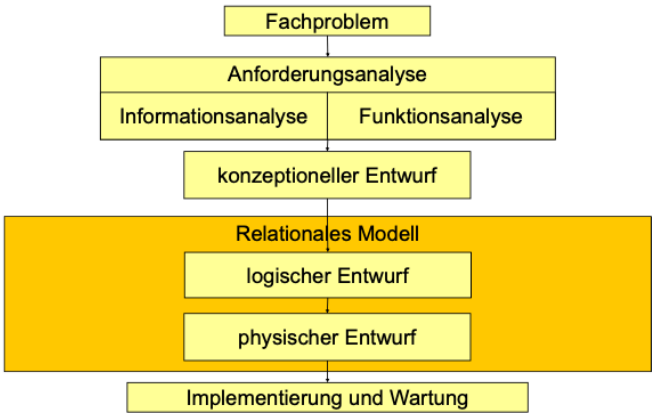
VL02_Konzeptionelles_Modell

1.0- Schema-Entwurf

Phasen des Datenbankentwurfs

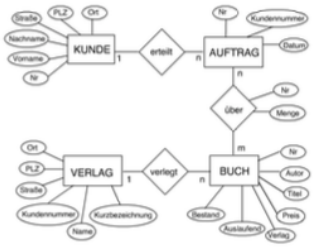


1. Das **Fachproblem** liegt normalerweise vor.
2. **Anforderungsanalyse**: Welche Informationen werden in der Datenbank gespeichert, welche Operationen werden auf den Daten ausgeführt werden, usw.
3. **Konzeptioneller Entwurf**: Beschreibe das Schema der Daten unabhängig von der späteren Implementierung
4. **Logischer Entwurf**: Übersetzen des konzeptionellen Schemas in ein Implementierungsmodell, z.B. das relationale Modell. Verbesserung des Modells durch z.B. Normalisierung.
5. **Physischer Entwurf:** Schema-Entwicklung für ein spezielles DBMS, Deklaration der Daten, Festlegung der (Speicher-)Zugriffstrukturen
6. **Implementierung und Wartung:** Installation des Datenbanksystems, Anpassung an neue Anforderungen.

Übersicht Entwurfsmodelle

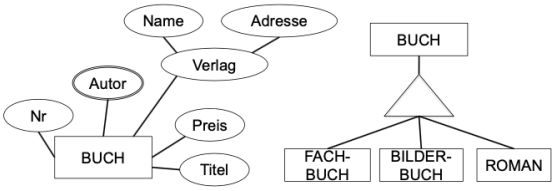
Entity Relationship Modell (ER Modell)

• Basiert auf den Grundkonzepten **Entity** (Informationseinheit), **Attribut** (Eigenschaft eins Entitys) und **Relation** (Beziehung zwischen Entities)



Erweitertes Entity Relationship Modell (EER Modell)

• Mehr Attributtypen (zusammengesetzt, mehrwertig, ...)
• Generalisierung (engl. is-a), Spezialisierung und Aggregation (engl. part-of)



Unified Modelling Language (UML)

• Allgemeine Sprache, nicht nur zur Modellierung im Datenbankenbereich

Übersicht Implementierungsmodelle

~~*Hierarchisches Modell (1968)*~~

• Schema ist eine Menge von Bäumen
• Beispiel: IBM IMS (Information Management System) – heute noch große Datenbestände in hierarchischen DBS

~~*Netzwerkmodell (1969)*~~

• CODASYL 1971, eingeschränktes Relationenmodell (keine n:m Beziehungen), darstellbar als Graph
• Beispiel: Siemens UDS

Relationenmodell (RDBMS) (1980)

• Basiert auf dem mathematischen Modell der Relation
• Beispiele: DB2, Oracle, MySQL, PostgreSQL, ...

***Objektorientiertes** (OODBMS) (1993) und **Objekt-Relationales Modell** (ORDBMS) (2003)*

• Mehr Konzepte (Mengen, Tupel, Listen, Vererbung)
• Beispiel: db4o, Objectivity, Versant, (Objektorientiert) DB2, Oracle, PostgreSQL (Objekt-Relational)

• Wozu benötige ich ein konzeptionelles Datenmodell?
• D.h.: Warum übersetze ich nicht die fachlichen Anforderungen direkt in ein Implementierungsmodell (also z.B. das relationale Modell)?

