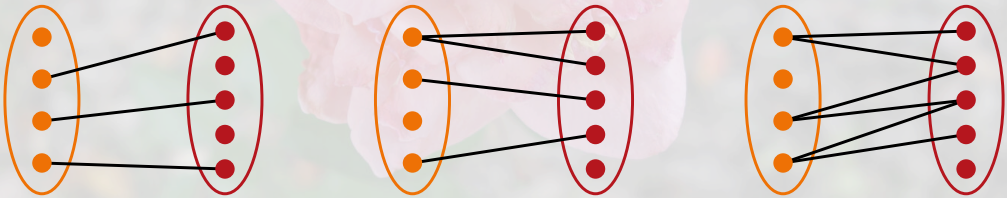


Einleitung

- Wir haben schon gelernt, dass Attribute verschiedene Kardinalitäten haben können: Sie können einwertig, mehrwertig und/oder optional sein.
- Beziehungen können auch Kardinalitäten haben: wir unterscheiden oft eins-zu-eins und eins-zu-mehreren Beziehungen.

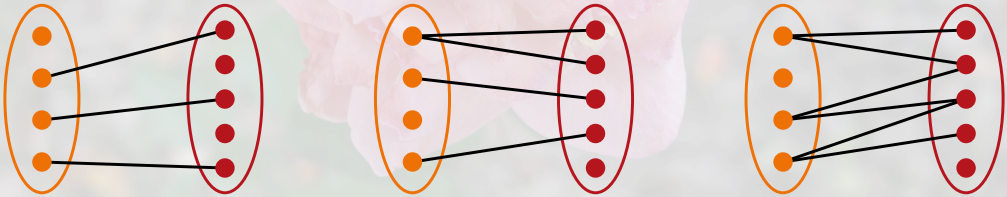
Einleitung

- Wir haben schon gelernt, dass Attribute verschiedene Kardinalitäten haben können: Sie können einwertig, mehrwertig und/oder optional sein.
- Beziehungen können auch Kardinalitäten haben: wir unterscheiden oft eins-zu-eins, eins-zu-mehreren und mehrere-zu-mehrere Beziehungen.



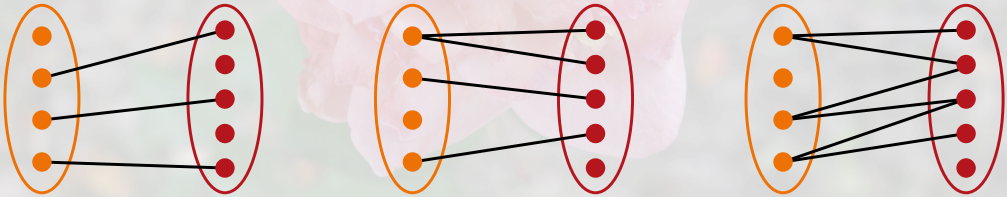
Einleitung

- Wir haben schon gelernt, dass Attribute verschiedene Kardinalitäten haben können: Sie können einwertig, mehrwertig und/oder optional sein.
- Beziehungen können auch Kardinalitäten haben: wir unterscheiden oft eins-zu-eins, eins-zu-mehreren und mehrere-zu-mehrere Beziehungen.
- In Chen's ursprünglicher ERD Notation[21] werden die Enden einer Beziehung mit 1, N, oder M annotiert, umd 1:1, 1:N, oder M:N Beziehungen darzustellen.



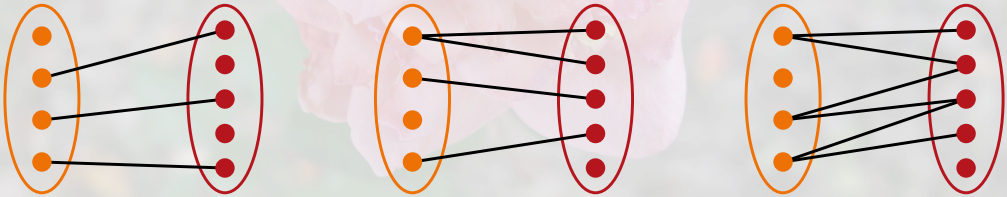
Einleitung

- Wir haben schon gelernt, dass Attribute verschiedene Kardinalitäten haben können: Sie können einwertig, mehrwertig und/oder optional sein.
- Beziehungen können auch Kardinalitäten haben: wir unterscheiden oft eins-zu-eins, eins-zu-mehreren und mehrere-zu-mehrere Beziehungen.
- In Chen's ursprünglicher ERD Notation[21] werden die Enden einer Beziehung mit 1, N, oder M annotiert, umd 1:1, 1:N, oder M:N Beziehungen darzustellen.
- In der Notation von Bachman[3] aus 1969 wird eine Beziehung mit einem Pfeil von einer Entität zu einer anderen dargestellt.



Einleitung

- Wir haben schon gelernt, dass Attribute verschiedene Kardinalitäten haben können: Sie können einwertig, mehrwertig und/oder optional sein.
- Beziehungen können auch Kardinalitäten haben: wir unterscheiden oft eins-zu-eins, eins-zu-mehreren und mehrere-zu-mehrere Beziehungen.
- In Chen's ursprünglicher ERD Notation[21] werden die Enden einer Beziehung mit 1, N, oder M annotiert, umd 1:1, 1:N, oder M:N Beziehungen darzustellen.
- In der Notation von Bachman[3] aus 1969 wird eine Beziehung mit einem Pfeil von einer Entität zu einer anderen dargestellt. Die Entität, auf die der Pfeil zeigt, taucht N mal auf, die anderen einmal.



Modalität und Kardinalität

Beziehungsmodalität (1)

- Wenn wir ein ERD erstellen, wollen wir hohe Genauigkeit.

Beziehungsmodalität (1)

- Wenn wir ein ERD erstellen, wollen wir hohe Genauigkeit.
- Wir wollen ausdrücken, ob Beziehungen optional oder zwingend (EN: *mandatory, required*) sind.

Beziehungsmodalität (1)

- Wenn wir ein ERD erstellen, wollen wir hohe Genauigkeit.
- Wir wollen ausdrücken, ob Beziehungen optional oder zwingend (EN: *mandatory, required*) sind.
- Deshalb werden die Enden von Beziehungen in ERDs oft mit einer Modalität (EN: *modality*) (optional?) und einer Kardinalität (EN: *cardinality*) (wie viel?) annotiert.[72].

Beziehungsmodalität (1)

- Wenn wir ein ERD erstellen, wollen wir hohe Genauigkeit.
- Wir wollen ausdrücken, ob Beziehungen optional oder zwingend (EN: *mandatory, required*) sind.
- Deshalb werden die Enden von Beziehungen in ERDs oft mit einer Modalität (EN: *modality*) (optional?) und einer Kardinalität (EN: *cardinality*) (wie viel?) annotiert.[72].

Definition: Beziehungsmodalität

Die *Modalität* (EN: *modality*) eines Beziehungendes definiert, ob die Teilnahme an der Beziehung optional oder erforerlich ist.

Beziehungsmodalität (1)

- Wenn wir ein ERD erstellen, wollen wir hohe Genauigkeit.
- Wir wollen ausdrücken, ob Beziehungen optional oder zwingend (EN: *mandatory, required*) sind.
- Deshalb werden die Enden von Beziehungen in ERDs oft mit einer Modalität (EN: *modality*) (optional?) und einer Kardinalität (EN: *cardinality*) (wie viel?) annotiert.[72].

Definition: Beziehungsmodalität

Die *Modalität* (EN: *modality*) eines Beziehungendes definiert, ob die Teilnahme an der Beziehung optional oder erforerlich ist. Man kann das als die Minimalzahl der teilnehmenden Entities interpretieren.

Beziehungsmodalität (2)

- Aus praktischer Sicht sind nur die Minimal-Teilnehmerzahlen 0 and 1 wirklich relevant.

Beziehungsmodalität (2)

- Aus praktischer Sicht sind nur die Minimal-Teilnehmerzahlen 0 and 1 wirklich relevant.
- Eine Modalität von „optionale Teilnahme“ bedeutet eine minimale Teilnehmerzahl von 0.

Beziehungsmodalität (2)

- Aus praktischer Sicht sind nur die Minimal-Teilnehmerzahlen 0 and 1 wirklich relevant.
- Eine Modalität von „optionale Teilnahme“ bedeutet eine minimale Teilnehmerzahl von 0.
- Eine Modalität von „erforderliche Teilnahme“ bedeutet eine minimale Teilnehmerzahl von 1.

Beziehungsmodalität (2)

- Aus praktischer Sicht sind nur die Minimal-Teilnehmerzahlen 0 and 1 wirklich relevant.
- Eine Modalität von „optionale Teilnahme“ bedeutet eine minimale Teilnehmerzahl von 0.
- Eine Modalität von „erforderliche Teilnahme“ bedeutet eine minimale Teilnehmerzahl von 1.
- So können wir totale und partielle Beziehungen unterscheiden[70,95].

Beziehungsmodalität (2)

- Aus praktischer Sicht sind nur die Minimal-Teilnehmerzahlen 0 and 1 wirklich relevant.
- Eine Modalität von „optionale Teilnahme“ bedeutet eine minimale Teilnehmerzahl von 0.
- Eine Modalität von „erforderliche Teilnahme“ bedeutet eine minimale Teilnehmerzahl von 1.
- So können wir totale und partielle Beziehungen unterscheiden[70,95].
- Wenn alle Entitäten einer Entitätsmenge an einer Beziehung teilnehmen *müssen*, dann ist die Beziehung total.

Beziehungsmodalität (2)

- Aus praktischer Sicht sind nur die Minimal-Teilnehmerzahlen 0 and 1 wirklich relevant.
- Eine Modalität von „optionale Teilnahme“ bedeutet eine minimale Teilnehmerzahl von 0.
- Eine Modalität von „erforderliche Teilnahme“ bedeutet eine minimale Teilnehmerzahl von 1.
- So können wir totale und partielle Beziehungen unterscheiden[70,95].
- Wenn alle Entitäten einer Entitätsmenge an einer Beziehung teilnehmen *müssen*, dann ist die Beziehung total.
- Wenn nur einige der Eintäten an der Beziehung teilnehmen, dann ist die Beziehung partiell.

Beziehungskardinalität

Definition: Beziehungskardinalität

Die Maximalzahl der teilnehmenden Entitäten eines Beziehungendes wird als die *Kardinalität* (EN: *cardinality*) bezeichnet.

Beziehungskardinalität

Definition: Beziehungskardinalität

Die Maximalzahl der teilnehmenden Entitäten eines Beziehungendes wird als die *Kardinalität* (EN: *cardinality*) bezeichnet.

- Praktisch wird meist nur zwischen den Kardinalitäten *eins* und *mehrere*, also unbegrenzt, unterschieden.

Beziehungskardinalität

Definition: Beziehungskardinalität

Die Maximalzahl der teilnehmenden Entitäten eines Beziehungendes wird als die *Kardinalität* (EN: *cardinality*) bezeichnet.

- Praktisch wird meist nur zwischen den Kardinalitäten *eins* und *mehrere*, also unbegrenzt, unterschieden.
- In der ähnlichen Modellierungssprache UML bezeichnet man die Kardinalität als Multiplizität (EN: *multiplicity*).

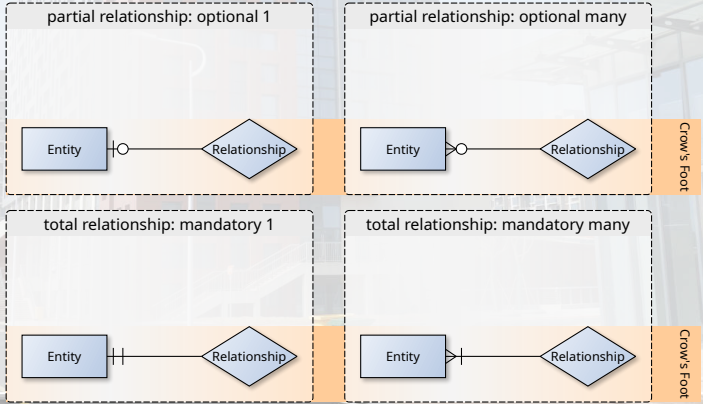
Modellierungskonventionen

Modellierungskonventionen (1)

- Es gibt mehrere Konventionen, wie die Modalität und Kardinalität von Beziehungsenden in eine ERD modelliert werden kann.

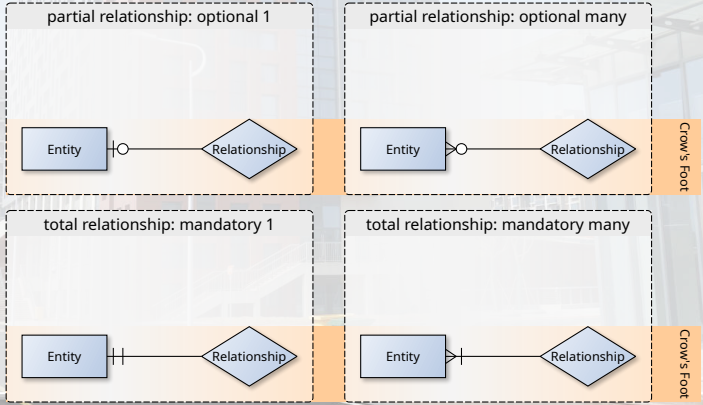
Modellierungskonventionen (1)

- Es gibt mehrere Konventionen, wie die Modalität und Kardinalität von Beziehungsenden in eine ERD modelliert werden kann.
- Die Krähenfußnotation (EN: *crow's foot notation*) verwendet graphische Zeichen[17,35,79].



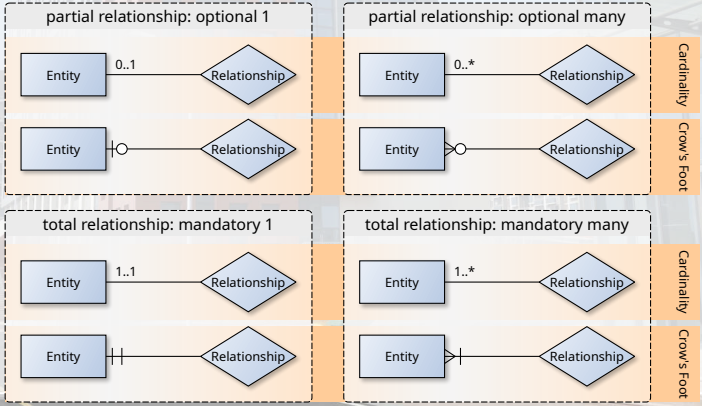
Modellierungskonventionen (1)

- Es gibt mehrere Konventionen, wie die Modalität und Kardinalität von Beziehungsenden in eine ERD modelliert werden kann.
- Die Krähenfußnotation (EN: *crow's foot notation*) verwendet graphische Zeichen[17,35,79].
- Sie benutzt die vier Symbole ⟙, ⟜, ⟛ und ⟝ um optional-eins und -mehrere bzw. erzwungen-eins und -mehrere Teilnehmer darzustellen.



Modellierungskonventionen (1)

- Es gibt mehrere Konventionen, wie die Modalität und Kardinalität von Beziehungsenden in eine ERD modelliert werden kann.
- Die Krähenfußnotation (EN: *crow's foot notation*) verwendet graphische Zeichen[17,35,79].
- Wir können die Minimal- und Maximalzahl der erlaubten Teilnehmer auch direkt als Labels hinschreiben[70].



Modellierungskonventionen (1)

- Es gibt mehrere Konventionen, wie die Modalität und Kardinalität von Beziehungsenden in eine ERD modelliert werden kann.
- Die Krähenfußnotation (EN: *crow's foot notation*) verwendet graphische Zeichen[17,35,79].
- Wir können die Minimal- und Maximalzahl der erlaubten Teilnehmer auch direkt als Labels hinschreiben[70].
- Ein Ganzzahlintervall $i..j$ bedeutet, dass mindestens i und höchstens j Entitäten teilnehmen, wobei $j = *$ für unbegrenzt steht.

Modellierungskonventionen (1)

- Es gibt mehrere Konventionen, wie die Modalität und Kardinalität von Beziehungsenden in eine ERD modelliert werden kann.
- Die Krähenfußnotation (EN: *crow's foot notation*) verwendet graphische Zeichen[17,35,79].
- Sie benutzt die vier Symbole  ,  ,  und um optional-eins und -mehrere bzw. erzwungen-eins und -mehrere Teilnehmer darzustellen.
- Wir können die Minimal- und Maximalzahl der erlaubten Teilnehmer auch direkt als Labels hinschreiben[70].
- Ein Ganzzahlintervall $i..j$ bedeutet, dass mindestens i und höchstens j Entitäten teilnehmen, wobei $j = *$ für unbegrenzt steht.
- Die zweite Methode hat den Vorteil, dass wir damit mehr verschiedene Kardinalitäten modellieren können.

Modellierungskonventionen (1)

- Es gibt mehrere Konventionen, wie die Modalität und Kardinalität von Beziehungsenden in eine ERD modelliert werden kann.
- Die Krähenfußnotation (EN: *crow's foot notation*) verwendet graphische Zeichen[17,35,79].
- Sie benutzt die vier Symbole  ,  ,  und  um optional-eins und -mehrere bzw. erzwungen-eins und -mehrere Teilnehmer darzustellen.
- Wir können die Minimal- und Maximalzahl der erlaubten Teilnehmer auch direkt als Labels hinschreiben[70].
- Ein Ganzzahlintervall $i..j$ bedeutet, dass mindestens i und höchstens j Entitäten teilnehmen, wobei $j = *$ für unbegrenzt steht.
- Die zweite Methode hat den Vorteil, dass wir damit mehr verschiedene Kardinalitäten modellieren können.
- Der Nachteil ist, dass sie etwas schwerer zu lesen ist, wenn man sie ausdruckt oder von weiter weg anguckt.

Modellierungskonventionen (1)

- Es gibt mehrere Konventionen, wie die Modalität und Kardinalität von Beziehungsenden in eine ERD modelliert werden kann.
- Die Krähenfußnotation (EN: *crow's foot notation*) verwendet graphische Zeichen[17,35,79].
- Sie benutzt die vier Symbole  ,  ,  und um optional-eins und -mehrere bzw. erzwungen-eins und -mehrere Teilnehmer darzustellen.
- Wir können die Minimal- und Maximalzahl der erlaubten Teilnehmer auch direkt als Labels hinschreiben[70].
- Ein Ganzzahlintervall $i..j$ bedeutet, dass mindestens i und höchstens j Entitäten teilnehmen, wobei $j = *$ für unbegrenzt steht.
- Die zweite Methode hat den Vorteil, dass wir damit mehr verschiedene Kardinalitäten modellieren können.
- Der Nachteil ist, dass sie etwas schwerer zu lesen ist, wenn man sie ausdruckt oder von weiter weg anguckt.
- Sie ist auch schwieriger zu malen, weil wir manuell Labels einfügen müssen.

Modellierungskonventionen (1)

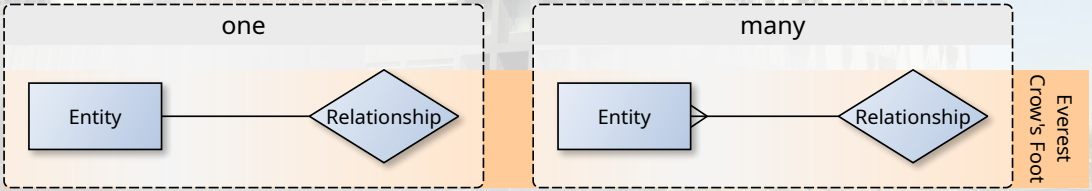
- Es gibt mehrere Konventionen, wie die Modalität und Kardinalität von Beziehungsenden in eine ERD modelliert werden kann.
- Die Krähenfußnotation (EN: *crow's foot notation*) verwendet graphische Zeichen[17,35,79].
- Sie benutzt die vier Symbole  ,  ,  und  um optional-eins und -mehrere bzw. erzwungen-eins und -mehrere Teilnehmer darzustellen.
- Wir können die Minimal- und Maximalzahl der erlaubten Teilnehmer auch direkt als Labels hinschreiben[70].
- Ein Ganzzahlintervall $i..j$ bedeutet, dass mindestens i und höchstens j Entitäten teilnehmen, wobei $j = *$ für unbegrenzt steht.
- Die zweite Methode hat den Vorteil, dass wir damit mehr verschiedene Kardinalitäten modellieren können.
- Der Nachteil ist, dass sie etwas schwerer zu lesen ist, wenn man sie ausdruckt oder von weiter weg anguckt.
- Sie ist auch schwieriger zu malen, weil wir manuell Labels einfügen müssen.
- Dazu kommt, dass kompliziertere Kardinalitäten sowieso selten gebraucht werden.

Modellierungskonventionen (1)

- Es gibt mehrere Konventionen, wie die Modalität und Kardinalität von Beziehungsenden in eine ERD modelliert werden kann.
- Die Krähenfußnotation (EN: *crow's foot notation*) verwendet graphische Zeichen[17,35,79].
- Sie benutzt die vier Symbole  ,  ,  und  um optional-eins und -mehrere bzw. erzwungen-eins und -mehrere Teilnehmer darzustellen.
- Wir können die Minimal- und Maximalzahl der erlaubten Teilnehmer auch direkt als Labels hinschreiben[70].
- Ein Ganzzahlintervall $i..j$ bedeutet, dass mindestens i und höchstens j Entitäten teilnehmen, wobei $j = *$ für unbegrenzt steht.
- Die zweite Methode hat den Vorteil, dass wir damit mehr verschiedene Kardinalitäten modellieren können.
- Der Nachteil ist, dass sie etwas schwerer zu lesen ist, wenn man sie ausdruckt oder von weiter weg anguckt.
- Sie ist auch schwieriger zu malen, weil wir manuell Labels einfügen müssen.
- Dazu kommt, dass kompliziertere Kardinalitäten sowieso selten gebraucht werden.
- So etwas wie 7..11 könnte auch später schwierig zu implementieren und durchzusetzen

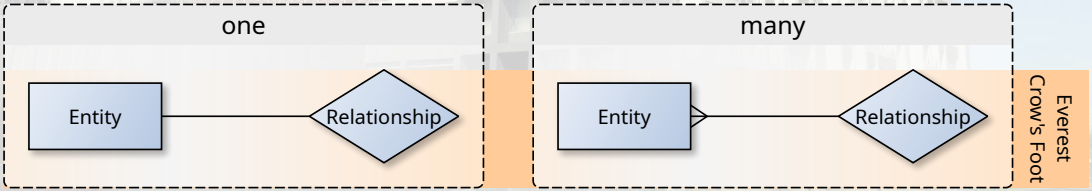
Modellierungskonventionen (2)

- Die ursprüngliche Krähenfußnotation von Everest aus 1976 hatte noch keine Symbole für optional/erforderlich[35].



Modellierungskonventionen (2)

- Die ursprüngliche Krähenfußnotation von Everest aus 1976 hatte noch keine Symbole für optional/erforderlich[35].
- Man kann sie immer noch verwenden, wenn die Modalität nicht so wichtig ist oder wenn man sie in Diskussionen später festlegen will.



Modellierungskonventionen (2)

- Die ursprüngliche Krähenfußnotation von Everest aus 1976 hatte noch keine Symbole für optional/erforderlich[35].
- Man kann sie immer noch verwenden, wenn die Modalität nicht so wichtig ist oder wenn man sie in Diskussionen später festlegen will.
- In LibreOffice Base kann man ERDs malen, die direkt mit der darunterliegenden Datenbank verbunden sind.

Modellierungskonventionen (2)

- Die ursprüngliche Krähenfußnotation von Everest aus 1976 hatte noch keine Symbole für optional/erforderlich[35].
- Man kann sie immer noch verwenden, wenn die Modalität nicht so wichtig ist oder wenn man sie in Diskussionen später festlegen will.
- In LibreOffice Base kann man ERDs malen, die direkt mit der darunterliegenden Datenbank verbunden sind.
- Hier benutzt man die 1:1/1:n Notation und ein kleines „n“ steht für „mehrere“.

Modellierungskonventionen (2)

- Die ursprüngliche Krähenfußnotation von Everest aus 1976 hatte noch keine Symbole für optional/erforderlich[35].
- Man kann sie immer noch verwenden, wenn die Modalität nicht so wichtig ist oder wenn man sie in Diskussionen später festlegen will.
- In LibreOffice Base kann man ERDs malen, die direkt mit der darunterliegenden Datenbank verbunden sind.
- Hier benutzt man die 1:1/1:n Notation und ein kleines „n“ steht für „mehrere“.
- Microsoft Access offers bietet eine ähnliche Funktionalität, aber die Beziehungsenden sind mit 1 oder ∞ annotiert.

Modellierungskonventionen (2)

- Die ursprüngliche Krähenfußnotation von Everest aus 1976 hatte noch keine Symbole für optional/erforderlich[35].
- Man kann sie immer noch verwenden, wenn die Modalität nicht so wichtig ist oder wenn man sie in Diskussionen später festlegen will.
- In LibreOffice Base kann man ERDs malen, die direkt mit der darunterliegenden Datenbank verbunden sind.
- Hier benutzt man die 1:1/1:n Notation und ein kleines „n“ steht für „mehrere“.
- Microsoft Access offers bietet eine ähnliche Funktionalität, aber die Beziehungsenden sind mit 1 oder ∞ annotiert.
- Wir werden die Krähenfußnotation für Beziehungsenden verwenden, weil sie in yEd einfacher zu malen ist.

Modellierungskonventionen (2)

- Die ursprüngliche Krähenfußnotation von Everest aus 1976 hatte noch keine Symbole für optional/erforderlich[35].
- Man kann sie immer noch verwenden, wenn die Modalität nicht so wichtig ist oder wenn man sie in Diskussionen später festlegen will.
- In LibreOffice Base kann man ERDs malen, die direkt mit der darunterliegenden Datenbank verbunden sind.
- Hier benutzt man die 1:1/1:n Notation und ein kleines „n“ steht für „mehrere“.
- Microsoft Access offers bietet eine ähnliche Funktionalität, aber die Beziehungsenden sind mit 1 oder ∞ annotiert.
- Wir werden die Krähenfußnotation für Beziehungsenden verwenden, weil sie in yEd einfacher zu malen ist.
- Wir behalten aber Chen's Notation für die Symbole bei.

Krähenfußnotation

Krähenfußnotation

- Wie gesagt, wir nehmen die Krähenfußnotation.

Krähenfußnotation

- Wie gesagt, wir nehmen die Krähenfußnotation.
- Wir werden uns alle möglichen Kombinationen von „Beziehungsenden“ anschauen.

Krähenfußnotation

- Wie gesagt, wir nehmen die Krähenfußnotation.
- Wir werden uns alle möglichen Kombinationen von „Beziehungsenden“ anschauen.
- Es gibt vier Möglichkeiten, ein Ende einer Beziehung zu annotieren.

Krähenfußnotation

- Wie gesagt, wir nehmen die Krähenfußnotation.
- Wir werden uns alle möglichen Kombinationen von „Beziehungsenden“ anschauen.
- Es gibt vier Möglichkeiten, ein Ende einer Beziehung zu annotieren:
  - Optional 1: , äquivalent zu 0..1.

Krähenfußnotation

- Wie gesagt, wir nehmen die Krähenfußnotation.
- Wir werden uns alle möglichen Kombinationen von „Beziehungsenden“ anschauen.
- Es gibt vier Möglichkeiten, ein Ende einer Beziehung zu annotieren:
  - Optional 1: ⟊○–, äquivalent zu 0..1,
  - Erforderlich 1: ⟊⟊–, äquivalent zu 1..1.

Krähenfußnotation

- Wie gesagt, wir nehmen die Krähenfußnotation.
- Wir werden uns alle möglichen Kombinationen von „Beziehungsenden“ anschauen.
- Es gibt vier Möglichkeiten, ein Ende einer Beziehung zu annotieren:
  - Optional 1: $\dashv\!\circ\!-$, äquivalent zu 0..1,
  - Erforderlich 1: $\dashv\!\dashv\!-$, äquivalent zu 1..1,
  - Optional Viele: $\rightarrowtail\!\circ\!-$, äquivalent zu 0..$*$, 0..N und 0..$\infty$.

