Zürcher Hochschule
für Angewandte Wissenschaften

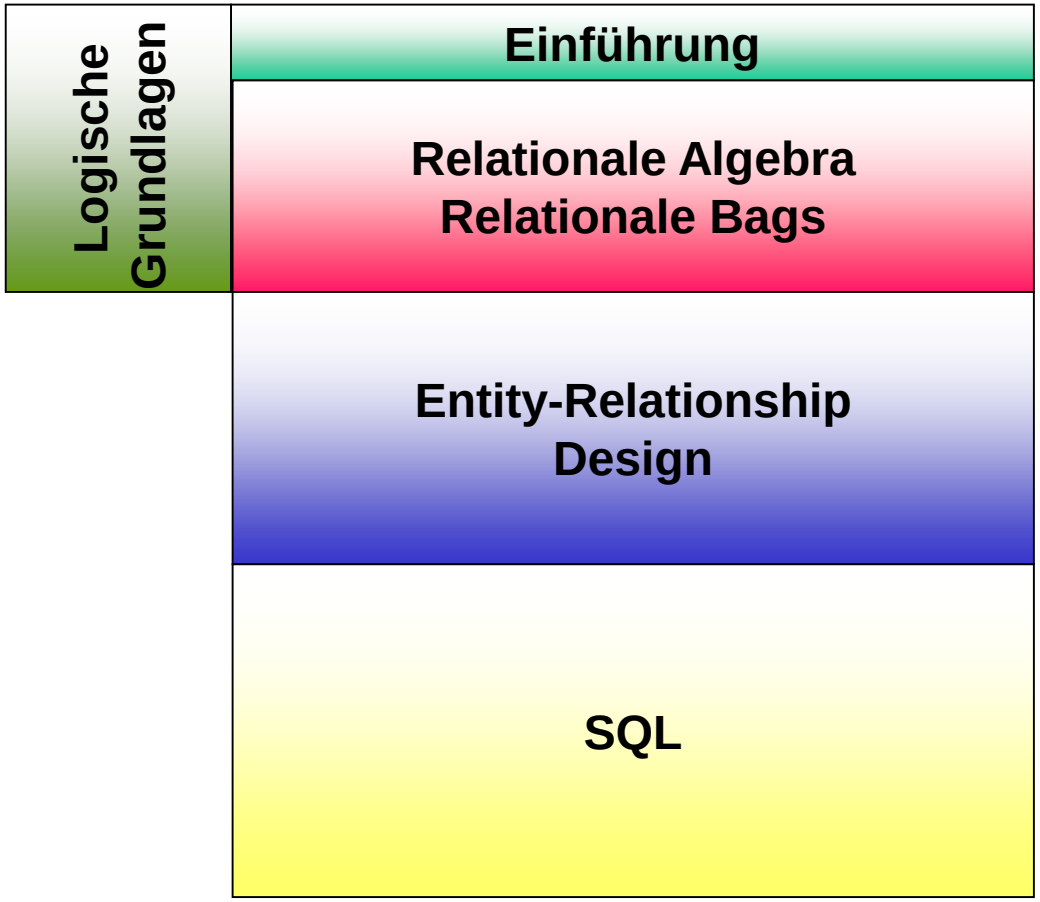
School of
Engineering
InIT Institut für angewandte
Informationstechnologie

DAB1 – Datenbanken 1

Dr. Daniel Aebi (aebd@zhaw.ch)

Lektion 5: Schlüssel, Entity-Relationship (ER) Modell

Wo stehen wir?



"You are here"

Rückblick

- ÷ Division
- Aggregatfunktionen (COUNT, SUM, MIN, MAX, AVG), Gruppierung
- "bag"-Theorie:
  - δ Duplikatelimination
  - σ Selektion: Multiplizitäten übernehmen
  - π Projektion: Multiplizitäten aufsummieren
  - ⋈ Join: Multiplizitäten multiplizieren
  - Vereinigung: 2 Varianten
    - ∪ "bag union": Grössere der beiden Multiplizitäten nehmen
    - ⊔ "bag concatenation": Summe der beiden Multiplizitäten nehmen
  - ∩ Durchschnitt: Kleinere der beiden Multiplizitäten nehmen
  - \ Differenz: Max(0,Differenz der beiden Multiplizitäten) nehmen

Und noch eine Bieridee…

- «Die Restaurants, welche alle Biersorten im Sortiment haben, welche überhaupt im Sortiment irgendeines Restaurants sind»
- Wo finden wir die zu Grunde liegenden Daten? → Tabelle Sortiment

- Zur Erinnerung: Relationenformat Sortiment(Restaurant, Biersorte) mit Relation s zu Sortiment
- Lösung: $s \div \pi_{Biersorte}(s)$

- Wie berechnet man das? → Division S ÷ U ist äquivalent zur folgenden Formel: $\pi_{Attr(S) \setminus Attr(U)}(S) \setminus \pi_{Attr(S) \setminus Attr(U)} ((\pi_{Attr(S) \setminus Attr(U)} (S) \times U) \setminus S)$

- Voraussetzung für Division?

Und noch eine Bieridee…

s	Restaurant	Biersorte
	Ochsen	Cardinal
	Hirschen	Leichtbier
	Ochsen	Feldschlösschen
	Krone	Leichtbier
	Krone	Cardinal
	Hirschen	Cardinal
	Ochsen	Leichtbier
	Hirschen	Feldschlösschen
	Löwen	Leichtbier
	Löwen	Cardinal
	Ochsen	Löwenbräu

$\pi_{Biersorte}(s) = u$	Biersorte
	Cardinal
	Leichtbier
	Feldschlösschen
	Löwenbräu

Und noch eine Bieridee…

- Also: $\pi_{Attr(s) \setminus Attr(u)}(s) \setminus \pi_{Attr(s) \setminus Attr(u)} ((\pi_{\mathbf{Attr(s) \setminus Attr(u)}} \mathbf{(s) \times u}) \setminus s)$

