Datenbanksysteme 1

Datenmodellierung - Relationales Modell

Prof. Stefan Keller

Datenmodellierung - Überblick

- Überblick
  - DB-Analyse und -Entwurfsprozess
  - Das Entity-Relationship-Modell
  - Datenmodellierung mit UML
  - Das Relationale Modell

- ‚Datenmodell' steht für:
  - Konzept zur Beschreibung von Datenbanksystemen: Datenbankmodell
  - semantisches Datenmodell (eng. *conceptual schema*): in der Datenmodellierung eine abstrakte, formale Beschreibung eines Ausschnittes der wahrgenommenen Welt
  - die Beschreibung der Struktur einer Datenbank, besser: Schema

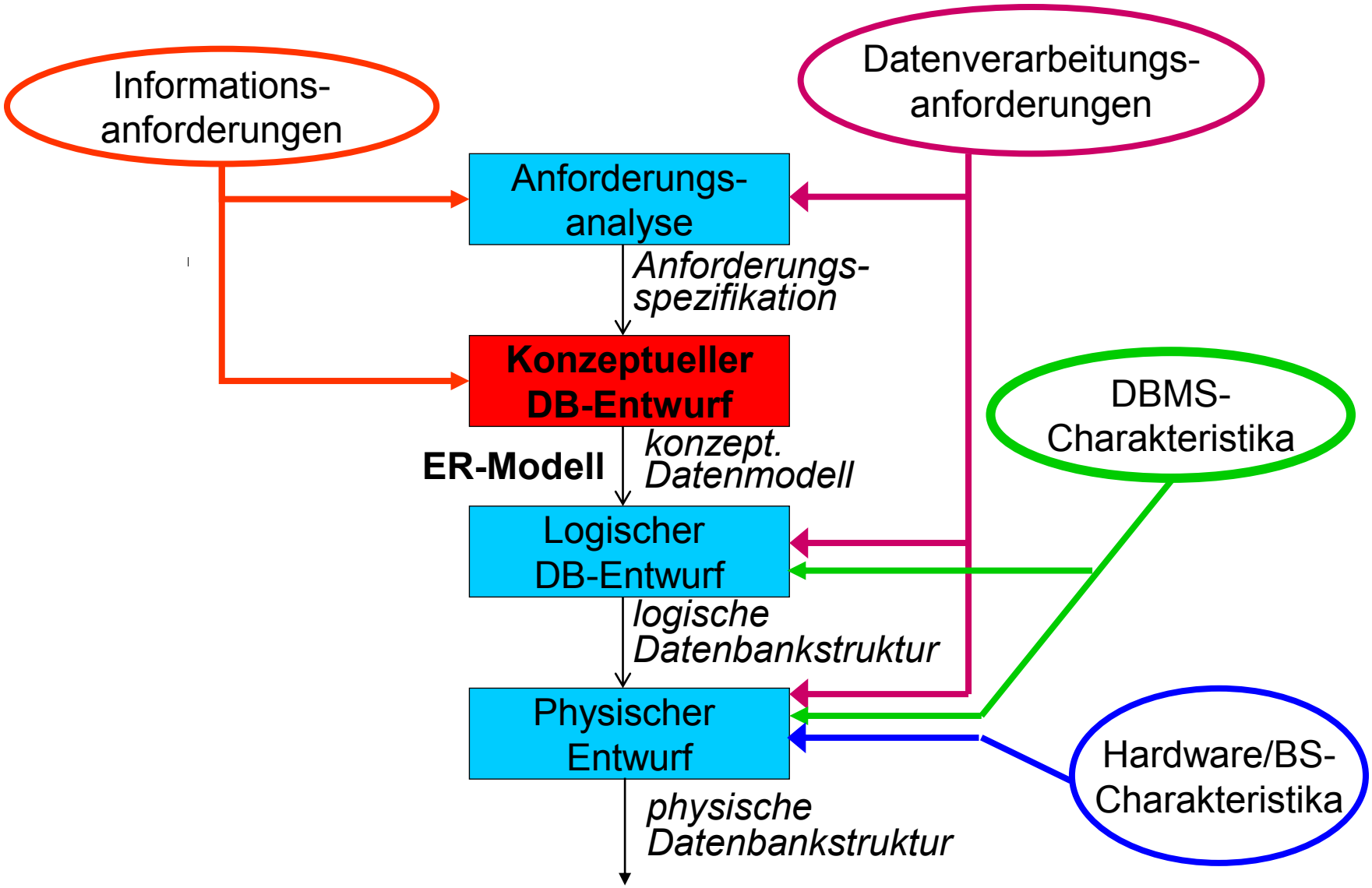
Analyse und Entwurf von relationalen Datenbanken

Datenmodellierung

Entwurf einer Datenbank

- Wie wird eine relationale Datenbank entworfen?
  - Genauer: Wie entwirft man die Tabellenstruktur für die Speicherung der Daten für eine spezifische Anwendung?
  - Der DB-Entwurf ist häufig Teil der Entwicklung eines SW-Entwicklungsvorhabens für ein IT-System (meist ein verteiltes Client-Server-System). Der DB-Entwurfsprozess ist eingebettet in den Software-Engineering-Prozess.
  - Wir betrachten hier den DB-Entwurf isoliert vom Software-Engineering-Prozess und nur im Überblick.

DB-Entwurfsprozess



Datenbank-Entwurfsprozess

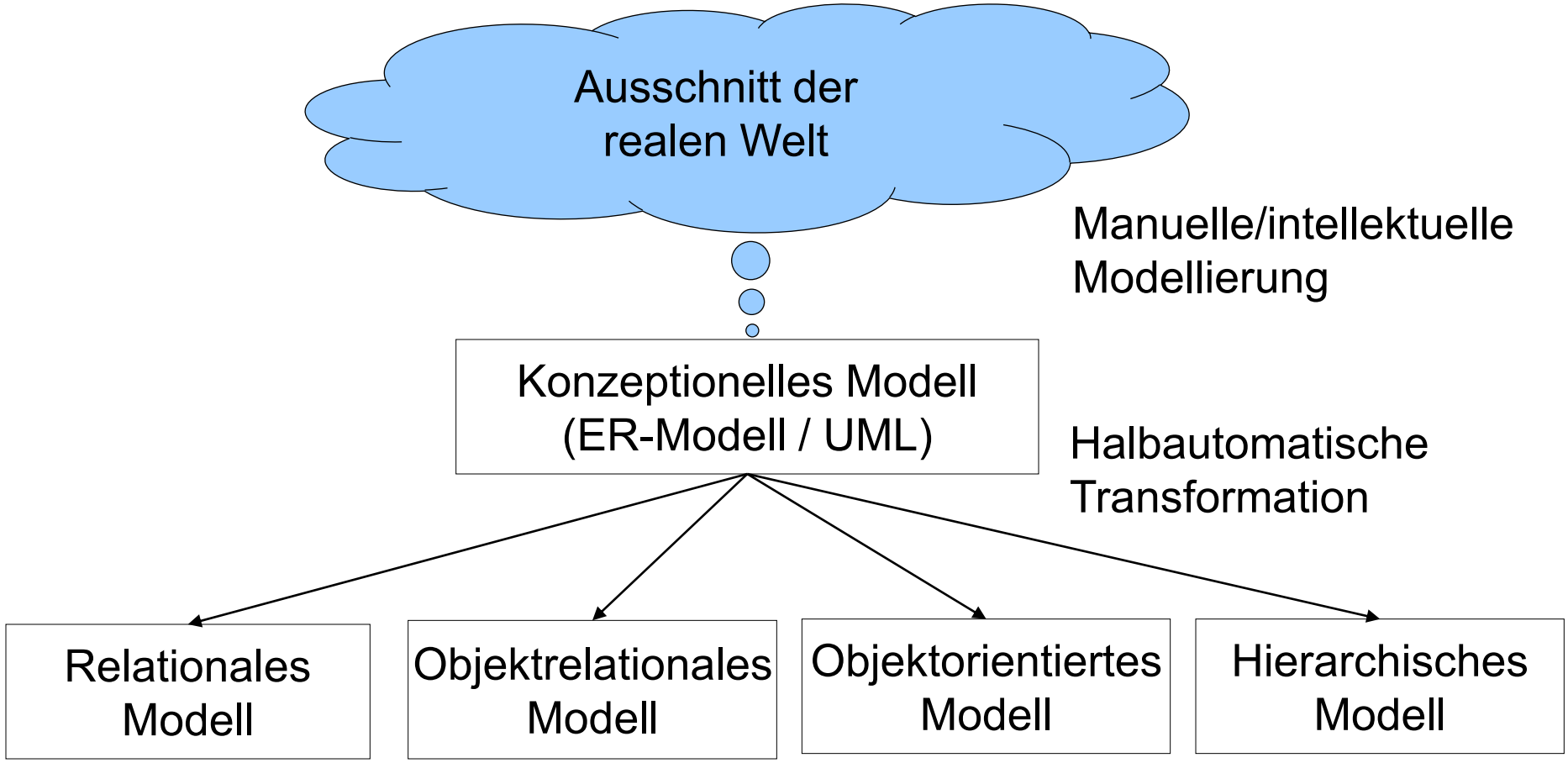
Anforderungsanalyse
- Resultat: Anforderungsspezifikation

Konzeptioneller DB-Entwurf
- Resultat: Konzeptionelles Datenmodell =
  Logisches Modell der in der Datenbank zu speichernden Objekte und Beziehungen
  - Darstellung als UML- oder Entity-Relationship-Modell
  - Modell ist Technologie-neutral

Logischer DB-Entwurf
- Resultat: Logisches Modell der DB-Objekte (z.B.: Tabellen)
- Entwurf abhängig vom verwendeten DB-Paradigma (relational, OO, etc.)