Krähenfußnotation

- Wie gesagt, wir nehmen die Krähenfußnotation.
- Wir werden uns alle möglichen Kombinationen von „Beziehungsenden“ anschauen.
- Es gibt vier Möglichkeiten, ein Ende einer Beziehung zu annotieren:
  - Optional 1: $\dashv\!\circ\!-$, äquivalent zu 0..1,
  - Erforderlich 1: $\dashv\!\vdash$, äquivalent zu 1..1,
  - Optional Viele: $\rightarrowtail\!\circ\!-$, äquivalent zu 0..$*$, 0..N und 0..$\infty$ und
  - Erforderlich Viele: $\rightarrowtail\!\vdash$, äquivalent zu 1..$*$, 1..N, und 1..∞.

Krähenfußnotation

- Wie gesagt, wir nehmen die Krähenfußnotation.
- Wir werden uns alle möglichen Kombinationen von „Beziehungsenden“ anschauen.
- Es gibt vier Möglichkeiten, ein Ende einer Beziehung zu annotieren:
  - Optional 1: $\dashv\!\circ\!-$, äquivalent zu 0..1,
  - Erforderlich 1: $\dashv\!\vdash$, äquivalent zu 1..1,
  - Optional Viele: $\rightarrow\!\circ\!-$, äquivalent zu 0..$*$, 0..N und 0..∞ und
  - Erforderlich Viele: $\rightarrow\!\vdash$, äquivalent zu 1..$*$, 1..N, und 1..∞.
- Jede Beziehung hat zwei Enden, also gibt es $10 = \frac{(4+2-1)!}{2!*(4-1)!}$ mögliche binäre Beziehungen.

Krähenfußnotation

- Wie gesagt, wir nehmen die Krähenfußnotation.
- Wir werden uns alle möglichen Kombinationen von „Beziehungsenden“ anschauen.
- Es gibt vier Möglichkeiten, ein Ende einer Beziehung zu annotieren:
  - Optional 1: $\dashv\!\circ\!-$, äquivalent zu 0..1,
  - Erforderlich 1: $\dashv\!\vdash$, äquivalent zu 1..1,
  - Optional Viele: $\rightarrow\!\circ\!-$, äquivalent zu 0..$*$, 0..N und 0..∞ und
  - Erforderlich Viele: $\rightarrow\!\vdash$, äquivalent zu 1..$*$, 1..N, und 1..∞.
- Jede Beziehung hat zwei Enden, also gibt es $10 = \frac{(4+2-1)!}{2!*(4-1)!}$ mögliche binäre Beziehungen. (Es gibt $\frac{(m+k-1)!}{k!*(n-1)!}$ verschiedene k-Kombinationen mit Wiederholungen von n Elementen ohne Beachtung der Reihenfolge.)

Lesen einer Beziehung

- Es ist wichtig, zu verstehen, wie man die Kardinalitäten und Modalitäten von Beziehungsenden interpretiert.

Lesen einer Beziehung

- Es ist wichtig, zu verstehen, wie man die Kardinalitäten und Modalitäten von Beziehungsenden interpretiert.
- Das kann man sehr leicht falsch machen!

Lesen einer Beziehung

- Es ist wichtig, zu verstehen, wie man die Kardinalitäten und Modalitäten von Beziehungsenden interpretiert.
- Das kann man sehr leicht falsch machen!
- Die Teilnahme einer Entität an einer Beziehung basiert auf dem *anderen* Ende.

Lesen einer Beziehung

- Es ist wichtig, zu verstehen, wie man die Kardinalitäten und Modalitäten von Beziehungsenden interpretiert.
- Das kann man sehr leicht falsch machen!
- Die Teilnahme einer Entität an einer Beziehung basiert auf dem *anderen* Ende.
- Schauen wir uns die Beziehung K ⫲—o< L als Beispiel an.

Lesen einer Beziehung

- Es ist wichtig, zu verstehen, wie man die Kardinalitäten und Modalitäten von Beziehungsenden interpretiert.
- Das kann man sehr leicht falsch machen!
- Die Teilnahme einer Entität an einer Beziehung basiert auf dem *anderen* Ende.
- Schauen wir uns die Beziehung K ⫲—o< L als Beispiel an.
- Platzieren Sie zuerst den Finger auf dem „K“ und sagen: „Jedes K hat. . . “.

Lesen einer Beziehung

- Es ist wichtig, zu verstehen, wie man die Kardinalitäten und Modalitäten von Beziehungsenden interpretiert.
- Das kann man sehr leicht falsch machen!
- Die Teilnahme einer Entität an einer Beziehung basiert auf dem *anderen* Ende.
- Schauen wir uns die Beziehung K ⫲—○≪ L als Beispiel an.
- Platzieren Sie zuerst den Finger auf dem „K“ und sagen: „Jedes K hat. . . “.
- Dann schieben wir den Finger zum „L“ und sagen „. . . kein, ein, oder mehrere L.“[47].

Lesen einer Beziehung

- Es ist wichtig, zu verstehen, wie man die Kardinalitäten und Modalitäten von Beziehungsenden interpretiert.
- Das kann man sehr leicht falsch machen!
- Die Teilnahme einer Entität an einer Beziehung basiert auf dem *anderen* Ende.
- Schauen wir uns die Beziehung K ⫲—o< L als Beispiel an.
- Platzieren Sie zuerst den Finger auf dem „K“ und sagen: „Jedes K hat. . . “.
- Dann schieben wir den Finger zum „L“ und sagen „. . . kein, ein, oder mehrere L.“[47].
- Nun platzieren wir den Finger auf dem „L“ und sagen „Jedes L hat. . . “.

Lesen einer Beziehung

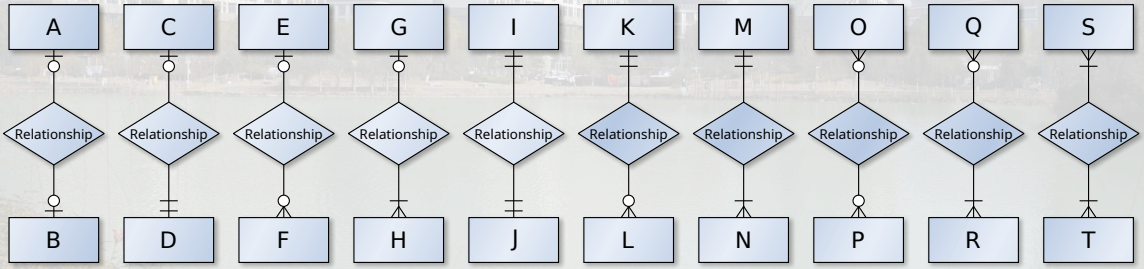
- Es ist wichtig, zu verstehen, wie man die Kardinalitäten und Modalitäten von Beziehungsenden interpretiert.
- Das kann man sehr leicht falsch machen!
- Die Teilnahme einer Entität an einer Beziehung basiert auf dem *anderen* Ende.
- Schauen wir uns die Beziehung K ⫲—o< L als Beispiel an.
- Platzieren Sie zuerst den Finger auf dem „K“ und sagen: „Jedes K hat. . . “.
- Dann schieben wir den Finger zum „L“ und sagen „. . . kein, ein, oder mehrere L.“[47].
- Nun platzieren wir den Finger auf dem „L“ und sagen „Jedes L hat. . . “.
- Wir schieben den Finger rüber zum „K“ und sagen „. . . genau ein K.“.

Lesen einer Beziehung

- Es ist wichtig, zu verstehen, wie man die Kardinalitäten und Modalitäten von Beziehungsenden interpretiert.
- Das kann man sehr leicht falsch machen!
- Die Teilnahme einer Entität an einer Beziehung basiert auf dem *anderen* Ende.
- Schauen wir uns die Beziehung K ⫲—o⫷ L als Beispiel an.
- Platzieren Sie zuerst den Finger auf dem „K“ und sagen: „Jedes K hat. . . “.
- Dann schieben wir den Finger zum „L“ und sagen „. . . kein, ein, oder mehrere L.“[47].
- Nun platzieren wir den Finger auf dem „L“ und sagen „Jedes L hat. . . “.
- Wir schieben den Finger rüber zum „K“ und sagen „. . . genau ein K.“.
- Beachten Sie, dass das ⫲— am K nicht bedeutet, dass jedes K mit einem L in Beziehung steht!

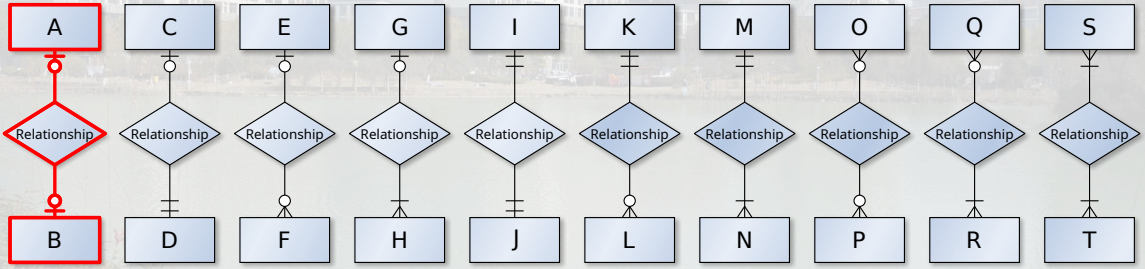
Mögliche Beziehungen in der Krähenfußnotation

- Schauen wir uns die zehn möglichen Beziehungen einmal an.



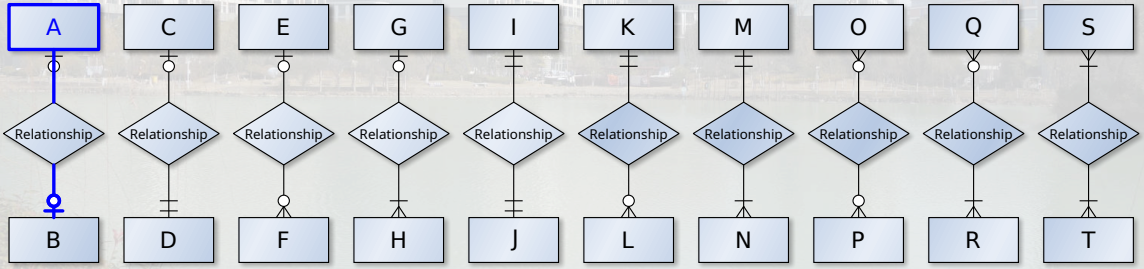
Mögliche Beziehungen in der Krähenfußnotation

- A ⟝○—○⟞ B



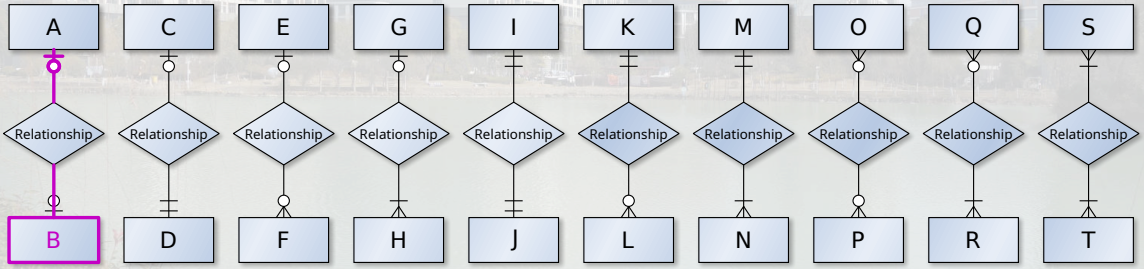
Mögliche Beziehungen in der Krähenfußnotation

- A ⟊○—○⟊ B:
  - Jede Entität vom Typ A kann mit keiner oder einer Entität vom Typ B verbunden sein.



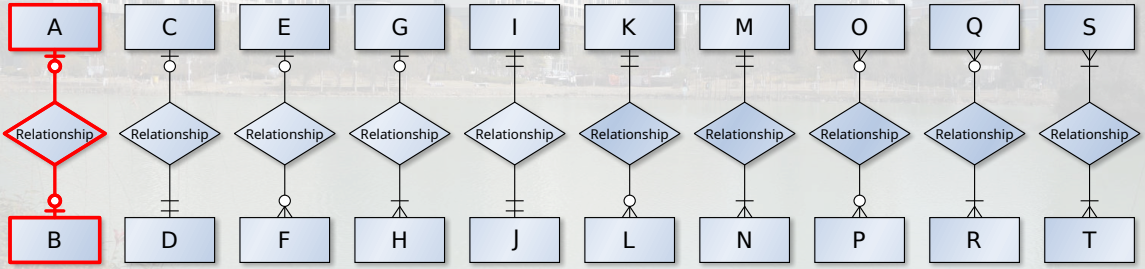
Mögliche Beziehungen in der Krähenfußnotation

- A ⟝○—○⟞ B:
  - Jede Entität vom Typ A kann mit keiner oder einer Entität vom Typ B verbunden sein.
  - Jede Entität vom Typ B kann mit keiner oder einer Entität vom Typ A verbunden sein.



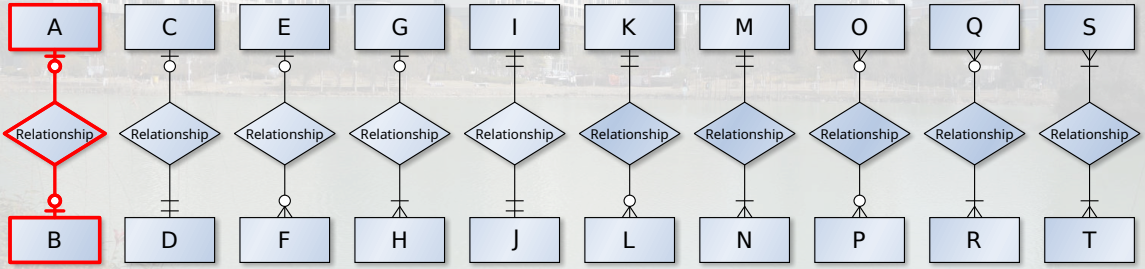
Mögliche Beziehungen in der Krähenfußnotation

- A ⟊—○⟊ B:
  - Jede Entität vom Typ A kann mit keiner oder einer Entität vom Typ B verbunden sein.
  - Jede Entität vom Typ B kann mit keiner oder einer Entität vom Typ A verbunden sein.
  - *Beispiel:* Eine Kundin kann einen Discount-Kode eingeben, we sie etwas online bestellt[63]. Jeder Discount-Kode kann höchstens einmal verwendet werden. Höchstens ein Discount-Kode kann pro Bestellung verwendet werden.



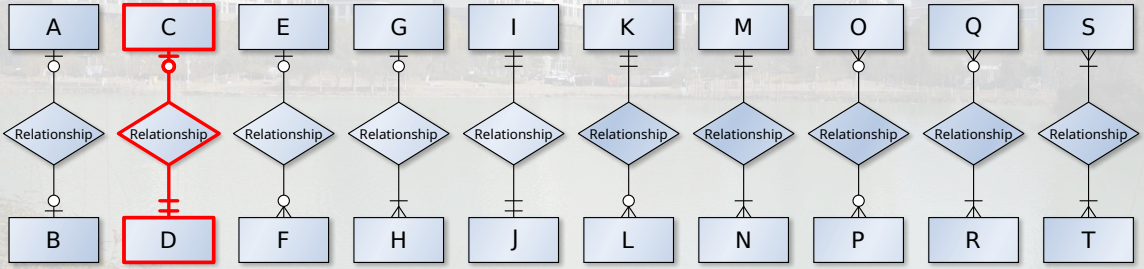
Mögliche Beziehungen in der Krähenfußnotation

- A ⟝⟝ B:
  - Jede Entität vom Typ A kann mit keiner oder einer Entität vom Typ B verbunden sein.
  - Jede Entität vom Typ B kann mit keiner oder einer Entität vom Typ A verbunden sein.
  - *Beispiel:* Eine Kundin kann einen Discount-Kode eingeben, we sie etwas online bestellt[63]. Jeder Discount-Kode kann höchstens einmal verwendet werden. Höchstens ein Discount-Kode kann pro Bestellung verwendet werden.
  - *Beispiel:* Eine Person kann mit höchstens einer anderen Person verheiratet sein[72].



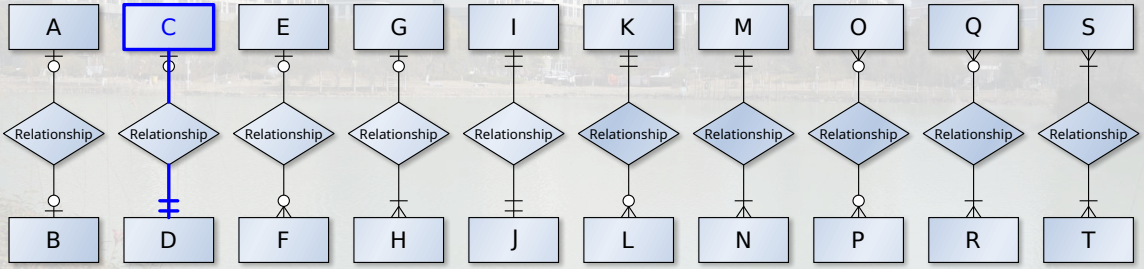
Mögliche Beziehungen in der Krähenfußnotation

- C ⟜—‖ D



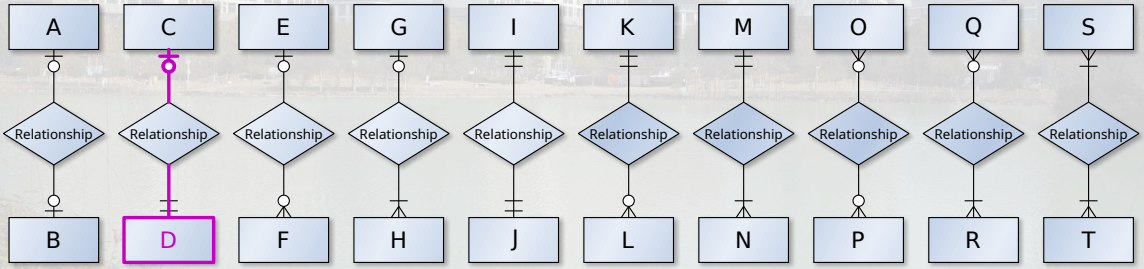
Mögliche Beziehungen in der Krähenfußnotation

- C ⟜⟞ D:
  - Jede Entität vom Typ C muss mit genau einer Entität vom Typ D verbunden sein.



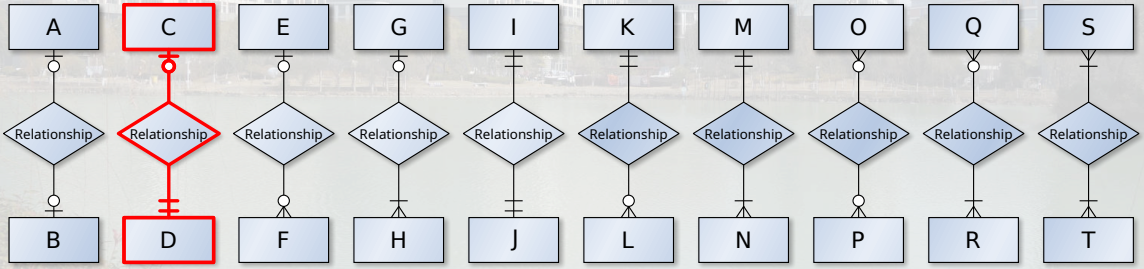
Mögliche Beziehungen in der Krähenfußnotation

- C ⟝ D:
  - Jede Entität vom Typ C muss mit genau einer Entität vom Typ D verbunden sein.
  - Jede Entität vom Typ D kann mit einer oder keiner Entität vom Typ C verbunden sein.



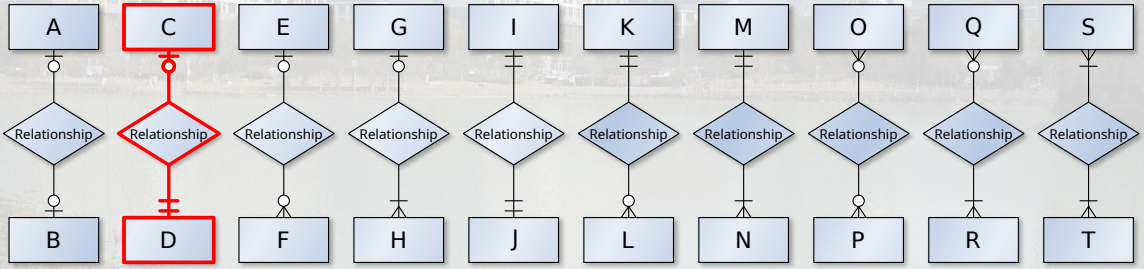
Mögliche Beziehungen in der Krähenfußnotation

- C ⟝ D:
  - Jede Entität vom Typ C muss mit genau einer Entität vom Typ D verbunden sein.
  - Jede Entität vom Typ D kann mit einer oder keiner Entität vom Typ C verbunden sein.
  - *Beispiel:* In jedem Büro des Bürogebäudes kann einem oder keinem Mitarbeiter zugewiesen sein.Jeder Mitarbeiter hat genau ein Büro[89].



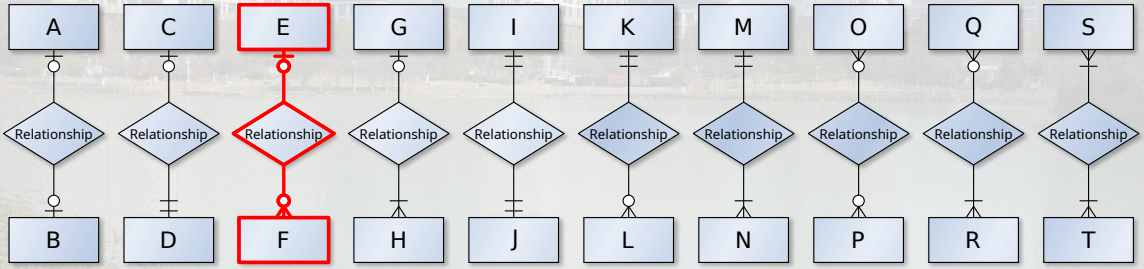
Mögliche Beziehungen in der Krähenfußnotation

- C ⟝ D:
  - Jede Entität vom Typ C muss mit genau einer Entität vom Typ D verbunden sein.
  - Jede Entität vom Typ D kann mit einer oder keiner Entität vom Typ C verbunden sein.
  - *Beispiel:* In jedem Büro des Bürogebäudes kann einem oder keinem Mitarbeiter zugewiesen sein.Jeder Mitarbeiter hat genau ein Büro[89].
  - *Beispiel:* Ein Professor kann entweder Dekan einer Fakultät sein oder nicht. Jede Fakultät hat genau einen Dekan[72].



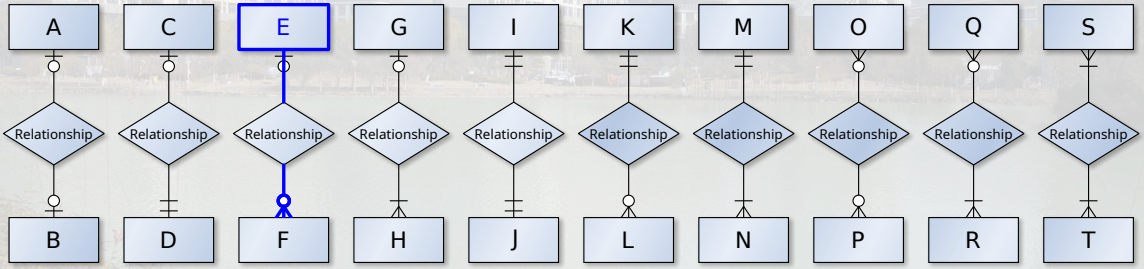
Mögliche Beziehungen in der Krähenfußnotation

- E ⟊○—○⟨ F



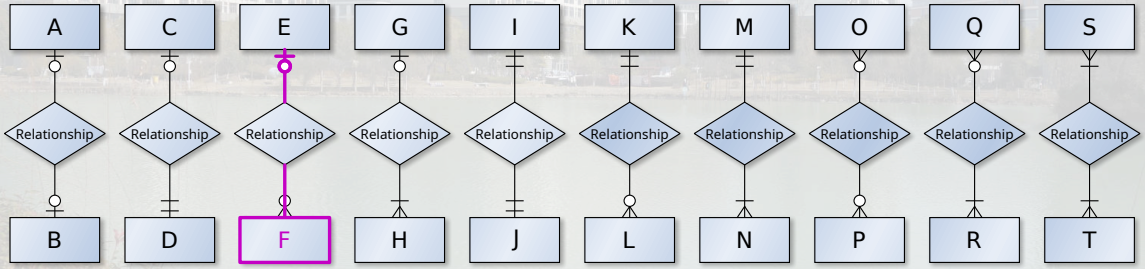
Mögliche Beziehungen in der Krähenfußnotation

- E ┼○─○< F:
  - Jede Entität vom Typ E kann mit keiner, einer, oder mehreren Entitäten vom Typ F.



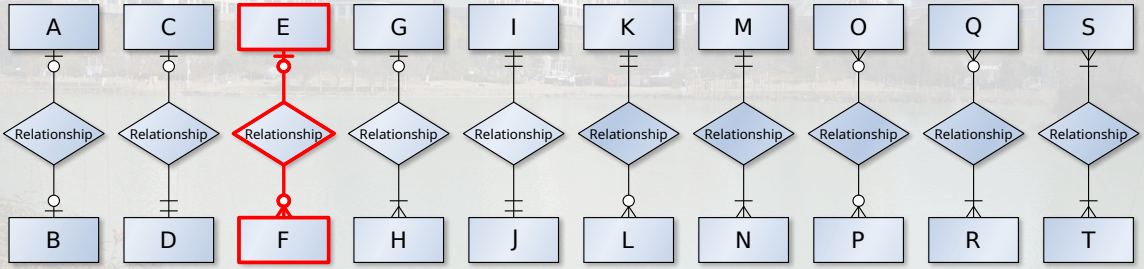
Mögliche Beziehungen in der Krähenfußnotation

- E ⟊○—○⟨ F:
  - Jede Entität vom Typ E kann mit keiner, einer, oder mehreren Entitäten vom Typ F.
  - Jede Entität vom Typ F kann mit keiner oder einer Entität vom Type E verbunden sein[56].



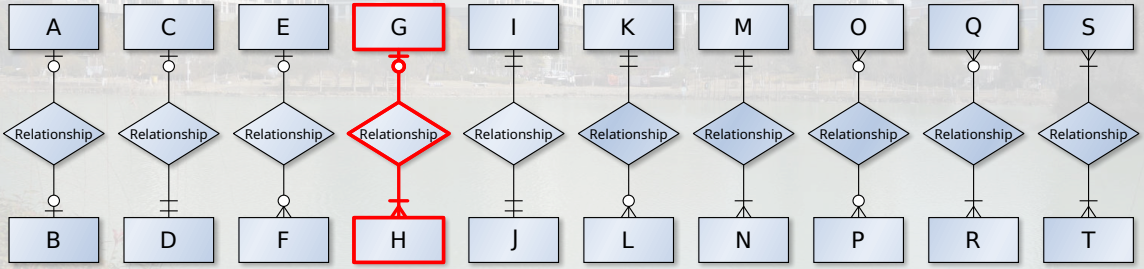
Mögliche Beziehungen in der Krähenfußnotation

- E ┼○─○< F:
  - Jede Entität vom Typ E kann mit keiner, einer, oder mehreren Entitäten vom Typ F.
  - Jede Entität vom Typ F kann mit keiner oder einer Entität vom Type E verbunden sein[56].
  - *Beispiel:* Jedem Bankkonto können zwei Addressen hinterlegt sein: Eine Wohnadresse wird immer benötigt, was für dieses Beispiel egal ist. Was wichtig ist, ist dass zusätzlich optional eine (also keine oder eine) Rechnungsadresse hinterlegt werden kann, Jede Adresse kann die Rechnungsadresse von keinem, einem, oder mehreren Bankkonten sein[56].