$\pi_{Attr(s) \setminus Attr(u)}$ (s)	Restaurant
	Ochsen
	Hirschen
	Krone
	Löwen

X

$\pi_{Biersorte}(s) = u$	Biersorte
	Cardinal
	Leichtbier
	Feldschlösschen
	Löwenbräu

Und noch eine Bieridee…

- Also: $\pi_{Attr(s) \setminus Attr(u)}(s) \setminus \pi_{Attr(s) \setminus Attr(u)} ((\pi_{\mathbf{Attr(s) \setminus Attr(u)}} \mathbf{(s) \times u}) \setminus s)$

$\pi_{Attr(s) \setminus Attr(u)}$ (s) x u	Restaurant	Biersorte
	Ochsen	Cardinal
	Ochsen	Leichtbier
	Ochsen	Feldschlösschen
	Ochsen	Löwenbräu
	Hirschen	Cardinal
	Hirschen	Leichtbier
	Hirschen	Feldschlösschen
	Hirschen	Löwenbräu
	Krone	Cardinal
	Krone	Leichtbier
	Krone	Feldschlösschen
	Krone	Löwenbräu
	Löwen	Cardinal
	Löwen	Leichtbier
	Löwen	Feldschlösschen
	Löwen	Löwenbräu

Und noch eine Bieridee…

- Also: $\pi_{Attr(s) \setminus Attr(u)}(s) \setminus \pi_{Attr(s) \setminus Attr(u)} ((\pi_{\mathbf{Attr(s) \setminus Attr(u)}} \mathbf{(s) \times u) \setminus s})$

$\pi_{Attr(s) \setminus Attr(u)}$ (s) x u \ s	Restaurant	Biersorte
	~~Ochsen~~	~~Cardinal~~
	~~Ochsen~~	~~Leichtbier~~
	~~Ochsen~~	~~Feldschlösschen~~
	~~Ochsen~~	~~Löwenbräu~~
	~~Hirschen~~	~~Cardinal~~
	~~Hirschen~~	~~Leichtbier~~
	~~Hirschen~~	~~Feldschlösschen~~
	Hirschen	Löwenbräu
	~~Krone~~	~~Cardinal~~
	~~Krone~~	~~Leichtbier~~
	Krone	Feldschlösschen
	Krone	Löwenbräu
	~~Löwen~~	~~Cardinal~~
	~~Löwen~~	~~Leichtbier~~
	Löwen	Feldschlösschen
	Löwen	Löwenbräu

Und noch eine Bieridee…

- Also: $\pi_{Attr(s) \setminus Attr(u)}(s) \setminus \pi_{Attr(s) \setminus Attr(u)} ((\pi_{Attr(s) \setminus Attr(u)} (s) \times u) \setminus s)$

$\pi_{Attr(s) \setminus Attr(u)}((\pi_{Attr(s) \setminus Attr(u)} (s) \times u) \setminus s)$	Restaurant
	Hirschen
	Krone
	Löwen

Und noch eine Bieridee…

- Also: $\pi_{Attr(s) \setminus Attr(u)}(s) \setminus \pi_{Attr(s) \setminus Attr(u)}((\pi_{Attr(s) \setminus Attr(u)}(s) \times u) \setminus s)$

$\pi_{Attr(s) \setminus Attr(u)}(s)$	Restaurant
	Ochsen
	Hirschen
	Krone
	Löwen

\

→

Restaurant
Ochsen

$\pi_{Attr(s) \setminus Attr(u)}((\pi_{Attr(s) \setminus Attr(u)}(s) \times u \setminus s)$	Restaurant
	Hirschen
	Krone
	Löwen

Lernziele Lektion 5

- Schlüsselbegriffe kennen:
  - Schlüssel
  - Superschlüssel
  - Primärschlüssel
  - Fremdschlüssel
- Grundsätzliches Vorgehen beim Entwurf einer Datenbank kennen
- Grundlagen der «Entity Relationship» Diagrammsprache kennen

Bemerkungen zur Notation

- Schreibweisen, Abkürzungen:
  - Für eine Menge M bezeichnet card(M) die Anzahl ihrer Elemente. Wenn M eine Relation ist dann bezeichnet card(M) also die Anzahl Tupel.
  - Für eine Relationenformat R(A,B,C,…) bezeichnet Attr(R) die Menge der Attribute. Also Attr(R) = {A,B,C,…}
  - Bemerkung: Für bags wird die Kurzschreibweise < a,b,c,m> (m = Multiplizität) verwendet statt <<a,b,c>,m>

Schlüsselbegriffe

- Schlüsselbegriffe sind im Datenbankbereich omnipräsent, allerdings oft unpräzise verwendet:
  - Suchschlüssel
  - Primärschlüssel
  - Surrogatschlüssel
  - Fremdschlüssel
  - Superschlüssel
  - Schlüsselkandidat
  - Verteilschlüssel, Sortierschlüssel, …
  - …
- Worum geht es?

Schlüssel

- Ursprüngliche und zentrale Bedeutung von «Schlüsseln» in DBMS sind ihre Rolle als «Suchschlüssel» («search key»).
- Aus dieser Überlegung folgt, dass es nicht zwei verschiedene Objekte mit dem gleichen Schlüssel geben soll.
- Bei Relationen geht es um die Frage, welche Attributwerte eines Tupels das ganze Tupel eindeutig «festlegen».

