3 Das Entity-Relationship-Modell

Inhalt

3.1 Grundlagen des Entity-Relationship-Modells

3.1.1 Eckdaten

- Entity-Relationship-Modell abgekürzt: ER-Modell oder ERM
- 1976 von Peter Chen in seiner Veröffentlichung *The Entity-Relationship Model* vorgestellt:

  Peter Pin-Shan Chen: The Entity-Relationship Model - Toward a Unified View of Data. In: ACM Transactions on Database Systems, 1/1/1976, ACM-Press, ISSN 0362-5915, S. 9-36
  Download über Uni Wien Rechner/VPN: http://doi.acm.org/10.1145/320434.320440

- Beschreibungsmittel für Generalisierung und Aggregation wurden 1977 von Smith and Smith eingeführt.

  J.M. Smith, D.C.P. Smith: Database Abstractions: Aggregation and Generalization, ACM Transactions on Database Systems, Vol. 2, No. 2 (1977), S. 105-133
  Download über Uni Wien Rechner/VPN: http://doi.acm.org/10.1145/320544.320546

J. M. Smith and D. C. P. Smith: Database Abstraction: Aggregation, Communications of the ACM, Vol. 20, Nr. 6, pp. 405-413, June 1977
Download über Uni Wien Rechner/VPN: http://doi.acm.org/10.1145/359605.359620

- Einige Weiterentwicklungen so z. B. Ende der 80er Jahre durch Wong und Katz.
- 1985 wurde IDEF1X als Standard zur ER-basierten Modellierung von der US Air Force veröffentlicht. 1994 wurde IDEF1X als U.S. Behördenstandard übernommen.
- Systemunabhängiges Datenmodell
- Dient in der konzeptionellen Phase der Anwendungsentwicklung der Verständigung zwischen Anwendern und Entwicklern, ermöglicht ein abstraktes Modell eines statischen Ausschnitts der realen Welt zu schaffen (WAS, aber nicht WIE)
- Dient in der Implementierungsphase als Grundlage für das Design von Datenbanken; weitgehend akzeptiertes Mittel zum konzeptionellen Datenbankentwurf
- Elemente des ER Modells finden sich in anderen Sprachen, auch in UML.

3.1.2 Nutzung des Entity-Relationship Modells

- ER-Modell wird vorzüglich bei der Erstellung von Datenbanken genutzt; dient oft zur Beschreibung von statischen Informationsmodellen.
- ER-Modell zur Konzeption der Datenbank dient als Grundlage für die Implementierung der Datenbank.
- Umsetzung der in der Realwelt erkannten Objekte und Beziehungen in ein Datenbank-Schema erfolgt dabei in mehreren Schritten:
  - Erkennen und Zusammenfassen von **Objekten** zu Entitätstypen durch Abstraktion (z. B. Die Kollegen Klas und Karagiannis und viele weitere zum Entitätstyp „Universitätsprofessor“).
  - Erkennen und Zusammenfassen von **Beziehungen** zwischen je zwei Objekten zu einem Beziehungstyp (z. B. Professor Klas koordiniert das Modul Grundl. d. Modellierung). Dieses führt zum Beziehungstyp „LV-Leiter koordiniert Modul“.
  - Bestimmung der **Kardinalitäten**, d. h. der Häufigkeit des Auftretens (z. B. wird ein Modul immer von genau einem LV-Leiter koordiniert und ein LV-Leiter darf mehrere Module koordinieren).
  - Bestimmung der relevanten **Attribute** der einzelnen Entitätstypen.

- Markierung bestimmter Attribute eines Entitätstyps als **identifizierende Attribute**, so genannte **Schlüsselattribute**.

- Weitere Schritte (nicht in dieser LV behandelt):

  - Um einen hohen **Gütegrad des Entwurfs** zu gewährleisten, sind Optimierungsschritte nötig: Durchführung des Prozesses der Normalisierung, um die Redundanz innerhalb der zu erstellenden Datenbank zu verringern und um die Datenintegrität zu erhöhen (siehe Datenbanksystemvorlesung).

  - **Generierung des Schemas** einer relationalen Datenbank mit all seinen Tabellen- und zugehörigen Felddefinitionen mit ihren jeweiligen Datentypen (siehe Datenbanksystemvorlesung).

3.1.3 Grundbegriffe

Grundlage der Entity-Relationship-Modelle ist die Typisierung von Objekten und deren Beziehungen untereinander:

- **Entität** (**Entity**): Objekt in der Realität, materiell oder abstrakt (zum Beispiel Student "Müller", Lehrveranstaltung "050039")

- **Entitätstyp**: Typisierung gleichartiger Entitäten (zum Beispiel Studierende, Lehrveranstaltung, Buch, Autor, Verlag)

- **Beziehung** (**Relationship**): semantische Beziehung zwischen zwei Objekten (zum Beispiel "Student Müller nimmt-teil-an Lehrveranstaltung "050039")

- **Beziehungstyp** (**Relationship-Type**): Typisierung gleichartiger Beziehungen (zum Beispiel Teilnahmebeziehung zwischen Studierenden und Lehrveranstaltung)

ACHTUNG: In der Regel wird bei einer Modellierung (in Diskussionen und Beispielen) mit konkreten Objekten gearbeitet (Entitäten und Beziehungen). Das Modell selbst hingegen besteht aber immer ausschließlich aus Entitätstypen und Beziehungtypen.