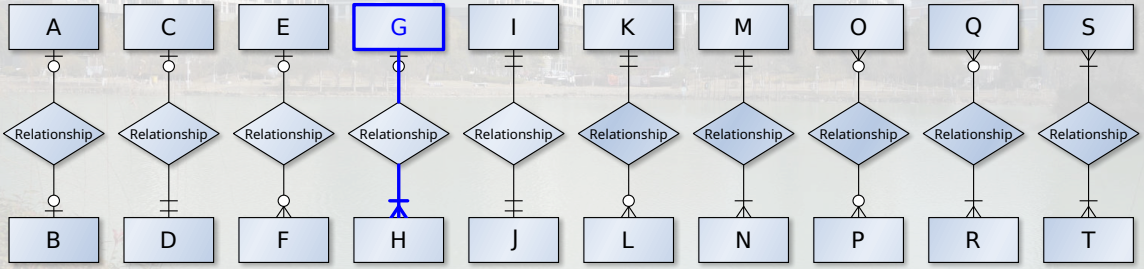
Mögliche Beziehungen in der Krähenfußnotation

- G ⟝ H



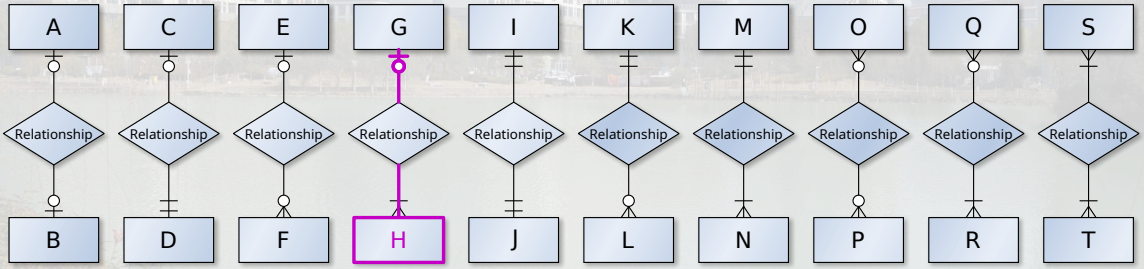
Mögliche Beziehungen in der Krähenfußnotation

- G ⟊○—⟊< H:
  - Jede Entität vom Typ G ist mit mindestens einer, möglicherweise aber auch mehreren Entitäten vom Type H verbunden.



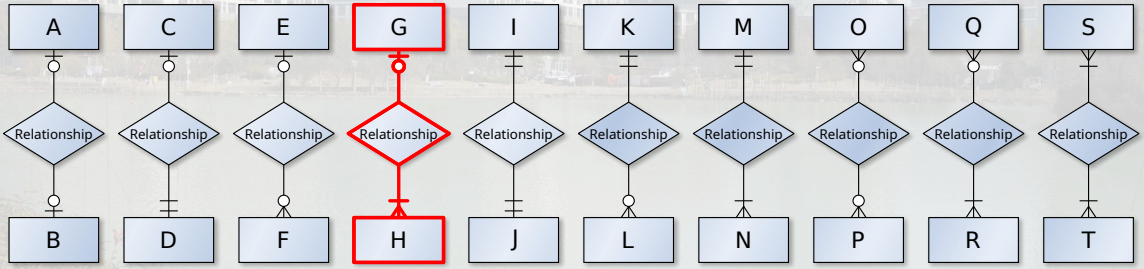
Mögliche Beziehungen in der Krähenfußnotation

- G ⟙—⟙< H:
  - Jede Entität vom Typ G ist mit mindestens einer, möglicherweise aber auch mehreren Entitäten vom Type H verbunden.
  - Jede Entität vom Typ H ist mit keiner oder einer Entität vom Type G verbunden.



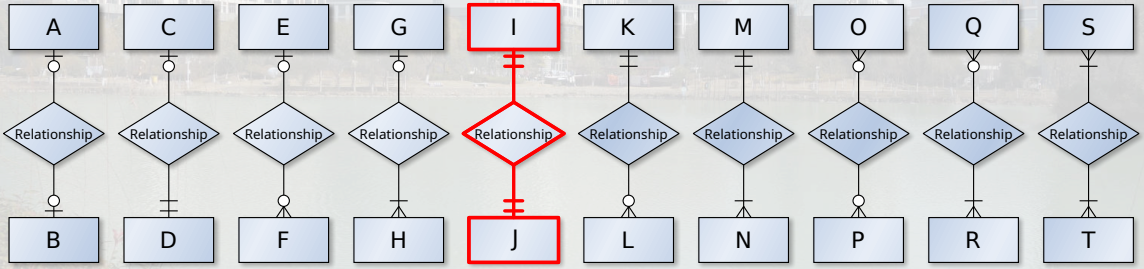
Mögliche Beziehungen in der Krähenfußnotation

- G ⟝ H:
  - Jede Entität vom Typ G ist mit mindestens einer, möglicherweise aber auch mehreren Entitäten vom Type H verbunden.
  - Jede Entität vom Typ H ist mit keiner oder einer Entität vom Type G verbunden.
  - *Beispiel:* Ein Trainer einer Fußballmannschaft kann mehrere Klubmitglieder trainieren, aber immer mindestens eins, sonst ist er kein Trainer. Ein Klubmitglied kann entweder von einem Trainer oder keinem Trainer trainiert werden (z. B. wenn das Mitglied andere Funktionen hat und nicht aktiv spielt)[46].



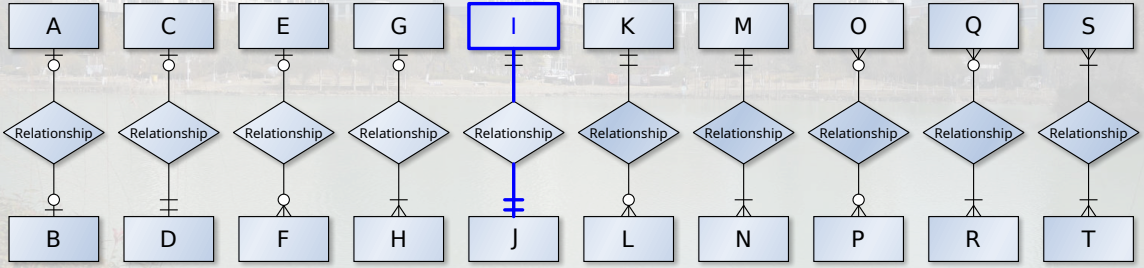
Mögliche Beziehungen in der Krähenfußnotation

- I ═╫──╫═ J



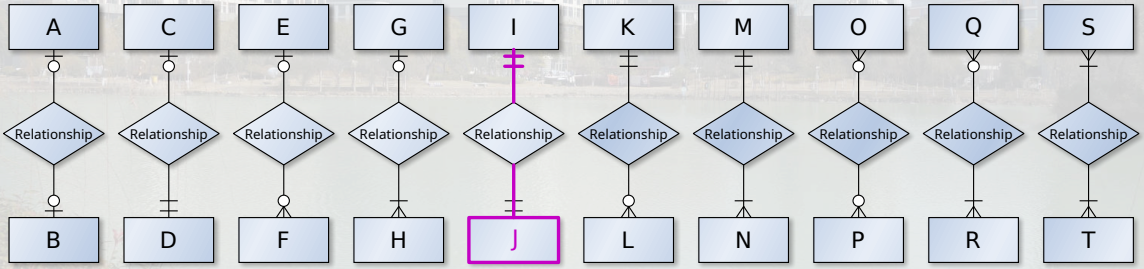
Mögliche Beziehungen in der Krähenfußnotation

- I ||—|| J:
  - Jede Entität vom Typ I muss mit genau einer Entität vom Type J verbunden sein.



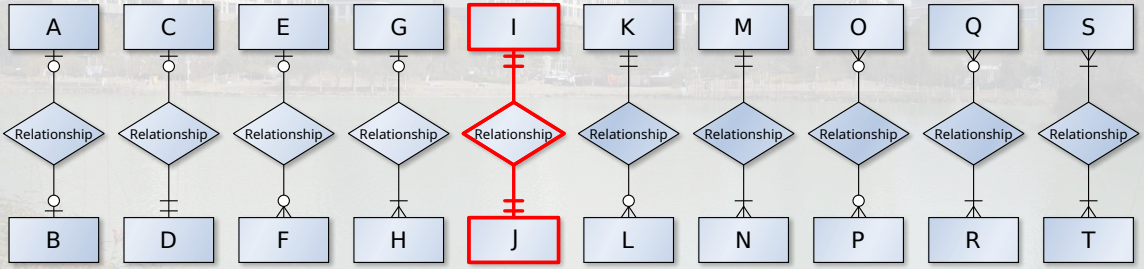
Mögliche Beziehungen in der Krähenfußnotation

- I ||—|| J:
  - Jede Entität vom Typ I muss mit genau einer Entität vom Type J verbunden sein.
  - Jede Entität vom Typ J muss mit genau einer Entität vom Type I verbunden sein.



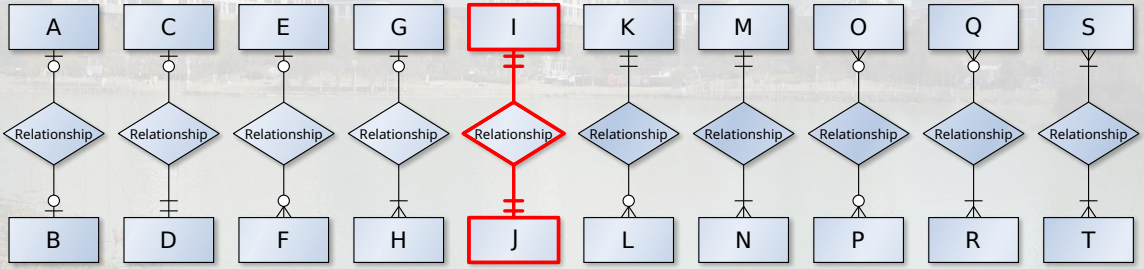
Mögliche Beziehungen in der Krähenfußnotation

- I ‖—‖ J:
  - Jede Entität vom Typ I muss mit genau einer Entität vom Type J verbunden sein.
  - Jede Entität vom Typ J muss mit genau einer Entität vom Type I verbunden sein.
  - *Beispiel:* Polizeipatrouillen werden von Teams zweier Polizeibeamte durchgeführt (zumindest in Filem. . .).



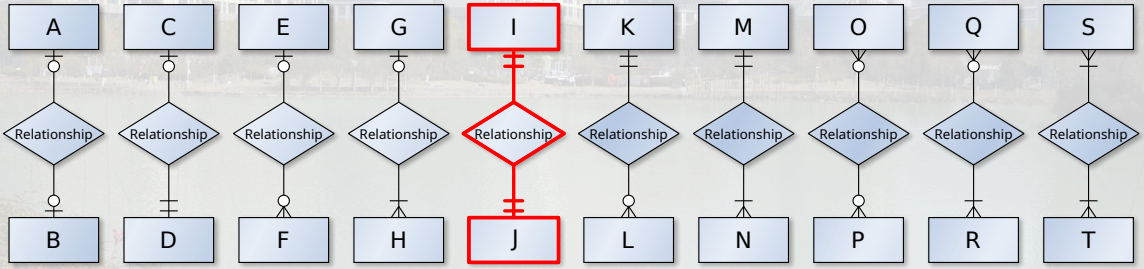
Mögliche Beziehungen in der Krähenfußnotation

- I ╫──╫ J:
  - Jede Entität vom Typ I muss mit genau einer Entität vom Type J verbunden sein.
  - Jede Entität vom Typ J muss mit genau einer Entität vom Type I verbunden sein.
  - *Beispiel:* Polizeipatrouillen werden von Teams zweier Polizeibeamte durchgeführt (zumindest in Filem. . .).
  - *Beispiel:* Für lange Busfahrten könnten es immer Teams von zwei Busfahrern geben, die sich abwechseln.



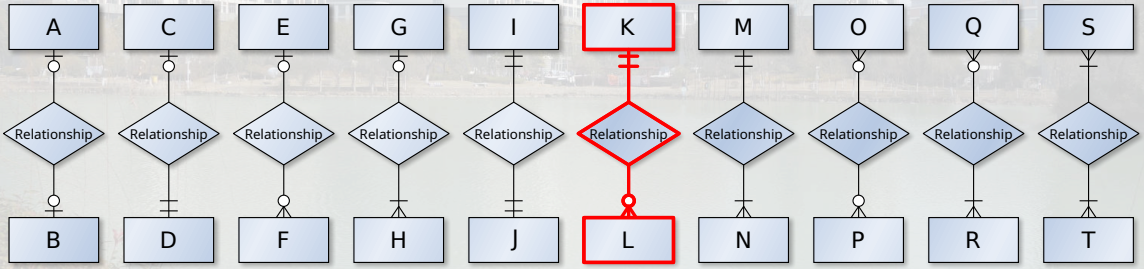
Mögliche Beziehungen in der Krähenfußnotation

- I ‖—‖ J:
  - Jede Entität vom Typ I muss mit genau einer Entität vom Type J verbunden sein.
  - Jede Entität vom Typ J muss mit genau einer Entität vom Type I verbunden sein.
  - *Beispiel:* Für lange Busfahrten könnten es immer Teams von zwei Busfahrern geben, die sich abwechseln.
  - *Beispiel:* In einer Firma könnte ein Verkäufer immer als Notfallersatz für genau einen anderen Verkäufer eingetragen sein und umgekehrt[89].



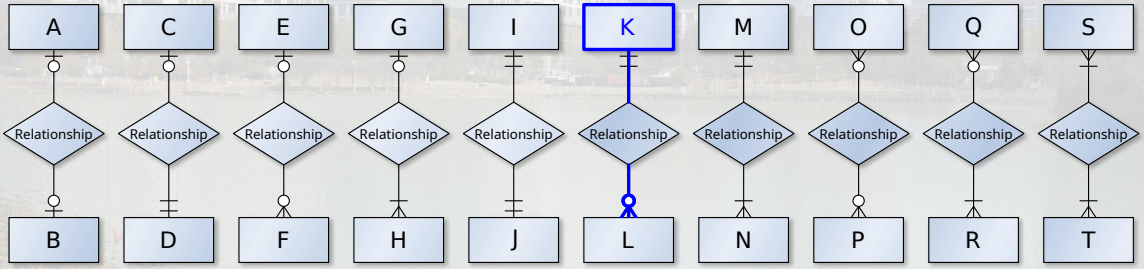
Mögliche Beziehungen in der Krähenfußnotation

- K ||—o< L



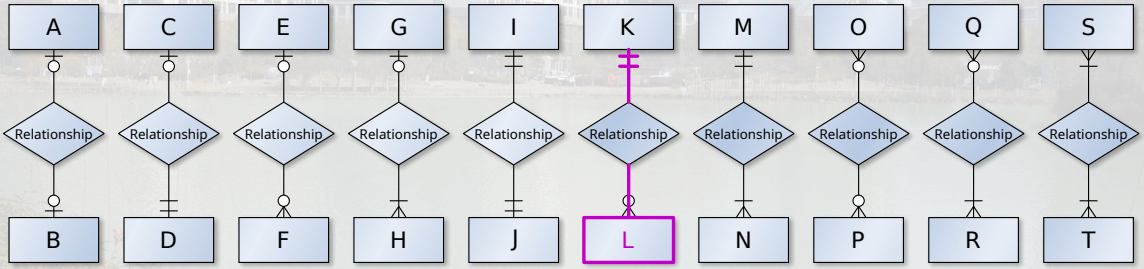
Mögliche Beziehungen in der Krähenfußnotation

- K ||—o< L:
  - Jede Entität vom Typ K kann mit keiner, einer, oder mehreren Entitäten vom Typ L verbunden sein.



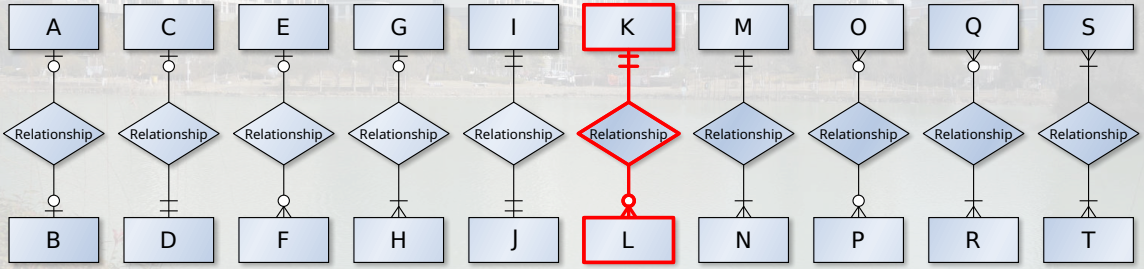
Mögliche Beziehungen in der Krähenfußnotation

- K ||—o< L:
  - Jede Entität vom Typ K kann mit keiner, einer, oder mehreren Entitäten vom Typ L verbunden sein.
  - Jede Entität vom Typ L muss mit genau einer Entität vom Typ K verbunden sein[56,63].



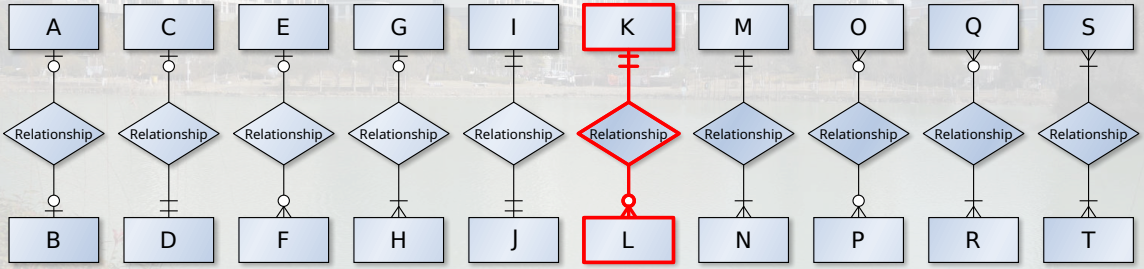
Mögliche Beziehungen in der Krähenfußnotation

- K ||—o< L:
  - Jede Entität vom Typ K kann mit keiner, einer, oder mehreren Entitäten vom Typ L verbunden sein.
  - Jede Entität vom Typ L muss mit genau einer Entität vom Typ K verbunden sein[56,63].
  - *Beispiel:* Eine Kundin kann keine oder beliebig viele Bestellungen aufgeben. Jede Bestellung muss aber mit genau einer Kundin verbunden sein[63].



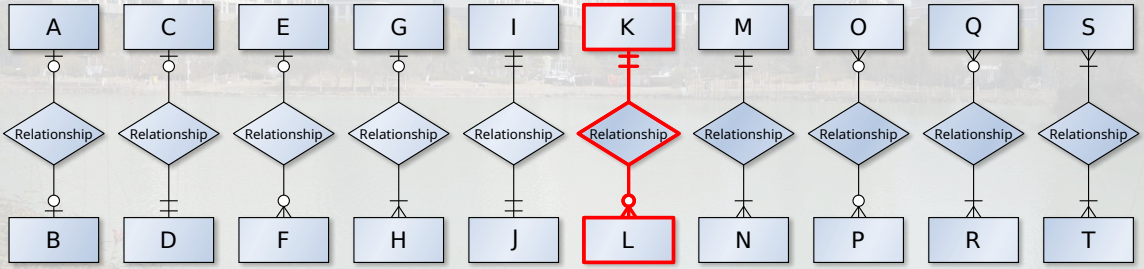
Mögliche Beziehungen in der Krähenfußnotation

- K ||—o< L:
  - Jede Entität vom Typ K kann mit keiner, einer, oder mehreren Entitäten vom Typ L verbunden sein.
  - Jede Entität vom Typ L muss mit genau einer Entität vom Typ K verbunden sein[56,63].
  - *Beispiel:* Eine Kundin kann keine oder beliebig viele Bestellungen aufgeben. Jede Bestellung muss aber mit genau einer Kundin verbunden sein[63].
  - *Beispiel:* Ein Bankkonto kann die Quelle von keiner, einer, oder mehreren Überweisungen sein. Jede Überweisung muss aber genau ein Quell-Konto haben[56].



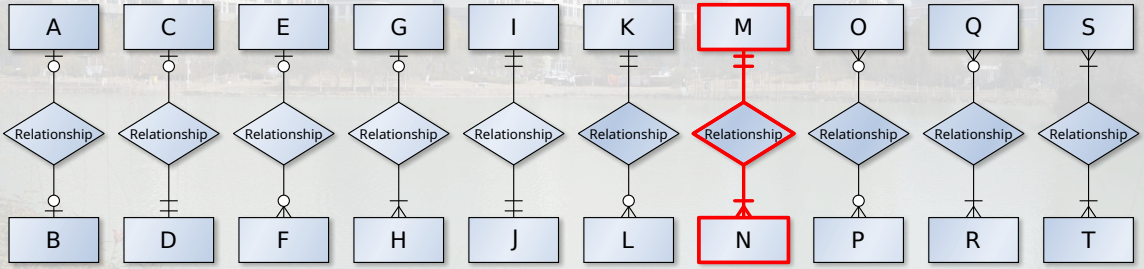
Mögliche Beziehungen in der Krähenfußnotation

- K ||—o< L:
  - Jede Entität vom Typ K kann mit keiner, einer, oder mehreren Entitäten vom Typ L verbunden sein.
  - Jede Entität vom Typ L muss mit genau einer Entität vom Typ K verbunden sein[56,63].
  - *Beispiel:* Ein Bankkonto kann die Quelle von keiner, einer, oder mehreren Überweisungen sein. Jede Überweisung muss aber genau ein Quell-Konto haben[56].
  - *Beispiel:* Ein Verkäufer hat entweder keinen, einen, oder mehrere Kunden. Jeder Kunde wir aber von genau einem Verkäufer unterstützt[89].



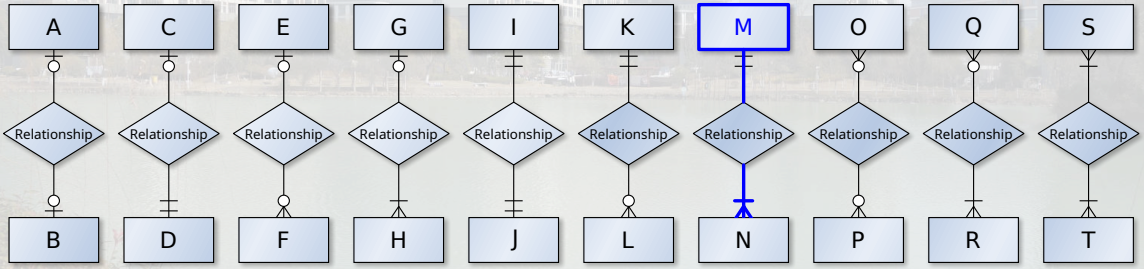
Mögliche Beziehungen in der Krähenfußnotation

- M ||—|< N



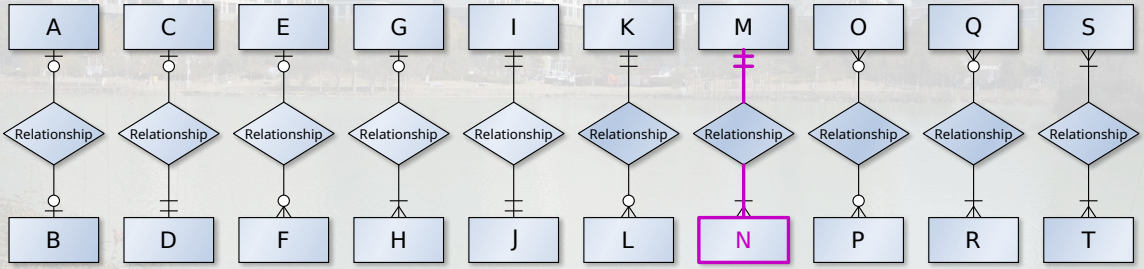
Mögliche Beziehungen in der Krähenfußnotation

- M ⫫⟝⫞⋖ N:
  - Jede Entität vom Typ M ist mit mindestens einer und möglicherweise mehreren Entitäten vom Type N verbunden.



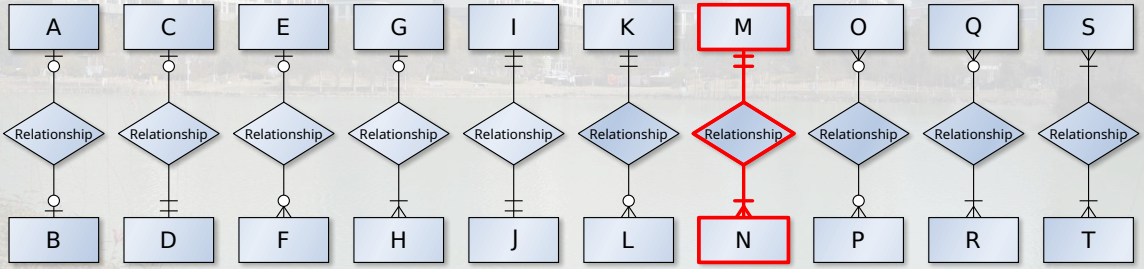
Mögliche Beziehungen in der Krähenfußnotation

- M ||—|< N:
  - Jede Entität vom Typ M ist mit mindestens einer und möglicherweise mehreren Entitäten vom Type N verbunden.
  - Jede Entität vom Typ N ist mit genau einer Entität vom Typ M verbunden.[64]



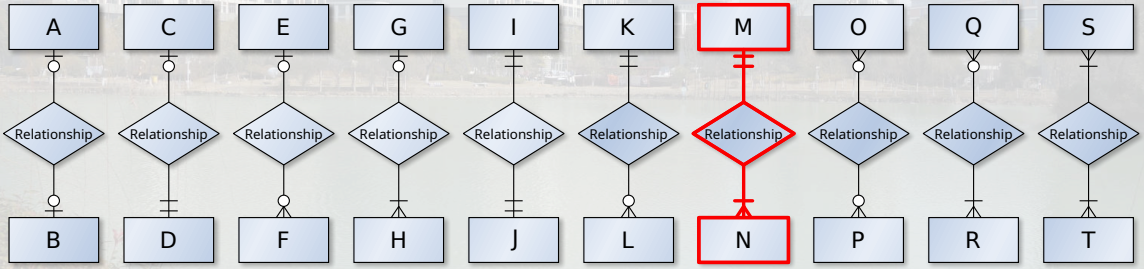
Mögliche Beziehungen in der Krähenfußnotation

- M ||—|< N:
  - Jede Entität vom Typ M ist mit mindestens einer und möglicherweise mehreren Entitäten vom Type N verbunden.
  - Jede Entität vom Typ N ist mit genau einer Entität vom Typ M verbunden.[64]
  - *Beispiel:* Ein Patient kann mehrere Termine mit Ärztinnen vereinbaren. Jeder Termin ist mit genau einem Partienten verbunden.[64]



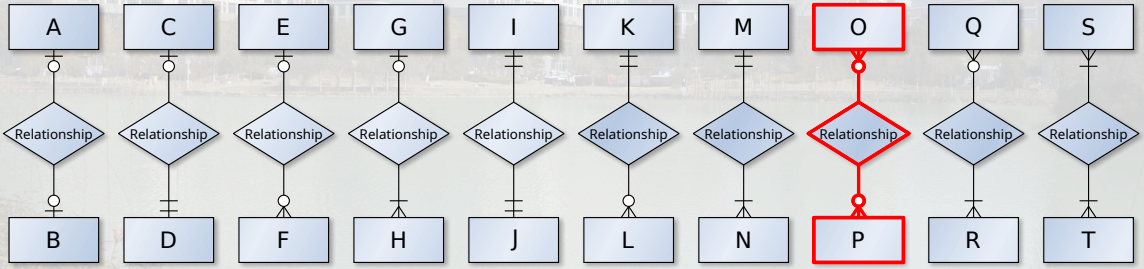
Mögliche Beziehungen in der Krähenfußnotation

- M ||—|< N:
  - Jede Entität vom Typ M ist mit mindestens einer und möglicherweise mehreren Entitäten vom Type N verbunden.
  - Jede Entität vom Typ N ist mit genau einer Entität vom Typ M verbunden.[64]
  - *Beispiel:* Ein Patient kann mehrere Termine mit Ärztinnen vereinbaren. Jeder Termin ist mit genau einem Partienten verbunden.[64]
  - *Beispiel:* Eine Fakultät einer Universität besteht aus mindestens einer, wahrscheinlich aber aus mehreren Abteilungen. Jede Abteilung gehört zu genau einer Fakultät.[72]



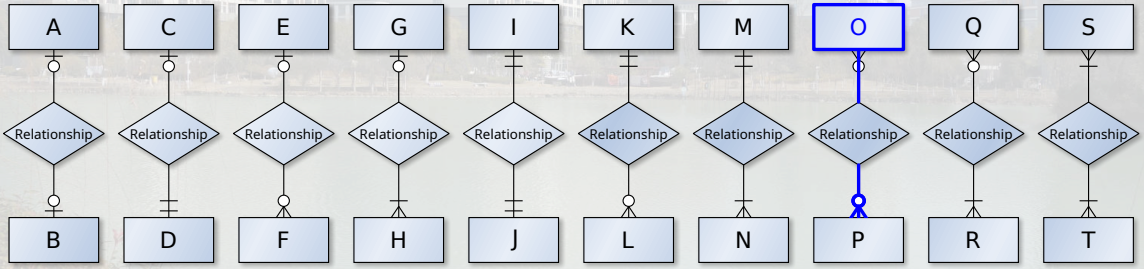
Mögliche Beziehungen in der Krähenfußnotation

- O ⟩○—○⟨ P



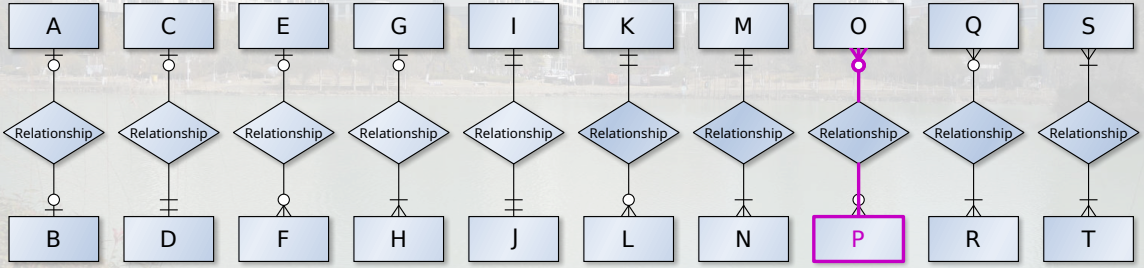
Mögliche Beziehungen in der Krähenfußnotation

- O ⟩○—○⟨ P:
  - Jede Entität vom Typ O kann mit keiner, einer, oder mehreren Entitäten vom Typ P verbunden sein.



