

Datenbanken und Informationssysteme

2. Das Entity-Relationship-Modell

Prof. Dr. Stefan Decker

📍 RWTH Aachen

Agenda für dieses Modul

1. **Der Datenbankentwurf** (Lebenszyklus & Qualität)
2. **Das Entity-Relationship-Modell** (Konzepte & Notation)
3. **Konzeptueller Entwurf** (Methodische Anwendung)



Wir starten mit Teil 1 und betrachten das Fundament – wie und warum modellieren wir Daten überhaupt?

Informationssysteme: Statik vs. Dynamik

Ein Informationssystem abstrahiert die Realität in zwei orthogonalen Dimensionen:

1. Statische Struktur (Datenbankschema)

- ▶ Definiert mögliche Systemzustände.
- ▶ Welche Objekte existieren?
- ▶ *Bsp: Studierende, Vorlesungen.*
- ▶ Ändert sich extrem selten.

2. Dynamisches Verhalten (Operationen)

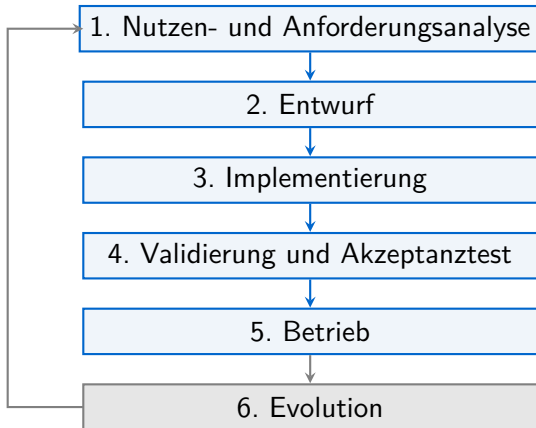
- ▶ Definiert gültige Zustandsübergänge.
- ▶ Wie ändern sich Daten?
- ▶ *Bsp: Klausuranmeldung.*
- ▶ Potentiell hochfrequente Ausführung.



Kerngedanke: Das statische Schema erzwingt die 'Spielregeln', an die sich alle dynamischen Operationen halten müssen!

Der Informationssystem-Lebenszyklus

Bevor wir programmieren, müssen wir den Prozess verstehen. Der Datenbankentwurf ist eingebettet in den Lebenszyklus des gesamten Systems:



Nutzen- und Anforderungsanalyse sind das Fundament

Diese beiden Phasen laufen parallel und bilden die wichtigste Basis des Projekts. Sie entscheiden über das „Was“, bevor wir uns um das „Wie“ kümmern.

Nutzenanalyse

- ▶ Ermittlung potenzieller Anwendungsgebiete.
- ▶ Durchführung von Kosten-Nutzen-Rechnungen.
- ▶ Setzen von Prioritäten für die Umsetzung.

Anforderungsanalyse

In enger Zusammenarbeit mit den **potenziellen Benutzern**:

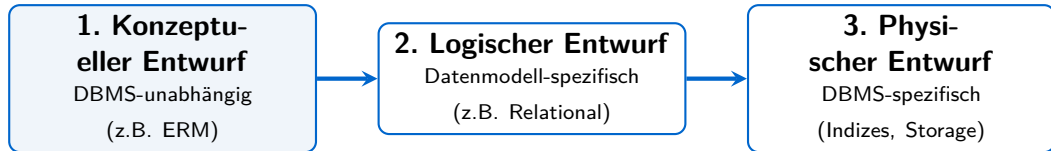
- ▶ **Informationsanforderungen:** Welche Daten müssen gespeichert werden?
- ▶ **Bearbeitungsanforderungen:** Welche Operationen (qualitativ/quantitativ)?
- ▶ **Benutzeranforderungen:** UI, Adaption bekannter Formate, Hilfesysteme.



Wichtigste Phase: Fehler, die hier gemacht werden, beeinflussen alle anderen Phasen und sind später am teuersten zu beheben!

Der Datenbankentwurf in drei Abstraktionsebenen

Wir übersetzen schrittweise von der Fachabteilung zur Speichermaschine:



- ▶ **Stufe 1:** Erfassung der Semantik (Miniwelt) unabhängig von der späteren Technik.
- ▶ **Stufe 2:** Übersetzung in ein mathematisch fundiertes Modell (Relationen, Dokumente).
- ▶ **Stufe 3:** Optimierung für die Hardware (B-Bäume, Caching).

Qualitätskriterien: Was macht ein gutes Schema aus?

► Korrektheit & Vollständigkeit

- Syntaktisch fehlerfrei.
- Alle relevanten Aspekte der Realität sind erfasst.

► Minimalität

- Keine Information wird ohne Not redundant (doppelt) gespeichert.

► Lesbarkeit

- Das Schema sollte selbsterklärend und dokumentiert sein.

► Modifizierbarkeit

- Leichte Anpassung an neue Anforderungen durch modularen Aufbau.



Ziel: Die Erstellung einer übersichtlichen Struktur unter Vermeidung von Redundanzen (später formalisiert als **Normalisierung**).

Agenda für dieses Modul

1. **Der Datenbankentwurf** (Lebenszyklus & Qualität)
2. **Das Entity-Relationship-Modell** (Konzepte & Notation)
3. **Konzeptueller Entwurf** (Methodische Anwendung)



Wir starten mit Teil 2. Was sind Entitäten, Attribute und Beziehungen, und wie zeichnet man sie formal korrekt?