Miniwelten

- Modellierung der Realität

Relevanter Ausschnitt aus der realen Welt (Realität)

Zentraler Punkt an dieser Stelle:

- Die Realität kann **niemals vollständig** abgebildet werden.
- Mit dem Kunden klären, welcher Ausschnitt benötigt wird.
- Welche Vereinfachungen gegenüber der Realität können vorgenommen werden?

Methoden zur Informationsgewinnung:

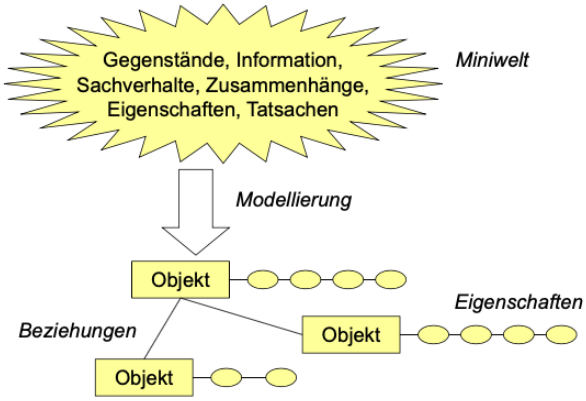
- Interviews
- Analyse bestehender Arbeitsabläufe (Vorgänge)
- Analyse genutzter Formulare oder Standarddokumente

Ergebnis: **nicht-formale** Beschreibungen des Fachproblems

- Texte, Tabellen, Formblätter

➔ Man erkennt Gegenstände/Objekte der realen Welt, die

- bestimmte **Eigenschaften** (Attribute) haben
- in bestimmten **Ve rbindungen** zueinanderstehen



Konzeptionelles Modell

Überblick

Eigenschaften des konzeptionellen Modells

- strukturiert den zukünftigen Anwendungsbereich
- beschreibt die Gesamtheit derjenigen Daten, die in der Datenbank verwaltet werden
- beschreibt die Integritätsbedingungen, also Bedingungen oder Vorschriften, die
  - für die Daten immer gelten oder für die Änderungen von Daten gelten

Modellierungskonzepte:

- Entity – Typen, Beziehungstypen, Attribute, Wertebereiche

Modellierungsebene:

- Das ER Modell modelliert auf der Typ-Ebene, nicht auf Instanz-Ebene
- Aussagen werden über Mengen getroffen, nicht über einzelne Elemente
  einer Menge (Menge der Studierenden vs. Studentin Winnie Wendig)
- Integritätsbedingungen / Constraints ergänzen das Modell

Beispiel Hochschule
Entity-Typen (Auswahl)
• Lehrveranstaltungen.
Attribute: Bezeichnung, Semester,
• Räume. Attribute: Raumnr., Bezeichnung, …
Beziehungstypen zwischen den Objekttypen (Auswahl)
• bietet_an zwischen Prof und Lehrveranstaltungen
• belegt zwischen Stud. und Lehrveranstaltungen
Integritätsbedingungen
• Matrikelnr 7 – stellig,
• Geburtsdatum nach 1900

Entities und Entity-Typen, Attribute

Entity (Entität): -Abstrakte Objekte wie z.B. Tische

- Basisobjekt, „etwas“ aus der realen Welt, physisch oder konzeptionell existierendes Objekt
- Entities besitzen Attribute, d.h. bestimmte Eigenschaften, die sie beschreiben und deren konkrete Ausprägung als Werte bezeichnet werden.