- **Attribut**: beschreibendes Merkmal einer Entität oder einer Beziehung (zum Beispiel Vorname und Nachname von Studierenden).

- **Kardinalität**: mögliche Anzahl der an einer Beziehung beteiligten Entitäten (zum Beispiel kann ein Student an mehreren Lehrveranstaltungen teilnehmen, während ein Modul von genau einem LV-Leiter koordiniert wird).

3.1.4 Sprachen zur Notation von ER-Modellen

- Die **grafische Darstellung** von Entitätstypen und Beziehungstypen wird **Entity-Relationship-Diagramm** (ERD) oder **ER-Diagramm** genannt.
  - Unterschiedliche Darstellungsformen.

- Vielzahl **unterschiedlicher Notationen**, die sich unter anderem in Klarheit, Umfang der grafischen Sprache, Unterstützung durch Standards und Werkzeuge unterscheiden. Kernaussage der ER-Diagramme aber jeweils nahezu identisch.

- Besondere - zum Teil historischer - Bedeutung:
  - **Chen-Notation** von Peter Chen
  - **IDEF1X** als langjähriger de-facto Standard bei U.S. amerikanischen Behörden.
  - **Bachman-Notation** von Charles Bachman als weit verbreitete Werkzeug-Diagramm-Sprache.
  - **Martin-Notation** (Krähenfuß-Notation) als weit verbreitete Werkzeug-Diagramm-Sprache (Information Engineering).
  - **Min-Max-Notation** als (kurzlebiger) ISO-Standard.
  - **UML** als aktueller Standard

3.1.5 Abstraktionskonzepte (allgemein)

- Abstraktion ist ein mentaler Prozess um einige Charakteristika und Eigenschaften einer Menge von Objekten zu deren Beschreibung auszuwählen und gleichzeitig andere, nicht relevante auszuschließen.

- Für die Datenmodellierung lassen sich 3 wesentliche Konzepte identifizieren:
  - Klassifikation
  - Aggregation
  - Verallgemeinerung bzw. Spezialisierung

- Eine **Klassifikation** dient der Definition bestimmter Konzepte als Klassen von Dingen bzw. Objekten mit gemeinsamen Eigenschaften.

  Eigenschaften werden in diesem Zusammenhang als Attribute bezeichnet

  Mathematisch geht es hierbei um Mengenbildung
  Ein Realwelt-Objekt kann auf viele Weisen klassifiziert werden. Welche Klassifikation gewünscht bzw. angemessen ist, muss im Einzelfall in Abhängigkeit von der Zielsetzung (Anwendung) entschieden werden.

- Eine **Aggregation** definiert eine neue Klasse aus anderen bereits existierenden, welche dann Komponenten repräsentieren, bzw. sie setzt bestehende Klassen zu neuen zusammen.

  Aus mathematischer Sicht handelt es sich um die Bildung kartesischer Produkte

- Eine **Generalisierung** bzw. **Spezialisierung** definiert eine Teilmengenbeziehung zwischen den Elementen verschiedener Klassen.

  Mathematisch geht es um Potenzmengenkonstruktion (bzw. Teilmengenbildung)

  Verallgemeinerungen bzw. Spezialisierungen haben eine Vererbungseigenschaft: Alle Konzepte (Attribute), welche für die allgemeinere Klasse definiert sind, werden an jede Teilmenge vererbt.

  Häufig (aber nicht immer) wird unter Spezialisierung eine disjunkte Zerlegung einer Klasse in Unterklassen verstanden.

3.1.6 Entities und Attribute

- Wohlunterscheidbare Dinge der realen Welt nennen wir im folgenden in Anlehnung an den englischen Sprachgebrauch ***Entities***. z.B. ein Auto.

- Einzelne Entities, welche “ähnlich”, “vergleichbar”, oder “zusammengehörig” sind, fassen wir zu einen ***Entity-Typ*** zusammen. z.B. alle Autos im System.

- Entities besitzen Eigenschaften, deren konkrete Ausprägungen wir als ***Werte*** bezeichnen. z.B. die Farbe meines Autos ist schwarz.

- Die Zusammenfassung aller möglichen bzw. zugelassenen Werte bezeichnet man als ***Wertebereich*** (**Domain**).

- Auf Ebene des Entity-Typs nennen wir die (bei allen Entities dieses Typs auftretenden) Eigenschaften ***Attribute***.

- Ein Attribut ordnet also jedem Entity des Entity-Typs einen Wert aus einem bestimmten Wertebereich (dem des Attributs) zu.

- Der Entity-Typ sowie dessen Attribute sind als zeitinvariant anzusehen. Diese zeitinvarianten Aspekte können grafisch dargestellt werden:
  - *Entity-Typen* werden als *Rechteck* dargestellt
  - *Attribute* werden als mit dem Rechteck verbundene *Kreise/Ellipsen* dargestellt

- Beispiel: Modell zu Bücher und Leser.

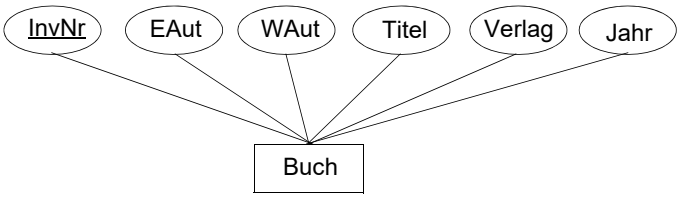


Abb. 3-1: Entity-Typ für Buchinformation

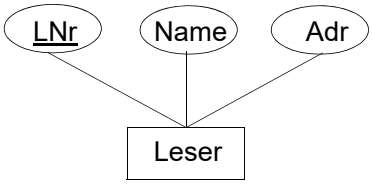


Abb. 3-2: Entity-Typ für Leserinformation

- Obige Beispiele sind unzureichend erfasst, da jedes Attribut als einwertig angenommen werden muss.