Das ER-Modell als maschinenunabhängige Skizze

Natürliche Sprache (Pflichtenheft) ist oft mehrdeutig. Code (z.B. SQL) ist für die frühe Entwurfsphase zu technisch und maschinennah.

Die Lösung: Das Entity-Relationship-Modell (ER-Modell)

- ▶ Ein *konzeptuelles*, graphisches Modell auf sehr hohem Abstraktionsniveau.
- ▶ Physische Effizienz (Speicher, Zugriffswege) spielt hier noch absolut keine Rolle!

1. Objekttypen

Entity-Typ

(Rechtecke)

2. Eigenschaften

Attribut

(Ellipsen)

3. Beziehungen

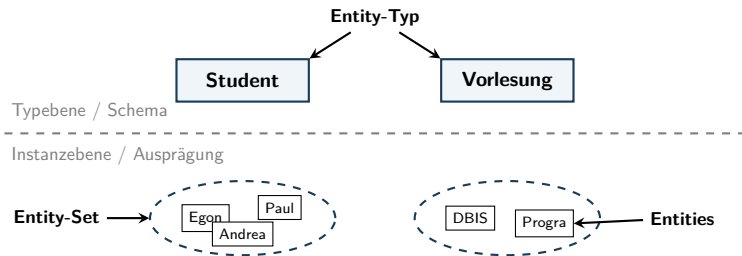
Relationship

(Rauten)

Wir trennen strikt zwischen Schema (Typ) und Ausprägung (Set)

- ▶ **Entity-Typ:** Die abstrakte Schablone im Schema (Rechteck).
- ▶ **Entity-Set:** Die Menge von Entities eines Entity-Typs.

Ein Entity-Set wird im Normalfall durch den gleichen Namen wie ein Entity-Typ bezeichnet.



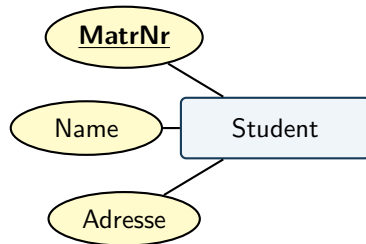
Merke: Das ER-Diagramm beschreibt die **Typeebene**. Die konkreten Daten liegen im **Entity-Set**.

Schlüssel identifizieren Entitäten eindeutig innerhalb ihres Typs

- ▶ **Attribute (Ellipsen):** Charakterisieren einen Entity-Typ (z.B. *Farbe*, *Gewicht*).
- ▶ Sie stammen aus definierten Wertebereichen (z.B. INTEGER, STRING).

Schlüssel identifizieren Entitäten eindeutig innerhalb ihres Typs

- ▶ **Attribute (Ellipsen):** Charakterisieren einen Entity-Typ (z.B. *Farbe*, *Gewicht*).
- ▶ Sie stammen aus definierten Wertebereichen (z.B. INTEGER, STRING).
- ▶ **Schlüsselkandidat:** Eine minimale Attributmenge, die eine Entität *eindeutig* identifiziert.
- ▶ **Primärschlüssel (Unterstrichen):** Der gewählte Identifikator.
- ▶ Oft nutzen wir *Künstliche Schlüssel* (z.B. MatrNr), da natürliche oft mehrdeutig sind.

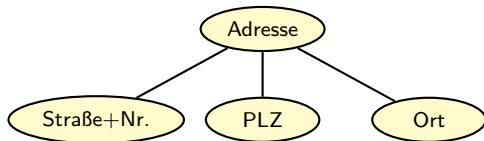


Komplexe Attribute modellieren innere Strukturen & Wertemengen

Die Realität erfordert oft Strukturen, die über einzelne (atomare) Werte hinausgehen:

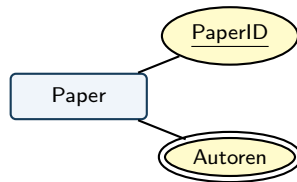
1. Zusammengesetzte Attribute

Ein Attribut besteht aus logischen Unterattributen.



2. Mehrwertige Attribute

Eine Entität besitzt eine Menge von Werten desselben Typs.



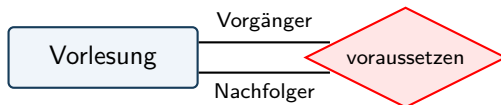
Beziehungen (Relationships) verknüpfen Entitäten

- ▶ Repräsentieren Zusammenhänge zwischen Entity-Typen.
- ▶ Darstellung durch **Rauten** (oft Verben zur Andeutung der Richtung).
- ▶ *Beispiel:* Student hört Vorlesung.



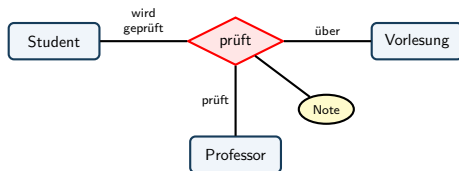
Sonderfall: Rekursive Beziehungen

- ▶ Verbindet denselben Entity-Typ mehrfach mit sich selbst.
- ▶ **Problem:** Ohne Beschriftung ist die Semantik völlig unklar!
- ▶ **Lösung:** Wir annotieren die Kanten zwingend mit **Rollen**.