Beispiele: Entity mit dem Attribut Nummer mit dem Wert 1.H.1.018

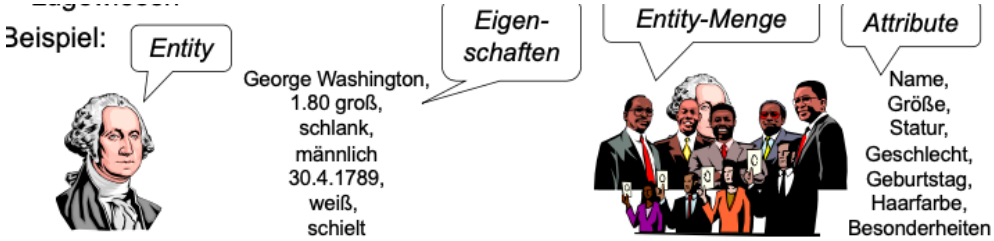
Entity-Typ:

- definiert eine Menge von gleichartigen Entities mit gleichen Attributen, also gemeinsamen Eigenschaften (diese Menge wird auch als Entitymenge bezeichnet)

Beispiel: Entity-Typ **Raum** mit den Attributen **Nummer**, **Plaetze**,...

Attribute

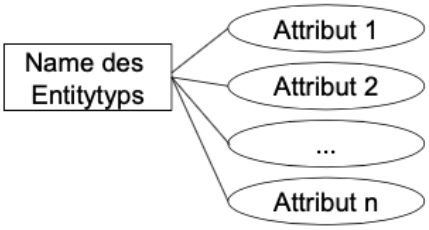
- Eigenschaft, die alle Elemente desselben Entitytyps besitzen (gemeinsame Eigenschaften).
- Die zulässigen Werte eines Attributes nennt man Wertebereich oder Domäne (engl. Domain).
- jedem Entity eines Entitytyps wird pro Attribut ein Wert aus einem Wertebereich zugewiesen



Beschreibung von Entity-Typen

Textuelle Beschreibung

Buch (Titel String,
{Autor} {n1,n2 },
Verlag (Name, Ort char(8)))



Attribut-Typen

Einwertige Attribute (klassisches ER-Modell)

- Nehmen genau einen Wert aus dem Wertebereich an

Mehrwertige Attribute (erweitertes ER-Modell)

- Können für ein konkretes Entity einen oder mehrere Werte aus dem Wertebereich annehmen.
  Beispiel: Ein Buch kann mehrere AutorInnen haben

Zusammengesetzte Attribute (erweitertes ER-Modell)

- Nehmen in einem konkreten Entity für jede ihrer Komponenten einen Wert an.
  Beispiel: Das Attribut Verlag besteht aus zwei Komponenten: Name und Ort. „Addison-Wesley, Bonn“

Ganze Zahlen	**Gleitkommazahlen**	**Zeichenketten**	**Datum**
int(Stellenzahl)	number(p,s)	string char(Zeichenzahl)	date

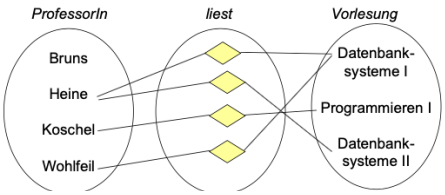
Schlüssel

- **Schlüssel** sind **Teilmengen** von Attributen, die ein Objekt **eindeutig identifizieren**

Mögliche Schlüssel - Zeitinvarianz beachten

- **Schlüssel müssen eindeutig und unveränderlich sein!**
- Schlüssel müssen **minimale** Mengen von Attributen sein!
  Keine Teilmenge des Schlüssels identifiziert die Entities eindeutig
- Es kann mehrere Schlüssel für einen Entitytypen geben.
  (**Schlüsselkandidaten**). Beispiel: {Autor, Titel, Verlag, Jahr }
- Wähle einen Schlüssel aus und nenne ihn **Primärschlüssel**.
  - durch Unterstreichen gekennzeichnet:

Beziehungen



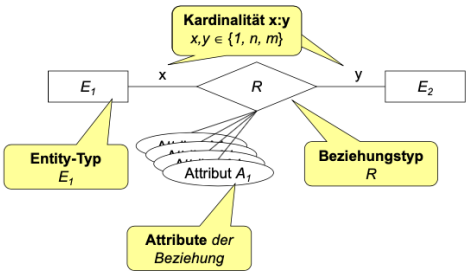
Beziehungen beschreiben **Zusammenhänge** zwischen Entities.
Jede Teilmenge des **kartesischen Produktes** ist eine mögliche Beziehungsmenge.