Physischer DB-Entwurf
- Resultat: produktspezifische SQL-Scripts mit Definitionen der DB-Objekte (Tabellen, Views), Konsistenzbedingungen, interner Speicher- und Indexstruktur
- Entwurf abhängig vom verwendeten SW-Produkt (Oracle, MS SQL Server, PostgreSQL, MySQL, ...)

Datenmodellierung



Konzeptioneller DB-Entwurf = Datenmodellierung

Der Entwurf des **konzeptionellen Modells** ist die zentrale und schwierigste Aufgabe im Entwicklungsprozess einer Datenbank. Da aus dem konzeptionellen Modell alle anderen Modelle und damit die Strukturierung der Daten im DBMS hergeleitet werden, können Fehler im konzeptionellen Entwurf unter Umständen nur mit grossem Aufwand korrigiert werden.

Der Datenbank-Entwerfer muss beim Entwurf des konzeptionellen Modelles herausfinden, welche Objekte (Entitäten) seiner Welt (z.B eines Unternehmens oder einer technischen Anlage) und welche Beziehungen und Abhängigkeiten zwischen diesen Objekten er datenmässig erfassen muss. Für diese **Datenmodellierung** benötigt er Beschreibungs-mechanismen und Methoden.

Das konzeptionelle Datenmodell

- Das konzeptionelle Datenmodell
  - definiert die (Daten-) Objekte und deren Eigenschaften
  - definiert die Zusammenhänge und die Beziehungen zwischen den (Daten-) Objekten
  - legt Konsistenzbedingungen fest
  - ist Lösungs- und Technologieunabhängig („Was“ und nicht „Wie“)
- Entity Relationship (ER)-Diagramm
  - Die klassische Datenmodellierung verwendet die ER-Technik.
  - Das (klassische) konzeptionelle Datenmodell ist statisch, es gibt keine Auskunft über funktionale und dynamische Eigenschaften des modellierten Systems.
- UML
  - Unified Modelling Language (UML) erweitert die ER-Technik um dynamische Aspekte. UML wird in der objektorientierten SW-Entwicklung eingesetzt und taugt auch für Datenmodellierung (als ER-Ersatz)

Analyse und Entwurf von relationalen Datenbanken

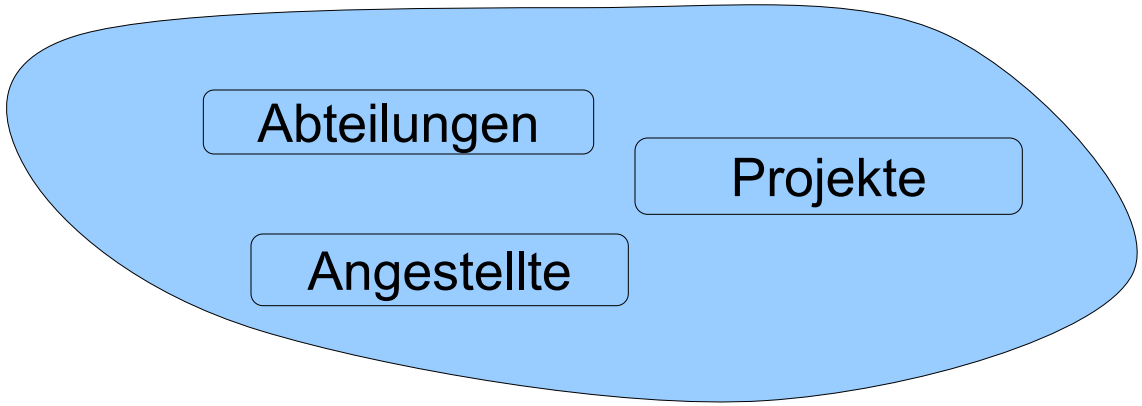
Das ER-Modell

ER-Modell

- Entity-Relationship-Technik (ER-Modell)
  - modelliert den interessierenden Ausschnitt der 'Welt' grafisch als Entitätsmengen (Typen) mit den Beziehungen zwischen diesen Mengen (Assoziationen).
  - vergleichbar mit „Domain Model“

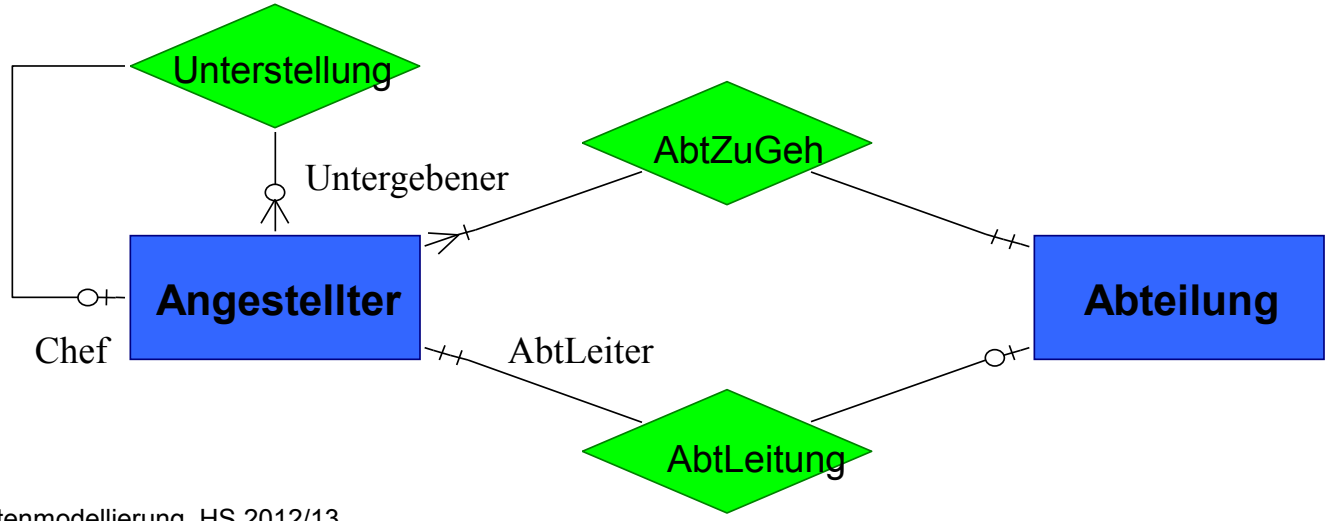
- ER-Diagramm:
  - graphische Darstellung der Entitätstypen und der Assoziationen.

Modellierung einer kleinen Beispielanwendung



Reale Welt: Unternehmen

Analyse und Entwurf,
konzeptionelle Modellierung
(Krähenfussnotation)



Grundbegriffe der Datenmodellierung I

- **Entität** Individuelles Element (Objekt) des betrachteten Systems, beschrieben durch Merkmale (Attribute), eindeutig identifizierbar.
  - Bsp. Der Angestellte ‚Fritz Meier'
- **Entitätsmenge** Menge von Entitäten, die bezüglich bestimmter Eigenschaften oder Beziehungen als gleichartig behandelt werden.
  - Bsp: Die Angestellten, die in ‚Zürich' wohnen
- **Attribut** Beschreibung eines bestimmten Merkmales (property) der Entitäten einer Entitätsmenge. Das Attribut definiert einen Namen für das Merkmal und einen Wertebereich für die möglichen Merkmalswerte.
  - Bsp: Das Attribut Geburtstag mit dem Wertebereich Datum
- **Wertebereich** Eine Menge (oder ein Bereich) möglicher Werte eines skalaren Wertebereichs, die zur Beschreibung einer Eigenschaft verwendet werden.
  - Bsp: Der Bereich von 1000..9999 zur Speicherung einer Postleitzahl.
- **Attributwert** Der Attributwert wird durch ein Attribut einer Entität zugeordnet. Er beschreibt eine Eigenschaft der Entität.
  - Bsp: 23.6.72 ist dem Angestellten 'Fritz Müller' als Wert für das Attribut 'Geburtstag' zugeordnet.