N-stellige Beziehungen sind Teilmengen des kartesischen Produkts

Beziehungen können mehr als zwei Entity-Typen einbinden (z.B. *ternär*).

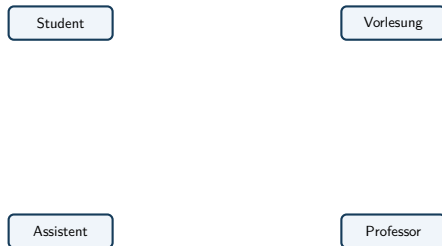


- **Formal:** Eine Ausprägung einer n -stelligen Beziehung R ist eine Menge von n -Tupeln:

$$r \in R \subseteq E_1 \times E_2 \times \cdots \times E_n$$

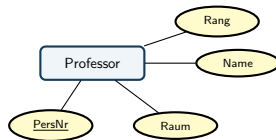
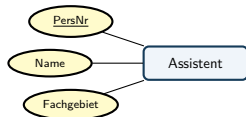
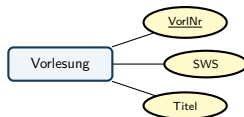
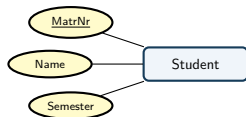
- Eigene Attribute (wie Note) hängen direkt an der *Beziehung*, da sie nur für diese exakte Kombination existieren.

Beispiel: ER-Diagramm (Universität): Schritt 1



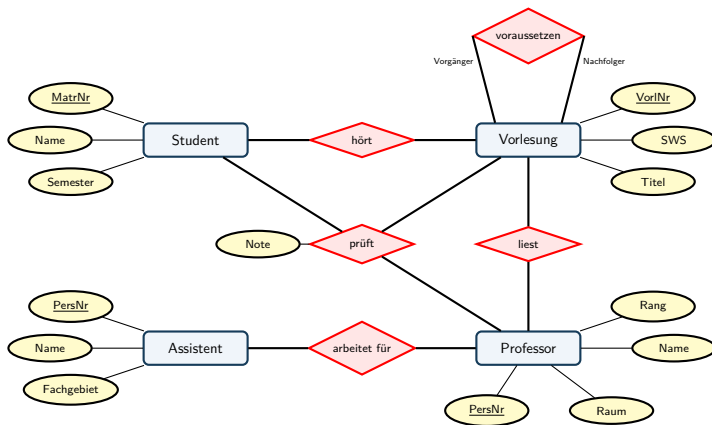
Schritt 1: Identifikation der zentralen **Entity-Typen**.

Beispiel: ER-Diagramm (Universität): Schritt 2



Schritt 2: Merkmale und **Primärschlüssel** festlegen.

Beispiel: ER-Diagramm (Universität): Schritt 3



Schritt 3: Verknüpfung der Typen durch **Beziehungen**.

Cliffhanger: Was fehlt unserem Modell noch?

Wir wissen bisher (Struktur):

- ▶ Studenten *hören* Vorlesungen.
- ▶ Professoren *lesen* Vorlesungen.

Cliffhanger: Was fehlt unserem Modell noch?

Wir wissen bisher (Struktur):

- ▶ Studenten *hören* Vorlesungen.
- ▶ Professoren *lesen* Vorlesungen.



Wir wissen NICHT (Geschäftsregeln):

- ▶ Darf ein Student **beliebig viele** Vorlesungen hören?
- ▶ Muss eine Vorlesung **mindestens einen** Hörer haben?
- ▶ Wie viele Assistenten hat ein Professor maximal?

Wie viele?

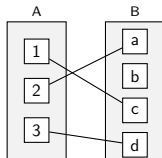
▶▶ **Ausblick:** Im nächsten Block lösen wir dies durch **Kardinalitäten** (Chen-Notation vs. Min/Max-Formalismus).

Grundtypen bei binären Beziehungen

1:1 (Eins-zu-Eins)

Eine Entität aus A ist mit *höchstens einer* Entität aus B verbunden (und umgekehrt).

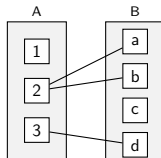
Bsp: Person $\leftrightarrow$ Ausweis.



1:N (Eins-zu-Viele)

Entität aus A kann mit *vielen* aus B verbunden sein, B aber mit *höchstens einer* aus A.

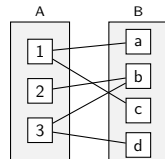
Bsp: Gebäude $\leftrightarrow$ Raum.



N:M (Viele-zu-Viele)

Eine Entität aus A kann mit *vielen* aus B verbunden sein (und umgekehrt).

Bsp: Student $\leftrightarrow$ Kurs.



Hinweis: N:1 ist die Umkehrung von 1:N.

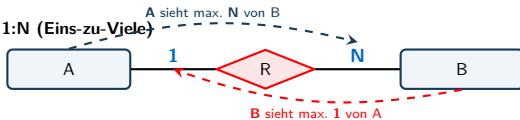
Notation: 1:N an den Verbindungslinien (Chen)

- ▶ Gibt die **maximale** Anzahl der Entitäten an, mit der eine Entität des **anderen** Typs in Beziehung stehen kann.
- ▶ **Die Über-Kreuz-Regel (Lookacross):** Die Beschriftung steht an der Linie **beim gegenüberliegenden** Entity-Typ!