  De facto kann jedoch ein Buch mehrere Autoren haben und die Verlagsinformation ist aus einem Verlagsnamen und einem Verlagsort zusammengesetzt. Ebenso besteht ein Name typischerweise aus einem Vor- und einem Nachnamen und Adressen sind aus Straßen- und Ortsangaben zusammengesetzt.

- Weitere Typen von Attributen
  - ***Mehrwertige Attribute*** werden durch einen *Doppelkreis/Doppelellipse* dargestellt
  - Komponenten eines ***zusammengesetzten Attributs*** auch über Ellipsen, die verbunden werden.

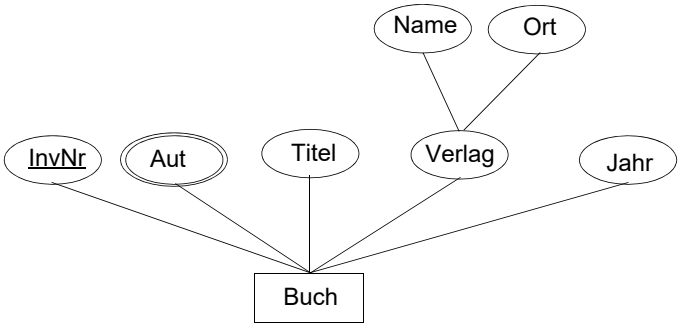


Abb. 3-3: Detaillierter Entity-Typ für Buchinformationen

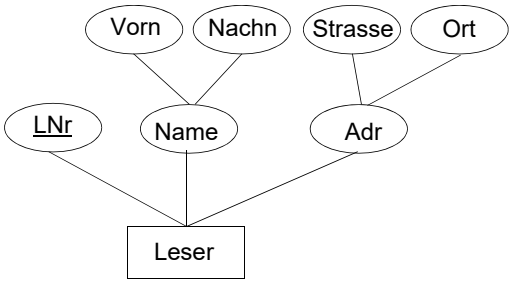


Abb. 3-4: Detaillierter Entity-Typ für Leserinformationen

- Es sei bemerkt, dass Mehrwertigkeit und Zusammensetzung nicht auf eine einmalige Anwendung beschränkt sein brauchen. Die Konzepte können theoretisch beliebig tief geschachtelt werden.

- Zur Beschreibung eines speziellen Entities ist häufig nicht die Angabe aller Attributwerte erforderlich. Es genügt oft für gewisse der einwertigen Attribute einen Wert anzugeben, durch welche das Entity eindeutig identifiziert wird.

- Attribute, die ein Entity eindeutig identifizieren, nennen wir **Schlüsselattribute** und ihre Zusammenfassung einen **Schlüssel** für das betreffende Entity

- Da wir die einzelnen Entities eines Entity-Typs als wohlunterscheidbar vorausgesetzt haben, kommt die Menge aller einwertigen Attribute des Typs immer als ein Schlüssel für den Typ in Frage. Im allgemeinen ist man jedoch in der Lage, bestimmte Attribute oder Attributkombinationen (aber nicht alle) als (zeitinvariante) Schlüssel auszuzeichnen.

- Es ist auch denkbar, dass sich für einen Entity-Typ mehr als ein Schlüssel bzw. genauer Schlüsselkandidat angeben lässt.

Beispiel:

Entity-Typ: Stadt
Attribute: PLZ, Name, Einwohner, Vorwahl

Schlüssel: PLZ oder Vorwahl

- Es wird stets ein Schlüssel als **Primärschlüssel** ausgezeichnet und in die Beschreibung eines Typs aufgenommen.

- Ein Schlüssel K ist immer eine **minimale, identifizierende Attributkombination**. Jede (echte) Obermenge von K ist ein Superschlüssel.

- **Definition**:
  Eine **Entity-Deklaration** hat die Form E = (X,K). Sie besteht aus einem Namen E, einem Format X und einem Primärschlüssel K, welcher aus (einwertigen) Elementen von X zusammengesetzt ist.

  Die Elemente eines Formats X werden dabei wie folgt notiert:
  (i) Einwertige Attribute: A
  (ii) Mehrwertige Attribute: {A}
  (iii) Zusammengesetzte Attribute: $A(B_1,...,B_k)$

  Nach gängigem Sprachgebrauch bezeichnet man eine Entity-Deklaration auch als **Entity-Typ** oder **Entity-Set**.

- **Beispiel**:
  Die Entity-Deklaration aus Abb. 3-3:
  Buch=((InvNr, {Autor}, Titel, Verlag(Name, Ort), Jahr), (InvNr))

  Die Entity-Deklaration aus Abb. 3-4:
  Leser=((LNr, Name(Vorn,Nachn), Adresse(Strasse,Ort)), (LNr))

3.1.7 Relationships

- Verschiedene Entity-Typen eines betrachteten Systems stehen im allgemeinen miteinander in Beziehung

  z.B.: Bücher werden von Lesern entliehen, sodass durch den Entlehnungsvorgang ein bestimmtes Buch mit einem bestimmten Leser in Beziehung gesetzt wird.