- Bsp. ProfessorInnen **lesen** Vorlesungen

Eigenschaften von Beziehungstypen
Stelligkeit oder **Grad** gibt an, wie viele Entity-Typen mit einem Beziehungstyp verbunden sind

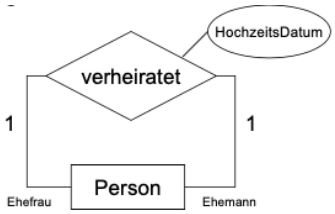
Kardinalität bzw. Funktionalität beschreibt, wie viele Instanzen eines Entity–Typs jeweils in eine Beziehung eingehen

- **1:1 Beziehung**
  - Jedem Entity des Typen E1 ist höchstens ein Entity des Typen E2 zugeordnet und umgekehrt
- **1:n Beziehung (n:1 analog)**
  - Jedem Entity des Typen E1 können beliebig viele Entities vom Typ E2 zugeordnet sein. Einem Entity des Typen E2 kann immer nur höchstens ein Entity des Typen E1 zugeordnet sein.
- **n:m Beziehung**
  - Jedes Entity vom Typ E1 kann mit beliebig vielen Entities vom Typ E2 in Beziehung stehen und umgekehrt.



Rekursive Beziehungstypen:

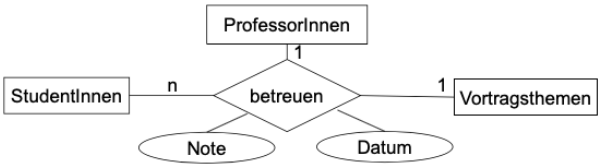
- Ein Entity-Typ kann auch mehrfach an dem gleichen Beziehungstyp teilnehmen.
- Beziehungen können auch innerhalb einer Entity-Menge vorhanden sein.



im Beispiel: istVerheiratetMit (Ehefrau:Person, Ehemann:Person)

Beispiel einer mehrstelligen Beziehung mit Attributen

Kardinalität n:1:1
Student sollen bei einer ProfessorIn nur **ein** Vortragsthema bearbeiten
StudentInnen dürfen ein Vortragsthema **nur einmal** bearbeiten
Vortragsthemen dürfen von Profs **mehrfach vergeben** werden

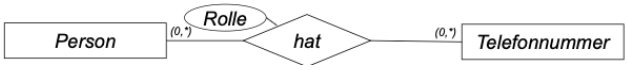


erzwungene Konsistenzbedingungen
StudentInnen dürfen bei demselben Professor nur ein Vortragsthema bearbeiten.
dennoch mögliche Datenbankzustände
ProfessorInnen können dasselbe Vortragsthema „wiederverwenden“.

Die (min, max) Notation bei Beziehungen

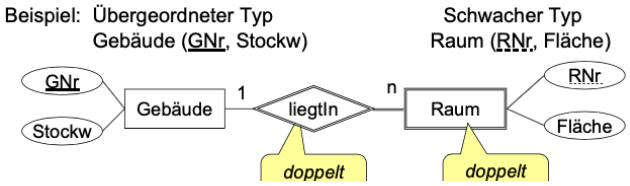
- Für jedes Entity in einer Beziehung werden in der
- (min, max) Notation **Ober- und Untergrenzen festgelegt**.
- Wenn es Entities in E2 geben darf, die gar nicht an der Beziehung teilnehmen, so wird min1 mit 0 angegeben
- Wenn ein Entity beliebig oft an einer Beziehung teilnehmen darf, so wird die max – Angabe durch * ersetzt

Bsp.
Eine Person hat viele Telefonnummern, eine Telefonnumer ist beliebig vielen, ggf. auch keiner Person zugeordnet



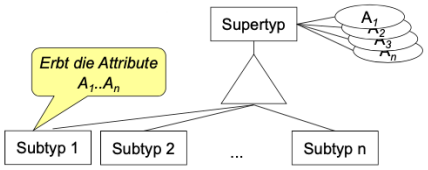
Erweiterte ER-Konzepte

Schwache Entities (existenzabhängige Entities)