1:1 (Eins-zu-Eins)



1:N (Eins-zu-Viele)



N:M (Viele-zu-Viele)



Limitierung: Zeigt **nur** die maximale Anzahl. Minimales Verhalten (Pflicht/Optional) fehlt!

Beispiele zur 1:N-Notation (Chen)



Annahme: Jede Person hat max. 1 Ausweis, jeder Ausweis gehört zu max. 1 Person.



Annahme: Ein Gebäude hat viele Räume, ein Raum gehört zu max. 1 Gebäude.



Annahme: Student hört viele Vorlesungen, Vorlesung wird von vielen Studenten gehört.

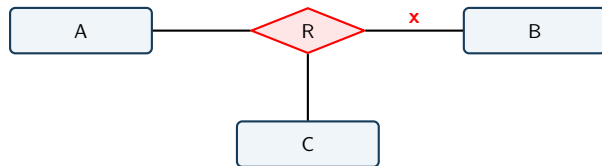
Wichtig: Die Wahl der Kardinalität (1, N, M) hängt **immer** von den Regeln der modellierten Realität bzw. den Anforderungen ab!

Kardinalitäten bei mehrstelligen Beziehungen

Die Über-Kreuz-Regel gilt auch hier, aber wir müssen **alle** anderen Partner festhalten, um die Kardinalität an einer Kante zu bestimmen!



x: Wie viele Entities des Typs B stehen in Beziehung zu **einer** Entity des Typs A?



x: Wie viele Entities des Typs B stehen in Beziehung zu einem **(a, c)-Paar** mit $a \in A$ und $c \in C$?

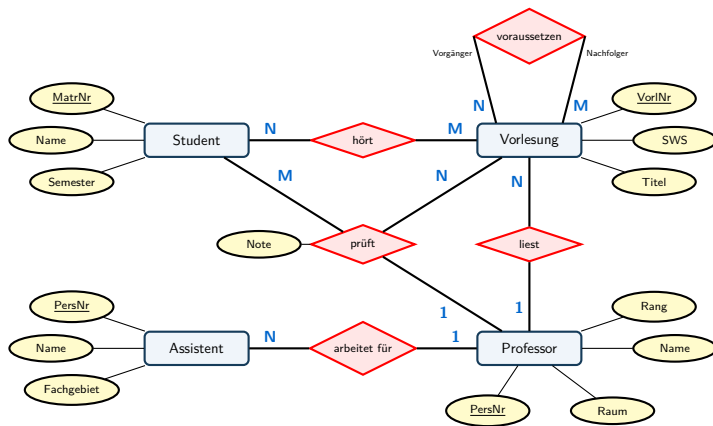
Erweiterung auf n-stellige Beziehungen:

► **$m : 1$ -Beziehung (many-to-one)**

- **Binäre Beziehungen:** Eine Beziehung $R(E_1, E_2)$ „von E_1 nach E_2 “, bei der jedes Entity aus E_1 zu *höchstens einem* Entity aus E_2 in Beziehung steht (nicht notwendig umgekehrt), heißt $m : 1$ -Beziehung.
- **n-stellige Beziehungen:** Eine Beziehung $R(E_1, \dots, E_j, \dots, E_n)$ heißt „ $m : 1$ von $E_1, \dots, E_{j-1}, E_{j+1}, \dots, E_n$ nach E_j “, falls jede Auswahl von Entities aus $E_1, \dots, E_{j-1}, E_{j+1}, \dots, E_n$ *höchstens ein* Entity in E_j bestimmt.

 **Kerneinsicht:** Es besteht eine **funktionale Abhängigkeit**. Deswegen wird die Kardinalität in diesem Kontext auch als **Funktionalität** bezeichnet.

Beispiel: ER-Diagramm (Universität) mit Kardinalitäten



Limitation / Offene Frage: Diese Notation sagt **nichts** über die **minimale** Teilnahme (Pflicht/Optional)! Muss jede Vorlesung gelesen werden?

Muss jede Entität teilnehmen? (Pflicht vs. Optional)

Die bisherige Chen-Notation (1, N, M) betrachtet **nur die maximale Anzahl** an Beziehungspartnern.

Aber: Was ist mit der *minimalen* Teilnahme?

- ▶ **Pflicht:** Muss eine Entität zwingend teilnehmen?
- ▶ **Optional:** Kann sie teilnehmen (darf aber auch fehlen)?

Beispiel: Student / Vorlesung (in Chen-Notation)



Fragen, die das 'N' und 'M' nicht beantworten:

? Muss *jeder* Student (mindestens) eine Vorlesung hören?

? Muss *jede* Vorlesung (mindestens) einen Hörer haben, damit sie stattfindet?

💡Erkenntnis: Die (1:N)-Notation reicht nicht aus, um Pflicht/Optional-Regeln darzustellen.
Lösung: Die (min, max)-Notation!

Die (min, max)-Notation: Präzisere Angaben

ACHTUNG PARADIGMENWECHSEL: Bei (min, max) wird die Teilnahme-Bedingung einer Entität direkt an **ihrer eigenen** Kante notiert (lokales Lesen)! Das ist eine **Positionsvertauschung** zur Chen-Notation.