- An einer Beziehung sind Entities beteiligt

- Beziehungen können Attribute besitzen, welche spezielle Eigenschaften zum Ausdruck bringen, die erst durch das Herstellen der Beziehung relevant werden, jedoch für die einzelnen Entity-Typen bedeutungslos sind. Z.B.: Rückgabedatum der Beziehung “entliehen”.

  Attribute können wie bei Entities einwertig, mehrwertig oder zusammengesetzt sein.

- Wie bei Entities unterscheidet man bei Relationships zwischen den zeitinvarianten Beschreibungen der Beziehungstypen und dem zeitveränderlichen Inhalt.

- **Definition**:
  Eine **Relationship-Deklaration** hat die Form
  R = (Ent,Y). Dabei ist R der Name der Deklaration (“Name der Beziehung”), und Ent die Folge der Namen der Entity-Deklarationen, zwischen denen eine Beziehung definiert werden soll, und Y ist eine (möglicherweise leere) Folge von Attributen (der Beziehung).

  Die Relationship-Deklaration wird auch als **Relationship-Typ** bzw. **Beziehungstyp** bezeichnet.

- **Beispiel**:
  Die Relationship-Deklaration aus Abb. 3-5:
  Ausleihe = ((Buch, Leser), (RDat))

  Die Relationship-Deklaration aus Abb. 3-7:
  liefert = ((Lieferant, Teil, Filiale), (Preis, Anz))

- Ein Beziehungstyp wird durch eine Raute, welche den Namen der Deklaration enthält, dargestellt. Diese wird durch Kanten mit den beteiligten Entity-Typen verbunden.

- Falls der Beziehungstyp eigene Attribute besitzt, werden diese durch Kreise/Ellipsen dargestellt und über Kanten mit der entsprechenden Raute verbunden.

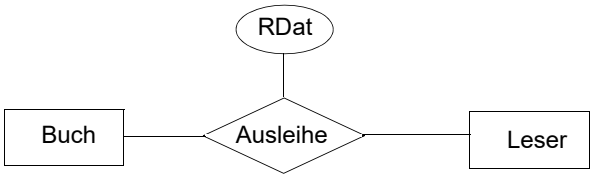


Abb. 3-5: Beziehung zwischen Büchern und Lesern

- Sei nun R = (Ent, Y) ein Beziehungstyp mit
  Ent = (E_1,...,E_k). Dann heißt k die **Stelligkeit** oder der **Grad** von R.
  Es sei bemerkt, dass der Fall k = 2 (“binäre Beziehung”) in praktischen Anwendungen der bei weitem häufigste Fall ist.

- Ein Entity kann jedoch auch mit sich selbst in Beziehung gesetzt werden. Man spricht auch von **rekursiven** oder **unären** Beziehungen. Auf jeder Seite der Beziehung übernimmt das Entity eine andere Rolle.

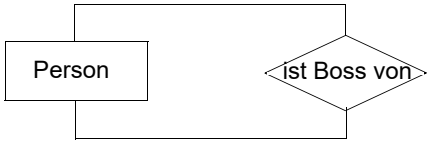


Abb. 3-6: Rekursive Beziehung zwischen Personen

- Jedoch können an einer Beziehung auch 3 (“ternäre Beziehung”) oder mehr (“n-äre Beziehungen”) Entity-Typen beteiligt sein.

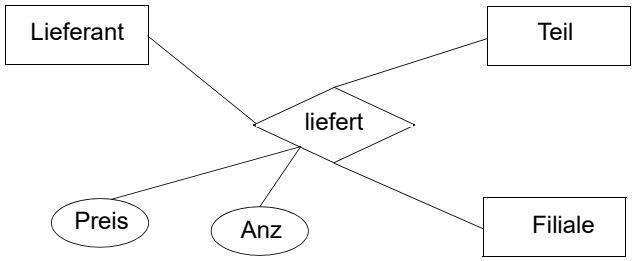


Abb. 3-7: ternäre Lieferantenbeziehung

- Beziehungstypen wird eine **Kardinalität** (Komplexität) zugeordnet, welche angibt, mit wievielen Entities des zweiten Typs (bei k = 2) ein bestimmtes Entity des ersten in einer konkreten Beziehung stehen kann, darf oder sogar muss.

- Ist R eine Relationship-Deklaration R = (Ent, Y) mit Ent = (E_1, ..., E_k), so tritt ein spezielles Entity $e_i \in E_i^t$ (zum Zeitpunkt t) in einem Relationship-Set R^t mindestens m-mal, aber höchstens n-mal auf. Dies wird durch die Schreibweise “comp(R,E_i) = (m,n)” ausgedrückt. Für den Fall k = 2 gilt:

R ist vom Typ	**lies**	**falls gilt**		**Bedeutung**
		comp(R,E_1) ∈	**comp(R,E_2) ∈**	
1:1	[0 oder 1] zu [1 oder 0]	{(0,1), (1,1)}	{(0,1), (1,1)}	Jede Entität aus der ersten Entitätsmenge kann mit höchstens einer Entität aus der zweiten Entitätsmenge in Beziehung stehen, und umgekehrt.
1:n	[0 oder 1] zu beliebig vielen	{(0,∞), (1,∞)}	{(0,1), (1,1)}	Jede Entität aus der ersten Entitätsmenge kann mit beliebig vielen Entitäten aus der zweiten Entitätsmenge in Beziehung stehen. Jede Entität aus der zweiten Entitätsmenge kann mit höchstens einer Entität aus der ersten Entitätsmenge in Beziehung stehen.
n:m	beliebig viele zu beliebig vielen	{(0,∞), (1,∞)}	{(0,∞), (1,∞)}	Jede Entität aus der ersten Entitätsmenge kann mit beliebig vielen Entitäten aus der zweiten Entitätsmenge in Beziehung stehen, und umgekehrt.