Beispiel

- Bsp.: Mitarbeiter(Personalnummer,Name,Vorname,Abteilung)

- Relationen (zu verschiedenen Zeitpunkten):

r_1={<17,Meier,Joseph,A>,<17,Meier,Joseph,B>}

r_2={<16,Meier,Joseph,A>,<17,Meier,Joseph,B>,
<18,Meier, Johannes,A>,<19,Müller,Joseph,A}

r_3={<17,Meier,Joseph,A>,<18,Meier,Joseph,A>}

r_4={<17,Meier,Joseph,A>,<17,Meier,Johannes,A>}

r_5={<17,Meier,Joseph,A>,<18,Meier,Johannes,A>}

r_6={<18,Meier,Joseph,A>,<18,Meier,Joseph,B>,
<18,Meier,Johannes,A>,<18,Müller,Joseph,A>,
<19,Meier,Joseph,A>}

r_7={<17,Meier,Joseph,A>}

Beispiel

- Identifizierende Attributmengen:

```
r1: {Abteilung}, {Name,Abteilung}, ...
r2: {Personalnummer}, {Name,Vorname,Abteilung}
r3: {Personalnummer}
r4: {Vorname}, {Name,Vorname}
r5: {Personalnummer}, {Vorname}
r6: {Personalnummer,Name,Vorname,Abteilung}
r7: jede nichtleere Teilmenge
```

Schlüssel

- Datenbank-Designer wollen eine zeitunabhängige Methode, um die Tupel zu identifizieren.
- Beispiel: Designer legt fest, dass {Personalnummer} und {Name,Vorname} je identifizierenden Charakter haben. Ein RDBMS kann das garantieren (siehe später: Integritätsbedingungen).
- Macht es Sinn, auch noch {Name,Vorname,Abteilung} als identifizierend festzulegen?

Superschlüssel

- Ein Schlüssel ist eine identifizierende Attributkombination (d.h. eine «nicht-leere Menge von Attributen»).
- Für Schlüssel wird aber im Allgemeinen nicht nur verlangt, dass sie identifizierend sind, sondern auch, dass sie minimal sind.
- Wenn zwei Attributmengen K_1 und K_2 als identifizierend festgelegt werden, wobei K_1 echt in K_2 enthalten ist (enthalten und nicht gleich), dann wird höchstens K_1 als Schlüssel bezeichnet, K_2 als Superschlüssel (super key, Oberschlüssel).
- Gibt es für jede Relation einen (Super)Schlüssel?

Minimalität

- Die Minimalitätseigenschaft bezieht sich immer auf die Wahl des Datenbankdesigners des Relationenformats, nie auf aktuelle oder zukünftige Relationen zu diesem Format.

- Sei r eine Relation und K eine (nichtleere) Teilmenge der Attribute

- K ist Superschlüssel von r, falls $card(\pi_K(r)) = card(r)$

- K ist ein Schlüssel, falls für alle (nichtleeren) $K' \subset K$ gilt:

$$card(\pi_{K'}(r)) < card(r)$$

(d.h. man kann keine Attribute weglassen).

Syntaktisch vs. semantisch

- Es gibt einen Unterschied zwischen syntaktischem Schlüssel (Relationenformat) und semantischem Schlüssel (auf eine konkrete Relation bezogen).

- Sei r eine beliebige Relation zum Relationenformat R(A,B,C,…) und K eine nicht-leere Teilmenge von Attr(R). Dann:

  - ist K ein semantischer Superschlüssel von r falls gilt:

    $card(\pi_K(r)) = card(r)$ (d.h. gilt für konkrete Attributwerte)

  - Ist K ein syntaktischer Superschlüssel von R falls für alle Relationen r zum Relationenformat R gilt:

    $card(\pi_K(r)) = card(r)$ (d.h. gilt IMMER)

Primär-/Fremdschlüssel

- Einen der Schlüssel(kandidaten) des Relationenformats kann der Designer als Primärschlüssel bezeichnen.
- Andere Relationenformate können in einem Fremdschlüssel darauf Bezug nehmen.
- Beispiel:
  - Seien im Relationenformat Mitarbeiter(Personalnummer,Name, Vorname,Abteilung) die Attributmengen {Personalnummer} und {Name, Vorname} Schlüssel. Es sei {Personalnummer} als Primärschlüssel gewählt.
  - In einem zweiten Format Kinder(Personalnummer, Vorname) kann {Personalnummer} als Fremdschlüssel zu Mitarbeiter gewählt werden.