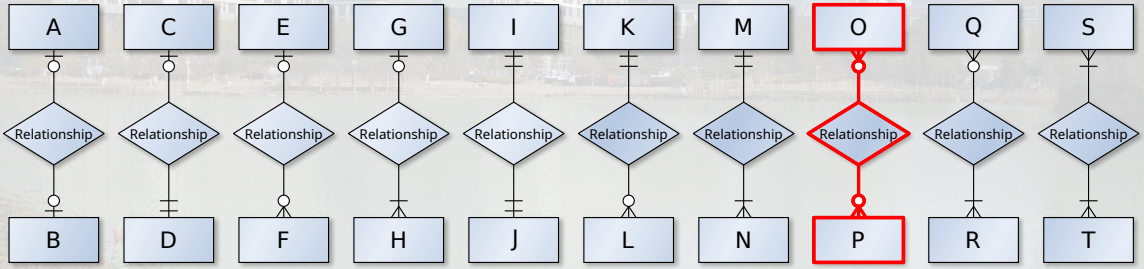
Mögliche Beziehungen in der Krähenfußnotation

- O ⟩○—○⟨ P:
  - Jede Entität vom Typ O kann mit keiner, einer, oder mehreren Entitäten vom Typ P verbunden sein.
  - Jede Entität vom Typ P kann mit keiner, einer, oder mehreren Entitäten vom Typ O verbunden sein.



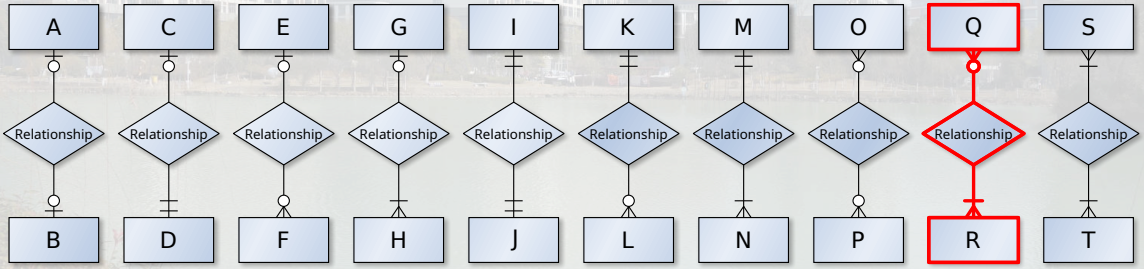
Mögliche Beziehungen in der Krähenfußnotation

- O ⟩○—○⟨ P:
  - Jede Entität vom Typ O kann mit keiner, einer, oder mehreren Entitäten vom Typ P verbunden sein.
  - Jede Entität vom Typ P kann mit keiner, einer, oder mehreren Entitäten vom Typ O verbunden sein.
  - *Beispiel:* Ein Pizzarestaurant verkauft Pizzen an Kunden. Eine Pizza kann von keinem, einem, oder mehreren Kunden bestellt werden. Ein Kunde bestellt keine (veielleicht will er ja Nudeln...), eine, oder mehrere Pizzen[26].



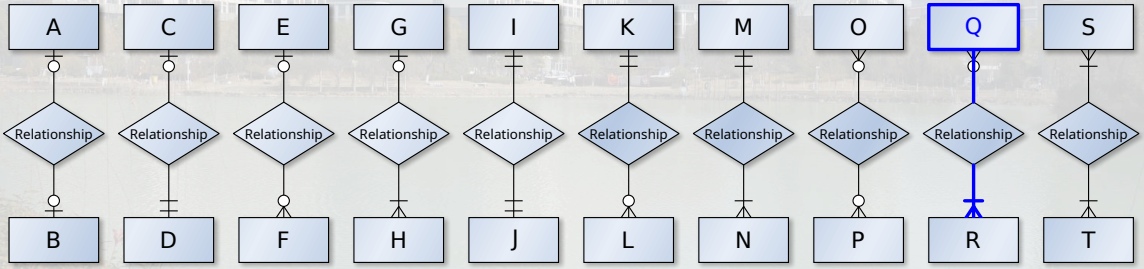
Mögliche Beziehungen in der Krähenfußnotation

- Q ⟩o—|< R



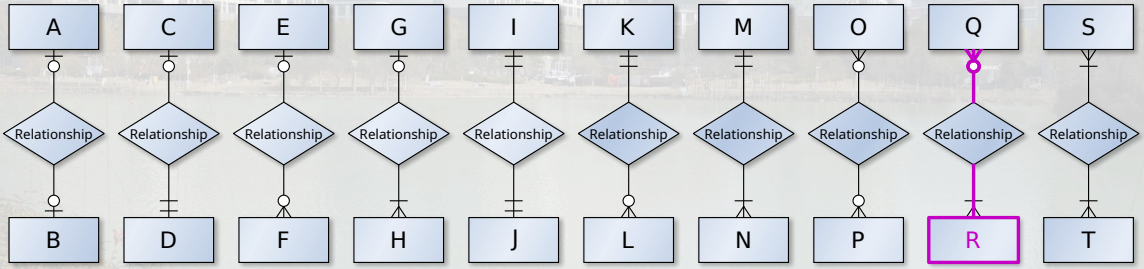
Mögliche Beziehungen in der Krähenfußnotation

- Q  R:
  - Jede Entität vom Typ Q ist mit mindestens einer, möglicherweise auch mehreren Entitäten vom Type R verbunden.



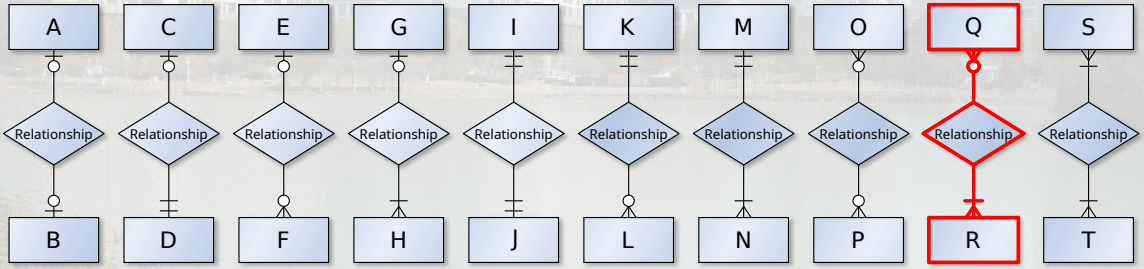
Mögliche Beziehungen in der Krähenfußnotation

- Q ⟩○—|⟨ R:
  - Jede Entität vom Typ Q ist mit mindestens einer, möglicherweise auch mehreren Entitäten vom Type R verbunden.
  - Jede Entität vom Typ R kann mit keiner, einer, oder mehreren Entitäten vom Typ Q verbunden sein[63].



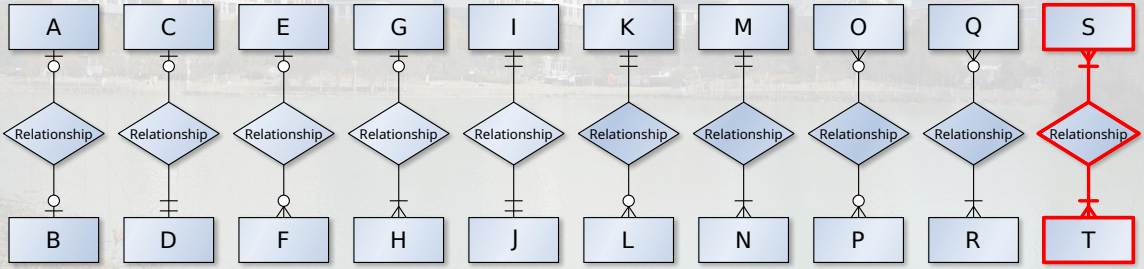
Mögliche Beziehungen in der Krähenfußnotation

- Q  R:
  - Jede Entität vom Typ Q ist mit mindestens einer, möglicherweise auch mehreren Entitäten vom Type R verbunden.
  - Jede Entität vom Typ R kann mit keiner, einer, oder mehreren Entitäten vom Typ Q verbunden sein[63].
  - *Beispiel:* Wenn wir online Produkte bestellen, muss jede Bestellung aus mindestens einem Produkt bestehen. Jedes Produkt kann von keiner, einer, oder mehreren Bestellungen referenziert werden[63].



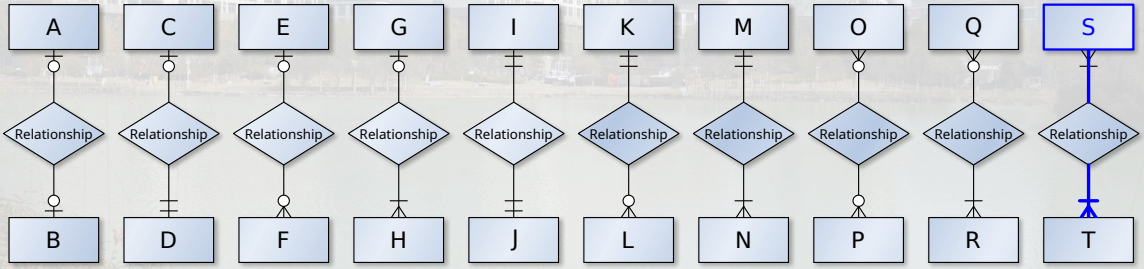
Mögliche Beziehungen in der Krähenfußnotation

- S >|—|< T



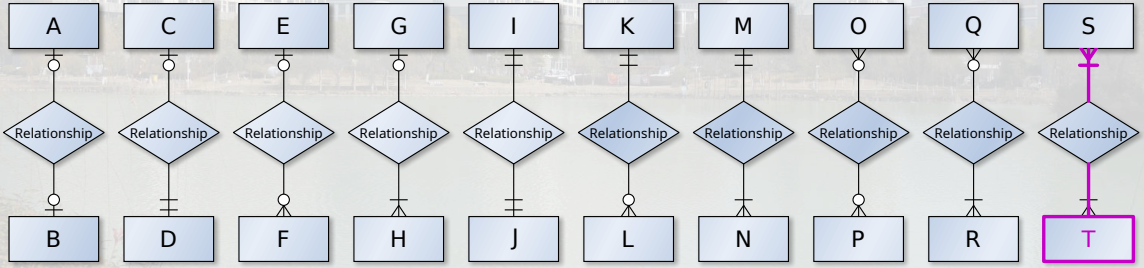
Mögliche Beziehungen in der Krähenfußnotation

- S >|—|< T:
  - Jede Entität vom Typ S ist mit mindestens einer, möglicherweise auch mehreren Entitäten vom Type T verbunden.



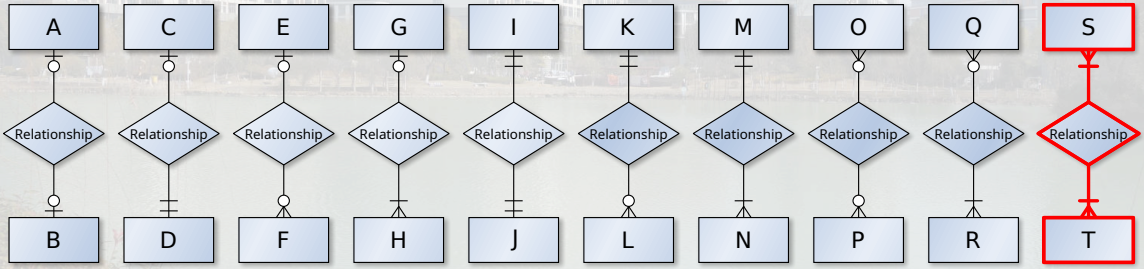
Mögliche Beziehungen in der Krähenfußnotation

- S >|—|< T:
  - Jede Entität vom Typ S ist mit mindestens einer, möglicherweise auch mehreren Entitäten vom Type T verbunden.
  - Jede Entität vom Typ T ist mit mindestens einer, möglicherweise auch mehreren Entitäten vom Type S verbunden.



Mögliche Beziehungen in der Krähenfußnotation

- S >|—|< T:
  - Jede Entität vom Typ S ist mit mindestens einer, möglicherweise auch mehreren Entitäten vom Type T verbunden.
  - Jede Entität vom Typ T ist mit mindestens einer, möglicherweise auch mehreren Entitäten vom Type S verbunden.
  - *Beispiel:* Jede Verkäuferin verkauft mindestens ein Produkt, kann aber auch für mehrere Produkte zuständig sein. Jedes Produkt wird von mindestens einer Verkäuferin verkauft, vielleicht aber auch von mehreren[89].



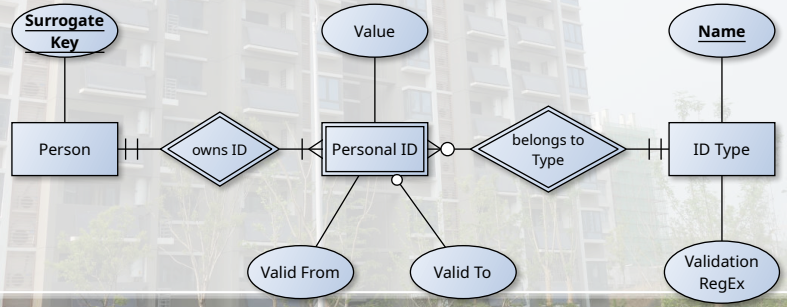
Beispiele

Person

- Annotieren wir nun unsere *Person*-Entitätstypen mit „Krähenfüßen“.

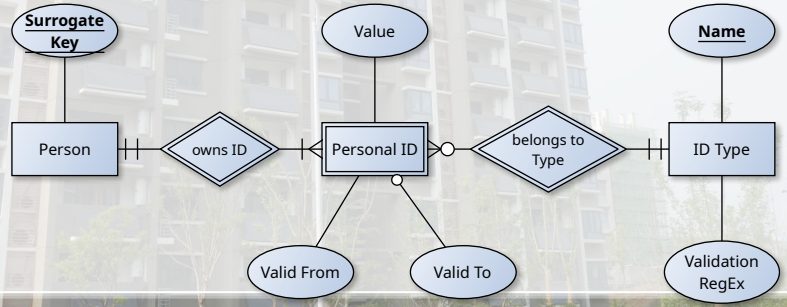
Person

- Annotieren wir nun unsere *Person*-Entitätstypen mit „Krähenfüßen“.
- Die Idee war damals, dass wir mehrere ID-Typen haben.



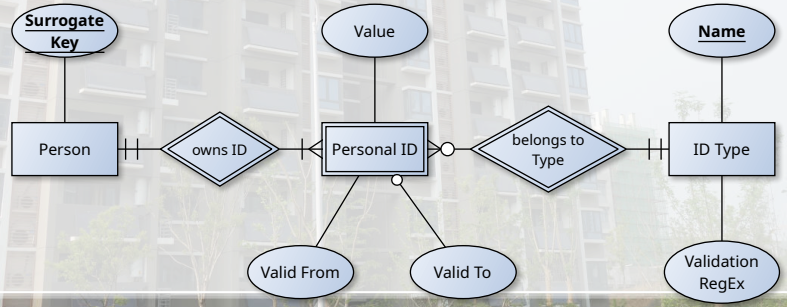
Person

- Annotieren wir nun unsere *Person*-Entitätstypen mit „Krähenfüßen“.
- Die Idee war damals, dass wir mehrere ID-Typen haben.
- Jede *Personal ID* gehört zu genau einem *ID Type*.



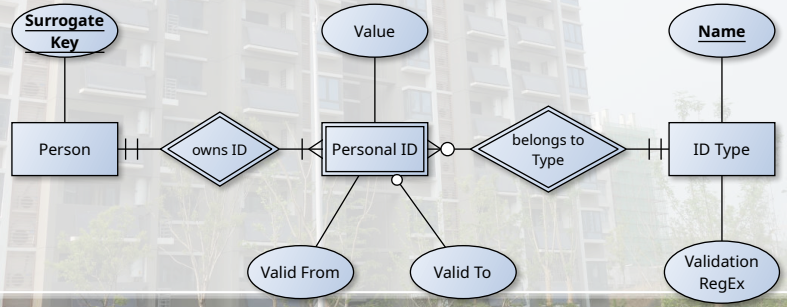
Person

- Annotieren wir nun unsere *Person*-Entitätstypen mit „Krähenfüßen“.
- Die Idee war damals, dass wir mehrere ID-Typen haben.
- Jede *Personal ID* gehört zu genau einem *ID Type*.
- Es kann beliebig viele *Personal ID*s für jeden *ID Type* in unserem System geben. Vielleicht keine, vielleicht viele.



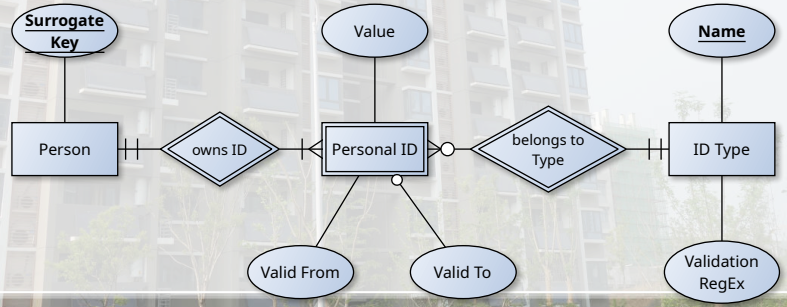
Person

- Annotieren wir nun unsere *Person*-Entitätstypen mit „Krähenfüßen“.
- Die Idee war damals, dass wir mehrere ID-Typen haben.
- Jede *Personal ID* gehört zu genau einem *ID Type*.
- Es kann beliebig viele *Personal ID*s für jeden *ID Type* in unserem System geben. Vielleicht keine, vielleicht viele.
- Die Beziehung ist also *Personal ID* >o—|| *ID Type*.



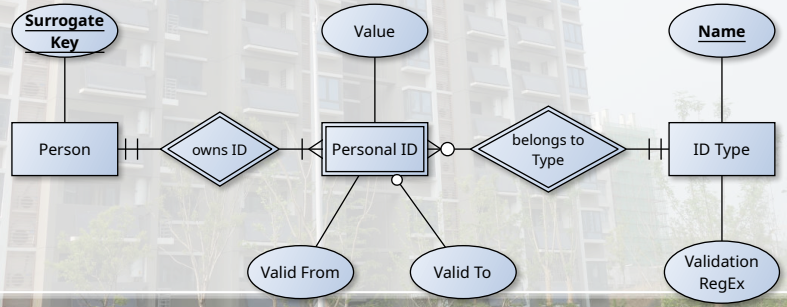
Person

- Die Idee war damals, dass wir mehrere ID-Typen haben.
- Jede *Personal ID* gehört zu genau einem *ID Type*.
- Es kann beliebig viele *Personal ID*s für jeden *ID Type* in unserem System geben. Vielleicht keine, vielleicht viele.
- Die Beziehung ist also *Personal ID* >o—|| *ID Type*.
- Jede *Personal ID* gehört zu genau einer *Person*.



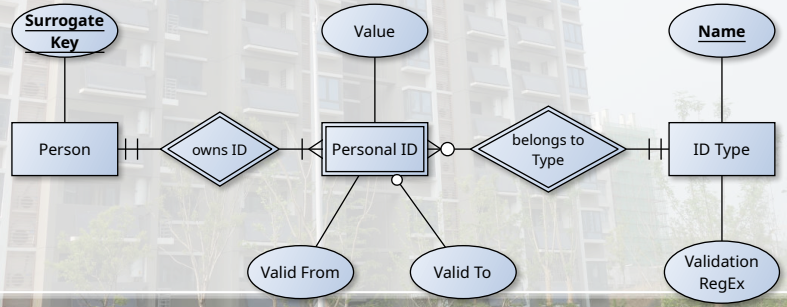
Person

- Jede *Personal ID* gehört zu genau einem *ID Type*.
- Es kann beliebig viele *Personal ID*s für jeden *ID Type* in unserem System geben. Vielleicht keine, vielleicht viele.
- Die Beziehung ist also *Personal ID* >o—|| *ID Type*.
- Jede *Personal ID* gehört zu genau einer *Person*.
- Zu jeder Zeit muss es mindestens eine *Personal ID* für jeden *Person*-Datensatz geben.



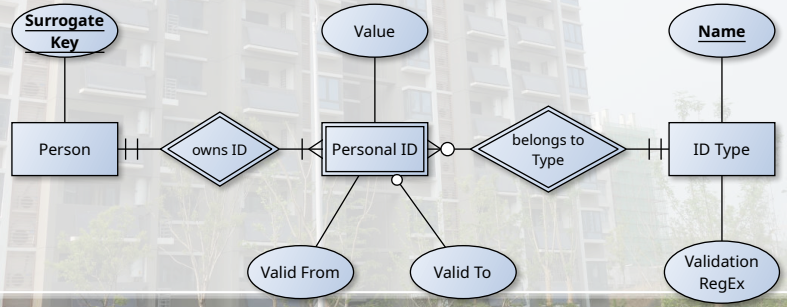
Person

- Es kann beliebig viele *Personal ID*s für jeden *ID Type* in unserem System geben. Vielleicht keine, vielleicht viele.
- Die Beziehung ist also *Personal ID* >o—|| *ID Type*.
- Jede *Personal ID* gehört zu genau einer *Person*.
- Zu jeder Zeit muss es mindestens eine *Personal ID* für jeden *Person*-Datensatz geben.
- Wir können keine Person in unserem System haben, deren Identität nicht auf mindestens eine offizielle Art bestätigt wurde.



Person

- Die Beziehung ist also *Personal ID* $\geq\!\!\circ\!\!-\!\!\|$ *ID Type*.
- Jede *Personal ID* gehört zu genau einer *Person*.
- Zu jeder Zeit muss es mindestens eine *Personal ID* für jeden *Person*-Datensatz geben.
- Wir können keine Person in unserem System haben, deren Identität nicht auf mindestens eine offizielle Art bestätigt wurde.
- Daher ist die Beziehung *Person* $\|\!\!-\!\!<$ *Personal ID*.

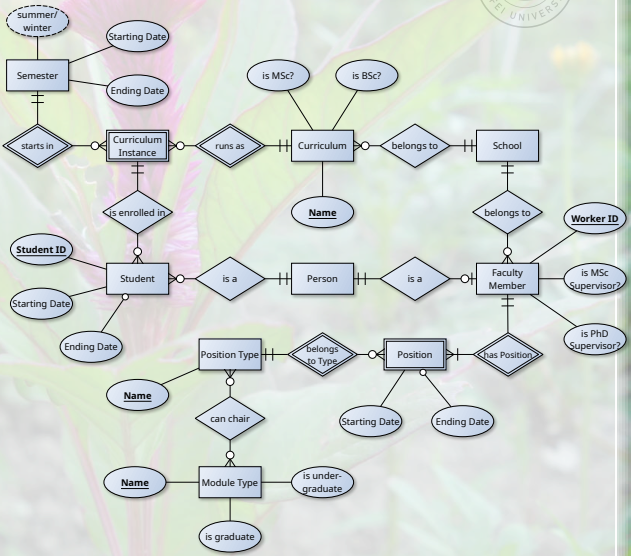


Bigger Picture

- Machen wir nun ein ERD eines viel größeren Ausschnitts unseres Systems.

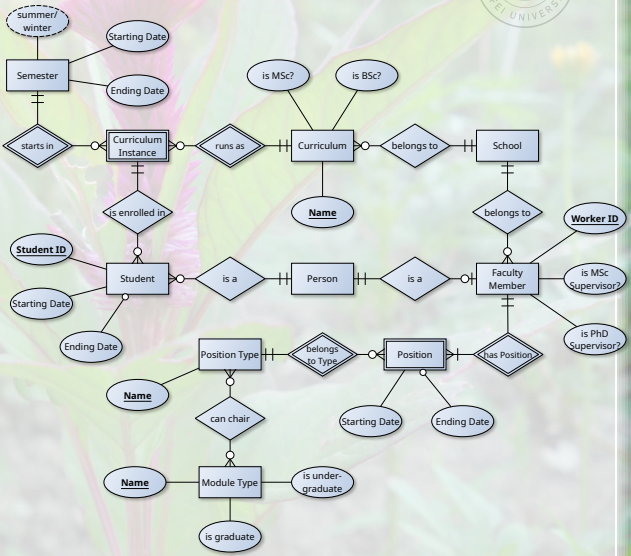
Bigger Picture

- Machen wir nun ein ERD eines viel größeren Ausschnitts unseres Systems.
- Jeder Studenten-Datensatz muss mit genau einem Personen-Datensatz assoziiert sind, denn jeder Student ist genau eine Person.