- Die Kardinalität wird an den Kanten der Beziehung angezeigt.
- Eine Beziehung ist vom Typ 1:1, falls jedes Entity vom Typ E_1 mit höchstens einem Entity vom Typ E_2 in Beziehung steht und umgekehrt.

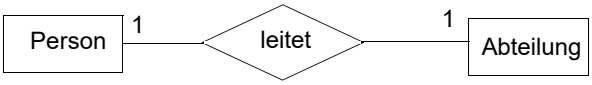


Abb. 3-8: 1:1-Beziehung zwischen Abteilung und ihrem Leiter

- Eine Beziehung ist vom Typ 1:n, falls ein Entity vom Typ E_1 mit $n \geq 0$ Entities vom Typ E_2 in Beziehung steht, andererseits jedoch jedes Entity vom Typ E_2 mit höchstens einem vom Typ E_1 assoziiert ist.

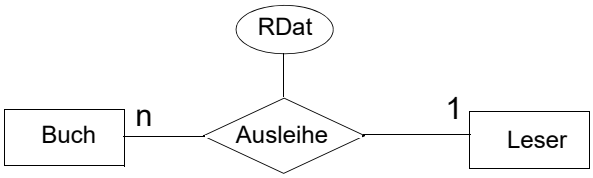


Abb. 3-9: Beziehung zwischen Büchern und Lesern

- Eine Beziehung vom Typ n:m stellt keine Restriktionen an die Entity-Paare eines Beziehungstyps.

Abb. 3-10: Beziehung zwischen Filialen und Artikeln

- Es können zwischen den selben Entity-Typen auch mehrere unterschiedliche Beziehungstypen bestehen.

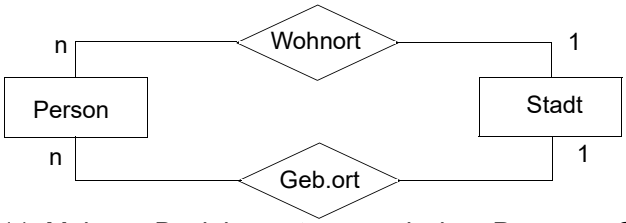


Abb. 3-11: Mehrere Beziehungstypen zwischen Person u. Stadt

- Für den Fall $k \geq 3$, gibt die Kardinalität an, mit wievielen Entities eines bestimmten Typs eine bestimmte Kombination von Entities der restlichen Typen (eine Entity pro Typ) in einer konkreten Beziehung stehen kann, darf oder sogar muss.

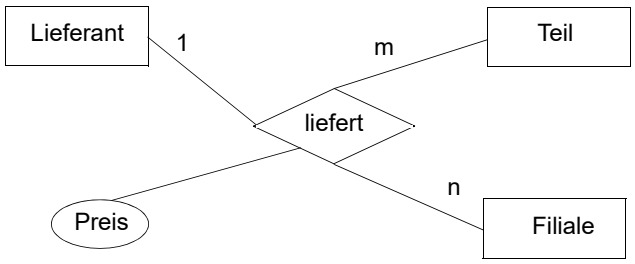


Abb. 3-12: ternäre Lieferantenbeziehung

3.1.8 IS-A-Beziehungen

- Es gibt den Fall, dass die Entities eines Typs nicht (allein) über ihre Attribute unterschieden werden, sondern über ihre Beziehung zu Entities eines anderen Typs.

- Beispiel:
  Angestellter = ((AngNr, Name, Adresse, Beruf, Gehalt),(AngNr))

  Für Piloten wird die Erfahrung in Flugstunden gemessen und sie besitzen eine Fluglizenz. Techniker gehören hingegen gewissen Technikerteams an. In beiden Fällen gibt es Attribute, welche für bestimmte Angestellte der Gesellschaft, aber nicht für alle relevant sind. In diesem Fall ist es sinnvoll, jeweils separate Entity-Deklarationen vorzunehmen:

  Pilot = ((AngNr, Std, Liz) , (AngNr))
  Techniker = ((AngNr, TNr) , (AngNr))

  In beiden Fällen handelt es sich um Spezialisierungen von Angestellter, welche zu diesen Deklarationen in einer sogenannten IS-A-Beziehung steht. Ein konkreter Pilot kann also als Entity vom Typ Angestellter und als Entity vom Typ Pilot aufgefasst werden.

- Eine konkrete Entity eines Subtyps (Spezialisierung) kann auch als Entity des Supertyps (Generalisierung) aufgefasst werden.

- Alle Attribute des Supertyps werden an die der Subtypen vererbt. Es ist nicht notwendig in einer grafischen Darstellung sämtliche Attribute eines Subtyps anzuzeigen, sondern es reichen die neuen Attribute.