Konkretes Beispiel: Student hört Vorlesung

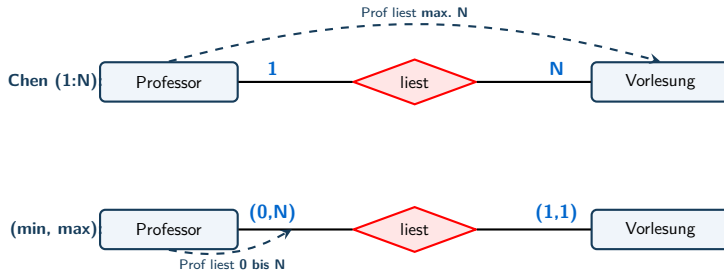


Abstrakte Regel:

- ▶ Die Angabe **(min, max)** bei **Entität A** definiert: Eine Entität von Typ A ist mit mindestens *min* und höchstens *max* Entitäten des anderen Typs verbunden.
- ▶ (Sie beschreibt also, wie viele Partner zu genau *einem* A gehören).

Vergleich: 1:N vs. (min, max) Notation

Beide Notationen haben eine unterschiedliche Ausdrucksstärke und eine **komplett gegensätzliche Leserichtung!**



⚠ ACHTUNG KLAUSURFALLE: Positionsvertauschung! Was bei Chen "drüben" steht, steht bei (min, max) an der eigenen Kante!

Systematischer Vergleich & Empfehlung

(1, N, M) Notation (Chen)

- ▶ Gibt **nur maximale** Teilnahme an.
- ▶ **Keine direkte Aussage** über Pflicht/Optional (min. Teilnahme).
- ▶ **Vorteil:** Evtl. schneller lesbar, wenn es nur um reine Obergrenzen geht.
- ▶ **Nachteil:** Weniger präzise, kann zu Mehrdeutigkeiten führen.
- ▶ **Verbreitung:** Eher ältere Notation.

(min, max) Notation

- ▶ Gibt minimale UND maximale Teilnahme an.
- ▶ **Direkte Aussage** über Pflicht/Optional:
 - ▶ $min = 0 \implies$ Optional
 - ▶ $min = 1 \implies$ Pflicht
- ▶ **Vorteil:** Präziser, deutlich höhere Ausdrucksstärke.
- ▶ **Nachteil:** Evtl. minimal mehr Schreibaufwand.
- ▶ **Verbreitung:** Moderner Standard.

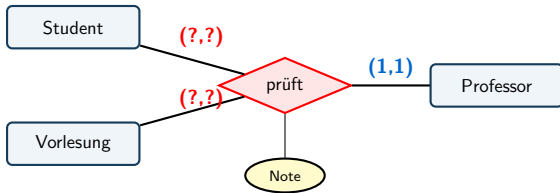
Fazit & Empfehlung für die Praxis:

- ▶ Für neue ER-Modelle **bevorzugt die (min, max)-Notation** verwenden!
- ▶ **WICHTIG:** Innerhalb eines Diagramms / Projekts **unbedingt konsistent bleiben!** (Niemals mischen!)

Kardinalitäten bei n -ären Beziehungen ($n > 2$)

Interpretation: Wenn Entitäten von $(n-1)$ Typen **festgelegt** sind, mit wie vielen Entitäten des n -ten Typs können diese in Beziehung stehen?

Beispiel: Ternäre Beziehung "Prüfung"



Warum (1,1) beim Professor?

Fixiere: 1 Student + 1 Vorlesung.

→ Dieses Paar wird von **genau 1** Professor geprüft.
Daher (1,1).

Was steht beim Studenten (?,?,)?

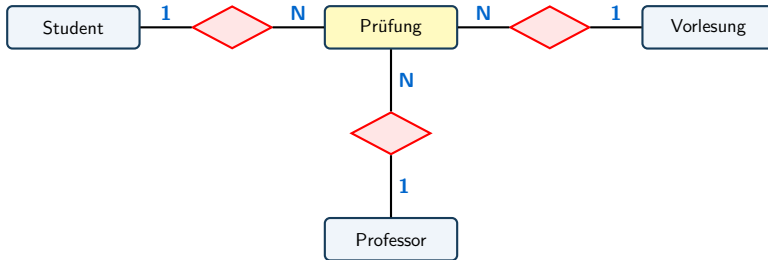
Fixiere: 1 Prof + 1 Vorlesung.

→ In dieser Konstellation können **0 bis N** Studenten geprüft werden. Daher (0,N).

Fazit: Die genaue Semantik erfordert eine **sehr sorgfältige Anforderungsanalyse!**

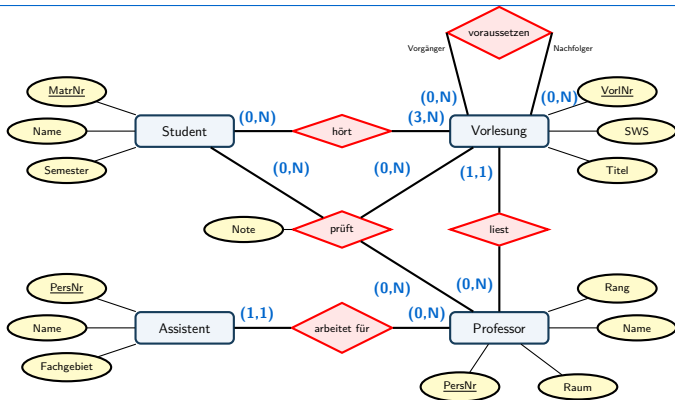
Alternative: Auflösung in binäre Beziehungen

Oft ist es einfacher, das "Ereignis" in einen **neuen Entity-Typ** (z.B. "Prüfung") aufzulösen und mehrere binäre Beziehungen zu ziehen.