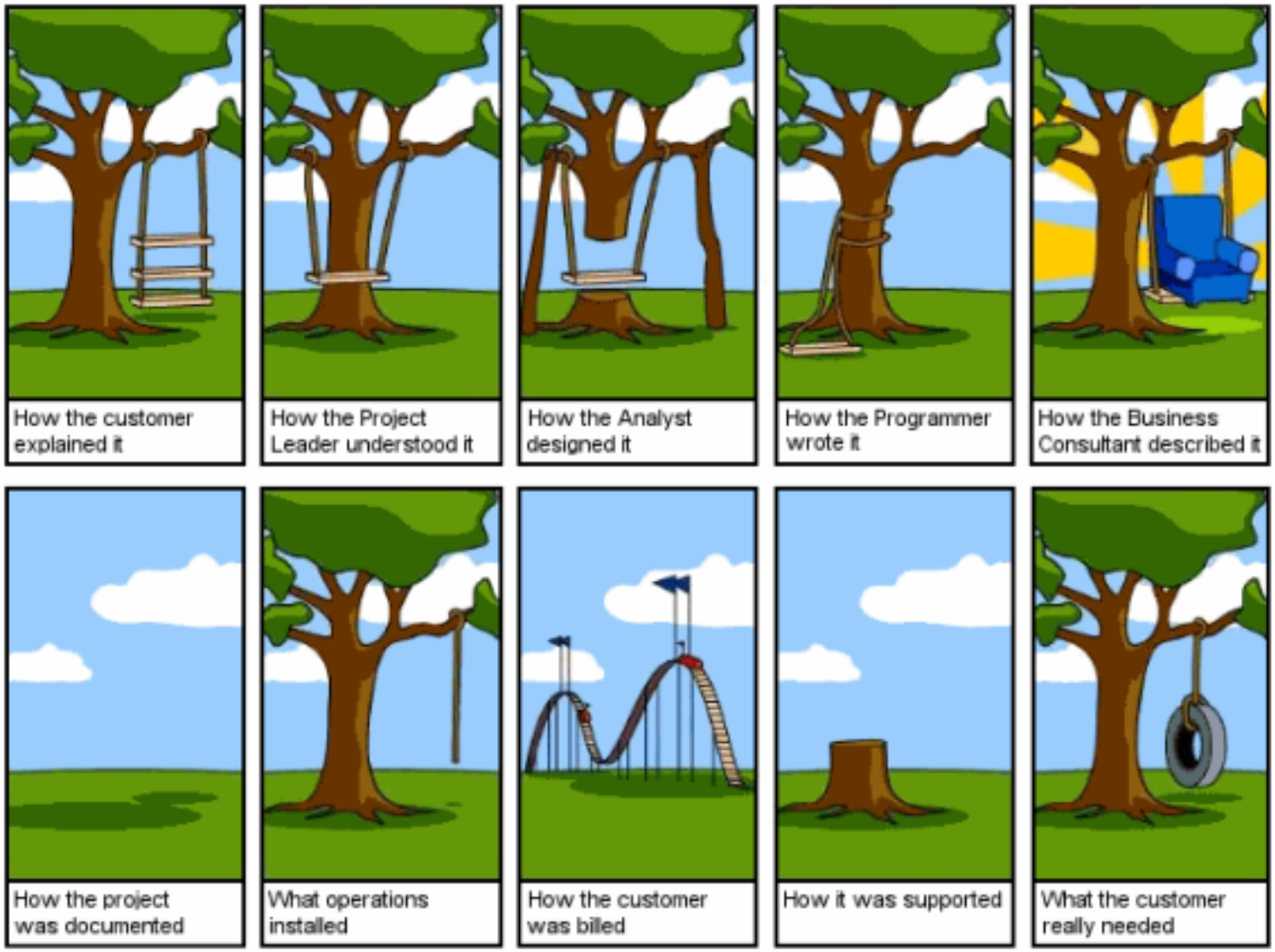
Primär-/Fremdschlüssel

- RDBMS garantiert die Fremdschlüssel-Primärschlüssel-Beziehung.
- Also (Relation m zu Mitarbeiter, Relation k zu Kinder):

  $\pi_{\text{Personalnummer}}(k) \subseteq \pi_{\text{Personalnummer}}(m)$

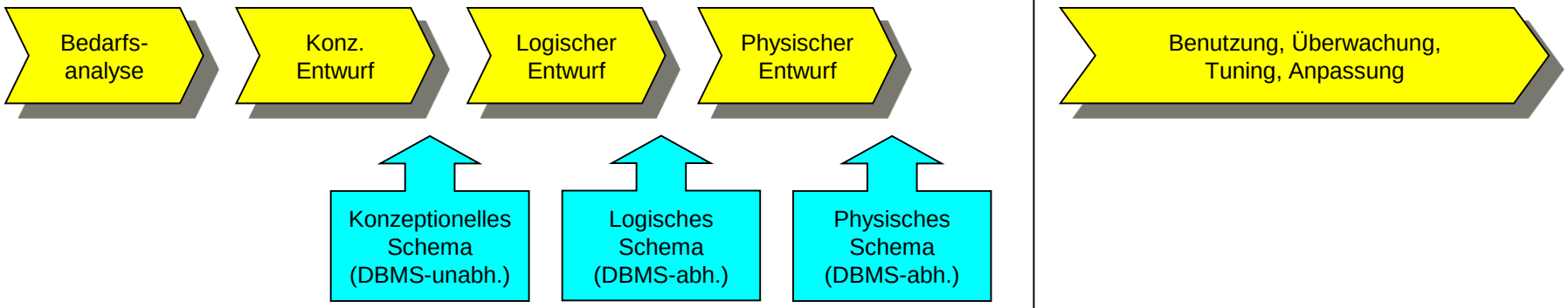
- Achtung: Fremdschlüssel sind nicht notwendigerweise Schlüssel (sondern zeigen auf einen Primärschlüssel „in der Fremde“).
- Bei Primär-Fremdschlüssel-Beziehungen spricht man auch von referentieller Integrität.

And now for something completely different!