- **Definition**:
  Sind $E_1=(X_1,K_1)$ und E2=(X_2,K_2) zwei Entity-Deklarationen, so besteht zwischen diesen eine **IS-A-Beziehung** der Form E_1 IS-A E_2, falls gilt:
  (i) Alle Elemente von X_2 kommen in X_1 vor;
  (ii) zu jedem Zeitpunkt t gilt: Für jedes $e_1 \in E_1^t$ existiert ein $e_2 \in E_2^t$ mit $e_1(A) = e_2(A)$ für jedes Attribut $A \in X_2$.

  Man schreibt auch kurz $E_1 \subseteq E_2$.

- Die (explizite) Bedingung (i) zusammen mit der (impliziten) Forderung, dass K_1 ein Schlüssel für X_1 (also für die Folge aller Attribute) ist, folgende Implikationen besitzen kann:
  - Es gilt $K_1 = K_2$; häufigster Fall
  - K_2 ist eine Teilfolge vom K_1; es gibt für die Spezialisierung neue Attribute, welche zur Identifikation eines Entities benötigt werden.

- Zur grafischen Darstellung von IS-A-Beziehungen verwenden wir wieder Rauten, in die jetzt “IS-A” eingetragen wird. Diese werden mit ungerichteten Kanten mit der oder den Subtypen und mit einer gerichteten Kante zum Supertyp verbunden.

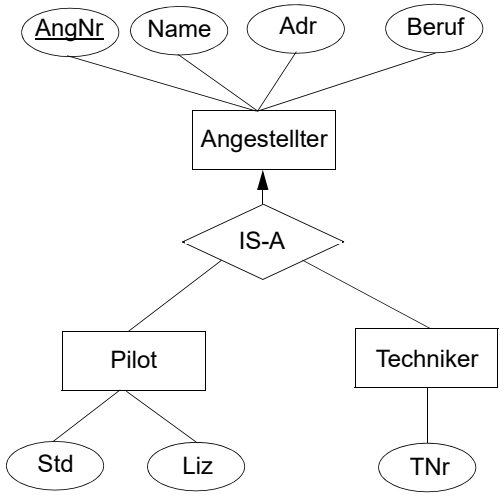


Abb. 3-13: IS-A-Beziehung zw. Angestellten, Piloten und Technikern

- Die Auszeichnung einer Reihe von Spezialisierungen einer einzelnen Entity-Deklaration kann
  - total (t) oder partiell (p) sein
  - disjunkt (d) oder nicht-disjunkt (n) sein

  Es ergeben sich 4 verschiedene Kombinationen

- **Definition**:
  Es seien E, E_1, ..., E_k Entity Deklarationen, $k \geq 2$, und es gelte, dass alle E_i, $1 \leq i \leq k$ zu derselben Spezialisierung von E gehören. Die IS-A-Beziehung heißt
  - **total**, falls gilt: $(\forall t) \cup_{i=1}^{k} E_i^t = E^t$

    andernsfalls **partiell**
  - **disjunkt**, falls gilt: $(\forall t)\ (\forall i, j \in \{1,...k\}, i \neq j) : E_i^t \cap E_j^t = \varnothing$

    andernfalls **nicht-disjunkt**

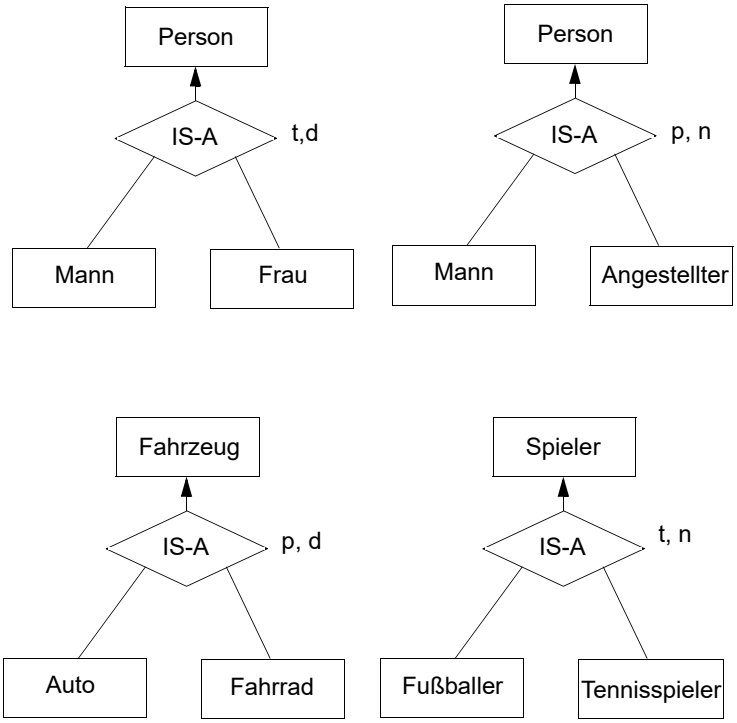
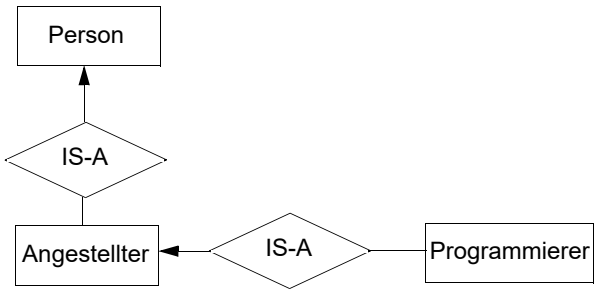