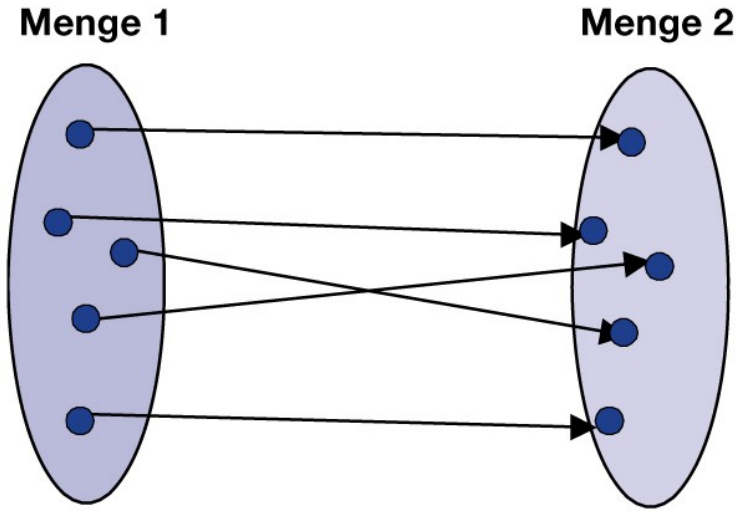
Grundbegriffe der Datenmodellierung II

- **Relation** Verknüpfung mehrerer Entitäten miteinander (Link). Eine Beziehung kann wie eine Entität Attributwerte aufweisen.
  - Bsp: 'Fritz Müller' gehört zur Abteilung 'Entwicklung'
  - Bsp: 'Fritz Müller' arbeitet zu 50% am Projekt 'Mars'
- **Entitätstyp** Zusammenfassung von Attributen, die auf einer Entitätsmenge definiert sind (Class). Eine Entitätstyp nimmt Bezug auf Gegenstände, die als gleichartig betrachtet werden, und beschreibt deren Eigenschaften. Die Elemente der Entitätsmenge sind Instanzen des Entitätstyps.
  - Bsp: Der Entitätstyp 'Angestellter' ist durch die Menge der Attribute 'Name', 'Vorname', 'Adresse', 'PersonalNummer', 'Abteilung', 'Salär' etc. definiert.
  - **Schwacher** Entitätstyp: Entitätstyp der (noch) kein eigenes Schlüsselattribut hat. (Ggs. normaler/starker Entitätstyp)
- **Assoziation** Verknüpft mehrere (in der Regel zwei) Entitätstypen. Binäre Assoziationen verknüpfen zwei Entitätstypen; n-wertige Assoziationen verknüpfen n Entitätstypen.

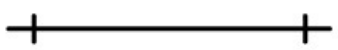
Beziehungen im ER-Modell

- Schlüssel
  - Künstliche Schlüssel
  - Primärschlüssel
  - Fremdschlüssel

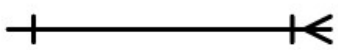
- Beziehungen im ER-Modell
  - 1:1
  - 1:n
  - n:m
  - rekursiv
  - Optionalität (c)
  - identifizierende



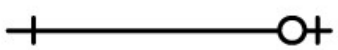
„Krähenfussnotation“



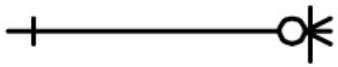
1 : 1 - Beziehung



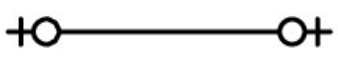
1 : n - Beziehung



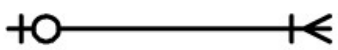
1 : c - Beziehung



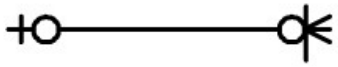
1 : cn - Beziehung



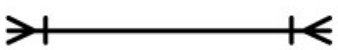
c : c - Beziehung



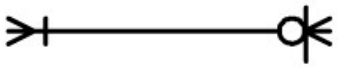
c : m - Beziehung



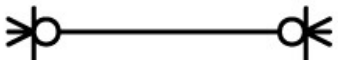
c : cm - Beziehung



n : m - Beziehung



n : cm - Beziehung



cn : cm - Beziehung

Erweiterungen des ER-Modells (EERM)

- Ergänzt ER-Modell um zusätzliche semantischen Eigenschaften:
  - mehrwertige Beziehungen
  - abhängige Entitäten (Weak Entity Type, Existence Dependency)
  - assoziative Klassen
  - Aggregationen
  - Generalisierungen

- Die meisten dieser Konzepte werden auch von UML unterstützt (siehe unten).

Entwurf von relationalen Datenbanken

Datenmodellierung mit UML

Datenmodellierung mit UML

- Unified Modelling Language UML

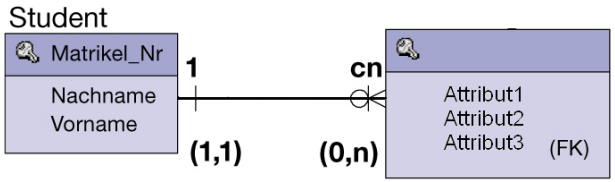
- UML-Klassendiagramm
  - Klasse (class) modelliert gleichartige Objekte hinsichtlich Struktur (~Attribute) und Verhalten (~Operationen)
  - Assoziationen zwischen Klassen modellieren Beziehungen (links) zwischen den Objekten der Klassen
  - Generalisierungshierarchien
  - Aggregation und Komposition
  - Assoziative Klassen

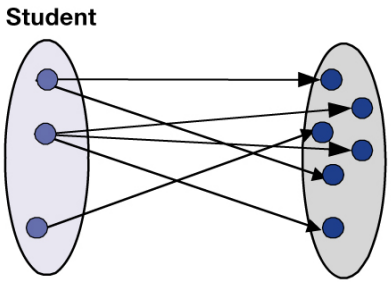
- UML-Objektdiagramm (Instanzdiagramm)

UML-Klassendiagr. vs. ER-Modell (Krähenfuss-Notation)

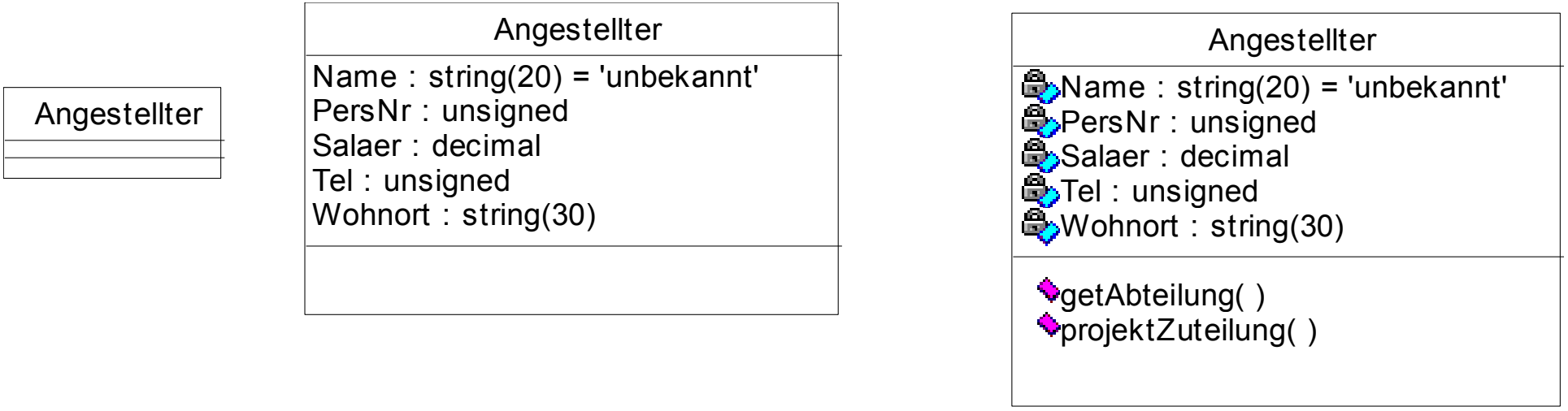
- UML
  - Vorteil: standardisiert, wird von vielen Tools unterstützt
  - Nachteil: in der DB-Community (noch) nicht überall bekannt
  - Klassendiagramme =~ ER-Diagramme + Operationen

- Achtung: Krähenfuss-Notation versus numerische UML-Notation von Beziehungen:
  - In UML wird angegeben, wie viele Beziehungen eine Entität min. und max. eingehen kann.
  - In K. bezeichnet 1:n (m) = 1:mehrere (nicht-optional), was in UML mit „1..1 zu 0..*“ bezeichnet wird!