Entwicklungsschritte

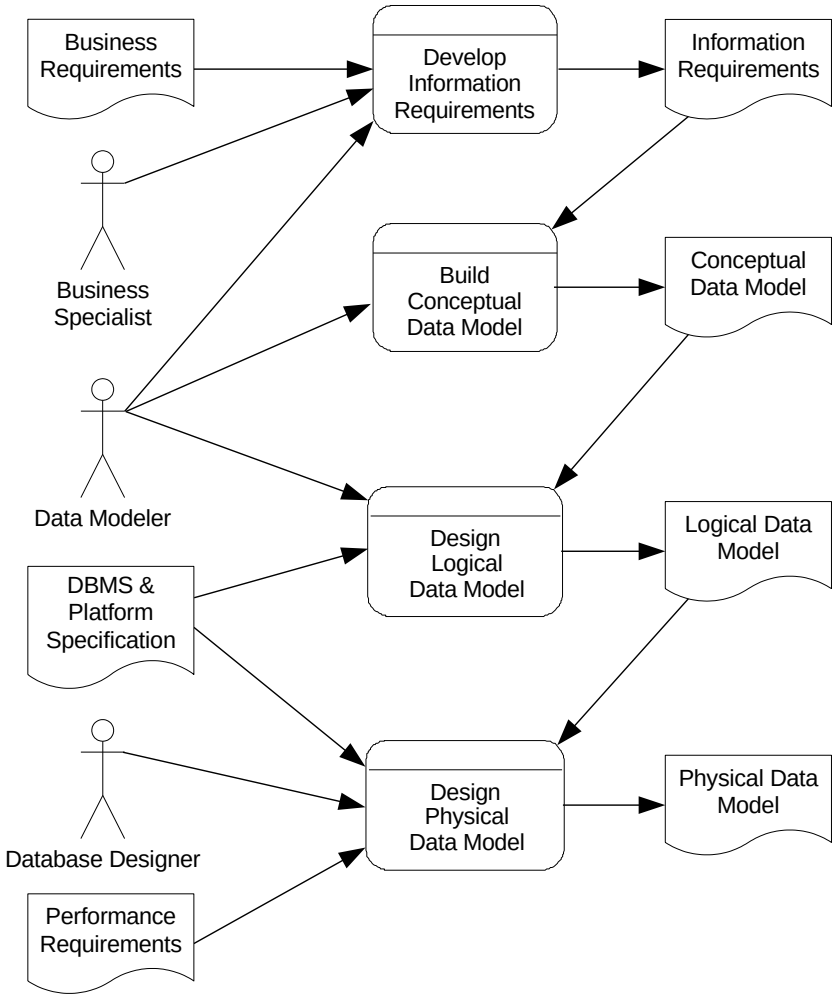
- Datenbank-Entwicklungsprozess



Datenmodellierung in der realen Welt

- Iterativer Prozess.
- Prozess in Phasen gliedern (konzeptionell → logisch → physisch).
- Grosse Projekte aufteilen.
- Einsatz von Werkzeugen zwingend → Einfluss auf Darstellungs-möglichkeiten.
- „Pflege“ eines Modelles (Wartung, „refactoring“) bei einer bereits existierenden Datenbank in der Regel sehr aufwändig → wird darum nur selten gemacht!

3-Ebenen: Konz. / logisch / physisch



3-Ebenen: Konz. / logisch / physisch

- Das konzeptionelle Datenmodell ist eine weitgehend technologie-unabhängige Spezifikation der Daten, die in der Datenbank gespeichert werden sollen.

- Das logische Datenmodell ist eine Übersetzung des konzeptionellen Schemas in Strukturen, die mit einem konkreten DBMS implementiert werden können.

- Das physische Datenmodell umfasst alle Anpassungen, die nötig sind um eine befriedigende Leistung im Betrieb zu erreichen (Datenverteilung, Indexierung, …).

Entity-Relationship-Modell (ERM)

- Erfunden 1976 von P.P.S. Chen

- Viele Varianten
  - Konzepte
  - Darstellungsarten, Notationen
- Sehr weit verbreitet.
- «Grafische Sprache», unabhängig von einem konkreten DBMS benutzbar.
- Für grössere Datenschemas nur mit Werkzeugunterstützung anwendbar (grafische Notation).
- Wenige Grundkonstrukte.

Entity-Relationship-Diagramme

Entity Relationship (ER)

- Im folgenden Darstellung eines ER-Dialektes, der sich für das Design von relationalen Datenstrukturen gut eignet.

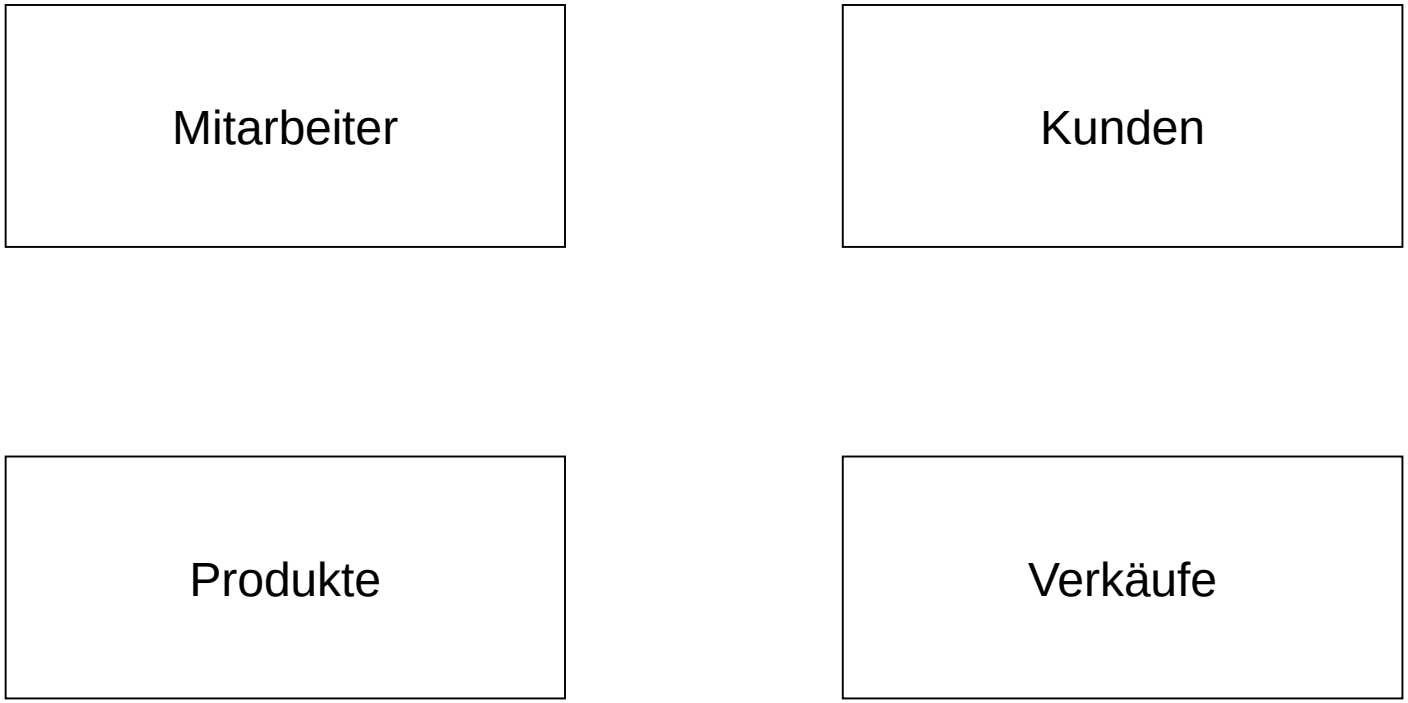
- Liefert bei sinnvoller Anwendung direkt Relationen in IDNF (Inclusion Dependency Normal Form), vermeidet NULLs, lässt nur Semantik zu, die von RDBMS überwacht werden kann.