⚠ ACHTUNG SEMANTIK-VERLUST: Diese Modellierung ändert die Constraints! Die harte Einschränkung, dass ein Student eine Vorlesung nur bei **einem** Professor prüfen lassen kann, ist hier **nicht mehr darstellbar!**

Beispiel: Universitätsschema mit (min, max)-Kardinalitäten

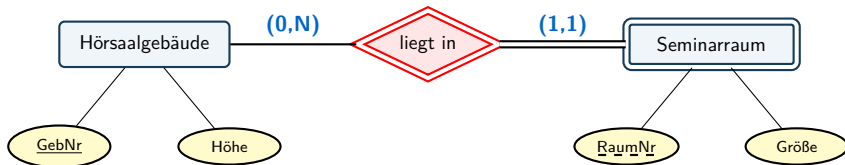


Die Regeln des Uni-Betriebs:

- ▶ **Prof/Vorlesung:** Vorlesung *muss* von exakt 1 Prof gelesen werden (1,1). Prof *kann* lesen (0,N).
- ▶ **prüft (ternär):** Alle haben (0,N). **Achtung:** Die (min,max)-Notation kann hier die funktionale Abhängigkeit (aus Chen) nicht abbilden!
- ▶ **Student/Vorlesung:** Vorlesung *braucht* mind. 3 Hörer (3,N). Student *kann* hören (0,N).
- ▶ **Assistent/Prof:** Assistent *muss* für exakt 1 Prof arbeiten (1,1).

Erweiterte Konzepte: Schwache Entity-Typen

Ein schwacher Entity-Typ ist von der Existenz eines übergeordneten Entity-Typs **abhängig** und kann ohne diesen nicht existieren.



Das "Warum?":

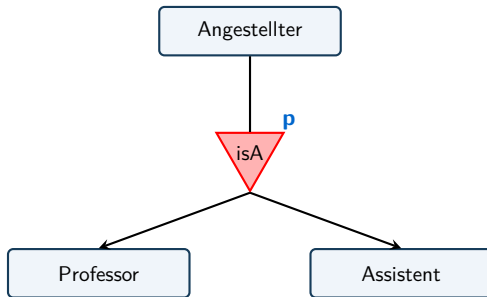
Ein Raum "101" ist nicht eindeutig. Er braucht das Gebäude "A", um eindeutig zu sein (A.101).

Die Kennzeichen:

1. Doppelte Ränder (Entität & Raute).
2. Gestrichelter Schlüssel (partiell).
3. Kardinalität zwingend (1,1)!

Erweiterte Konzepte: Generalisierung & Spezialisierung

- ▶ **Vererbungsbeziehung (isA)** zwischen einem allgemeinen Entity-Typ und spezialisierten Entity-Typen.
- ▶ **Die goldene Regel:** Ein spezialisierter Typ erbt **alle** Attribute und Beziehungen des allgemeinen Typs!



💡 **Lesehilfe:** Ein Professor "is a" (ist ein) Angestellter. Er hat eigene Attribute (z.B. Raum), besitzt aber automatisch auch die des Angestellten (z.B. Name, PersNr).

Merkmale der Spezialisierung (Constraints)

Wir können das IS-A-Dreieck durch zwei Merkmals-Paare präzisieren:

1. Überlappung (Pfeilrichtung!)

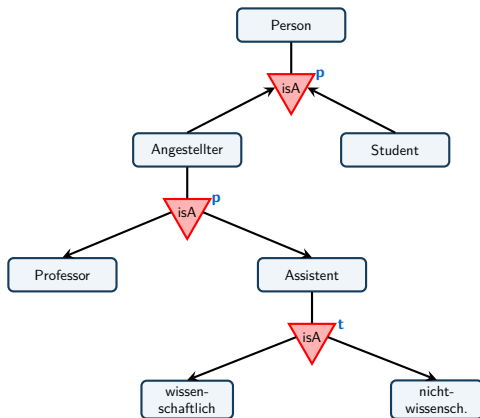
- ▶ **Disjunkt ($\downarrow$):** Pfeile zeigen *AUF* die Spezialisierung. Ein Element kann **nicht** zu mehreren Sub-Typen gleichzeitig gehören. (Bsp: Angestellter ist entweder Assi oder Prof, nicht beides).
- ▶ **Nicht disjunkt ($\uparrow$):** Pfeile zeigen *AUF* die Generalisierung. Ein Element **kann** in mehreren Sub-Typen liegen. (Bsp: Person kann gleichzeitig Angestellter und Student sein).

2. Vollständigkeit (Buchstaben)

- ▶ **Total (t):** Die Zerlegung ist vollständig. Jedes Element des Supertyps **muss** in (mindestens) einem Subtyp vorkommen. (Bsp: Assistent ist *zwingend* entweder wiss. oder nicht-wiss.).
- ▶ **Partiell (p):** Die Subtypen sind eine echte Teilmenge. Es gibt Elemente im Supertyp, die in **keinem** Subtyp liegen. (Bsp: Es gibt Angestellte, die weder Prof noch Assi sind, z.B. Sekretariat).

Beispiel: Generalisierung / Spezialisierung

Ein komplexer Vererbungsbaum mit wechselnden Constraints:



Leseanleitung:

Ebene 1: Person

↑ (nicht disjunkt): Man kann Student *und* Angestellter sein.

p (partiell): Es gibt Personen, die keines von beidem sind.