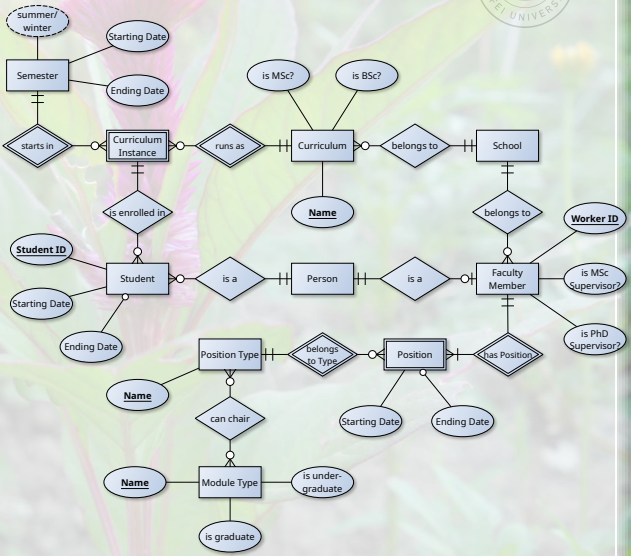
Bigger Picture

- Machen wir nun ein ERD eines viel größeren Ausschnitts unseres Systems.
- Jeder Studenten-Datensatz muss mit genau einem Personen-Datensatz assoziiert sind, denn jeder Student ist genau eine Person.
- Eine Person kann dagegen mit keinem Studenten-Datensatz assoziiert sein (wenn die Person kein Student ist), mit einem Studenten-Datensatz (wenn die Person ein Student is oder war), oder mit mehreren Studenten-Datensätzen (wenn die Person z. B. erst den Bachelor und dann den Master macht).



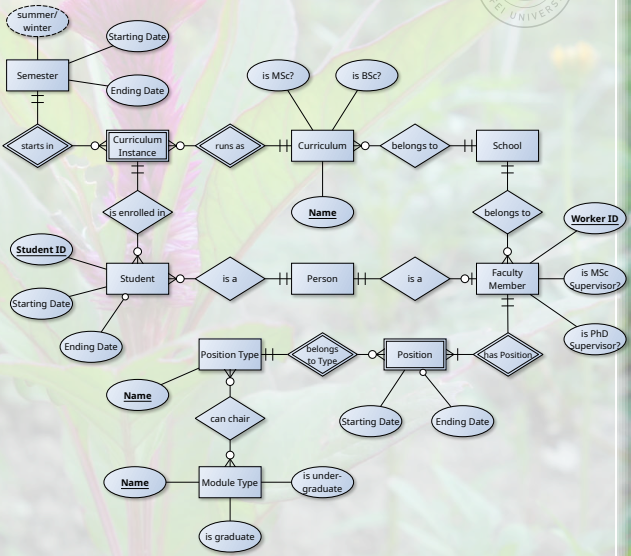
Bigger Picture

- Jeder Studenten-Datensatz muss mit genau einem Personen-Datensatz assoziiert sind, denn jeder Student ist genau eine Person.
- Eine Person kann dagegen mit keinem Studenten-Datensatz assoziiert sein (wenn die Person kein Student ist), mit einem Studenten-Datensatz (wenn die Person ein Student is oder war), oder mit mehreren Studenten-Datensätzen (wenn die Person z. B. erst den Bachelor und dann den Master macht).
- Jeder Studenten-Datensatz hat eine eindeutige Student ID, ein Start- und ein Enddatum.



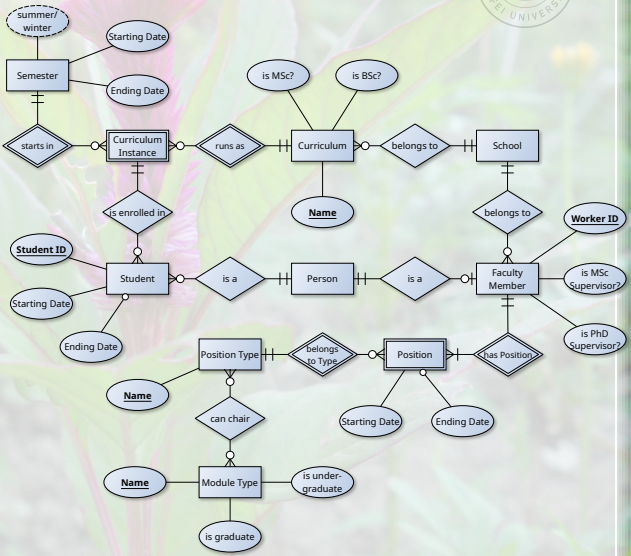
Bigger Picture

- Eine Person kann dagegen mit keinem Studenten-Datensatz assoziiert sein (wenn die Person kein Student ist), mit einem Studenten-Datensatz (wenn die Person ein Student is oder war), oder mit mehreren Studenten-Datensätzen (wenn die Person z. B. erst den Bachelor und dann den Master macht).
- Jeder Studenten-Datensatz hat eine eindeutige Student ID, ein Start- und ein Enddatum.
- Jeder Student ist in genau eine Studiengang-Instanz (EN: *curriculum instance*) eingeschrieben.



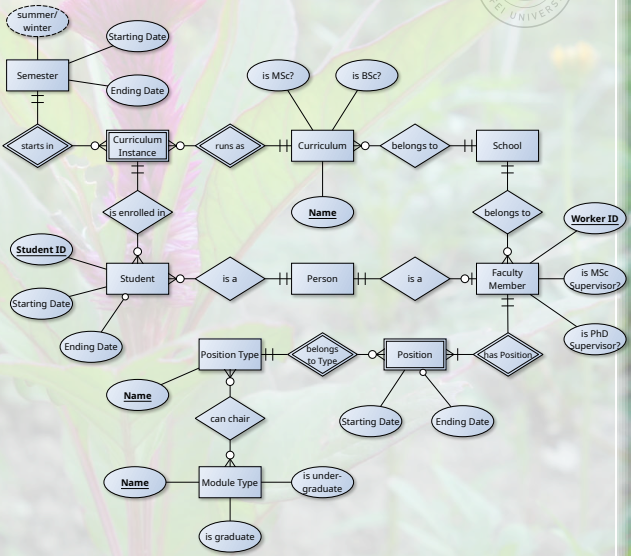
Bigger Picture

- Jeder Studenten-Datensatz hat eine eindeutige Student ID, ein Start- und ein Enddatum.
- Jeder Student ist in genau eine Studiengang-Instanz (EN: *curriculum instance*) eingeschrieben.
- Zu einer Studiengang-Instanz gehören entweder keine, ein, oder viele Studenten.



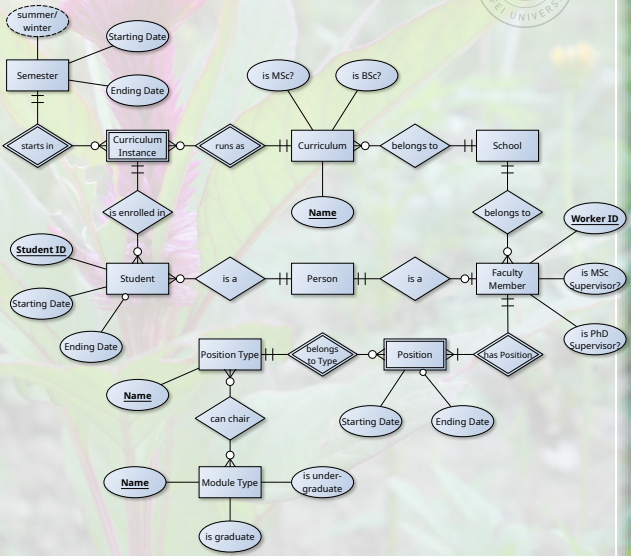
Bigger Picture

- Jeder Studenten-Datensatz hat eine eindeutige Student ID, ein Start- und ein Enddatum.
- Jeder Student ist in genau eine Studiengang-Instanz (EN: *curriculum instance*) eingeschrieben.
- Zu einer Studiengang-Instanz gehören entweder keine, ein, oder viele Studenten.
- Was ist eine Studiengang-Instanz?



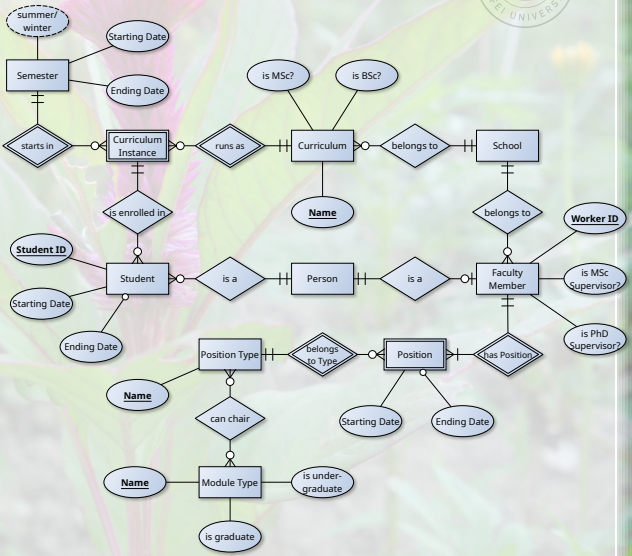
Bigger Picture

- Jeder Student ist in genau eine Studiengang-Instanz (EN: *curriculum instance*) eingeschrieben.
- Zu einer Studiengang-Instanz gehören entweder keine, ein, oder viele Studenten.
- Was ist eine Studiengang-Instanz?
- Wenn die Studenten sich in den Bachelor-Studiengang „Informatik“ einschreiben, dann tun sie das in einer bestimmten Realisierung des Studiengangs (EN: *curriculum*), die in einem bestimmten Semester anfängt – sagen wir in den Informatikstudiengang der im Winter-Semester 2024 anfängt.



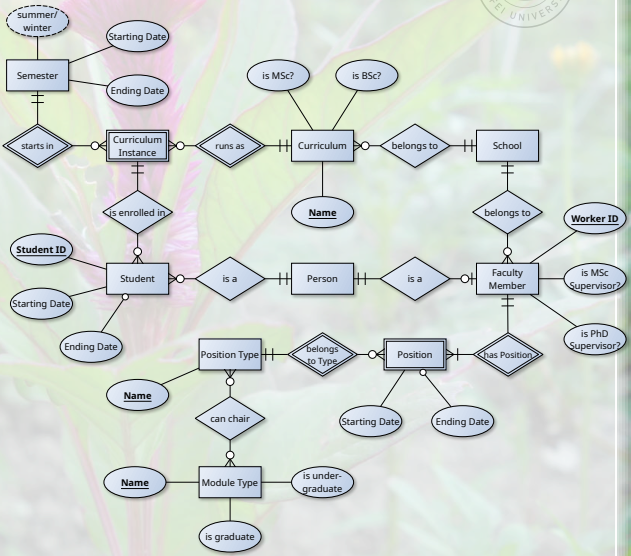
Bigger Picture

- Was ist eine Studiengang-Instanz?
- Wenn die Studenten sich in den Bachelor-Studiengang „Informatik“ einschreiben, dann tun sie das in einer bestimmten Realisierung des Studiengangs (EN: *curriculum*), die in einem bestimmten Semester anfängt – sagen wir in den Informatikstudiengang der im Winter-Semester 2024 anfängt.
- Aus der Perspektive der ERD-modellierung, könnten wir sagen *Ein Studiengang kann von der Universität null, einmal, oder mehrere Male als Studiengangs-Instanz ausgeführt werden.*



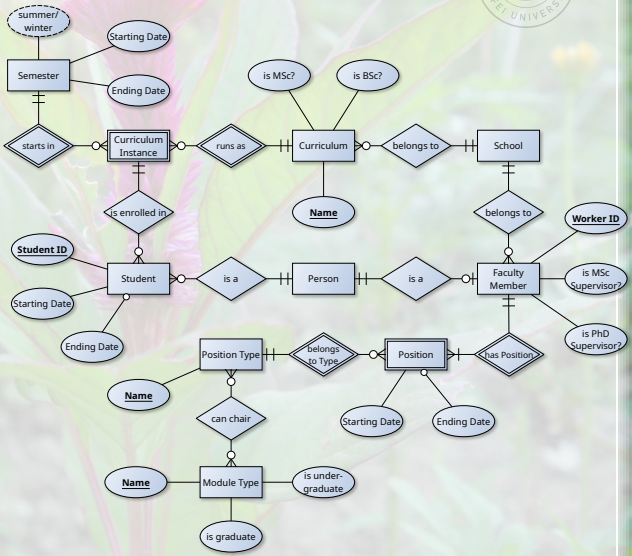
Bigger Picture

- Wenn die Studenten sich in den Bachelor-Studiengang „Informatik“ einschreiben, dann tun sie das in einer bestimmten Realisierung des Studiengangs (EN: *curriculum*), die in einem bestimmten Semester anfängt – sagen wir in den Informatikstudiengang der im Winter-Semester 2024 anfängt.
- Aus der Perspektive der ERD-modellierung, könnten wir sagen *Ein Studiengang kann von der Universität null, einmal, oder mehrere Male als Studiengangs-Instanz ausgeführt werden.*
- Diese Instanz ist meinem bestimmten Anfangssemester assoziiert.



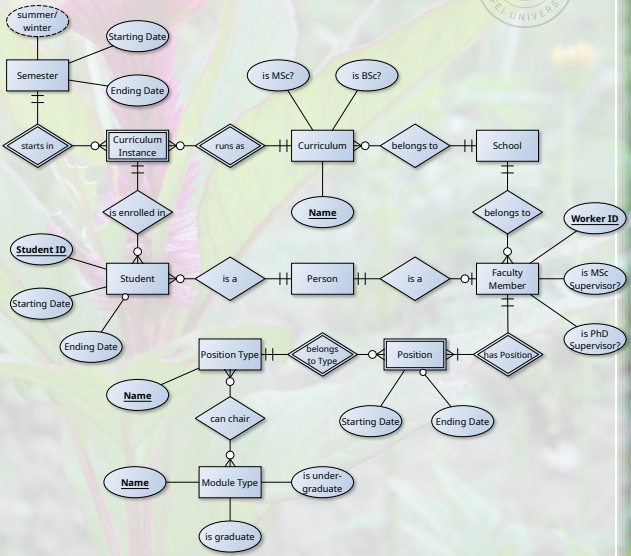
Bigger Picture

- Aus der Perspektive der ERD-modellierung, könnten wir sagen *Ein Studiengang kann von der Universität null, einmal, oder mehrere Male als Studiengangs-Instanz ausgeführt werden.*
- Diese Instanz ist meinem bestimmten Anfangssemester assoziiert.
- Weil Studiengangsinstanzen keine anderen identifizierenden Charakteristika haben und nicht „losgelöst“ existieren können, sind sie schwache Entitäten.



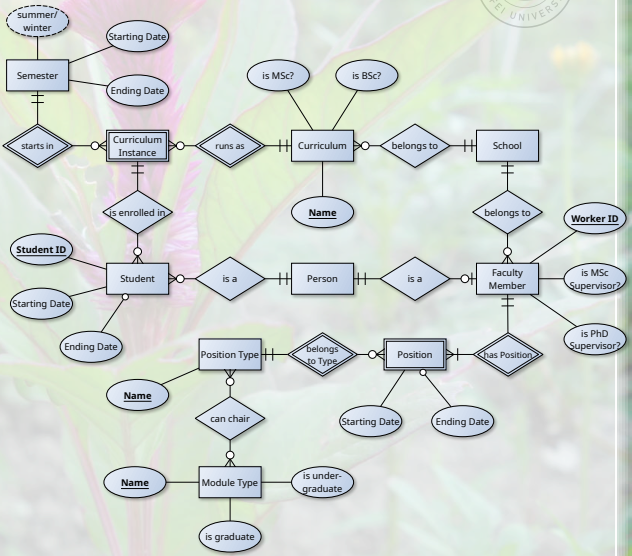
Bigger Picture

- Diese Instanz ist meinem bestimmten Anfangssemester assoziiert.
- Weil Studiengangsinstanzen keine anderen identifizierenden Charakteristika haben und nicht „losgelöst“ existieren können, sind sie schwache Entitäten.
- Sie kommen nur in die Existenz durch ihre identifizierenden Beziehungen mit dem Studiengang, zu dem sie gehören und dem Semester, in dem sie anfangen.



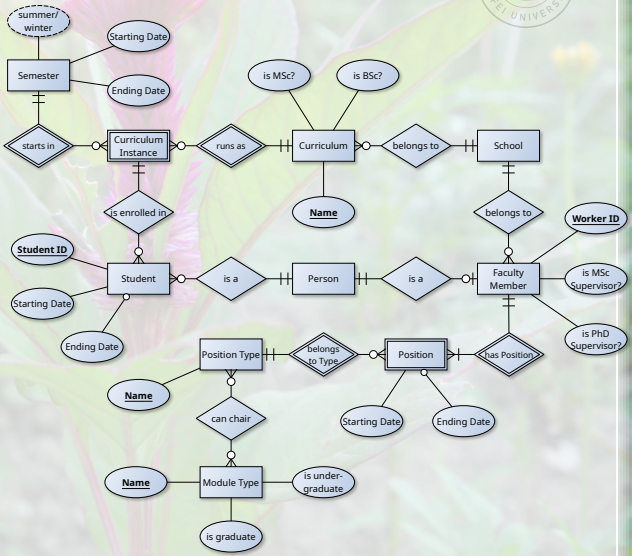
Bigger Picture

- Weil Studiengangsinstanzen keine anderen identifizierenden Charakteristika haben und nicht „losgelöst“ existieren können, sind sie schwache Entitäten.
- Sie kommen nur in die Existenz durch ihre identifizierenden Beziehungen mit dem Studiengang, zu dem sie gehören und dem Semester, in dem sie anfangen.
- Wir haben uns entschieden, Semester als unabhängige Entitäten zu modellieren, denn so können wir Attribute da dran hängen.



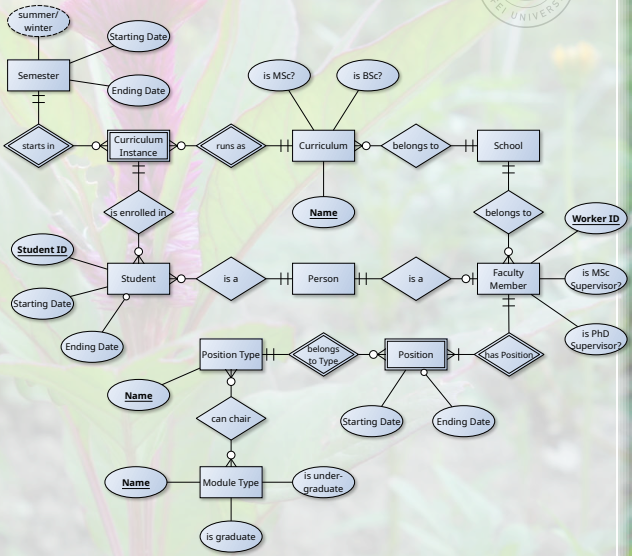
Bigger Picture

- Sie kommen nur in die Existenz durch ihre identifizierenden Beziehungen mit dem Studiengang, zu dem sie gehören und dem Semester, in dem sie anfangen.
- Wir haben uns entschieden, Semester als unabhängige Entitäten zu modellieren, denn so können wir Attribute da dran hängen.
- Ein Semester kann z. B. ein Start- und ein End-Datum haben und auch eine Periode für Prüfungen.



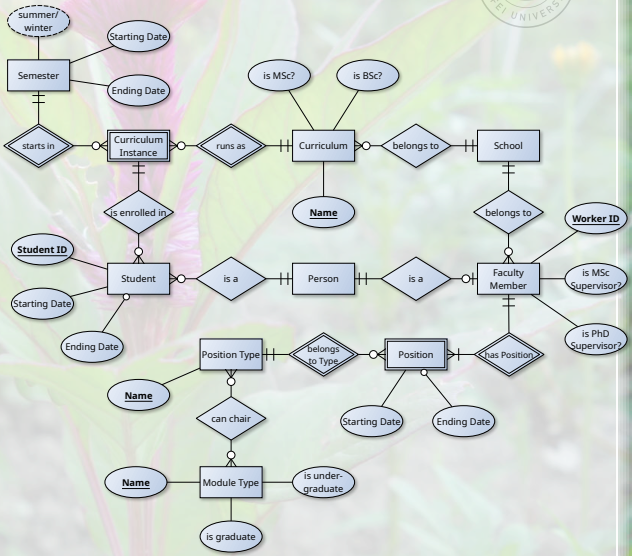
Bigger Picture

- Sie kommen nur in die Existenz durch ihre identifizierenden Beziehungen mit dem Studiengang, zu dem sie gehören und dem Semester, in dem sie anfangen.
- Wir haben uns entschieden, Semester als unabhängige Entitäten zu modellieren, denn so können wir Attribute da dran hängen.
- Ein Semester kann z. B. ein Start- und ein End-Datum haben und auch eine Periode für Prüfungen.
- Aus Start- und End-Datum können wir ableiten, ob ein Semester ein Sommer- oder ein Winter-Semester ist.



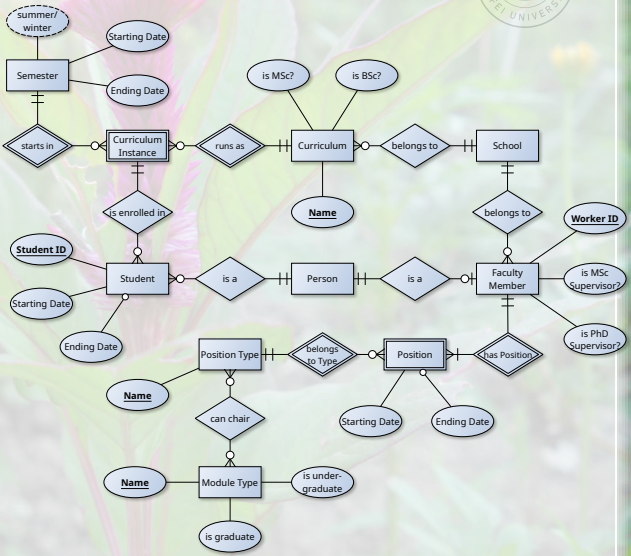
Bigger Picture

- Wir haben uns entschieden, Semester als unabhängige Entitäten zu modellieren, denn so können wir Attribute da dran hängen.
- Ein Semester kann z. B. ein Start- und ein End-Datum haben und auch eine Periode für Prüfungen.
- Aus Start- und End-Datum können wir ableiten, ob ein Semester ein Sommer- oder ein Winter-Semester ist.
- In jedem Semester kann unsere Universität keine, ein, oder viele Instanzen von Studiengängen starten.



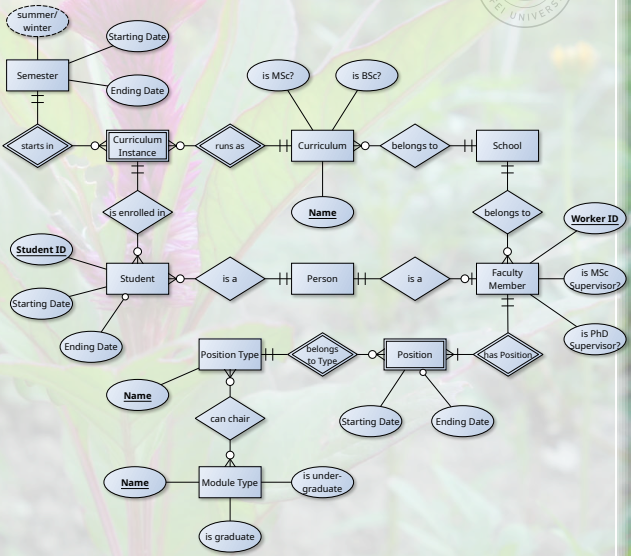
Bigger Picture

- Ein Semester kann z. B. ein Start- und ein End-Datum haben und auch eine Periode für Prüfungen.
- Aus Start- und End-Datum können wir ableiten, ob ein Semester ein Sommer- oder ein Winter-Semester ist.
- In jedem Semester kann unsere Universität keine, ein, oder viele Instanzen von Studiengängen starten.
- Ein Studiengang (EN: *curriculum*) hat einen einmaligen Name, der als Primärschlüssel dient.



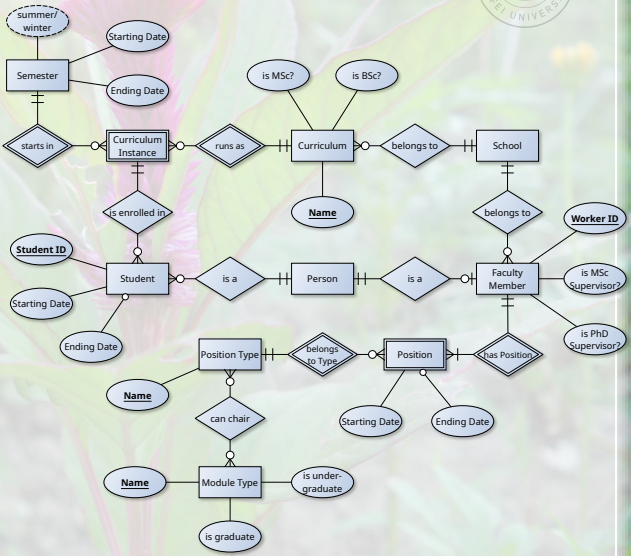
Bigger Picture

- Ein Semester kann z. B. ein Start- und ein End-Datum haben und auch eine Periode für Prüfungen.
- Aus Start- und End-Datum können wir ableiten, ob ein Semester ein Sommer- oder ein Winter-Semester ist.
- In jedem Semester kann unsere Universität keine, ein, oder viele Instanzen von Studiengängen starten.
- Ein Studiengang (EN: *curriculum*) hat einen einmaligen Name, der als Primärschlüssel dient.
- Studiengänge haben auch weitere Attribute, z. B. ob sie Bachelor- oder Master-Programme sind.



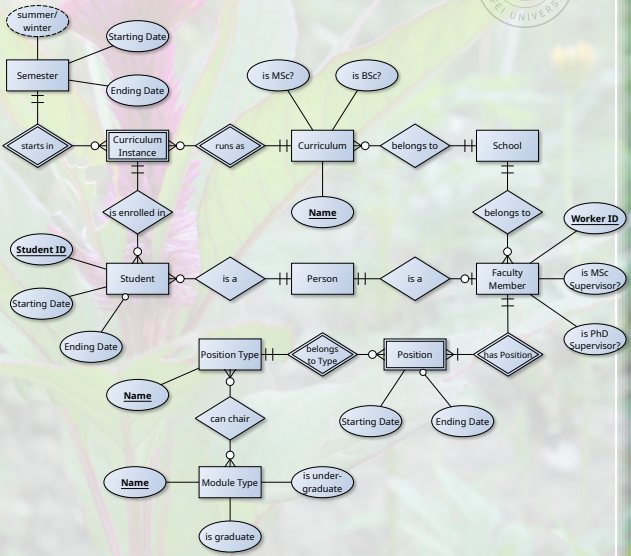
Bigger Picture

- Aus Start- und End-Datum können wir ableiten, ob ein Semester ein Sommer- oder ein Winter-Semester ist.
- In jedem Semester kann unsere Universität keine, ein, oder viele Instanzen von Studiengängen starten.
- Ein Studiengang (EN: *curriculum*) hat einen einmaligen Name, der als Primärschlüssel dient.
- Studiengänge haben auch weitere Attribute, z. B. ob sie Bachelor- oder Master-Programme sind.
- Jeder Studiengang gehört zu genau einer Fakultät (EN: *school*) unserer Universität.



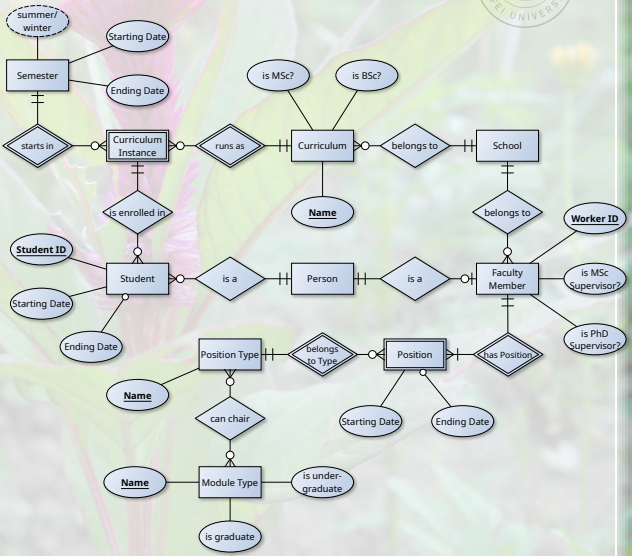
Bigger Picture

- In jedem Semester kann unsere Universität keine, ein, oder viele Instanzen von Studiengängen starten.
- Ein Studiengang (EN: *curriculum*) hat einen einmaligen Name, der als Primärschlüssel dient.
- Studiengänge haben auch weitere Attribute, z. B. ob sie Bachelor- oder Master-Programme sind.
- Jeder Studiengang gehört zu genau einer Fakultät (EN: *school*) unserer Universität.
- Jede Fakultät kann keinen, einen, oder mehrere Studiengänge haben.