Entität, Entitätstyp

- Entitätstyp als «undefined notion» (siehe auch Punkt in Geometrie)
  = «Etwas» über das wir Daten in einer Datenbank speichern wollen.
- Darstellung durch Rechteck mit eindeutigem Namen.
- Konkrete Entitätstypen sollen möglichst genau definiert werden.
- Ein Entitätstyp steht für Mengen von Entitäten.
  Analogie in Java: Klassen/Objekte
- Ein Entitätstyp wird in eine Relation abgebildet werden, also eine Tabelle mit Schlüssel(n), und die Entitäten werden die Zeilen der Tabelle sein.

Entitätstyp: Darstellung

- Beispiele:



Attribut, Attributswert

- Entitätstypen haben Attribute
- Entitäten haben Attributwerte

- Bsp. "Meier" könnte z.B. Attributwert des Attributs "Name" einer Entität des Entitätstyps "Person" sein

- Wir stellen Attribute als Ovale dar.

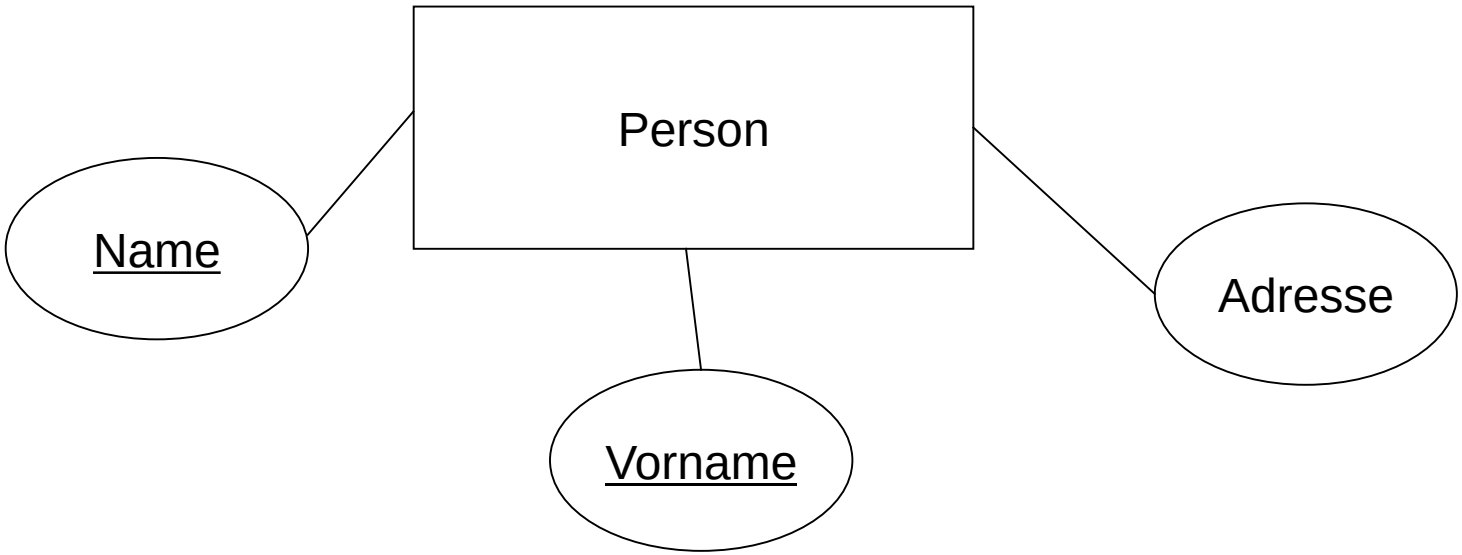
- Attribute sind mit Linien mit dem zugehörigen Entitätstyp verbunden.

- Wir lassen grundsätzlich nur "einfache" Attribute zu. D.h., hinter "Name" steckt ein Wert, nicht eine Liste von Werten.

Attribute: Darstellung

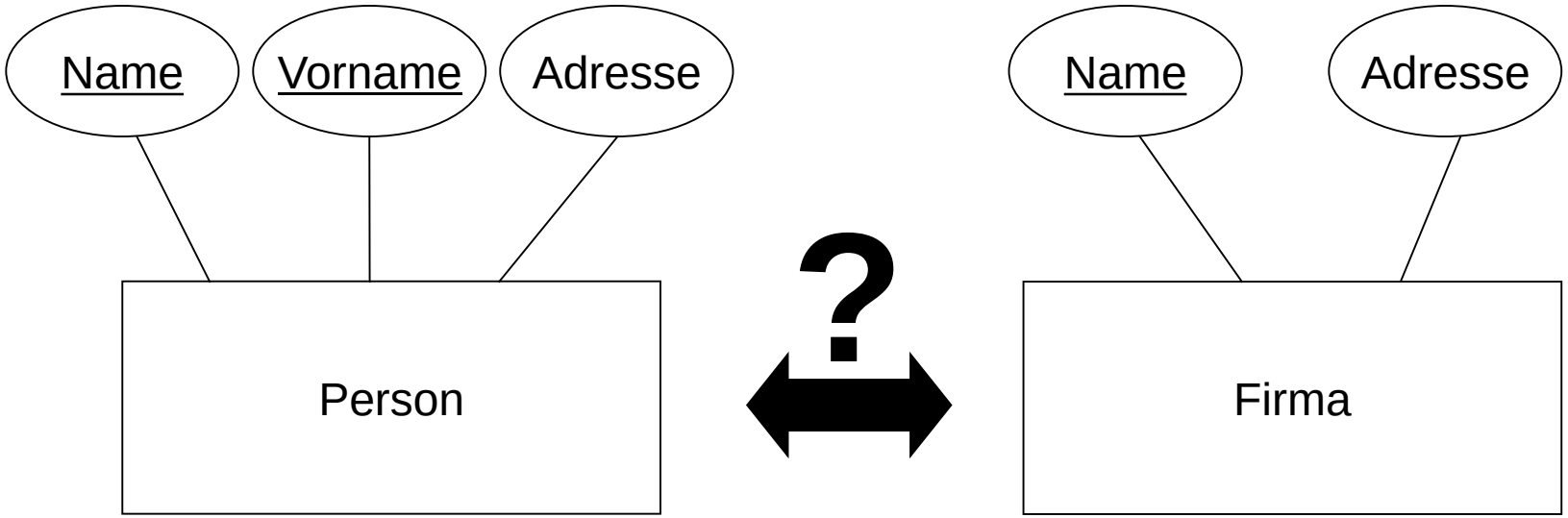
- Mit Hilfe der Unterstreichung wird angezeigt, dass die entsprechenden Attribute als Primärschlüssel gewählt wurden.