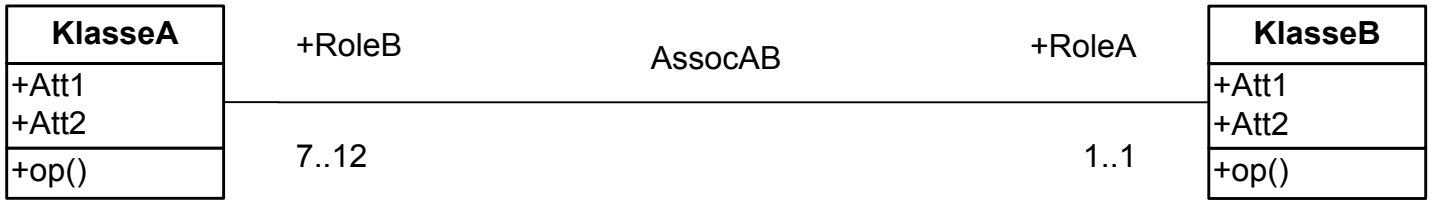
Klassen



Verschiedene Sichten der Klasse Angestellter

Manchmal ist UML-Klasse explizit als sog. Stereotyp gekennzeichnet:
<<Entity>> bzw. <<Entityset>>

Kardinalität



Jedes Element der KlasseA steht mit genau 1 Elementen der KlasseB (in der Rolle RoleA) in Beziehung

Jedes Element der KlasseB steht mit mindestens 7 und mit maximal 12 Elementen der KlasseA (in der Rolle RoleB) in Beziehung

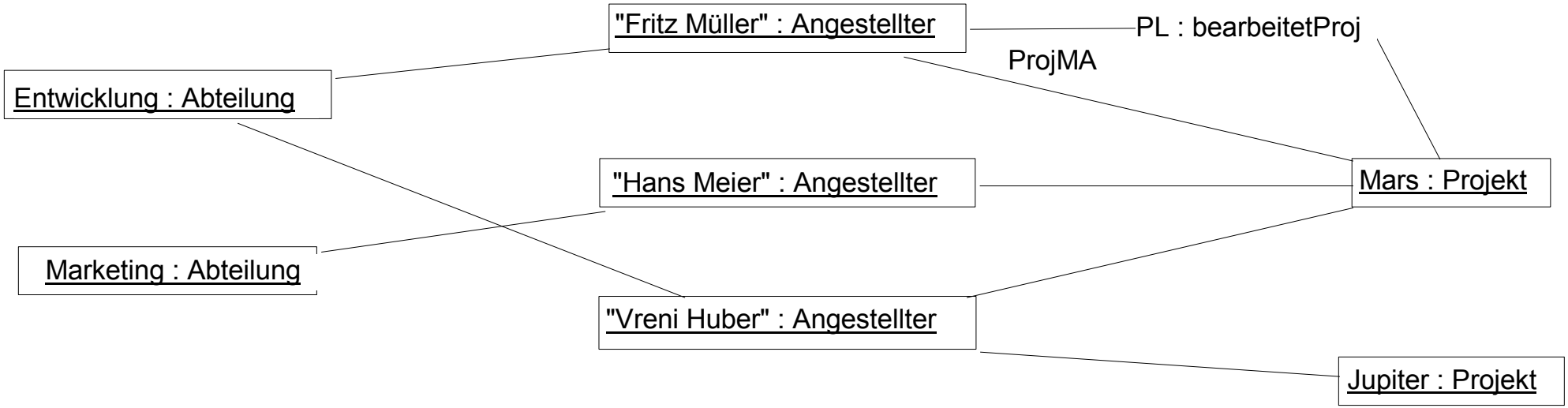
Kardinalitätsangaben

*	beliebig
0..*	0 oder mehrere
0..1	0 bis 1
i..j	i bis j
1..*	1 oder mehrere

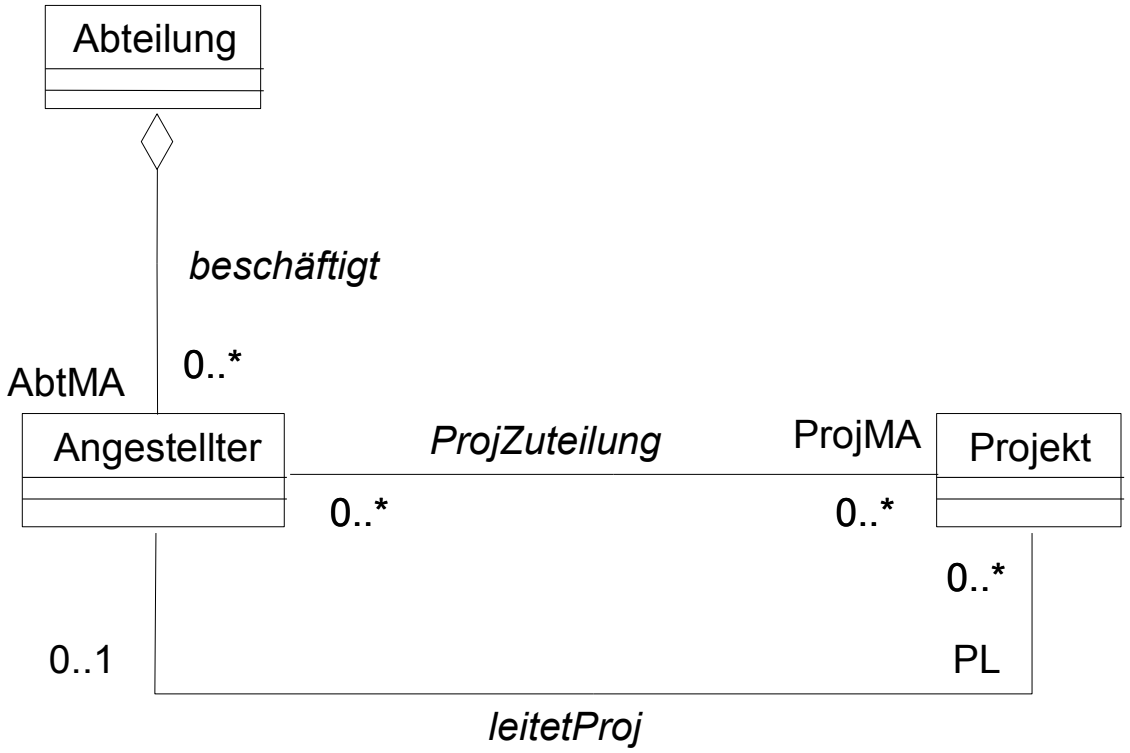
'Angestellter-Abteilung-Projekt'-Datenmodell

- Für jeden Angestellten sind Name, Wohnort und Salär zu speichern.
- Jeder Angestellte arbeitet in einer Abteilung. Eine Abteilung hat einen Namen und eine Abteilungsnummer, sie wird von einem Abteilungsleiter geführt (dieser ist natürlich auch Angestellter).
- Innerhalb der Angestellten einer Abteilung sind weitere Hierarchien möglich: Ein Mitarbeiter kann mehrere Untergebene führen. Es ist sogar möglich, dass einzelne dieser Untergebenen wiederum Chefs von anderen Mitarbeitern sind.
- Die Datenbank soll so konzipiert werden, dass neben den Angestellten- und Abteilungsdaten auch die Abteilungszugehörigkeit und die Unterstellungsbeziehungen erfasst werden können.
- Die Firma bearbeitet Projekte. Projekte werden von einem Projektleiter geleitet. Ein Angestellter arbeitet in einem oder mehreren Projekten zu einem bestimmten Prozentsatz seiner Arbeitszeit (zwischen 10 und 90%)

Objektdiagramm



Klassendiagramm

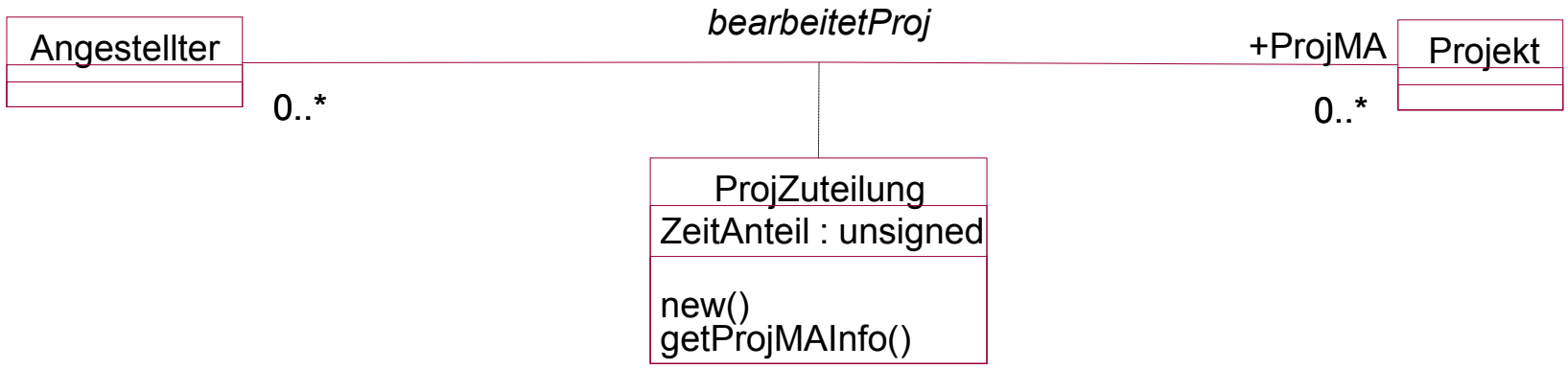


Assoziative Klassen

Assoziative Klassen sollten immer dann verwendet werden,

- wenn Beziehungen (von n: m Assoziationen) Eigenschaften haben
- wenn auf Beziehungen Operationen definiert sind

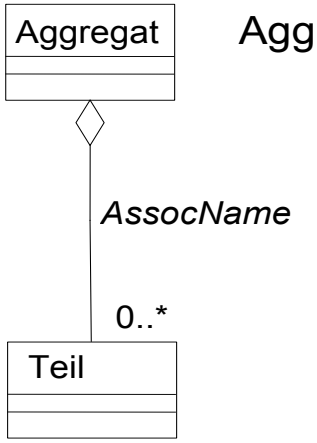
Die Beziehungsobjekte einer assoziativen Klasse sind existentiell abhängig von ihren verknüpften Objekten.