Ebene 2: Angestellter

↓ (disjunkt): Entweder Prof. *oder* Assi, nie beides.

p (partiell): Es gibt Angestellte (z.B. Sekretariat), die weder noch sind.

Ebene 3: Assistent

↓ (disjunkt): Nur eins von beidem.

t (total): Ein Assi *muss zwingend* wiss. oder nicht-wiss. sein!

Beispiele: Constraints der Spezialisierung

Die vier Kombinationen aus Überlappung und Vollständigkeit auf den Punkt gebracht:

	Disjunkt (↓)	Nicht disjunkt (↑)
Total (t)	Raumarten: Hörsaal, Büro ... ➤ Jeder Raum hat exakt eine Funktion.	Student: Bachelor, Master ... ➤ Jeder studiert, Mehrfach-Einschreibung möglich.
Partiell (p)	Klausur: Bestanden, Durchgefallen ➤ Einige fehlen (p), niemand hat beides (↓).	Student (Fächerauswahl): Info, Physik ➤ Andere Fächer fehlen (p), Doppelstudium möglich (↑).

💡Tipp: Merken Sie sich für jede Box **ein** Alltagsbeispiel. Wenn Sie in der Klausur unsicher sind, wenden Sie die Regel auf Ihr gemerktes Beispiel an!

1. **Der Datenbankentwurf** (Lebenszyklus & Qualität)
2. **Das Entity-Relationship-Modell** (Konzepte & Notation)
3. **Konzeptueller Entwurf** (Methodische Anwendung)



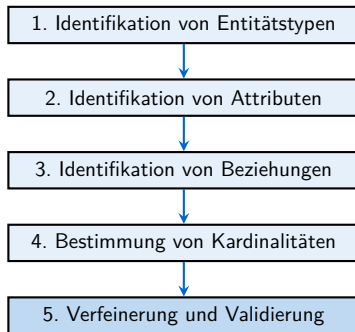
Wir starten mit Teil 3. Wie wenden wir die Konzepte systematisch an, um vom Text zum fertigen Diagramm zu gelangen?

Was ist das Ziel?

- ▶ **Hochrangiges** Modell
- ▶ **Technik-unabhängig**
- ▶ Fokus auf **Kern-Infos**

Nutzen:

- ▶ **Verständigung** Fach/IT
- ▶ **Stabile Basis** Logik
- ▶ **Vollständigkeit**



Schritt 1: Identifikation von Entitätstypen

Was ist ein Entitätstyp?

- ▶ Klasse von realen/abstrakten Objekten
- ▶ Abstraktion durch **Klassifikation**
- ▶ Eindeutig identifizierbar

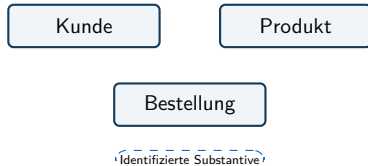
Wie identifizieren?

- ▶ **Substantiv-Methode:** Nomen im Text sind oft Entitäten.
- ▶ **Leitfrage:** Über welche "Dinge" müssen wir Daten speichern?

Fallbeispiel: Online-Shop

Anforderungstext

"Kunden können Produkte bestellen. Jede Bestellung enthält mehrere Produkte."



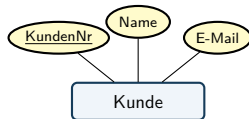
Weitere Beispiele:

Student, Vorlesung, Buch, Professor.

Schritt 2: Identifikation von Attributen

Eigenschaften festlegen

- ▶ Welche Daten müssen wir pro Entität speichern?
- ▶ **Primärschlüssel (PK):** Jede Entität braucht ein Attribut zur eindeutigen Identifizierung (unterstrichen).



Merke:
In der Praxis werden oft nur Schlüssel-Attribute im Diagramm gezeichnet, um Platz zu sparen.

Beispiele im Shop:

- ▶ **Kunde:** KundenNr, Name, Adresse, E-Mail.
- ▶ **Produkt:** ProduktNr, Bezeichnung, Preis.
- ▶ **Bestellung:** BestellNr, Datum, Status.

Design-Entscheidung: Überlegen Sie genau, welche Attribute *atomar* sein müssen (z.B. Name → Vorname, Nachname), um spätere Sortierungen zu ermöglichen.

Schritt 3a: Identifikation von Beziehungen

Beziehungen drücken Interaktionen zwischen Entitätstypen aus. In der Anforderungsanalyse suchen wir gezielt nach **Verben**.



Beziehungsarten:

- ▶ **Binär:** Verbindet zwei Typen.
- ▶ **Rekursiv:** Typ bezieht sich auf sich selbst.
- ▶ **Ternär:** Verbindet drei Typen gleichzeitig.

Identifikation im Shop-Text:

*"...Kunden können Produkte **bestellen** (tätigen). Jede Bestellung **enthält** mehrere Produkte."*

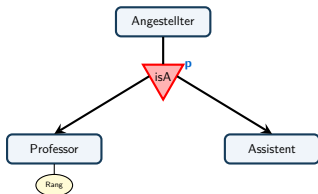
Wichtig bei rekursiven Beziehungen: Hier müssen zwingend **Rollen** definiert werden (z.B. Vorgänger/Nachfolger), um die Kanten semantisch zu unterscheiden!