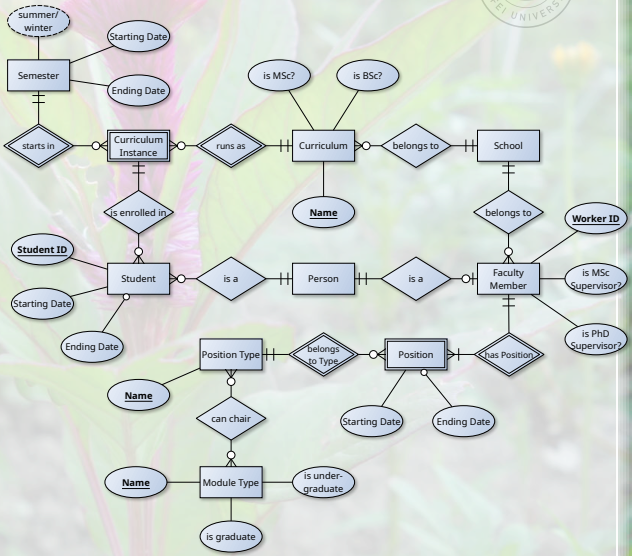
Bigger Picture

- Ein Studiengang (EN: *curriculum*) hat einen einmaligen Name, der als Primärschlüssel dient.
- Studiengänge haben auch weitere Attribute, z. B. ob sie Bachelor- oder Master-Programme sind.
- Jeder Studiengang gehört zu genau einer Fakultät (EN: *school*) unserer Universität.
- Jede Fakultät kann keinen, einen, oder mehrere Studiengänge haben.
- Dieses Modell beschreibt die Situation der Studenten schon recht gut.



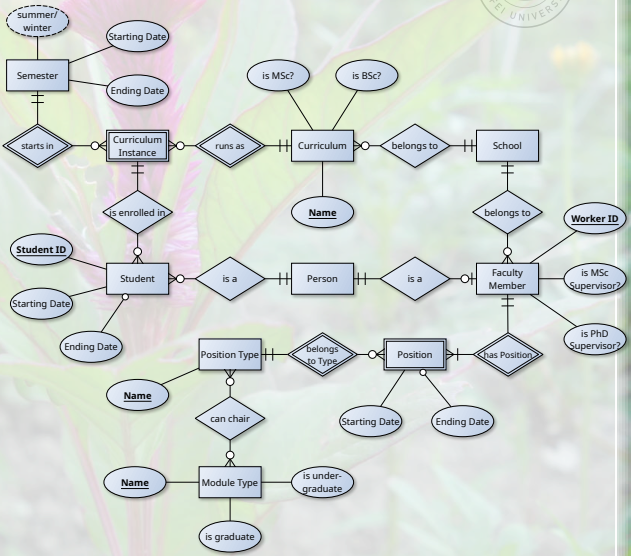
Bigger Picture

- Studiengänge haben auch weitere Attribute, z. B. ob sie Bachelor- oder Master-Programme sind.
- Jeder Studiengang gehört zu genau einer Fakultät (EN: *school*) unserer Universität.
- Jede Fakultät kann keinen, einen, oder mehrere Studiengänge haben.
- Dieses Modell beschreibt die Situation der Studenten schon recht gut.
- In dem wir die Beziehungen verfolgen, wissen wir, welchen Studiengang ein Student studiert.



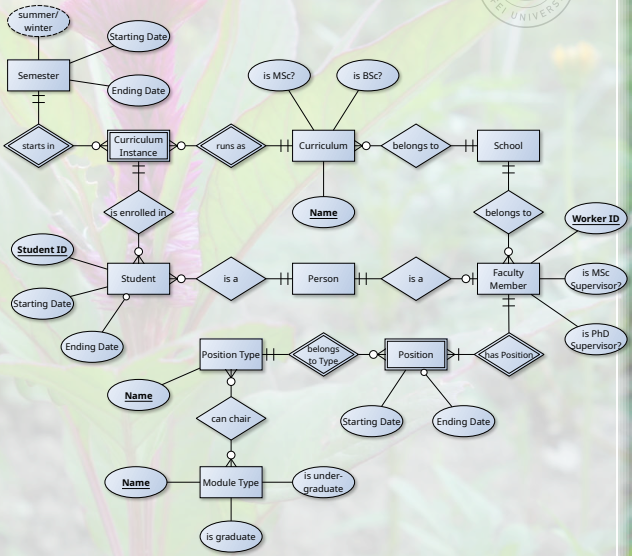
Bigger Picture

- Jeder Studiengang gehört zu genau einer Fakultät (EN: *school*) unserer Universität.
- Jede Fakultät kann keinen, einen, oder mehrere Studiengänge haben.
- Dieses Modell beschreibt die Situation der Studenten schon recht gut.
- In dem wir die Beziehungen verfolgen, wissen wir, welchen Studiengang ein Student studiert.
- Wir haben das hier nicht modelliert, aber Sie können sich vorstellen, dass wir auch wissen können, welche Module zu welchem Studiengang gehören und im wievielten Semester des Studiengangs sie stattfinden.



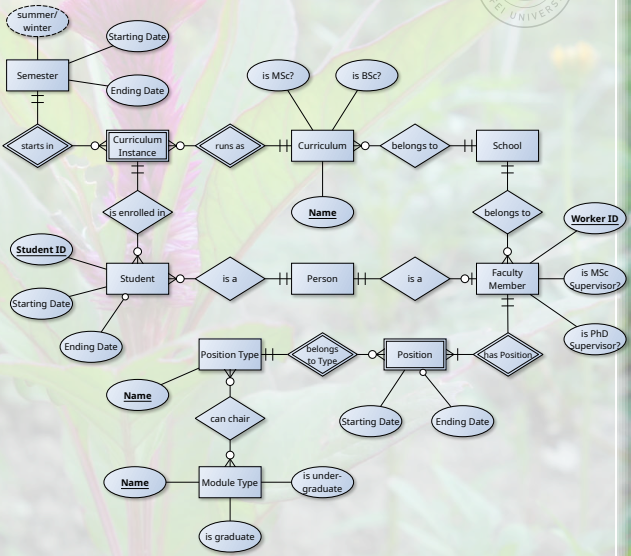
Bigger Picture

- Jede Fakultät kann keinen, einen, oder mehrere Studiengänge haben.
- Dieses Modell beschreibt die Situation der Studenten schon recht gut.
- In dem wir die Beziehungen verfolgen, wissen wir, welchen Studiengang ein Student studiert.
- Wir haben das hier nicht modelliert, aber Sie können sich vorstellen, dass wir auch wissen können, welche Module zu welchem Studiengang gehören und im wievielten Semester des Studiengangs sie stattfinden.
- Daher wüssten wir, wann welcher Student welches Modul anhören muss.



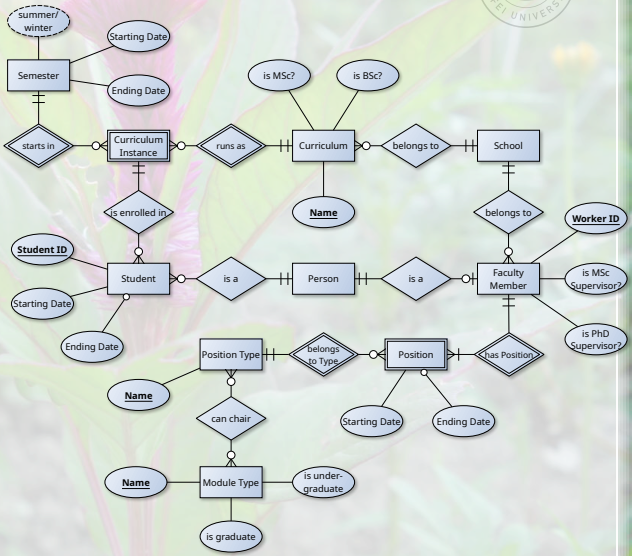
Bigger Picture

- Dieses Modell beschreibt die Situation der Studenten schon recht gut.
- In dem wir die Beziehungen verfolgen, wissen wir, welchen Studiengang ein Student studiert.
- Wir haben das hier nicht modelliert, aber Sie können sich vorstellen, dass wir auch wissen können, welche Module zu welchem Studiengang gehören und im wievielten Semester des Studiengangs sie stattfinden.
- Daher wüssten wir, wann welcher Student welches Modul anhören muss.
- Wir wissen auch, zu welcher Fakultät der Student gehört.



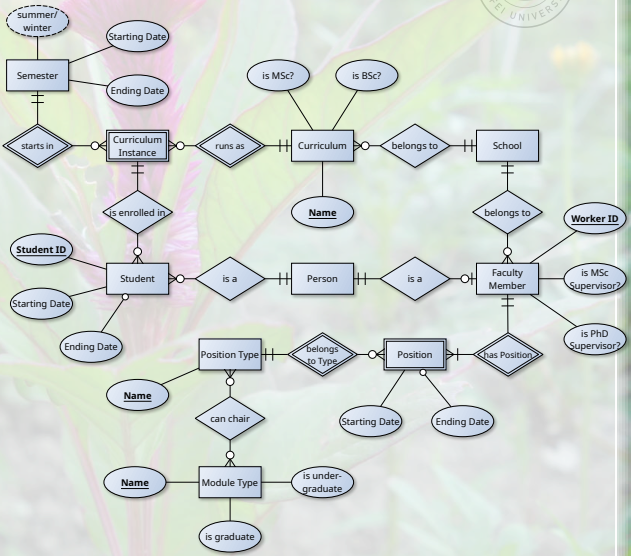
Bigger Picture

- In dem wir die Beziehungen verfolgen, wissen wir, welchen Studiengang ein Student studiert.
- Wir haben das hier nicht modelliert, aber Sie können sich vorstellen, dass wir auch wissen können, welche Module zu welchem Studiengang gehören und im wievielten Semester des Studiengangs sie stattfinden.
- Daher wüssten wir, wann welcher Student welches Modul anhören muss.
- Wir wissen auch, zu welcher Fakultät der Student gehört.
- Daher können wir auch die Menge der Professoren, die ihn betreuen können konstruieren.



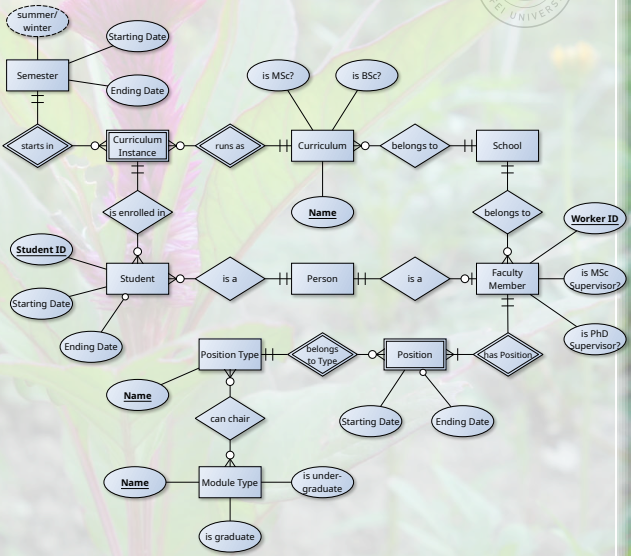
Bigger Picture

- Wir haben das hier nicht modelliert, aber Sie können sich vorstellen, dass wir auch wissen können, welche Module zu welchem Studiengang gehören und im wievielten Semester des Studiengangs sie stattfinden.
- Daher wüssten wir, wann welcher Student welches Modul anhören muss.
- Wir wissen auch, zu welcher Fakultät der Student gehört.
- Daher können wir auch die Menge der Professoren, die ihn betreuen können konstruieren.
- Modellieren wir nun auch etwas die Situation der Mitarbeiter (EN: *faculty members*) der Uni.



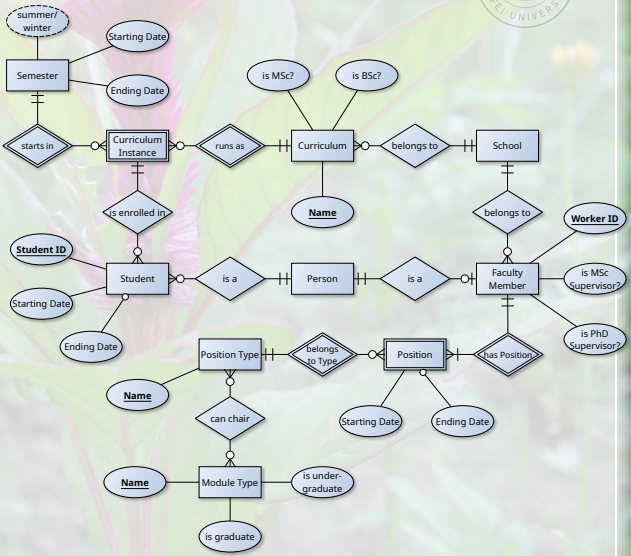
Bigger Picture

- Daher wüssten wir, wann welcher Student welches Modul anhören muss.
- Wir wissen auch, zu welcher Fakultät der Student gehört.
- Daher können wir auch die Menge der Professoren, die ihn betreuen können konstruieren.
- Modellieren wir nun auch etwas die Situation der Mitarbeiter (EN: *faculty members*) der Uni.
- Jede Person kann höchstens ein Mitarbeiter sein.



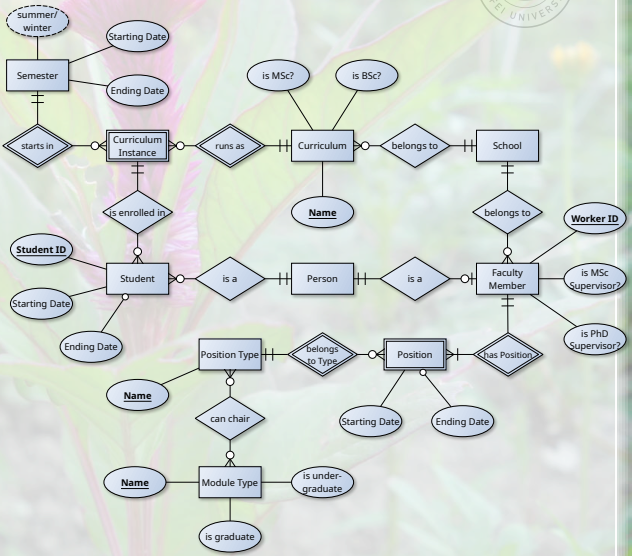
Bigger Picture

- Wir wissen auch, zu welcher Fakultät der Student gehört.
- Daher können wir auch die Menge der Professoren, die ihn betreuen können konstruieren.
- Modellieren wir nun auch etwas die Situation der Mitarbeiter (EN: *faculty members*) der Uni.
- Jede Person kann höchstens ein Mitarbeiter sein.
- Jeder Mitarbeiter ist genau eine Person und hat eine einmalige Arbeiternummer (EN: *worker's ID*).



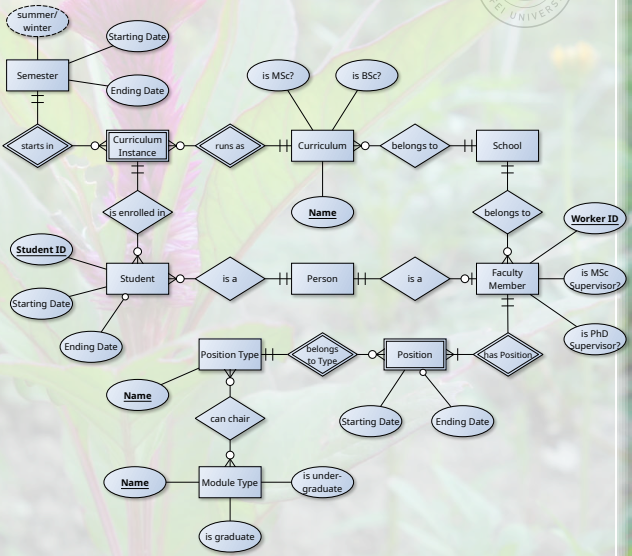
Bigger Picture

- Daher können wir auch die Menge der Professoren, die ihn betreuen können konstruieren.
- Modellieren wir nun auch etwas die Situation der Mitarbeiter (EN: *faculty members*) der Uni.
- Jede Person kann höchstens ein Mitarbeiter sein.
- Jeder Mitarbeiter ist genau eine Person und hat eine einmalige Arbeiternummer (EN: *worker's ID*).
- Das ist anders als die Situation der Studenten.



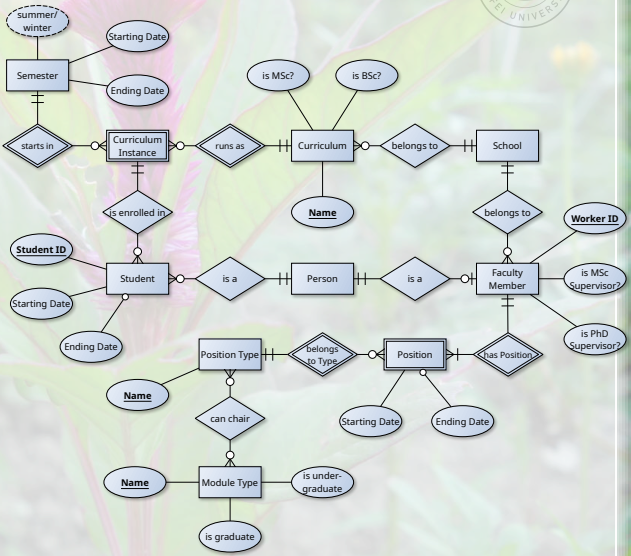
Bigger Picture

- Modellieren wir nun auch etwas die Situation der Mitarbeiter (EN: *faculty members*) der Uni.
- Jede Person kann höchstens ein Mitarbeiter sein.
- Jeder Mitarbeiter ist genau eine Person und hat eine einmalige Arbeiternummer (EN: *worker's ID*).
- Das ist anders als die Situation der Studenten.
- Ein Student bekommt eine neue Studenten-ID, jedesmal, wenn er sich in einen Studiengang einschreibt.



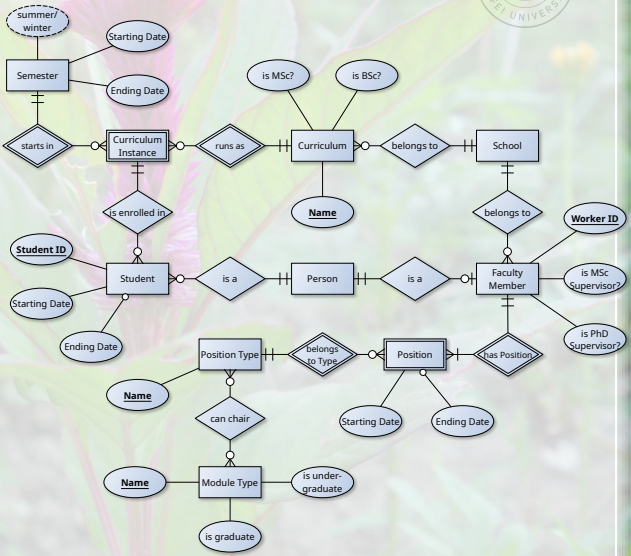
Bigger Picture

- Jede Person kann höchstens ein Mitarbeiter sein.
- Jeder Mitarbeiter ist genau eine Person und hat eine einmalige Arbeiternummer (EN: *worker's ID*).
- Das ist anders als die Situation der Studenten.
- Ein Student bekommt eine neue Studenten-ID, jedesmal, wenn er sich in einen Studiengang einschreibt.
- Normalerweise studiert ein Student nur einen Studiengang in unserer Uni.



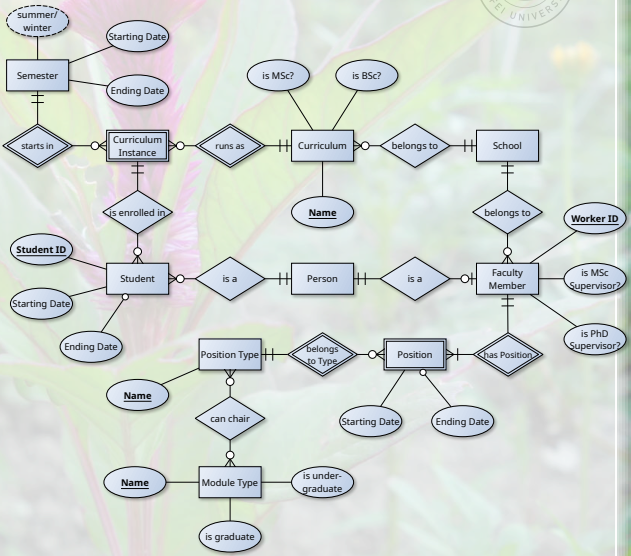
Bigger Picture

- Jeder Mitarbeiter ist genau eine Person und hat eine einmalige Arbeiternummer (EN: *worker's ID*).
- Das ist anders als die Situation der Studenten.
- Ein Student bekommt eine neue Studenten-ID, jedesmal, wenn er sich in einen Studiengang einschreibt.
- Normalerweise studiert ein Student nur einen Studiengang in unserer Uni.
- Ein Mitarbeiter wird immer die selbe Arbeiternummer behalten.



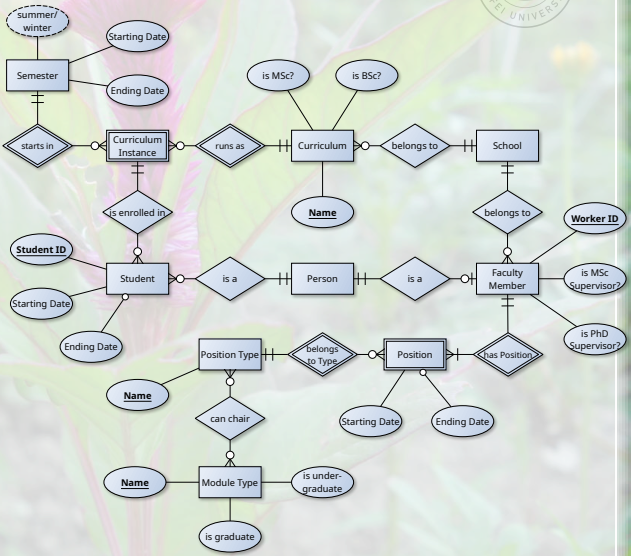
Bigger Picture

- Das ist anders als die Situation der Studenten.
- Ein Student bekommt eine neue Studenten-ID, jedesmal, wenn er sich in einen Studiengang einschreibt.
- Normalerweise studiert ein Student nur einen Studiengang in unserer Uni.
- Ein Mitarbeiter wird immer die selbe Arbeiternummer behalten.
- Ein Mitarbeiter kann befördert werden oder seine Position ändern, behält aber die gleich Arbeiternummer.



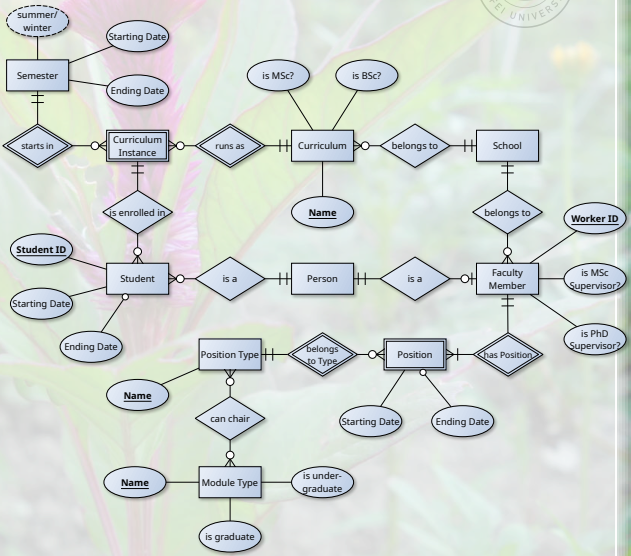
Bigger Picture

- Ein Student bekommt eine neue Studenten-ID, jedesmal, wenn er sich in einen Studiengang einschreibt.
- Normalerweise studiert ein Student nur einen Studiengang in unserer Uni.
- Ein Mitarbeiter wird immer die selbe Arbeiternummer behalten.
- Ein Mitarbeiter kann befördert werden oder seine Position ändern, behält aber die gleich Arbeiternummer.
- Es gibt verschiedene Arten von Positionen (EN: *position types*) in unserer Uni.



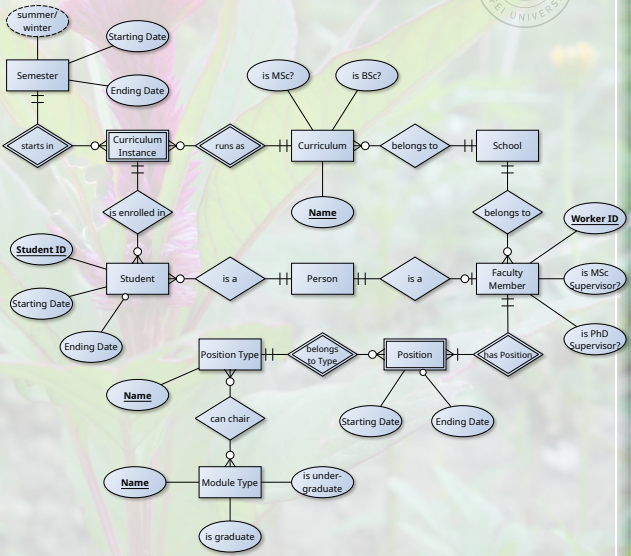
Bigger Picture

- Ein Mitarbeiter wird immer die selbe Arbeiternummer behalten.
- Ein Mitarbeiter kann befördert werden oder seine Position ändern, behält aber die gleich Arbeiternummer.
- Es gibt verschiedene Arten von Positionen (EN: *position types*) in unserer Uni.
- Z. B. Lehrer (EN: *lecturer*), Assitenzprofessor (EN: *assistant professor*), außerordentlicher Professor (EN: *associate professor*), Professor (EN: *full professor*) und vielleicht sogar verschiedene Level von Professor.



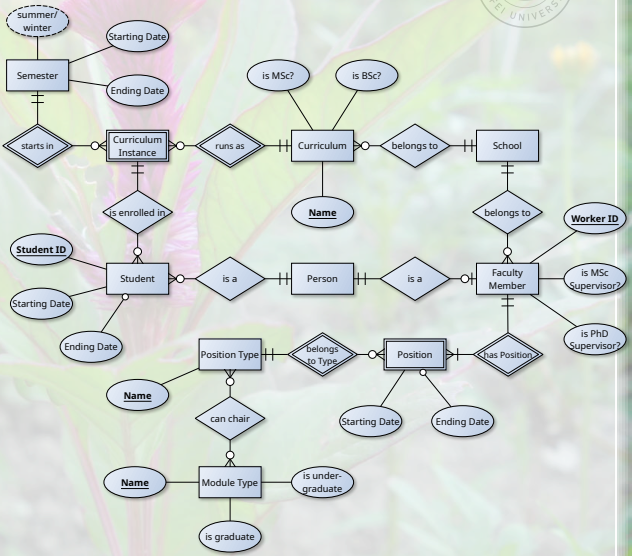
Bigger Picture

- Ein Mitarbeiter kann befördert werden oder seine Position ändern, behält aber die gleich Arbeiternummer.
- Es gibt verschiedene Arten von Positionen (EN: *position types*) in unserer Uni.
- Z. B. Lehrer (EN: *lecturer*), Assitenzprofessor (EN: *assistant professor*), außerordentlicher Professor (EN: *associate professor*), Professor (EN: *full professor*) und vielleicht sogar verschiedene Level von Professor.
- Jede Position hat einen eindeutigen Namen und bestimmt die Dinge, die ein Mitarbeiter machen darf.