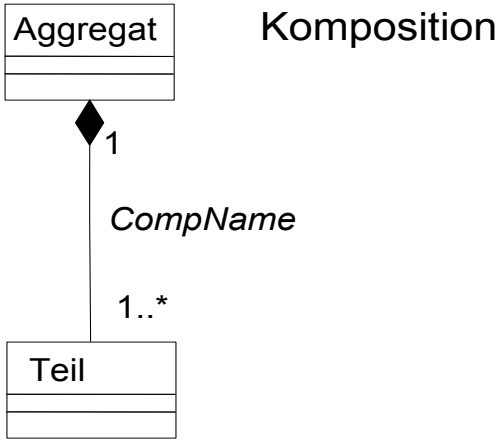


Beispiel: Die assoziative Klasse ‚ProjZuteilung' modelliert die Beziehungsattribute der n:m - Assoziation ProjZuteilung

A. Klasse stellt sicher, dass zwischen einem Angestellten A und einem Projekt P nur ein Link (=Instanz der A. Klasse) existiert,

Aggregation und Komposition





Aggregation und Komposition

- Modellierung von hierarchisch strukturierten Objekten: “part-of”
- gerichtet, asymmetrisch und transitiv.
- spezielle Assoziationen

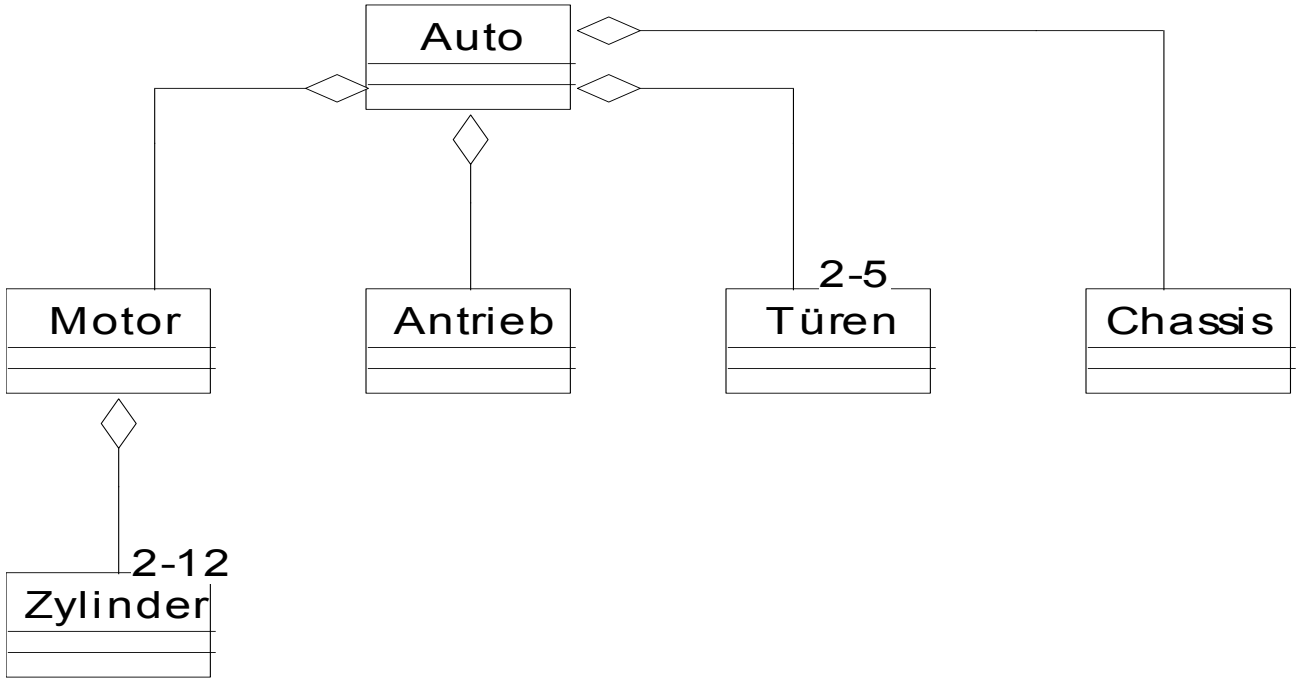
Aggregation

- Objekt kann Teil von mehreren Aggregatobjekten sein (Kardinalität > 1)
- Teil muss nicht existentiell abhängig sein vom Aggregat

Komposition = Spezialfall der Aggregation

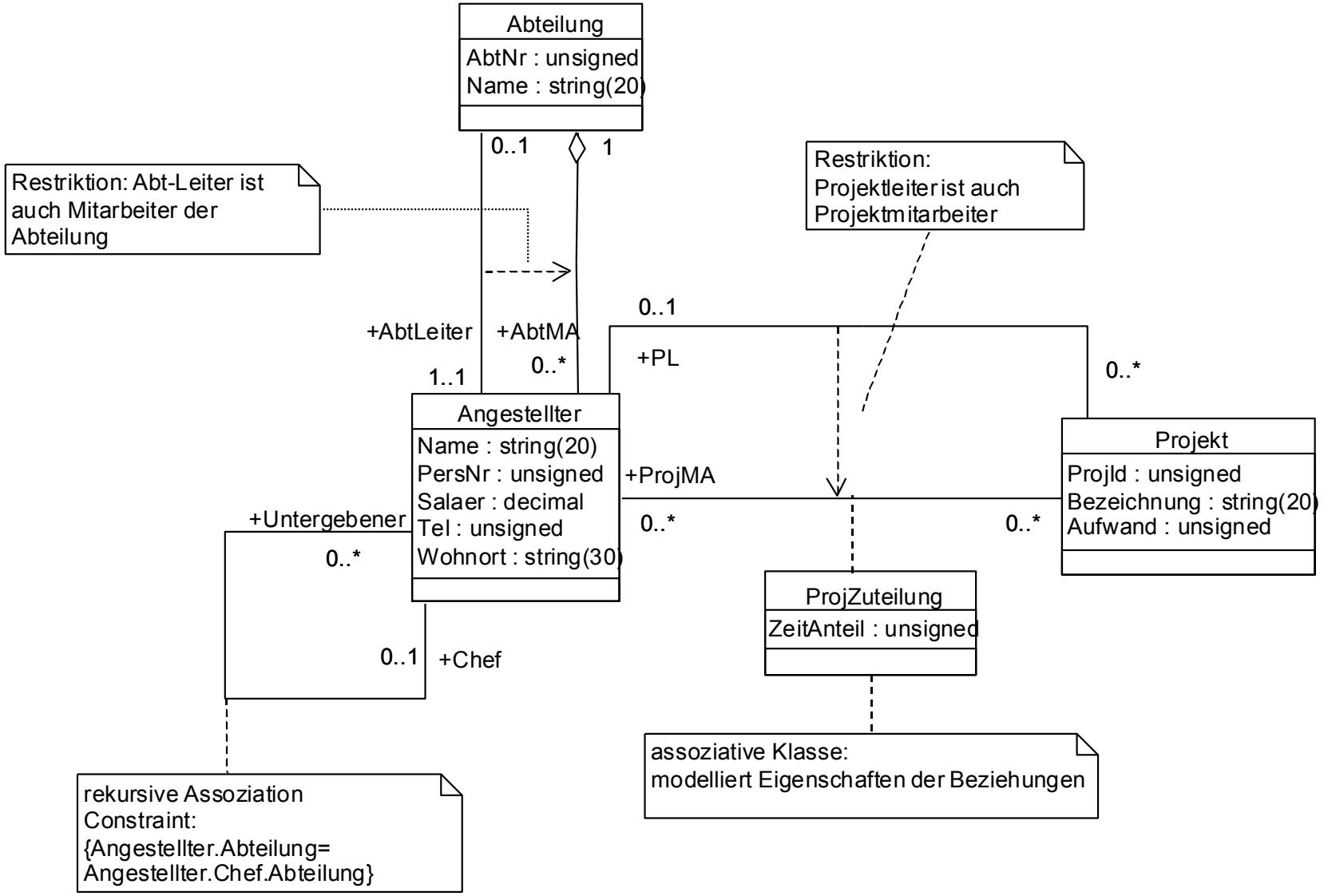
- Teile sind existentiell abhängig vom Aggregat
- Rekursive Aggregationen beschreiben Strukturen mit beliebiger Anzahl Verschachtelungsstufen.