Schritt 3b: Spezielle Beziehungen

Wie identifizieren wir Hierarchien und Zusammensetzungen im Anforderungstext?

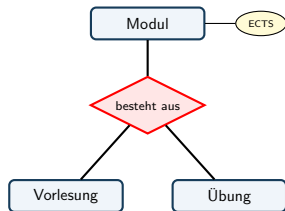
Identifikation: Generalisierung

- **Text-Signal:** "X ist ein spezieller Y" oder "Untergruppe".
- **Beispiel:** *"Professoren sind Angestellte mit einem Rang."*



Identifikation: Aggregation

- **Text-Signal:** "X besteht aus Y" oder "X bildet Einheit Z".
- **Beispiel:** *"Ein Modul bündelt Vorlesung und Übung."*



Design-Regel: Verwenden Sie Aggregation nur, wenn das "Ganze" (Modul) als eigenständiges Objekt mit eigenen Attributen/Beziehungen fungiert.

Schritt 4: Bestimmung von Kardinalitäten

Kardinalitäten legen die min/max Teilnahme fest. Ausgangspunkt sind immer Geschäftsregeln & Anforderungen!



Geschäftsregeln (Beispiele):

- ▶ Kunde → beliebig viele Bestellungen (0,N).
- ▶ Jede Bestellung gehört zu **einem** Kunden (1,1).
- ▶ Bestellung muss **mind. ein** Produkt enthalten (1,N).
- ▶ Produkt kann in **0 bis N** Bestellungen sein.

Merke: Jede (min, max)-Angabe muss begründbar sein! Annahmen müssen explizit validiert werden.

Schritt 5: Verfeinerung & Design-Entscheidungen

Ein Modell ist selten im ersten Wurf perfekt. Wir müssen kritische Fragen stellen:

1. Entität vs. Attribut?

Wann wird ein Begriff eine eigene Box?

- ▶ **Attribut:** Wenn es nur ein einzelner Wert ist (z.B. *Farbe*).
- ▶ **Entität:** Wenn das Objekt selbst weitere Attribute hat oder in mehreren Beziehungen steht (z.B. *Adresse* als eigene Entität, wenn sie geteilt wird).

2. Beziehung vs. Entität?

Wann brauche ich eine Raute oder eine Box?

- ▶ **Beziehung:** Dokumentiert die Interaktion.
- ▶ **Gerichtete Entität:** Wenn die Beziehung selbst Attribute trägt (z.B. *Note* in einer Prüfung), kann sie oft als schwache Entität oder Aggregation gesehen werden.

Leitregel zur Komplexität

Vermeiden Sie Redundanzen! Wenn Informationen mehrfach gespeichert werden müssen, ist das ein Indiz für ein fehlendes Objekt oder eine falsche Zuordnung.

⚠ Klausur-Tipp: Achten Sie auf Begriffe wie "Anschrift" oder "Kategorie". Prüfen Sie, ob diese nur beschreibend sind oder eine eigene Identität benötigen.

Häufige Fehler beim konzeptuellen Entwurf

Typische Stolperfallen, die Sie vor dem Abschluss prüfen sollten:

1. Entität vs. Attribut

Problem: Strukturierte Daten als einfachen Wert versteckt.

Bsp.: *Adresse* als String statt eigener Entität.

2. Redundanz (Transitivität)

Problem: Ableitbare Fakten unnötig direkt modelliert.

Bsp.: Kante $A \rightarrow C$, obwohl $A \rightarrow B \rightarrow C$ existiert.

3. Falsche Kardinalitäten

Problem: (min, max)-Werte verletzen die Geschäftsregeln.

Bsp.: (1,1) statt (1,N) für Produkte im Warenkorb.

4. Schlüssel & Benennung

Problem: Fehlende Eindeutigkeit oder unklare Begriffe.

Bsp.: Kein Primärschlüssel; Nomen statt Verben an den Rauten.

Praxis-Tipp: Lesen Sie das fertige Diagramm laut als Satz vor. Ergibt der Satz keinen Sinn, ist das Modell fehlerhaft!

Vergleich: ER-Modell vs. UML-Klassendiagramm

Beide modellieren Strukturen, aber mit unterschiedlichem Fokus: **Daten** (ER) vs. **Objekte** (UML).

ER-Modell: Fokus auf Daten & Integrität



UML-Klassendiagramm: Fokus auf Struktur & Verhalten



Wichtigster Unterschied: ER zeigt nur **Zustände**, UML zeigt auch **Operationen** (Methoden). In der Datenbanklehre bleiben wir beim ER-Fokus!

Zusammenfassung: ER-Modell

- ▶ **Ziel:** Strukturierte, plattformunabhängige Abbildung der Miniwelt.
- ▶ **Konzepte:** Entitäten (Nomen), Beziehungen (Verben), Attribute.
- ▶ **Regeln:** Präzise Vorgaben durch (min,max)-Kardinalitäten.
- ▶ **Erweiterungen:** *isA*-Hierarchien und Aggregation.
- ▶ **Qualitätskriterien:** Minimalität und Redundanzfreiheit.

Ausblick: Nächstes Modul

- ▶ **Relationales Modell:** Von runden Diagrammen zu harten **Tabellen**.

Fazit: Die Qualität der finalen Datenbank steht und fällt mit einem sauberen ER-Modell!