- Wir wählen im Allgemeinen aber erst dann einen Primärschlüssel, wenn der Entitätstyp referenziert wird (ansonsten nur Schlüssel).



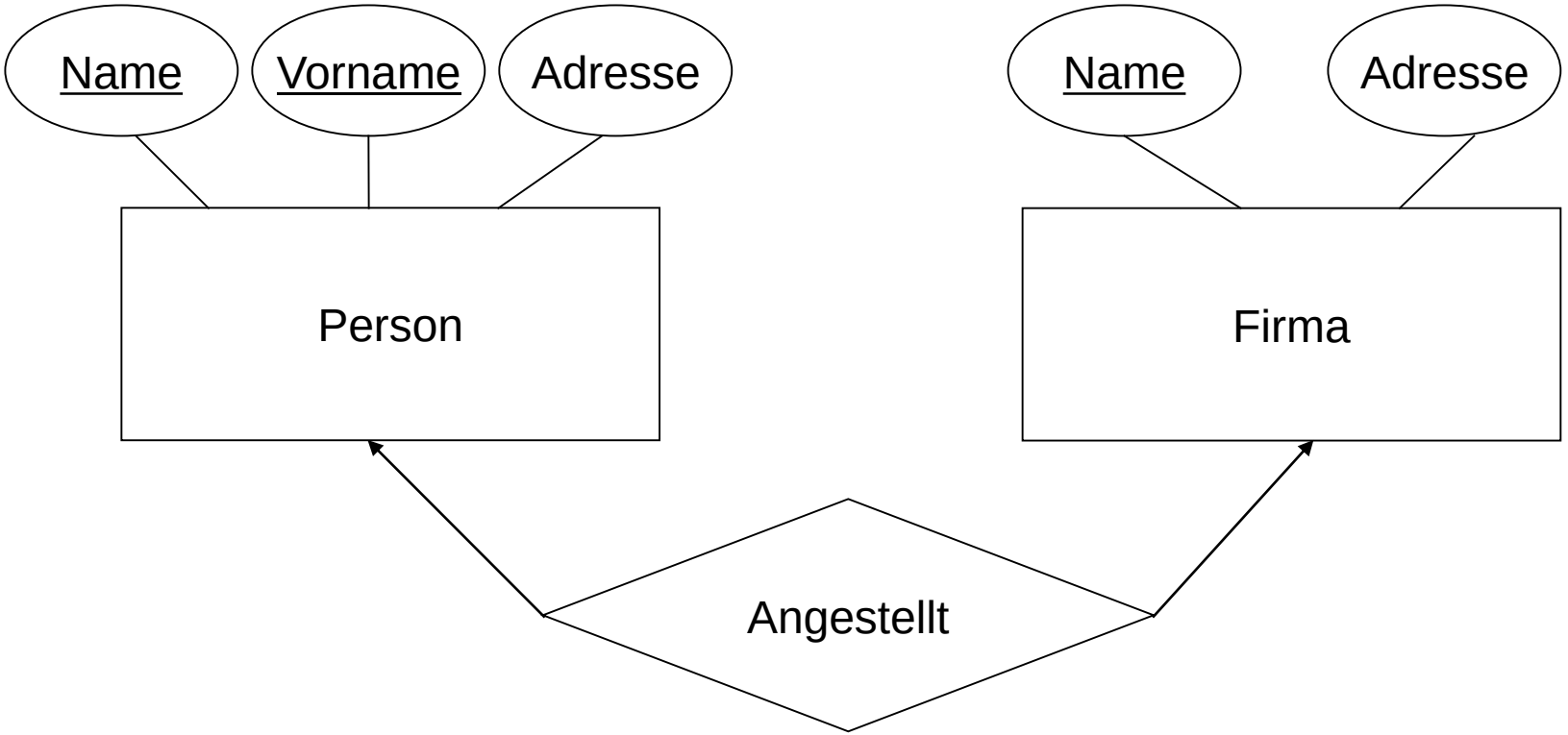
Beziehungen, Beziehungstypen

- Gegeben sei ein zweiter Entitätstyp «Firma», mit Primärschlüssel {Name}. Wir wollen nun ausdrücken, dass Personen in Firmen angestellt sein können.