Aggregation Beispiel

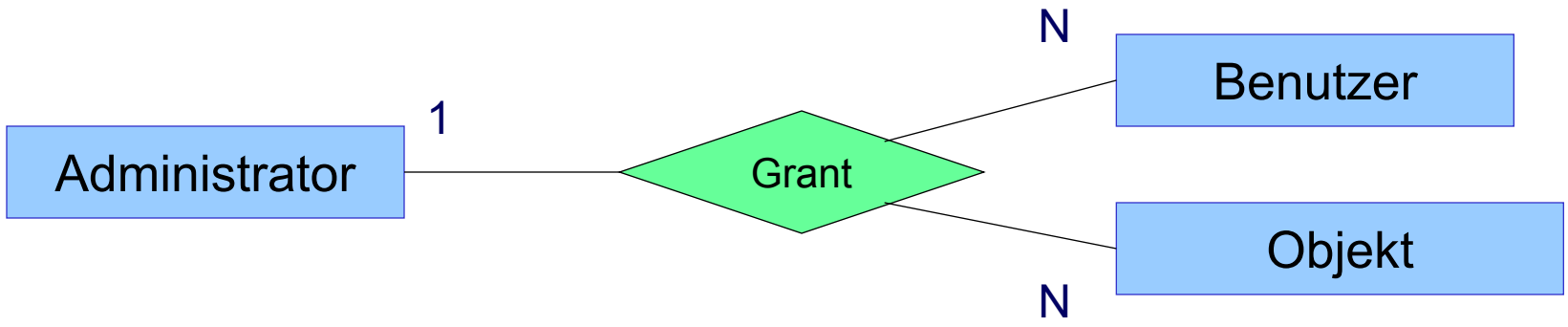


- Ein Auto ist ein komplexes, zusammengesetztes Objekt bestehend aus Motor, Antrieb, 2 bis 5 Türen und einem Chassis
- Ein Motor besteht aus 2 bis 12 Zylindern

Konzeptionelles Modell: Angestellter-Abteilung-Projekt



3-wertige-Assoziation: Grant



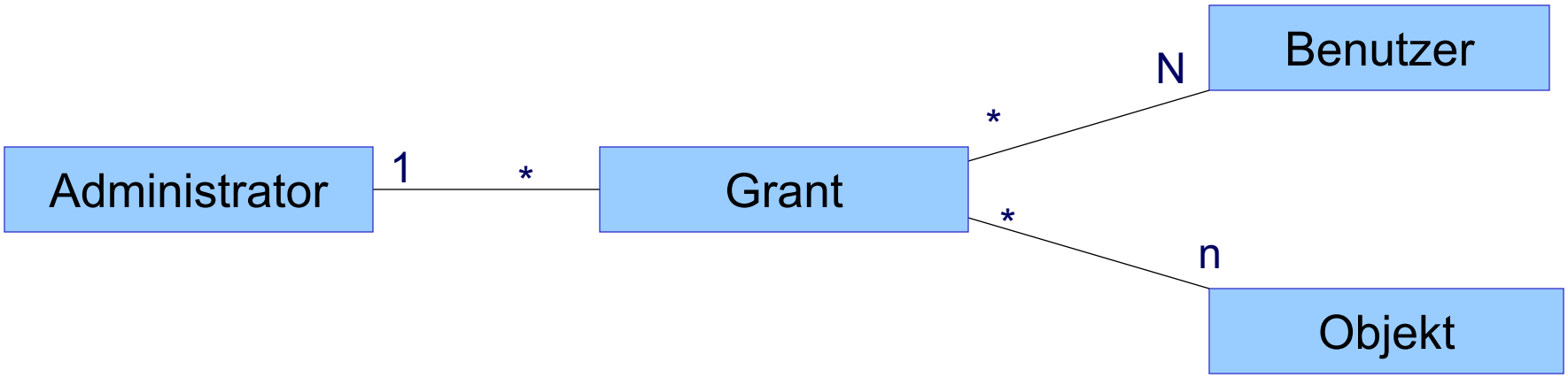
Grant: Admin x Benutzer -> Objekte (Kardinalität N)

Grant: Benutzer x Objekt -> Admin (Kardinalität 1)

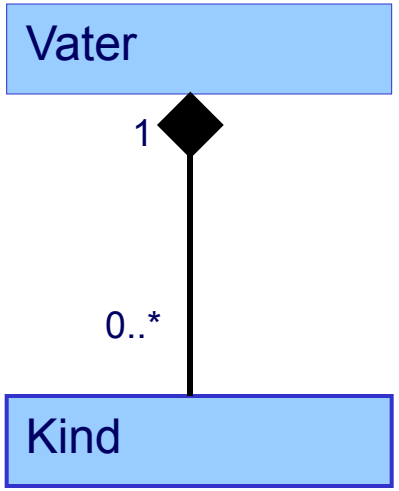
Grant: Admin x Objekt -> Benutzer (Kardinalität n)

3-wertige-Assoziation: Grant

Alternative Modellierung mit einer Beziehungs-Klasse



Abhängige Objekte



Ein Kind wird identifiziert durch (PersNr des Vaters, VorName)

Ein Kind ist existentiell abhängig von seinem Vater

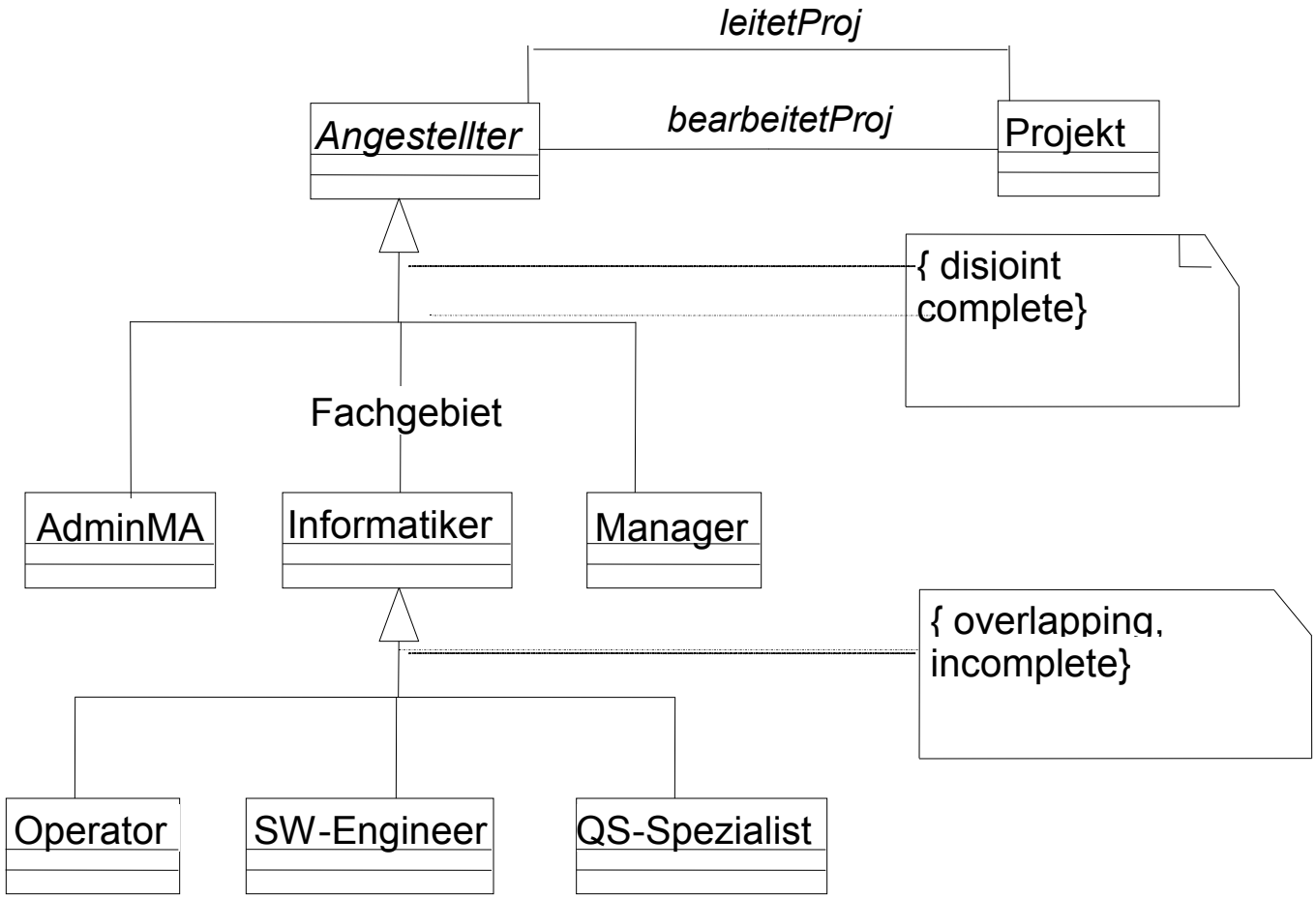
Wenn der Vater gelöscht wird, muss auch alle Kinder dieses Vaters gelöscht werden