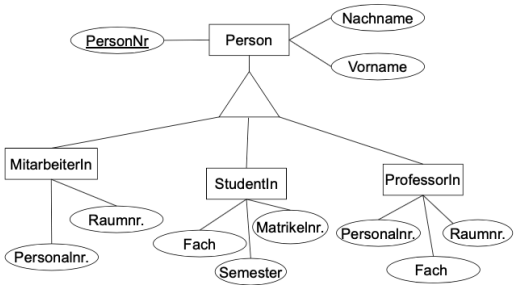
- sind von der **Existenz eines übergeordneten** Entities **abhängig**
- nur in Kombination mit dem **Schlüssel** des übergeordneten Entities eindeutig identifizierbar.
- **Doppelt umrandet und Schlüssel gestrichelt!**

Generalisierung



- **Abstraktion** auf Typ-Ebene
- Gemeinsame Attribute werden "**herausfaktorisiert**" und dem Supertypen (Obertypen) **zugeordnet**
- **Subtypen** (Untertypen) erben die **Attribute** ihrer Supertypen

Spezialisierung



Generalisierung geht

- von unten nach oben (bottom up)
- Vom Speziellen zum Allgemeinen

Spezialisierung ist im Prinzip dasselbe, die Vorgehensweise ändert sich:

- Von oben nach unten (top down)
- Vom Allgemeinen zum Speziellen

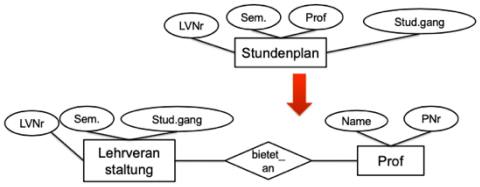
Besondere Eigenschaften von Spezialisierungen

Überlappende vs. disjunkte Spezialisierungen

Totale vs. partielle Spezialisierungen

Vorgehen beim Modellentwurf Zusammenfassung

Top-down – Strategie

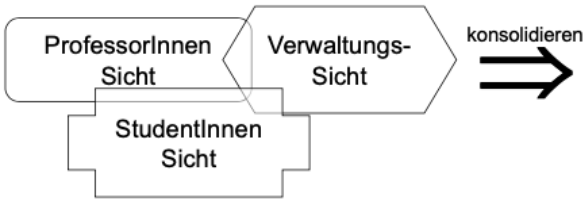


- beginnen mit einem Schema, das hohe Abstraktionen enthält
- Sukzessive Verfeinerungen

Bottom-up-Strategie

- beginnen mit einem Schema, das grundlegende Abstraktionen enthält
- kombinieren der Abstraktionen
- mit den Attributen beginnen und sie in Entitytypen und Beziehungstypen gruppieren; anschließend neue Beziehungstypen zwischen Entitytypen hinzufügen (Generalisierung in allgemeinere Superklassen)

Schema- bzw. View-Integration



- die einzelnen Anwendersichten modellieren
- diese Sichten zu einem gemeinsamen Modell zusammenführen

Benennungskonflikte

- **Synonyme** - Mehrere Bezeichnungen für das gleiche Konzept: Chef / Vorgesetzer
- **Homonyme** - Gleiche Bezeichnung für unterschiedliche Konzepte Blatt (Papier, am Baum), -> Projektglossar anlegen!

Typkonflikte

- das gleiche Konzept in zwei Schemas unterschiedlich modelliert (Abteilung in einem Schema als ET und in einem anderen Schema als Attribut)

Konflikte zwischen Wertemengen

- verschieden Wertemengen eines Attributs

Konflikte zwischen Einschränkungen

- verschiedene Schlüssel, verschiedene Kardinalitäten

Beziehungen können auf verschiedene Weisen modelliert werden:

- als Beziehungstyp im ER-Diagramm
- als Entitätstyp, der nur in Beziehung zu anderen Entities existieren kann