Abb. 3-14: Arten von IS-A-Beziehungen

- Spezialisierungen bzw. Generalisierungen können auch über mehrere Ebenen vorgenommen werden



3.1.9 Beispiel

Das Beispiel behandelt einen Ausschnitt einer konzeptionellen Modellierung für eine Fluggesellschaft. Die Gesellschaft will Daten über Personen, insbesondere Passagiere und die Angestellten, ihre Flugzeuge und die durchgeführten Flüge speichern. Wir betrachten zunächst die Personen genauer: Jede Person habe zur Identifikation eine Nummer, einen Namen, eine Adresse und ein Geburtsdatum. Ein Passagier hat zusätzlich ein Geschlecht und eine Angabe, ob er Raucher ist oder nicht. Angestellte haben eine Angestelltennummer, einen Beruf und ein Gehalt. Unter ihnen sind die Piloten und die Techniker ausgezeichnet.
Bei der Modellierung von Flugzeugen unterscheiden wir zwischen Flugzeugen, welche die Gesellschaft besitzt, gekennzeichnet durch Seriennummer, Kaufdatum, die Summe der bisherigen Flugstunden, den Typ des Ausbaus, eine Menge in jüngster Zeit vorgekommener Störfälle und die zuletzt durchgeführte Wartung und solchen Angaben, die für alle Maschinen eines Typs gültig sind, wie Hersteller, Typ, Länge, Reisegeschwindigkeit, Sitzanzahl, Wartungsintervalle und Reichweite. (So hat z.B. jeder Airbus 330-400 die gleiche Reisegeschwindigkeit; falls die Gesellschaft jedoch mehrere Maschinen dieses Typs besitzt, so sind diese über ihre Seriennummer unterscheidbar.)
Analog unterscheiden wir zwischen bestimmten, im allgemeinen regelmäßig verkehrenden Flügen (z.B. OS 873 von Wien nach Athen) und den einzelnen, tatsächlich an einem bestimmten Datum durchgeführten Abflügen (OS 873 am 28. 3. 2001), welche unter der Leitung eines bestimmten Piloten mit einer bestimmten Maschine der Gesellschaft durchgeführt wird und Passagiere an Bord hat. Die Unterscheidung zwischen “realen” und “virtuellen” Maschinen bzw. Flügen ermöglicht auch eine Unterscheidung zwischen der konkret ausgeführten und der möglichen Tätigkeit eines Piloten: Ein Pilot kann z.B. nicht nur eine spezielle, sondern jeden Airbus 330-400 fliegen; real eingesetzt wird er hingegen nur bei speziellen Flügen, aber nicht notwendig bei jedem Flug mit der gleichen Bezeichnung. Analog sind die Beziehungen “kann warten” und wartet zwischen Technikern und Fluggerät bzw. Flugzeugen zu verstehen.

Es sei bemerkt, dass die Lösung keinen Anspruch auf Vollständigkeit erhebt.

.