Beziehungstypen

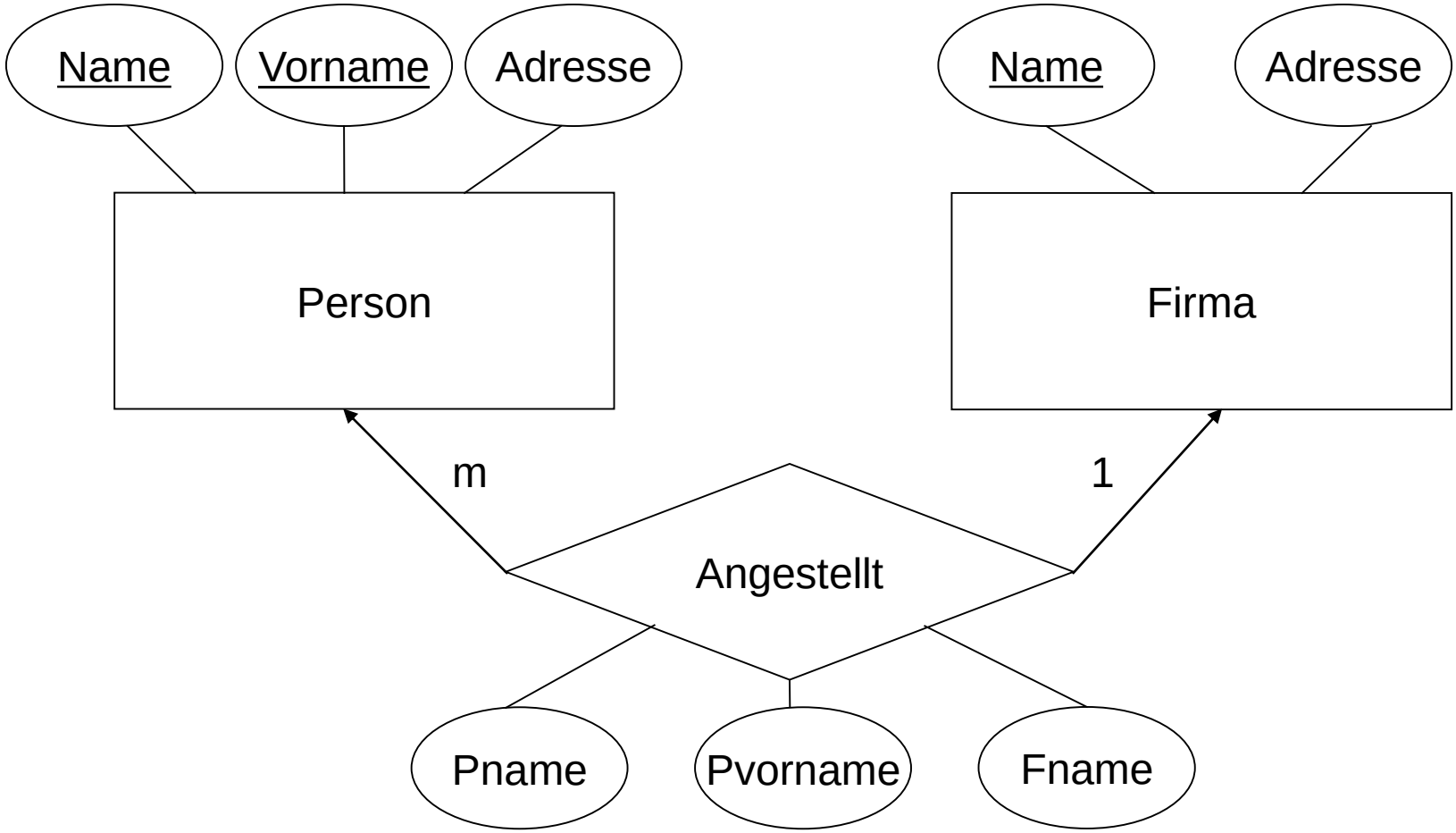
- Wir stellen einen Beziehungstyp (hier: «Angestellt») durch einen Rhombus dar.



Attribute

- Welche Attribute werden mindestens benötigt, damit "Angestellt" die Verbindung zwischen "Person" und "Firma" bilden kann?
- Was müssen wir sonst noch wissen, damit wir die Beziehung zwischen "Person" und "Firma" spezifizieren können?
- Inwiefern ist "Angestellt" andersartig als "Person" oder Firma"? Warum führen wir einen neuen Begriff "Beziehungstyp" ein?

Beziehungstyp



Beziehungstyp

- Der Beziehungstyp "Angestellt" erbt die Primärschlüsselattribute der Entitätstypen "Person" und "Firma", von denen er abhängig ist.
- Er kann auch noch weitere, "eigene" Attribute haben (z.B. "seit" – um auszudrücken, seit wann die Anstellung besteht).
- Die Fremdschlüssel wählt man wenn möglich mit gleichem Namen, ausser bei Namenskonflikten.

- Die Pfeilmarkierungen 1,m (Kardinalitäten) drücken aus, dass pro Person höchstens eine Firma als Arbeitgeber existiert, während eine Firma beliebig viele (auch 0!) Angestellte haben kann.
- "m" bedeutet also beliebig viele, " unbestimmt" o.ä.

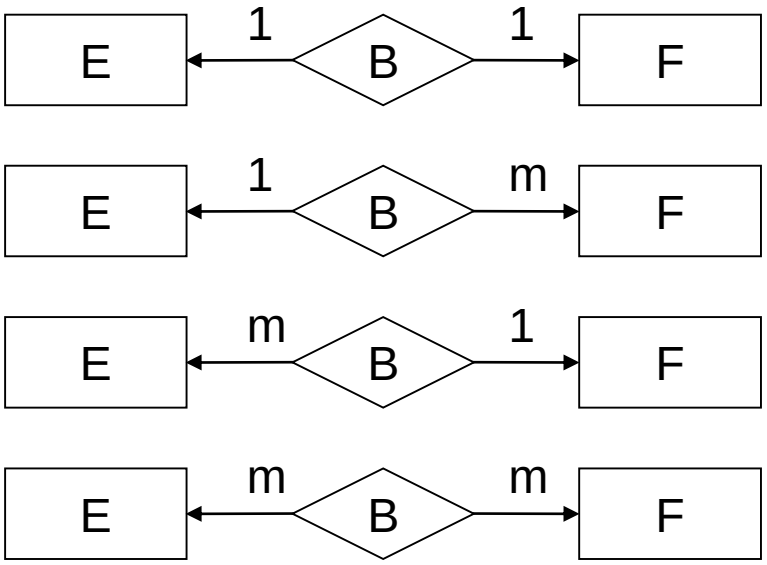
- Will man "Angestellt" als Relation abbilden, muss {Pname,Pvorname} als Schlüssel gewählt werden.

Beziehungstyp

- Warum ist der Schlüssel nicht {Fname}?
- Oder {Pname, Pvorname, Fname}?

Beziehungstyp

- Beziehungstypen haben Schlüssel, keine Primärschlüssel
  (Es sei denn, sie werden referenziert; siehe hierzu später)
- Mögliche Kombinationen: (inkl. passendem Schlüssel in B)



Schlüssel von E und F

Schlüssel von F

Schlüssel von E

Schlüssel von E komb. F

- Der Beziehungstyp ist existentiell abhängig von den Entitätstypen, welche er referenziert

Und weiter..

Das nächste Mal: ERM, komplexeres Beispiel