Beispiele

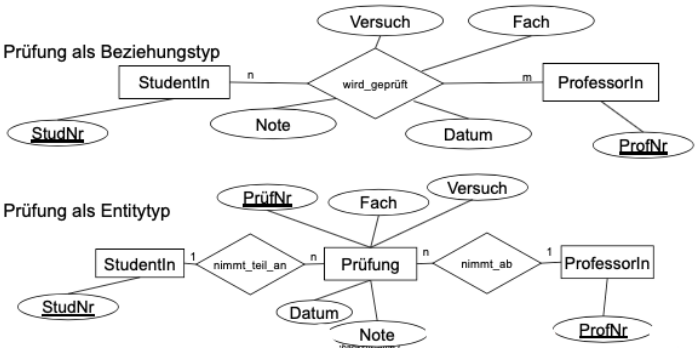
- Prüfung als Beziehung zwischen StudentIn und ProfessorIn
- Bestellung als Beziehung zwischen Teil, Projekt und Lieferant

Es gibt an der Stelle kein generelles besser oder schlechter

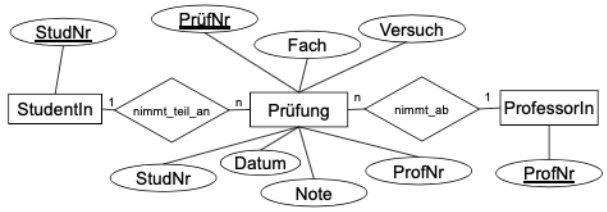
Entwurfsprogramm erfordert manchmal die eine oder andere Modellierungsvariante

- abhängig davon, ob eine Beziehung auch Attribute hat
- abhängig davon, wie viele Entity-Typen an der Beziehung beteiligt sind

Prüfung als Beziehungstyp vs Entityp



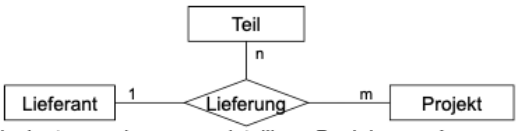
Redundanzen im Modell



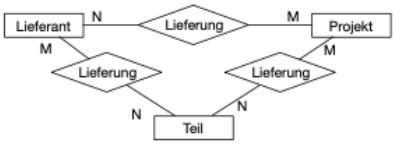
ProfNr und StudNr in Prüfung überflüssig!

Mehrstellige Beziehungen

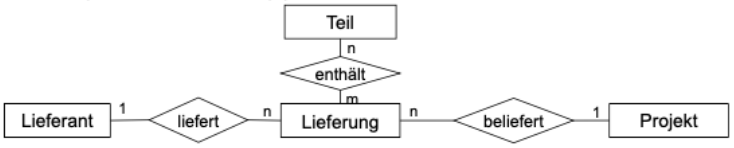
Als Beziehungstyp einfach zu modellieren



Aber nicht äquivalent zu mehreren zweistelligen Beziehungen!



Beziehung kann aber als Entitytyp modelliert werden



Miniwelt

- relevanter Ausschnitt der Realität besteht aus "Objekten" die bestimmten Eigenschaften haben.

Konzeptionelle Modelle

- Strukturieren die Miniwelt und beschreiben die relevanten Daten unabhängig von Implementierung oder einzelnen Anwendunge
- Integritätsbedingungen verbessern Übereinstimmung zwischen Realität und Modell

Entity: „etwas“ aus der realen Welt, physisch oder konzeptionell existierendes Objekt

Entity-Typ:

- definiert eine Menge von gleichartigen Entities mit gleichen Attributen, also gemeinsamen Eigenschaften (diese Menge wird auch als Entitymenge bezeichnet)
- Für einen Entity-Typ werden **Schlüssel** definiert

Beziehungtyp

- Ein Beziehungstyp deklariert eine Beziehung zwischen Entity-Typen.
- Es kann eine beliebige Anzahl von Entity-Typen an einem Beziehungstyp teilhaben (Stelligkeit)
- 1:1, 1:n, n:m Notation modelliert die Anzahl der beteiligten Entities an einer Beziehung (Kardinalität)
- (min, max) Notation modelliert, wie oft ein Entity in der Beziehung vorkommen kann.

Erweiterte Konzepte

- Mengenwertige und zusammengesetzte Attribute
- Schwache Entity-Typen
- Klassenhierarchien Vorgehensweise beim Modellentwurf