Generalisierung und Spezialisierung I

AngProj Ausbau

- Die Angestellten einer Abteilung werden weiter unterteilt in administrative Mitarbeiter, Manager und Informatiker (keine Überschneidungen).
  - Die Informatiker erhalten für die zusätzlichen interessierenden Merkmale die Attribute 'Fachausbildung' und 'Praxiserfahrung'.
  - Die Informatiker werden weiter unterteilt in Operators, SW-Ingenieure und QS-Spezialisten.
  - Bei den SW-Ingenieuren möchte man zusätzlich wissen, welche Programmiersprachen und SE-Methoden der Ingenieur anwenden kann.
  - Bei dieser Kategorisierung ist es möglich, dass ein Informatiker gleichzeitig in verschieden Funktionen (z.B. im Projekt A als SW-Ingenieur und im Projekt B als QS-Spezialist) tätig ist.
- Erweitern Sie das konzeptionelle Modell so, dass diese Zusatzinformationen verwaltet werden können.

Generalisierung in UML



Generalisierung – Spezialisierung: Begriffe I

- Falls mehrere Klassen gemeinsame Eigenschaften (Attribute, Operationen, Assoziationen) aufweisen, dann werden diese in einer Oberklasse (Generalisierung) zusammengefasst.
- Die Unterklassen (genannt Spezialisierungen) erben alle Eigenschaften ihrer (direkten und indirekten) Oberklassen
- Wenn ein Objekt eine Instanz einer Unterklasse Si ist, dann ist es auch Instanz der Ober-klasse G.
- Ein Unterklasse Si von G hat alle (vererbten) Eigenschaften von G, sowie alle speziellen, durch die Unterklasse gegebenen Merkmale (Attribute, Operationen, Assoziationen).
- "is-a" Beziehung zwischen Spezialisierung und Generalisierung.

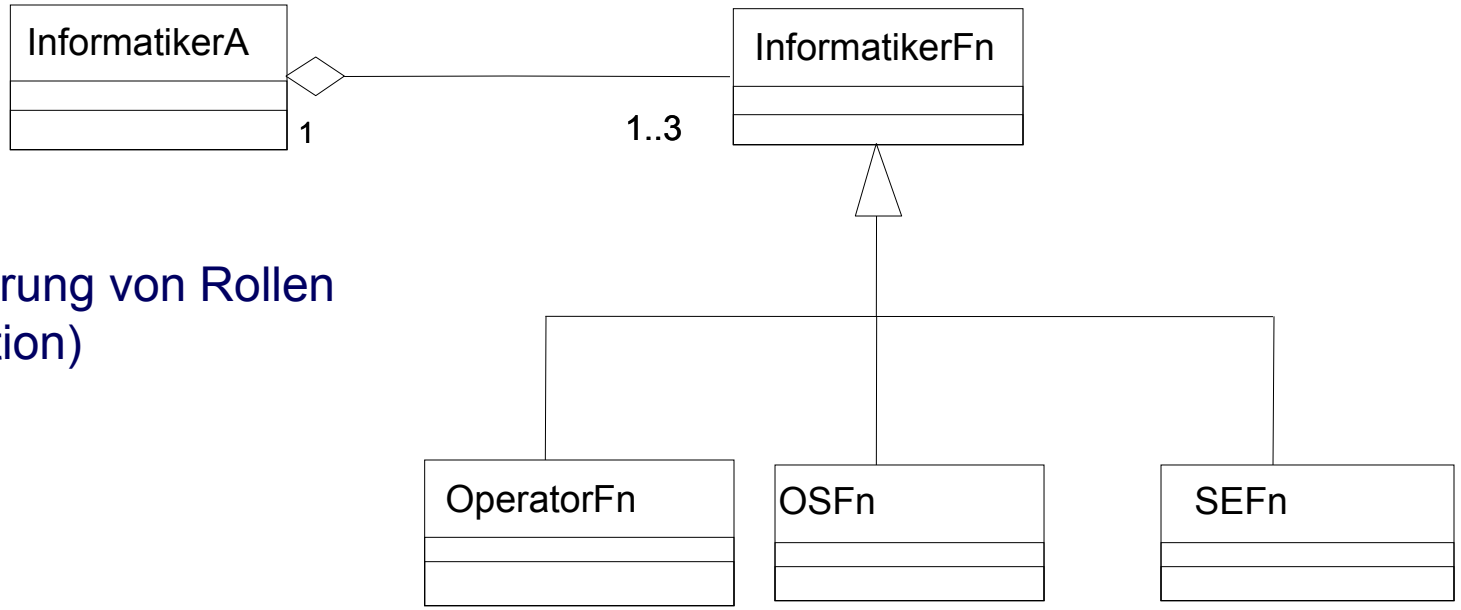
Generalisierung – Spezialisierung: Begriffe II

- Disjunkte und überlappende Generalisierungen {disjoint} bzw. {overlapping}
  - disjunkt: (Subtype Generalization)
    - Objekt ist Instanz von genau einer Unterklasse
    - Diskriminator: Attribut mit einem Aufzählungstyp; legt fest, welche Eigenschaft einer Klasse durch die Generalisierung abstrahiert wird (vgl. Tagfield bei Case-Records).
  - überlappend: (Dynamic Generalization)
    - Objekt kann Instanz von mehreren überlappenden Unterklassen sein, es kann im Laufe seines Lebens die Unterklassenzugehörigkeit wechseln.
- Vollständig- / Nichtvollständige Generalisierungen: {complete} bzw. {incomplete}
  - vollständig: alle Subklassen sind definiert.
  - nicht- vollständig: zusätzliche Subklassen sind erlaubt
- Abstrakte Basisklasse (Abstract Generalization)
  - Die Basisklasse legt einen Kontrakt für die Spezialisierungen fest, keine Instanzen der Basisklasse erlaubt

Delegation statt Vererbung

- Semantik der überlappenden Generalisierung ist sehr komplex
  - z.B. Wechseln der Unterklassenzugehörigkeit
  - Wird von OO-Sprachen nicht unterstützt
  - Überlappende Generalisierungen sollten nicht verwendet oder spätestens in der Design-Phase durch die Modellierung von Rollen basierend auf dem Prinzip der Delegation ersetzt werden.
- Bsp. AngProj
  - Für jede Funktion (oder Rolle) wird ein Objekt der Klasse , InformatikerFn‘ (‘Operator-Funkt‘, ‘QS-Funkt‘, ‘SW-Eng-Funkt’) erzeugt. Ein zusätzliche Einschränkung wäre, dass die Funktions-Objekte eines bestimmten Informatikers in verschiedenen Klassen sein müssen.

Delegation statt Vererbung II



Modellierung von Rollen (Delegation)

Fazit:

- klassisches Konzept der Generalisierung nur für die Modellierung von statischen Typenhierachien verwenden!
- Rollen für dynamische Klassifikation

Grundlagen des relationalen Modells

Relationales Modell

- Erstmals beschrieben durch E.T. Codd um 1970
- Basiert auf dem mathematischen Konzept der Relationen
- Benutzt Tabellen zur Darstellung von Daten und Beziehungen
- Ermöglicht saubere Trennung zwischen logischen und internen Sichten
- Flexible Abfragen der gespeicherten Daten möglich
- Nachteil:

  „Impedance Mismatch“ zwischen DB-Typen (Tabellen) und den Typen und Strukturen der Programmiersprachen.
  Komplizierte Abfragen über mehrere Tabellen können zu Problemen mit Zugriffszeiten führen.