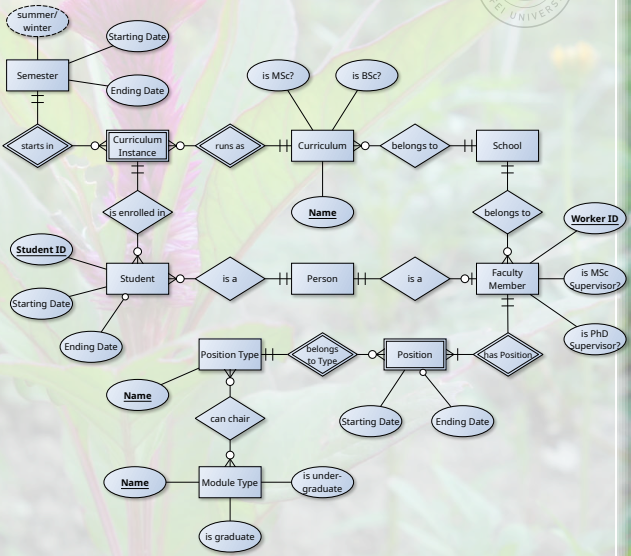
Bigger Picture

- Jede Position hat einen eindeutigen Namen und bestimmt die Dinge, die ein Mitarbeiter machen darf.
- Wir modellieren drei Arten von Modulen, z. B. Fachgrundlagenmodule (学科基础课), Allgemeinbildende Pflichtfächer (公共基础课), Fachspezifische Basismodule (专业基础课), Fachspezifische Wahlpflichtmodule (专业选修课), oder Allgemeinbildende Wahlfächer (公共选修课).



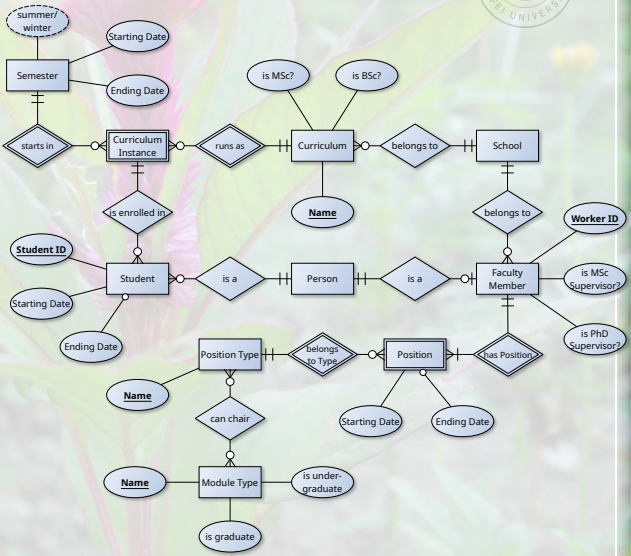
Bigger Picture

- Jede Position hat einen eindeutigen Namen und bestimmt die Dinge, die ein Mitarbeiter machen darf.
- Wir modellieren drei Arten von Modulen, z. B. Fachgrundlagenmodule (学科基础课), Allgemeinbildende Pflichtfächer (公共基础课), Fachspezifische Basismodule (专业基础课), Fachspezifische Wahlpflichtmodule (专业选修课), oder Allgemeinbildende Wahlfächer (公共选修课).
- Eine Position erlaubt einer Person in der Position, Module bestimmter Modultypen zu leiten.



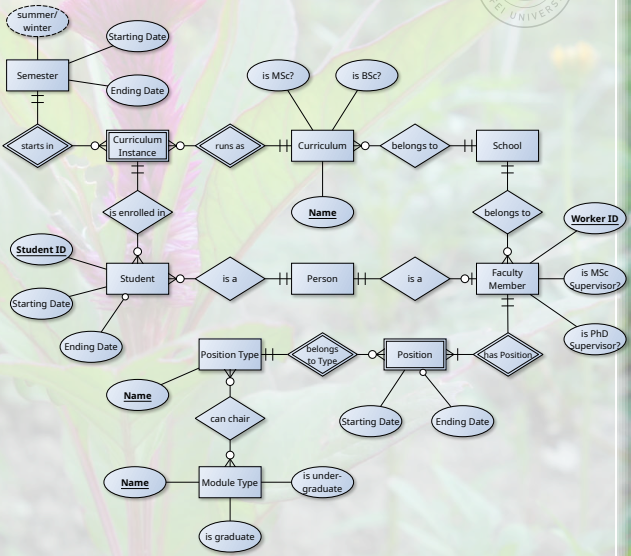
Bigger Picture

- Wir modellieren drei Arten von Modulen, z. B. Fachgrundlagenmodule (学科基础课), Allgemeinbildende Pflichtfächer (公共基础课), Fachspezifische Basismodule (专业基础课), Fachspezifische Wahlpflichtmodule (专业选修课), oder Allgemeinbildende Wahlfächer (公共选修课).
- Eine Position erlaubt einer Person in der Position, Module bestimmter Modultypen zu leiten.
- (Module jedes) Modultyps können von Mitarbeitern von keinem, einem, oder mehreren Positionstypen geleitet werden.



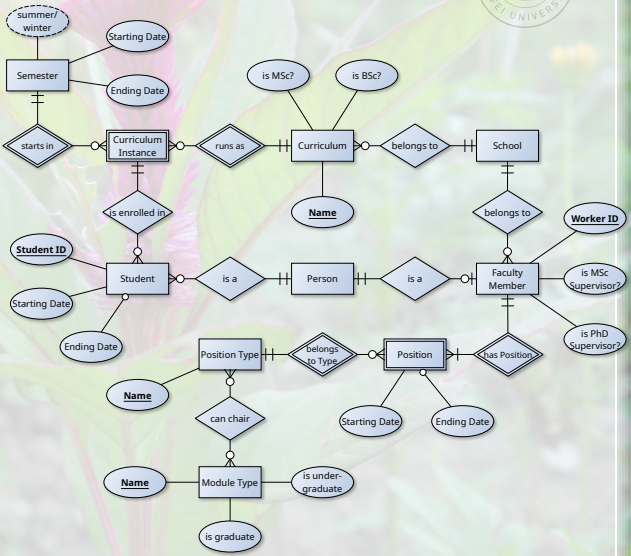
Bigger Picture

- Eine Position erlaubt einer Person in der Position, Module bestimmter Modultypen zu leiten.
- (Module jedes) Modultyps können von Mitarbeitern von keinem, einem, oder mehreren Positionstypen geleitet werden.
- Wir führen die schwache Entität Position ein, um Mitarbeiter und Positionstypen zu verbinden.



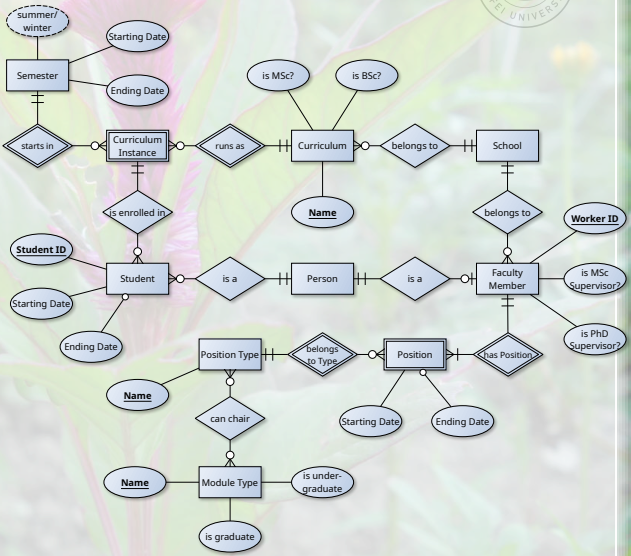
Bigger Picture

- Eine Position erlaubt einer Person in der Position, Module bestimmter Modultypen zu leiten.
- (Module jedes) Modultyps können von Mitarbeitern von keinem, einem, oder mehreren Positionstypen geleitet werden.
- Wir führen die schwache Entität Position ein, um Mitarbeiter und Positionstypen zu verbinden.
- Jede Instanz dieses schwachen Entitätstyps gehört zu genau einem Mitarbeiter und genau einem Positionstyp.



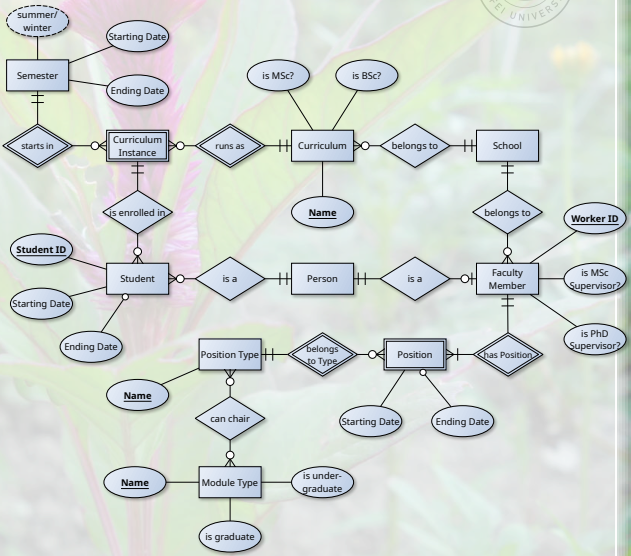
Bigger Picture

- Eine Position erlaubt einer Person in der Position, Module bestimmter Modultypen zu leiten.
- (Module jedes) Modultyps können von Mitarbeitern von keinem, einem, oder mehreren Positionstypen geleitet werden.
- Wir führen die schwache Entität Position ein, um Mitarbeiter und Positionstypen zu verbinden.
- Jede Instanz dieses schwachen Entitätstyps gehört zu genau einem Mitarbeiter und genau einem Positionstyp.
- Jeder Mitarbeiter hat mindestens eine Position.



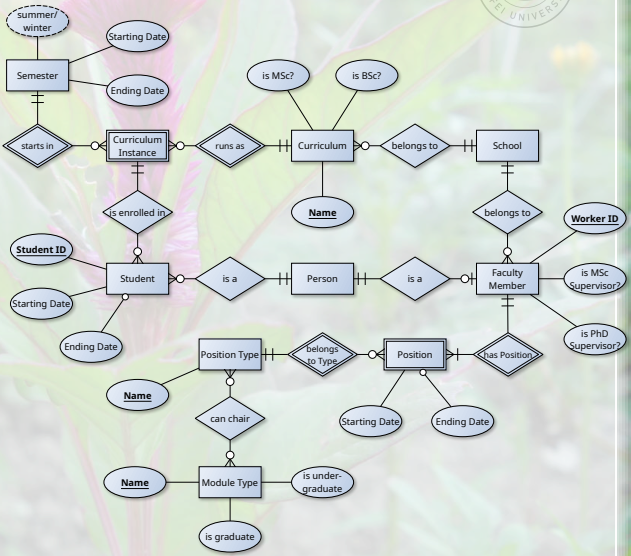
Bigger Picture

- (Module jedes) Modultyps können von Mitarbeitern von keinem, einem, oder mehreren Positionstypen geleitet werden.
- Wir führen die schwache Entität Position ein, um Mitarbeiter und Positionstypen zu verbinden.
- Jede Instanz dieses schwachen Entitätstyps gehört zu genau einem Mitarbeiter und genau einem Positionstyp.
- Jeder Mitarbeiter hat mindestens eine Position.
- Normalerweise hat ein Mitarbeiter genau eine Position zu einer Zeit.



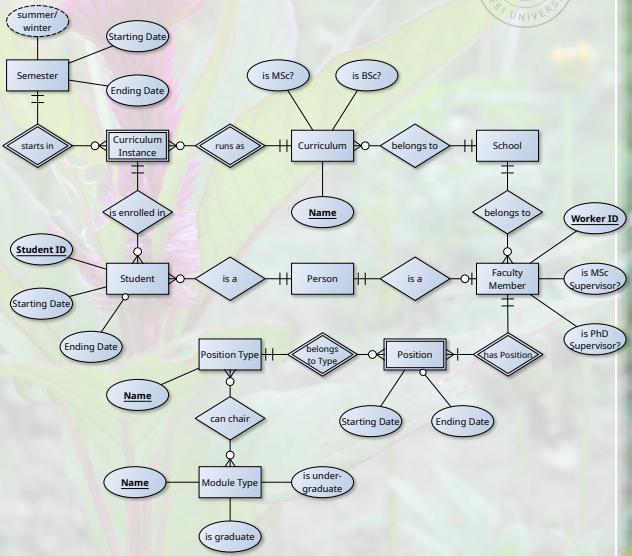
Bigger Picture

- Wir führen die schwache Entität Position ein, um Mitarbeiter und Positionstypen zu verbinden.
- Jede Instanz dieses schwachen Entitätstyps gehört zu genau einem Mitarbeiter und genau einem Positionstyp.
- Jeder Mitarbeiter hat mindestens eine Position.
- Normalerweise hat ein Mitarbeiter genau eine Position zu einer Zeit.
- Deshalb hat jede Position ein Start- und End-Datum.



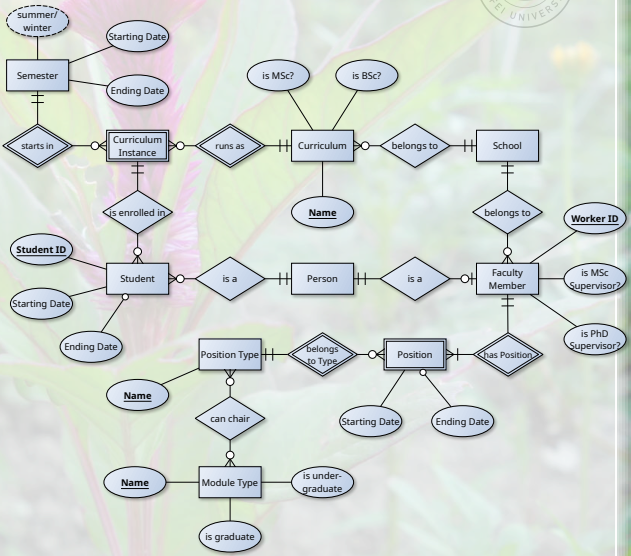
Bigger Picture

- Jede Instanz dieses schwachen Entitätstyps gehört zu genau einem Mitarbeiter und genau einem Positionstyp.
- Jeder Mitarbeiter hat mindestens eine Position.
- Normalerweise hat ein Mitarbeiter genau eine Position zu einer Zeit.
- Deshalb hat jede Position ein Start- und End-Datum.
- Mitarbeiter können über die Zeit hinweg verschiedene Positionen begleiten.



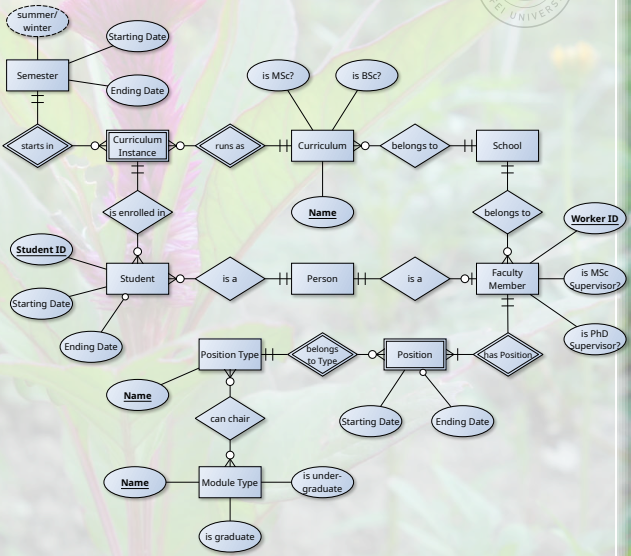
Bigger Picture

- Jeder Mitarbeiter hat mindestens eine Position.
- Normalerweise hat ein Mitarbeiter genau eine Position zu einer Zeit.
- Deshalb hat jede Position ein Start- und End-Datum.
- Mitarbeiter können über die Zeit hinweg verschiedene Positionen begleiten.
- Vielleicht fangen sie 2022 als Lecturer an und werden 2025 zum außerordentlichen Professor befördert.



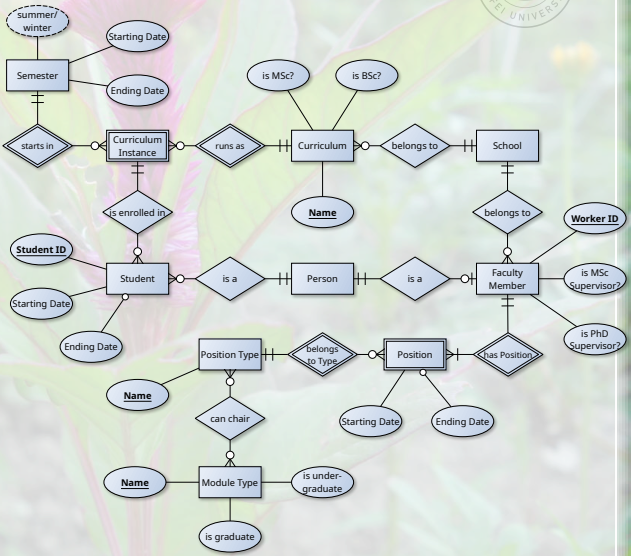
Bigger Picture

- Normalerweise hat ein Mitarbeiter genau eine Position zu einer Zeit.
- Deshalb hat jede Position ein Start- und End-Datum.
- Mitarbeiter können über die Zeit hinweg verschiedene Positionen begleiten.
- Vielleicht fangen sie 2022 als Lecturer an und werden 2025 zum außerordentlichen Professor befördert.
- Für jeden Positionstyp kann es beliebig viele assoziierte Positions-Datensätze geben.



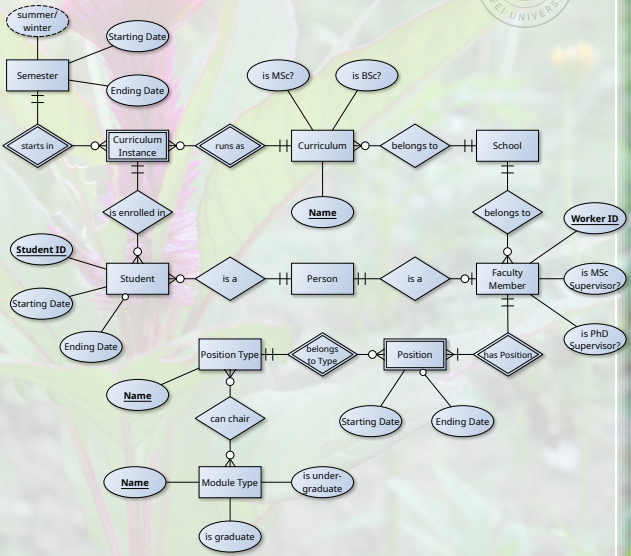
Bigger Picture

- Deshalb hat jede Position ein Start- und End-Datum.
- Mitarbeiter können über die Zeit hinweg verschiedene Positionen begleiten.
- Vielleicht fangen sie 2022 als Lecturer an und werden 2025 zum außerordentlichen Professor befördert.
- Für jeden Positionstyp kann es beliebig viele assoziierte Positions-Datensätze geben.
- Interessanterweise hängt es nicht unbedingt von dem Position ab, ob ein Mitarbeiter bestimmte Studenten betreuen kann.



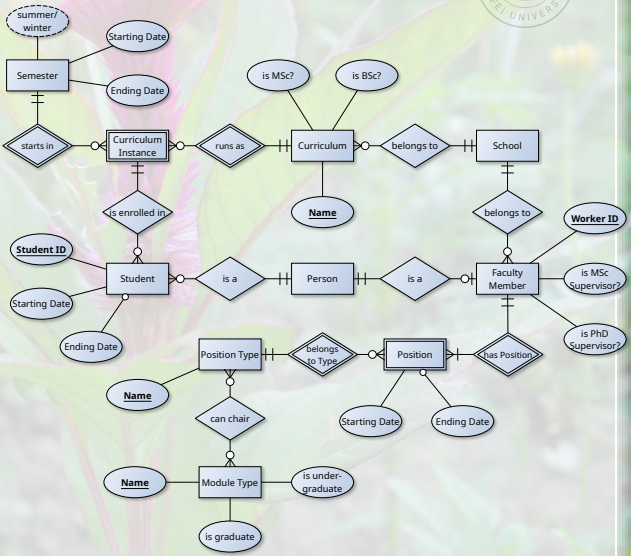
Bigger Picture

- Mitarbeiter können über die Zeit hinweg verschiedene Positionen begleiten.
- Vielleicht fangen sie 2022 als Lecturer an und werden 2025 zum außerordentlichen Professor befördert.
- Für jeden Positionstyp kann es beliebig viele assoziierte Positions-Datensätze geben.
- Interessanterweise hängt es nicht unbedingt von dem Position ab, ob ein Mitarbeiter bestimmte Studenten betreuen kann.
- Diese Fähigkeiten sind Attribute der Person.



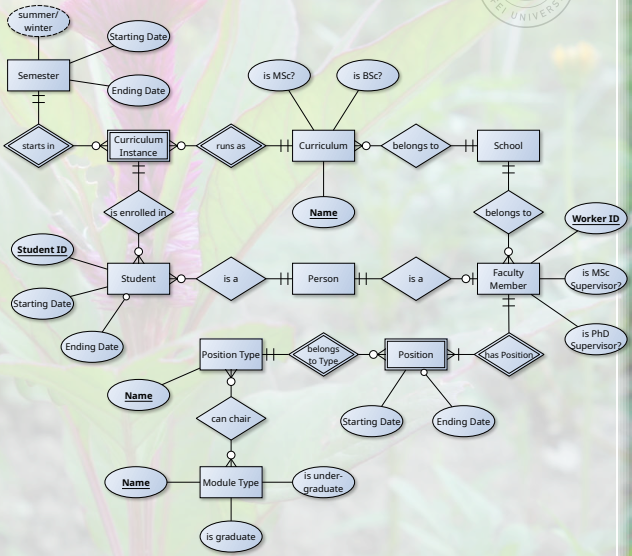
Bigger Picture

- Vielleicht fangen sie 2022 als Lecturer an und werden 2025 zum außerordentlichen Professor befördert.
- Für jeden Positionstyp kann es beliebig viele assoziierte Positions-Datensätze geben.
- Interessanterweise hängt es nicht unbedingt von dem Position ab, ob ein Mitarbeiter bestimmte Studenten betreuen kann.
- Diese Fähigkeiten sind Attribute der Person.
- Es könnten bestimmte Positionen sein, wie z. B. Professor für MSc Studenten-Betreuung.



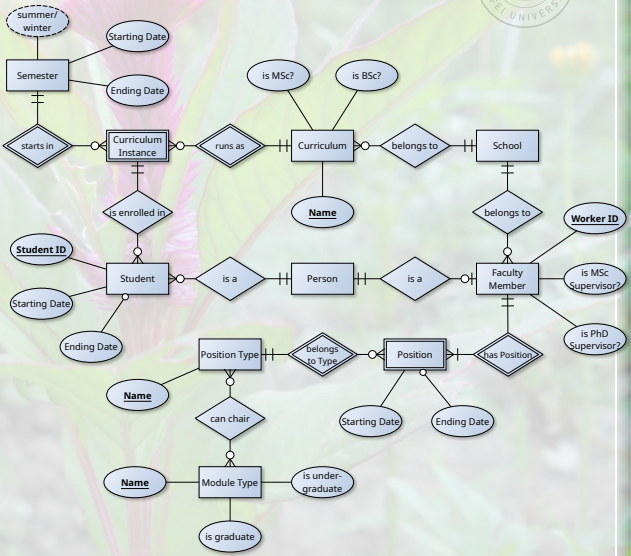
Bigger Picture

- Für jeden Positionstyp kann es beliebig viele assoziierte Positions-Datensätze geben.
- Interessanterweise hängt es nicht unbedingt von dem Position ab, ob ein Mitarbeiter bestimmte Studenten betreuen kann.
- Diese Fähigkeiten sind Attribute der Person.
- Es könnten bestimmte Positionen sein, wie z. B. Professor für MSc Studenten-Betreuung.
- Es kann zusätzliche Anforderungen geben, wie Publikationen, Lehre, usw.



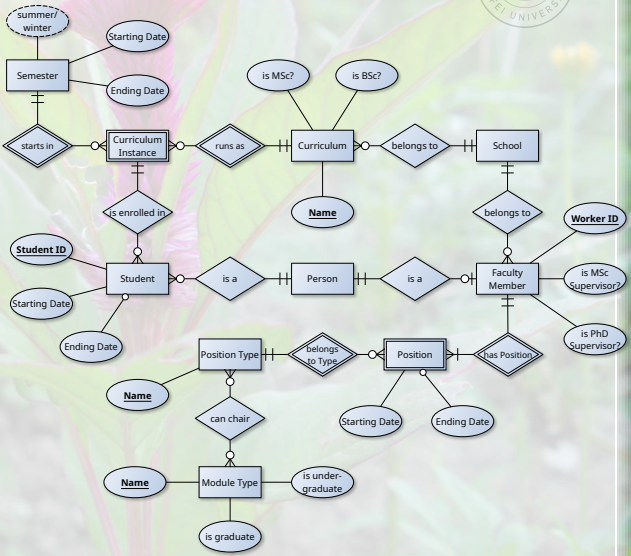
Bigger Picture

- Interessanterweise hängt es nicht unbedingt von dem Position ab, ob ein Mitarbeiter bestimmte Studenten betreuen kann.
- Diese Fähigkeiten sind Attribute der Person.
- Es könnten bestimmte Positionen sein, wie z. B. Professor für MSc Studenten-Betreuung.
- Es kann zusätzliche Anforderungen geben, wie Publikationen, Lehre, usw.
- Zuletzt modellieren wir noch, dass ein Mitarbeiter zu genau einer Fakultät gehört.



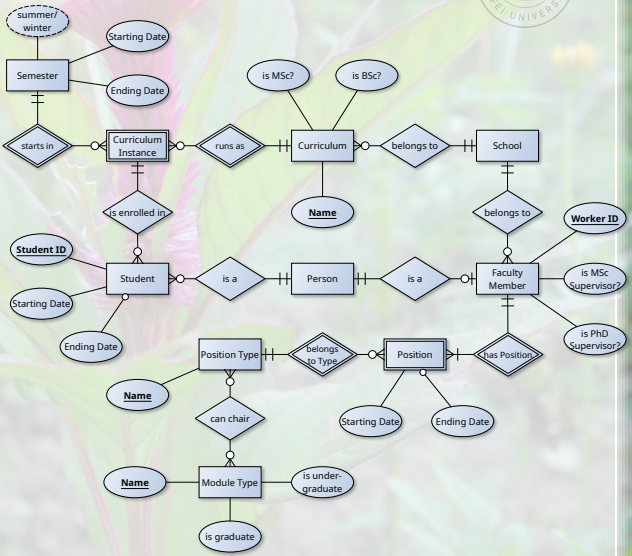
Bigger Picture

- Diese Fähigkeiten sind Attribute der Person.
- Es könnten bestimmte Positionen sein, wie z. B. Professor für MSc Studenten-Betreuung.
- Es kann zusätzliche Anforderungen geben, wie Publikationen, Lehre, usw.
- Zuletzt modellieren wir noch, dass ein Mitarbeiter zu genau einer Fakultät gehört.
- Eine Fakultät kann beliebig viele Mitarbeiter haben.



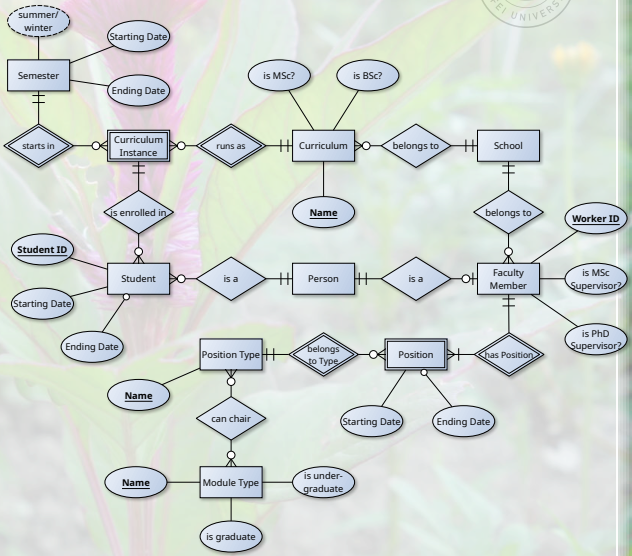
Bigger Picture

- Es könnten bestimmte Positionen sein, wie z. B. Professor für MSc Studenten-Betreuung.
- Es kann zusätzliche Anforderungen geben, wie Publikationen, Lehre, usw.
- Zuletzt modellieren wir noch, dass ein Mitarbeiter zu genau einer Fakultät gehört.
- Eine Fakultät kann beliebig viele Mitarbeiter haben.
- Wenn wir unser neues ERD anschauen, dann ist es wunderschön.