Relationales Modell: Begriffe

- Entität
  - Individuelles Element (Objekt) des betrachteten Systems
- Relation
  - beschreibt eine Entitätsmenge
  - Bsp: Ang (PersNr, Name, Chef, Salaer)
- Tupel
  - repräsentiert eine Entität
  - Bsp: (1001, “Marxer“, 1000, NULL, 10580)
- Darstellung im relationalen Modell durch Tabellen

PersNr	Name	Chef	Salär
...			
1001	Marxer	1000	10580

⇐ Zeile == Tupel

⇑ Spalte == Attribut
Feld == Attributwert

Relationales Modell im Überblick

Tabelle:
- definiert Struktur der Tuples
- speichert Entitätsmenge

Fremdschlüssel-Beziehung:
Angestellter.AbtNr: REFERENCES (Abteilung.AbtNr)

Tabelle Angestellter

PersNr	**Name**	***AbtNr***	***Chef***	**Salaer**
1001	Marxer	1		10580
1002	Widmer	2		12010
1019	Affolter	1	1001	4123
1123	Meier	1	1001	9765
2098	Zuercher	1	1001	10019
2109	Heiniger	2	1002	4098
2345	Becker	2	1002	6346
3333	Wernli	1	1001	8978
4000	Rey	4		15000
010	Danuser	4	4000	5100

Attributname:
PersNr: Primärschlüssel
AbtNr: Fremdschlüssel

Zeile: Tupel

Spalte: Attributwerte
Spaltenname: Attributname

referenziertes Tupel

AbtNr	**Name**
1	Verkauf
2	Marketing
4	Finanzen
5	QS

Tabelle Abteilung

Regeln für Tabellen

- Eine Tabelle beschreibt alle Entitäten einer Entitätsmenge.
- Jede Zeile beschreibt eine Entität und entspricht einem Tupel
- Jede Kolonne entspricht einem (unstrukturierten) Attribut.
- Jede Kolonne hat eine eindeutige Überschrift, sie entspricht dem Namen des Attributes, das durch diese Kolonne repräsentiert wird.
- Die Werte in einer Kolonne müssen im Wertebereich des dazugehörigen Attributes liegen. Die Attributwerte dürfen nicht strukturiert sein (1. Normalform).
- Die Reihenfolge der Kolonnen und Zeilen hat keine Bedeutung.
- Jede Zeile in einer Tabelle ist verschieden von den anderen Zeilen.
- Folgerung: Zu jeder Tabelle ist ein Primärschlüssel definiert.

Schlüssel I

- Schlüssel
  - Attribute oder eine Kombination von Attributen, die ein Tupel eindeutig identifizieren, heissen Schlüssel
  - Diese Kombination muss minimal sein, d.h. wenn ein Attribut weggelassen wird, dann darf der Schlüssel nicht mehr eindeutig sein.
- Primärschlüssel
  - Der aus den möglichen Schlüsseln (= Schlüsselkandidaten) ausgewählte identifizierende Schlüssel
  - Wird für Fremdschlüsselbeziehungen verwendet
- Schlüsselkandidat
  - Attribut oder Kombination davon, das/die ein Tupel identifiziert(en)

Schlüssel II

- Eigenschaften eines Primärschlüssels
  - eindeutig (innerhalb Tabelle)
  - laufend zuteilbar
  - möglichst kurz und nicht zusammengesetzt (z.B. Datentyp NUMBER(10))
  - invariant
- Surrogatschlüssel
  - künstlicher Schlüssel, wird häufig eingeführt, wenn keiner der Schlüsselkandidaten obige Eigenschaften erfüllt.
  - Keine Semantik, kein ‚sprechender' Schlüssel
- Beispiele aus dem Alltag
  - AHV-Nummer
  - Personalnummer
  - (Sind dies 'gute' Primärschlüssel?)

Entitäts-Integrität

- NULL
  - Undefinierter Attribut-Wert in einer Tabelle, ausserhalb des Wertebereichs des Attributes
  - Ob ein Attribut NULL sein darf, kann bei der Definition des Attributes festgelegt werden.
  - Vorsicht: die Zahl 0, der Nullstring, der Null-Pointer etc. sind alles definierte Werte!
  - Anschaulich: Das Feld der Tabelle ist leer, es enthält keinen Eintrag.

- Entitäts-Integrität
  - Da der Primärschlüssel minimal ist und ein Tupel eindeutig identifiziert, darf kein Attribut des Primärschlüssels NULL sein.

Referentielle Integrität

Angestellter

PersNr	Name	AbtNr
1001	Marxer	1
1002	Widmer	2
1010	Steiner	3
1019	Affolter	1
1100	Widmer	3
1123	Meier	1

Abteilung

AbtNr	Name
1	Verkauf
2	Marketing
3	Entwicklung
4	Finanzen
5	QS

Bedingung für referentielle Integrität:

Wertebereich von `Angestellter.AbtNr` = Menge der Attributwerte von `Abteilung.AbtNr` (dynamisch!)

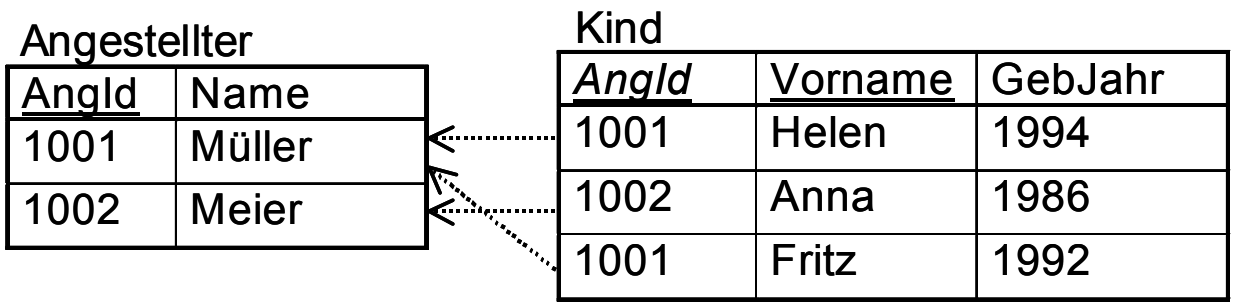
Kritische Operationen:

- Angestellter: `INSERT, UPDATE AbtNr`
- Abteilung: `DELETE, UPDATE AbtNr`

Referentielle Integrität

- Fremdschlüssel
  - Falls in einer Tabelle Attribute vorkommen, welche gleichzeitig Primärschlüssel einer anderen Tabelle sind, dann spricht man von einem Fremdschlüssel.
  - Fremdschlüssel werden für die Modellierung von Beziehungen verwendet.
- Referentielle Integrität
  - Der Wertebereich eines Fremdschlüsselattributes ist dynamisch: er besteht aus der aktuellen Menge der Primärschlüsselwerte der referenzierten Tabelle; d.h. Fremdschlüssel dürfen nur dann ungleich NULL sein, wenn das durch den Fremdschlüssel identifizierte Tupel existiert.
  - Diese referentielle Integritätsbedingung kann bei Einfüge-, Lösch- und Änderungsoperationen verletzt werden und sollte daher vom DBMS überprüft werden.

Abhängige Tabellen



Abhängige Tabelle:

Der Primärschlüssel einer abhängigen Tabelle ist entweder ein Fremdschlüssel oder er enthält Attribute eines Fremdschlüssels einer übergeordneten Tabelle.

Folgerung: Ein Tupel in der abhängigen Tabelle ist existentiell vom referenzierten Tupel in der übergeordneten Tabelle abhängig.

1. Wie wird ein Kind identifiziert?
2. Was passiert mit einem Kind, wenn 'sein' Angestellter gelöscht wird?
3. Wie könnte das Modell geändert werden, dass ein Kind ohne 'seinen' Angestellten existieren kann.

Zusammenfassung Tabelle

- Eine Tabelle besitzt einen eindeutigen Tabellennamen
- Innerhalb der T. ist jeder Merkmalsname eindeutig und bezeichnet eine bestimmte Spalte mit der gewünschten Eigenschaft
- Die Anzahl der Merkmale ist beliebig, die Ordnung der Spalten innerhalb der Tabelle ist bedeutungslos
- Eines der Merkmale oder eine Merkmalskombination identifiziert eindeutig die Tupel innerhalb der T. und wird als Primärschlüssel bezeichnet
- Die Anzahl der Tupel einer T. ist beliebig, die Ordnung der Tupel innerhalb der Tabelle ist bedeutungslos

Begriffe

Terminologischer "Konflikt": Tabelle oder Relation?

- Praxis : Betonung auf opt. Darstellung in Tabellenform
- Wissenschaft: Betonung der mathematischen Grundlage als Relation

Unterschiede zwischen beiden Bezeichnungen:

Tabelle:
2-dim. Array aus Zeilen/Spalten
geordnet
doppelte Zeilen möglich

Relation:
Menge von Tupeln
ungeordnet
duplikatfrei

aber:

- Relationen lassen sich in Tabellenform visualisieren, wenn man eine bestimmte, Anordnung wählt.
- Tabellen lassen sich als Relationen formalisieren, wenn man keine Duplikate, zulässt und von der Anordnung absieht.

UML-Klassen-diagramm	**Entity Relationship Model**	**Relationales Modell**	**SQL (Postgres)**	**Access**
Objekt, Instanz	Entität	Tupel	Zeile	Datensatz
Objektmenge, Instanzmenge, Klasse	Entitätsmenge	Relation	Tabelle	Datenblatt
Attribut	Attribut (Property)	Attribut	Feld / Spalte / Attribut / Column	Feld-bezeichner
Datentyp	Wertebereich	Wertebereich	Datentyp	Felddatentyp