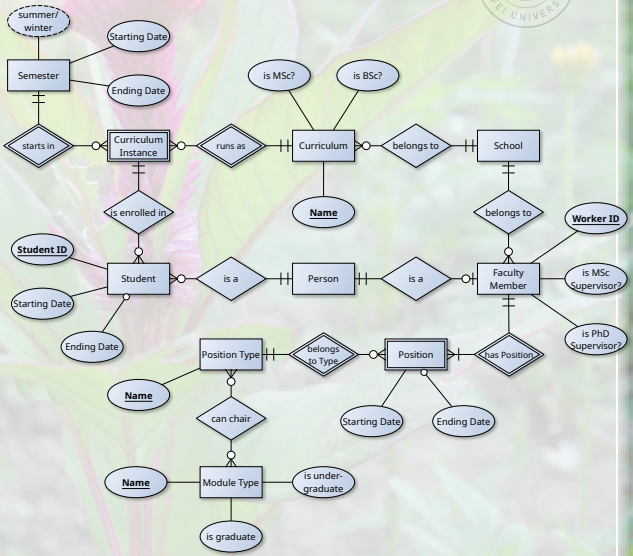
Bigger Picture

- Es kann zusätzliche Anforderungen geben, wie Publikationen, Lehre, usw.
- Zuletzt modellieren wir noch, dass ein Mitarbeiter zu genau einer Fakultät gehört.
- Eine Fakultät kann beliebig viele Mitarbeiter haben.
- Wenn wir unser neues ERD anschauen, dann ist es wunderschön.
- Wir sehen aber auch, dass viele Dinge noch fehlen.



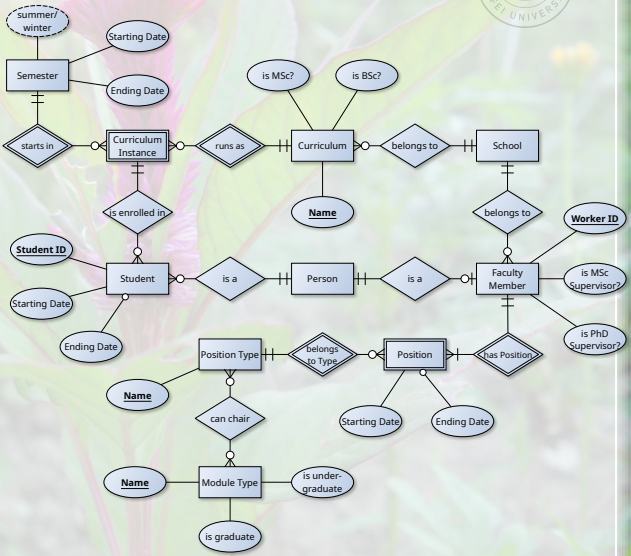
Bigger Picture

- Zuletzt modellieren wir noch, dass ein Mitarbeiter zu genau einer Fakultät gehört.
- Eine Fakultät kann beliebig viele Mitarbeiter haben.
- Wenn wir unser neues ERD anschauen, dann ist es wunderschön.
- Wir sehen aber auch, dass viele Dinge noch fehlen.
- Die Beziehungen zwischen Studenten, Vorlesungen, Modulen, Studiengängen, Räumen, Raumeigenschaften, Prüfungen, Abschlussanforderungen, usw. wurden noch nicht modelliert.



Bigger Picture

- Eine Fakultät kann beliebig viele Mitarbeiter haben.
- Wenn wir unser neues ERD anschauen, dann ist es wunderschön.
- Wir sehen aber auch, dass viele Dinge noch fehlen.
- Die Beziehungen zwischen Studenten, Vorlesungen, Modulen, Studiengängen, Räumen, Raumeigenschaften, Prüfungen, Abschlussanforderungen, usw. wurden noch nicht modelliert.
- Wir sind aber schon viel weiter als vorher.



Zusammenfassung

Zusammenfassung

- Die Fähigkeit, Beziehungstypen mit Modalitäten und Kardinalitäten zu annotieren ist sehr wichtig.

Zusammenfassung

- Die Fähigkeit, Beziehungstypen mit Modalitäten und Kardinalitäten zu annotieren ist sehr wichtig.
- Sie macht unsere Modelle sowohl klarer als auch genauer.

Zusammenfassung

- Die Fähigkeit, Beziehungstypen mit Modalitäten und Kardinalitäten zu annotieren ist sehr wichtig.
- Sie macht unsere Modelle sowohl klarer als auch genauer.
- Ohne sie können wir nicht ausdrücken, ob eine Person mehrere Studiengänge studieren kann oder nicht.

Zusammenfassung

- Die Fähigkeit, Beziehungstypen mit Modalitäten und Kardinalitäten zu annotieren ist sehr wichtig.
- Sie macht unsere Modelle sowohl klarer als auch genauer.
- Ohne sie können wir nicht ausdrücken, ob eine Person mehrere Studiengänge studieren kann oder nicht.
- Ohne sie können wir nicht ausdrücken, dass eine Person beliebig viele IDs eines ID-Typs haben kann aber das jede ID zu genau einem ID-Typ gehört.

Zusammenfassung

- Die Fähigkeit, Beziehungstypen mit Modalitäten und Kardinalitäten zu annotieren ist sehr wichtig.
- Sie macht unsere Modelle sowohl klarer als auch genauer.
- Ohne sie können wir nicht ausdrücken, ob eine Person mehrere Studiengänge studieren kann oder nicht.
- Ohne sie können wir nicht ausdrücken, dass eine Person beliebig viele IDs eines ID-Typs haben kann aber das jede ID zu genau einem ID-Typ gehört.
- Wir machen große Schritte in die Richtung, echte Realwelt-Szenarios modellieren zu können.

谢谢您们！

Thank you!

Vielen Dank!

References I

[1] Adam Aspin und Karine Aspin. *Query Answers with MariaDB – Volume I: Introduction to SQL Queries*. Tetras Publishing, Okt. 2018. ISBN: 978-1-9996172-4-0. See also[2] (siehe S. 189, 202).

[2] Adam Aspin und Karine Aspin. *Query Answers with MariaDB – Volume II: In-Depth Querying*. Tetras Publishing, Okt. 2018. ISBN: 978-1-9996172-5-7. See also[1] (siehe S. 189, 202).

[3] 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. 5–13).

[4] 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. 201).

[5] Daniel J. Barrett. *Efficient Linux at the Command Line*. Sebastopol, CA, USA: O'Reilly Media, Inc., Feb. 2022. ISBN: 978-1-0981-1340-7 (siehe S. 202, 204).

[6] Daniel Bartholomew. *Learning the MariaDB Ecosystem: Enterprise-level Features for Scalability and Availability*. New York, NY, USA: Apress Media, LLC, Okt. 2019. ISBN: 978-1-4842-5514-8 (siehe S. 202).

[7] Ben Beitler. *Hands-On Microsoft Access 2019*. Birmingham, England, UK: Packt Publishing Ltd, März 2020. ISBN: 978-1-83898-747-3 (siehe S. 202).

[8] 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. 204).

[9] 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. 201).

[10] Bernard Obeng Boateng. *Data Modeling with Microsoft Excel*. Birmingham, England, UK: Packt Publishing Ltd, Nov. 2023. ISBN: 978-1-80324-028-2 (siehe S. 202).

References II

[11] 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. 204).

[12] 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. 203).

[13] Ed Bott. *Windows 11 Inside Out*. Hoboken, NJ, USA: Microsoft Press, Pearson Education, Inc., Feb. 2023. ISBN: 978-0-13-769132-6 (siehe S. 203).

[14] Ron Brash und Ganesh Naik. *Bash Cookbook*. Birmingham, England, UK: Packt Publishing Ltd, Juli 2018. ISBN: 978-1-78862-936-2 (siehe S. 201).

[15] 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. 201).

[16] Jason Cannon. *High Availability for the LAMP Stack*. Shelter Island, NY, USA: Manning Publications, Juni 2022 (siehe S. 202, 203).

[17] 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. 30–39).

[18] Noureddine Chabini und Rachid Beguenane. "FPGA-Based Designs of the Factorial Function". In: *IEEE Canadian Conference on Electrical and Computer Engineering (CCECE'2022)*. 18.–20. Sep. 2022, Halifax, NS, Canada. Piscataway, NJ, USA: Institute of Electrical and Electronics Engineers (IEEE), 2022, S. 16–20. ISBN: 978-1-6654-8432-9. doi:10.1109/CCECE49351.2022.9918302 (siehe S. 205).

[19] 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. 204).

References III

[20] 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. 201).

[21] 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. 5–13, 40–46, 191, 201).

[22] Peter Pin-Shan Chen. "The Entity-Relationship Model: Toward a Unified View of Data". In: *1st International Conference on Very Large Data Bases (VLDB'1975)*. 22.–24. Sep. 1975, Framingham, MA, USA. Hrsg. von Douglas S. Kerr. New York, NY, USA: Association for Computing Machinery (ACM), 1975, S. 173. ISBN: 978-1-4503-3920-9. doi:10.1145/1282480.1282492. See[21] for a more comprehensive introduction. (Siehe S. 201).

[23] 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. 202).

[24] 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. 204).

[25] 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. 203).

[26] "Crow's Foot Notation – Relationship Symbols and How to Read Diagrams". In: *#DATABASE*. Oakland, CA, USA: Free Code Camp, Inc., 6. Juni 2022. URL: https://www.freecodecamp.org/news/crows-foot-notation-relationship-symbols-and-how-to-read-diagrams (besucht am 2025-04-06) (siehe S. 66–105).

[27] *Database Language SQL*. Techn. Ber. ANSI X3.135-1986. Washington, D.C., USA: American National Standards Institute (ANSI), 1986 (siehe S. 204).

References IV

[28] 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. 204).

[29] *Database Administrators*. New York, NY, USA: Stack Exchange Inc. URL: https://dba.stackexchange.com (besucht am 2025-02-27).

[30] *Database Language SQL*. International Standard ISO 9075-1987. Geneva, Switzerland: International Organization for Standardization (ISO), 1987 (siehe S. 204).

[31] Paul Deitel, Harvey Deitel und Abbey Deitel. *Internet & World Wide WebW[: How to Program*. 5. Aufl. Hoboken, NJ, USA: Pearson Education, Inc., Nov. 2011. ISBN: 978-0-13-299045-5 (siehe S. 204).

[32] Pooyan Doozandeh und Frank E. Ritter. "Some Tips for Academic Writing and Using Microsoft Word". *XRDS: Crossroads, The ACM Magazine for Students* 26(1):10–11, Herbst 2019. New York, NY, USA: Association for Computing Machinery (ACM). ISSN: 1528-4972. doi:10.1145/3351470 (siehe S. 203).

[33] Jacques Dutka. "The Early History of the Factorial Function". *Archive for History of Exact Sciences* 43(3):225–249, Sep. 1991. Berlin/Heidelberg, Germany: Springer-Verlag GmbH Germany. ISSN: 0003-9519. doi:10.1007/BF00389433. Communicated by Umberto Bottazzini (siehe S. 205).

[34] 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. 202, 203).

[35] 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. 30–46).

[36] Steve Fanning, Vasudev Narayanan, „flywire", Olivier Hallot, Jean Hollis Weber, Jenna Sargent, Pulkit Krishna, Dan Lewis, Peter Schofield, Jochen Schiffers, Robert Großkopf, Jost Lange, Martin Fox, Hazel Russman, Steve Schwettman, Alain Romedenne, Andrew Pitonyak, Jean-Pierre Ledure, Drew Jensen und Randolph Gam. *Base Guide 7.3. Revision 1. Based on LibreOffice 7.3 Community*. Berlin, Germany: The Document Foundation, Aug. 2022. URL: https://books.libreoffice.org/en/BG73/BG73-BaseGuide.pdf (besucht am 2025-01-13) (siehe S. 202).

References V

[37] 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. 203).

[38] 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. 202).

[39] *Gleek.io*. Prague, Czech Republic: Blocshop s.r.o., 2024. URL: https://www.gleek.io/blog (besucht am 2025-04-05).

[40] 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. 202).

[41] 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. 203).

[42] 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. 203).

[43] Michael Hausenblas. *Learning Modern Linux*. Sebastopol, CA, USA: O'Reilly Media, Inc., Apr. 2022. ISBN: 978-1-0981-0894-6 (siehe S. 202).

[44] 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. 202, 204).

[45] Manuel Hoffmann, Frank Nagle und Yanuo Zhou. *The Value of Open Source Software*. Working Paper 24-038. Boston, MA, USA: Harvard Business School, 1. Jan. 2024. URL: https://www.hbs.edu/ris/Publication%20Files/24-038_51f8444f-502c-4139-8bf2-56eb4b65c58a.pdf (besucht am 2025-06-04) (siehe S. 203).

References VI

[46] ."Use the following business rules to create a Crow's Foot ERD". In: *Database Administrators*. Hrsg. von zz Z. New York, NY, USA: Stack Exchange Inc., 13. Apr. 2020–27. Okt. 2024. URL: https://dba.stackexchange.com/questions/264891 (besucht am 2025-04-25) (siehe S. 66–84).

[47] ."How do I read ERD Notation (Crow's Feet) to convert to Natural Language?" In: *Database Administrators*. Hrsg. von raddevus. New York, NY, USA: Stack Exchange Inc., 17. Feb. 2016–22. Juni 2018. URL: https://dba.stackexchange.com/questions/129551 (besucht am 2025-04-06) (siehe S. 57–65).

[48] John Hunt. *A Beginners Guide to Python 3 Programming*. 2. Aufl. Undergraduate Topics in Computer Science (UTICS). Cham, Switzerland: Springer, 2023. ISBN: 978-3-031-35121-1. doi:10.1007/978-3-031-35122-8 (siehe S. 203).

[49] *Information Technology – Database Languages – SQL – Part 1: Framework (SQL/Framework), Part 1*. International Standard ISO/IEC 9075-1:2023(E), Sixth Edition, (ANSI X3.135). Geneva, Switzerland: International Organization for Standardization (ISO) und International Electrotechnical Commission (IEC), Juni 2023. URL: https://standards.iso.org/ittf/PubliclyAvailableStandards/ISO_IEC_9075-1_2023_ed_6_-_id_76583_Publication_PDF_(en).zip (besucht am 2025-01-08). Consists of several parts, see https://modern-sql.com/standard for information where to obtain them. (Siehe S. 204).

[50] 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. 201).

[51] Jay LaCroix. *Mastering Ubuntu Server*. 4. Aufl. Birmingham, England, UK: Packt Publishing Ltd, Sep. 2022. ISBN: 978-1-80323-424-3 (siehe S. 203).

[52] 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. 202, 203).

[53] 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. 203).

References VII

[54] *LibreOffice – The Document Foundation*. Berlin, Germany: The Document Foundation, 2024. URL: https://www.libreoffice.org (besucht am 2024-12-12) (siehe S. 202).

[55] 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. 201).

[56] Tony Loton. "Data Modeling: Entity-Relationship Diagram (ER Diagram)". In: *ModernAnalyst.com*. Calabasas, CA, USA: Modern Analyst Media, LLC, 2006–2025. URL: https://www.modernanalyst.com/Resources/Articles/tabid/115/ID/2008/Data-Modeling-Entity-Relationship-Diagram-ER-Diagram.aspx (besucht am 2025-04-05) (siehe S. 66–96).

[57] Peter Luschny. *A New Kind of Factorial Function*. Highland Park, NJ, USA: The OEIS Foundation Inc., 4. Okt. 2015. URL: https://oeis.org/A000142/a000142.pdf (besucht am 2024-09-29) (siehe S. 205).

[58] Mark Lutz. *Learning Python*. 6. Aufl. Sebastopol, CA, USA: O'Reilly Media, Inc., März 2025. ISBN: 978-1-0981-7130-8 (siehe S. 203).

[59] *MariaDB Server Documentation*. Milpitas, CA, USA: MariaDB, 2025. URL: https://mariadb.com/kb/en/documentation (besucht am 2025-04-24) (siehe S. 202).

[60] 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. 202).

[61] 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. 204).

[62] *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. 203).

[63] Constantine Nalimov. *Crow's Foot Notation in Entity-Relationship Diagrams*. Prague, Czech Republic: Blocshop s.r.o., 2. Sep. 2020. URL: https://www.gleek.io/blog/crows-foot-notation (besucht am 2025-04-05) (siehe S. 66–109).

References VIII

[64] Constantine Nalimov. *ER Diagram for a Hospital Management System (Crow's Foot Notation)*. Prague, Czech Republic: Blocshop s.r.o., 3. Dez. 2012. URL: https://www.gleek.io/blog/erd-hospital-management (besucht am 2025-04-05) (siehe S. 66–101).

[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. 201).

[66] 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. 203).

[67] *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. 204).

[68] 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. 201).

[69] Yasset Pérez-Riverol, Laurent Gatto, Rui Wang, Timo Sachsenberg, Julian Uszkoreit, Felipe da Veiga Leprevost, Christian Fufezan, Tobias Ternent, Stephen J. Eglen, Daniel S. Katz, Tom J. Pollard, Alexander Konovalov, Robert M. Flight, Kai Blin und Juan Antonio Vizcaíno. "Ten Simple Rules for Taking Advantage of Git and GitHub". *PLOS Computational Biology* 12(7), 14. Juli 2016. San Francisco, CA, USA: Public Library of Science (PLOS). ISSN: 1553-7358. doi:10.1371/JOURNAL.PCBI.1004947 (siehe S. 202).

[70] Donnie Pinkston. "Entity-Relationship Model II". In: *CS101b – Introduction to Relational Databases*. Pasadena, CA, USA: California Institute of Technology (Caltech), Winter 2007. Kap. 15. URL: http://users.cms.caltech.edu/~donnie/dbcourse/intro0607/lectures/Lecture15.pdf (besucht am 2025-04-04) (siehe S. 20–25, 30–39).

[71] *PostgreSQL Essentials: Leveling Up Your Data Work*. Sebastopol, CA, USA: O'Reilly Media, Inc., März 2024 (siehe S. 203).

[72] Saty Raghavachary. "ER". In: *CSCI 585: Database Systems*. Los Angeles, CA, USA: University of Southern California (UCS), Frühling 2024. URL: https://bytes.usc.edu/cs585/s24-d-a-t-aaa/lectures/ER/slides.html (besucht am 2025-04-06) (siehe S. 15–19, 66–101).

[73] 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. 201).

References IX

[74] Federico Razzoli. *Mastering MariaDB*. Birmingham, England, UK: Packt Publishing Ltd, Sep. 2014. ISBN: 978-1-78398-154-0 (siehe S. 202).

[75] 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 (siehe S. 203).

[76] 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. 201).

[77] 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 (siehe S. 204).

[78] 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. 202).

[79] 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. 30–39, 201).

[80] 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. 202).

[81] Anna Skoulikari. *Learning Git*. Sebastopol, CA, USA: O'Reilly Media, Inc., Mai 2023. ISBN: 978-1-0981-3391-7 (siehe S. 201).

[82] John Miles Smith und Philip Yen-Tang Chang. "Optimizing the Performance of a Relational Algebra Database Interface". *Communications of the ACM (CACM)* 18(10):568–579, Okt. 1975. New York, NY, USA: Association for Computing Machinery (ACM). ISSN: 0001-0782. doi:10.1145/361020.361025 (siehe S. 203).

References X

[83] "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. 204).

[84] 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. 198, 204).

[85] 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[84] (siehe S. 204).

[86] Allen Taylor. *Introducing SQL and Relational Databases*. New York, NY, USA: Apress Media, LLC, Sep. 2018. ISBN: 978-1-4842-3841-7 (siehe S. 203, 204).

[87] 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. 203).

[88] 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. 202).

[89] Kevin Treu. "Entity Relationship Modeling". In: *CSC-341: Database Management Systems*. Greenville, SC, USA: Furman University, Frühling 2025. Kap. 4. URL: https://cs.furman.edu/~ktreu/csc341/lectures/chapter04.pdf (besucht am 2025-04-05) (siehe S. 66–113).

[90] Mariot Tsitoara. *Beginning Git and GitHub: Version Control, Project Management and Teamwork for the New Developer*. New York, NY, USA: Apress Media, LLC, März 2024. ISBN: 979-8-8688-0215-7 (siehe S. 201, 202, 204).

[91] 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. 202).

[92] *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. 204).

References XI

[93] *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. 204).

[94] Sander van Vugt. *Linux Fundamentals*. 2. Aufl. Hoboken, NJ, USA: Pearson IT Certification, Juni 2022. ISBN: 978-0-13-792931-3 (siehe S. 202).

[95] 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. 20–25).

[96] 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. 201–203).

[97] 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. 203).

[98] 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. 201).

[99] *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. 203).

References XII

[100] 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. 203).

[101] 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. 201).

[102] *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. 204).

[103] 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. 202, 203).

[104] Giorgio Zarrelli. *Mastering Bash*. Birmingham, England, UK: Packt Publishing Ltd, Juni 2017. ISBN: 978-1-78439-687-9 (siehe S. 201).

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[14,65,104]. 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[9,55,68,73,76]. These requests and responses are usually sent over network connections. A typical example for such a system is the World Wide Web (WWW), where web servers host websites and make them available to web browsers, the clients. Another typical example is the structure of database (DB) software, where a central server, the DBMS, offers access to the DB to the different clients. Here, the client can be some terminal software shipping with the DBMS, such as psql, or the different applications that access the DBs.

DB A *database* is an organized collection of structured information or data, typically stored electronically in a computer system. Databases are discussed in our book *Databases*[96].

DBA A *database administrator* is the person or group responsible for the effective use of database technology in an organization or enterprise.

DBMS A *database management system* is the software layer located between the user or application and the DB. The DBMS allows the user/application to create, read, write, update, delete, and otherwise manipulate the data in the DB[101].

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

ERD Entity relationship diagrams show the relationships between objects, e.g., between the tables in a DB and how they reference each other[4,15,20–22,50,79,98]

Git is a distributed Version Control Systems (VCS) which allows multiple users to work on the same code while preserving the history of the code changes[81,90]. Learn more at https://git-scm.com.

Glossary (in English) II

GitHub	is a website where software projects can be hosted and managed via the Git VCS[69,90]. Learn more at https://github.com.
IT	information technology
LAMP Stack	A system setup for web applications: Linux, Apache (a web server), MySQL, and the server-side scripting language PHP[16,44].
LibreOffice	is on open source office suite[38,54,78] which is a good and free alternative to Microsoft Office. It offers software such as LibreOffice Writer, LibreOffice Calc, and LibreOffice Base. See[96] 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[36,78]. It is part of LibreOffice[38,54,78] and has functionality that is comparable to Microsoft Access[7,23,91].
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[54,78], i.e., an alternative to Microsoft Excel. It is part of LibreOffice[38,54,78].
LibreOffice Writer	is a free and open source text writing program[103] and part of LibreOffice[38,54,78]. It is a good alternative to Microsoft Word.
Linux	is the leading open source operating system, i.e., a free alternative for Microsoft Windows[5,43,80,88,94]. We recommend using it for this course, for software development, and for research. Learn more at https://www.linux.org. Its variant Ubuntu is particularly easy to use and install.
MariaDB	An open source relational database management system that has forked off from MySQL[1,2,6,34,59,74]. See https://mariadb.org for more information.
Microsoft Access	is a DBMS that can work on DBs stored in single, stand-alone files but also connect to other popular relational databases[7,23,60,91]. It is part of Microsoft Office. A free and open source alternative to this commercial software is LibreOffice Base.
Microsoft Excel	is a spreadsheet program that allows users to store, organize, manipulate, and calculate data in tabular structures[10,40,52]. It is part of Microsoft Office. A free alternative to this commercial software is LibreOffice Calc[54,78].

Glossary (in English) III

Microsoft Office is a commercial suite of office software, including Microsoft Excel, Microsoft Word, and Microsoft Access[52]. LibreOffice is a free and open source alternative.

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

Microsoft Word is one of the leading text writing programs[32,62,103] and part of Microsoft Office. A free alternative to this commercial software is the LibreOffice Writer.

MySQL An open source relational database management system[12,34,75,87,100]. MySQL is famous for its use in the LAMP Stack. See https://www.mysql.com for more information.

OSS Open source software, i.e., software that can freely be used, whose source code is made availabe in the internet, and which is usually developed cooperatively over the internet as well[45]. Typical examples are Python, Linux, Git, and PostgreSQL.

PostgreSQL An open source object-relational DBMS[37,66,71,87]. See https://postgresql.org for more information.

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

Python The Python programming language[48,53,58,97], i.e., what you will learn about in our book[97]. Learn more at https://python.org.

relational database A relational DB is a database that organizes data into rows (tuples, records) and columns (attributes), which collectively form tables (relations) where the data points are related to each other[25,41,42,82,86,96,99].

server In a client-server architecture, the server is a process that fulfills the requests of the clients. It usually waits for incoming communication carring the requests from the clients. For each request, it takes the necessary actions, performs the required computations, and then sends a response with the result of the request. Typical examples for servers are web servers[16] in the internet as well as DBMSes. It is also common to refer to the computer running the server processes as server as well, i.e., to call it the „server computer“[51].

Glossary (in English) IV

SQL The *Structured Query Language* is basically a programming language for querying and manipulating relational databases[19,27,28,30,49,61,83–86]. It is understood by many DBMSes. You find the Structured Query Language (SQL) commands supported by PostgreSQL in the reference[83].

terminal A terminal is a text-based window where you can enter commands and execute them[5,24]. 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 ⊞+R, dann Schreiben von `cmd`, dann Druck auf ↵. Under Ubuntu Linux, Ctrl+Alt+T opens a terminal, which then runs a Bash shell inside.

Ubuntu is a variant of the open source operating system Linux[24,44]. We recommend that you use this operating system to follow this class, for software development, and for research. Learn more at https://ubuntu.com. If you are in China, you can download it from https://mirrors.ustc.edu.cn/ubuntu-releases.

UML The Unified Modeling Language (UML) is a graphical language for visualizing, specifying, constructing, and documenting the artifacts of distributed object systems[11,67,92,93]

VCS A *Version Control System* is a software which allows you to manage and preserve the historical development of your program code[90]. A distributed VCS allows multiple users to work on the same code and upload their changes to the server, which then preserves the change history. The most popular distributed VCS is Git.

WWW World Wide Web[8,31]

yEd is a graph editor for high-quality graph-based diagrams[77,102], suitable to draw, e.g., technology-independent ERDs, control flow charts, or 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.

Glossary (in English) V

$i!$ The factorial $a!$ of a natural number $a \in \mathbb{N}_1$ is the product of all positive natural numbers less than or equal to a, i.e., $a! = 1 * 2 * 3 * 4 * \cdots * (a-1) * a$[18,33,57].

$i..j$ with $i, j \in \mathbb{Z}$ and $i \leq j$ is the set that contains all integer numbers in the inclusive range from i to j. For example, 5..9 is equivalent to $\{5, 6, 7, 8, 9\}$

$\mathbb{N}_1$ the set of the natural numbers *excluding* 0, i.e., 1, 2, 3, 4, and so on. It holds that $\mathbb{N}_1 \subset \mathbb{Z}$.

$\mathbb{R}$ the set of the real numbers.

$\mathbb{Z}$ the set of the integers numbers including positive and negative numbers and 0, i.e., . . . , -3, -2, -1, 0, 1, 2, 3, . . . , and so on. It holds that $\mathbb{Z} \subset \mathbb{R}$.