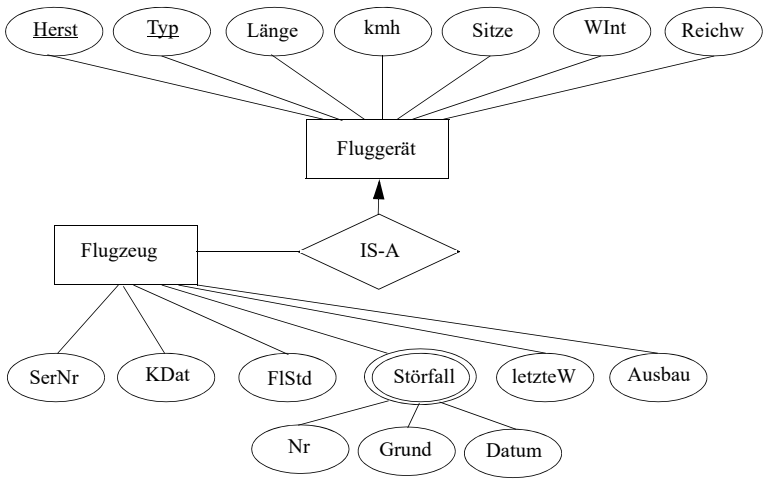


Abb. 3-15: Erstes Teildiagramm für die Fluggesellschaft

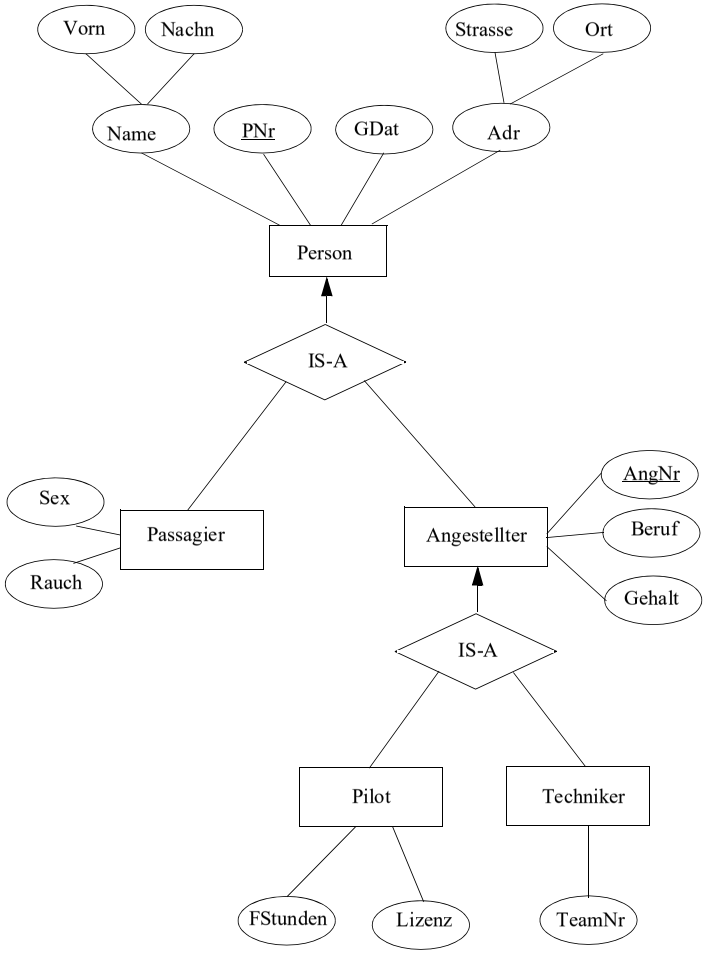


Abb. 3-16: Zweites Teildiagramm für die Fluggesellschaft

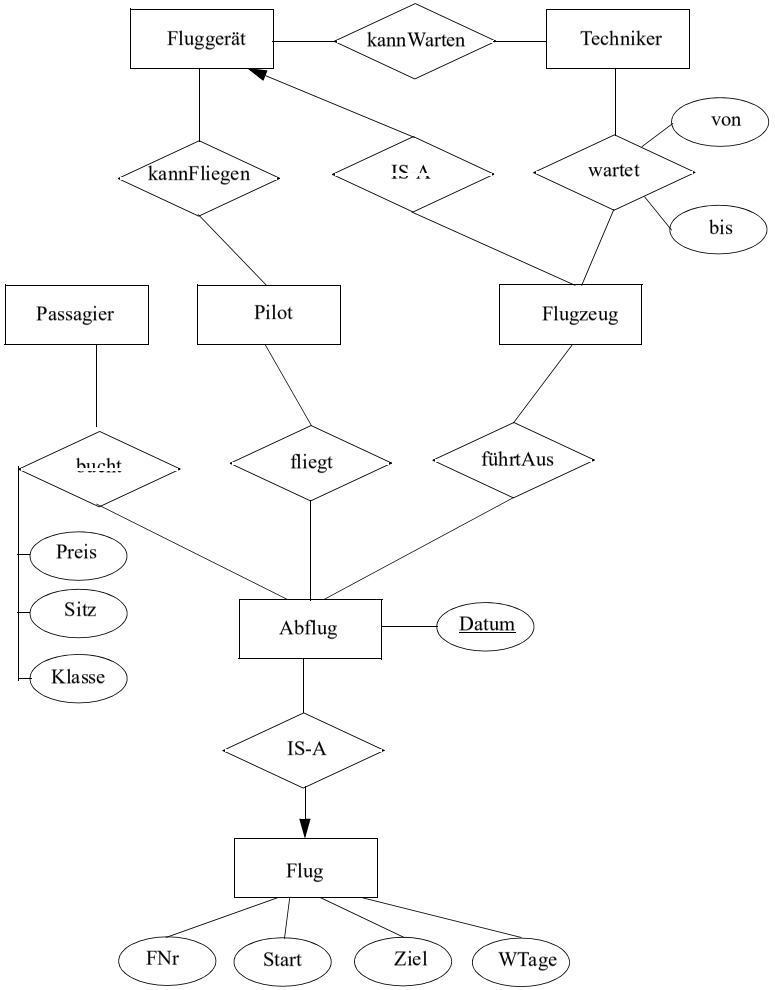


Abb. 3-17: Drittes Teildiagramm für die Fluggesellschaft

3.2 Erweiterungen und Spezialfälle der ER-Modellierung

3.2.1 Min,Max-Notation

- Bei der Min,Max-Notation für einen Beziehungstyp wird nicht nur die maximale Komplexität in der Kardinalität der beteiligten Entity-Typen angegeben, sondern die Kardinalität wird in Form eines Intervalls [u,o] angegeben.

- Eine Beziehung hat die Kardinalität (u, o), wenn zu jeder Entität aus der Entitätsmenge X mindestens u, aber höchstens o verschiedene Entitäten aus der Entitätsmenge Y in Beziehung gebracht werden können

- Die Grenzen des Intervalls bedeuten daher, dass zwischen u und o Entitäten des betroffenen Entity-Typs mit dem in Relation stehenden Entity-Typ (bei binären Relationen) in Beziehung stehen können.

- Notation der Kardinalität (u, o) als Intervall [u, o]
  - 0 steht für kein Vorkommen
  - 1 steht für einmaliges Vorkommen
  - * steht für beliebig oftes Vorkommen (0.. ∞)

- Vorteil: sowohl Unter- als auch Obergrenzen können präzise ausgedrückt werden

- Nachteil: keine Modellierung von Beziehungen, an denen drei Entity-Typen beteiligt sind. Daher wird die min,max-Notation oft eingeschränkt im Sinne der Chen-Notation verwendet

- Für die Überführung in Relationenschemata gilt, dass eine Oberschranke $o \geq 2$, wie ein “n” zu behandeln ist (siehe dazu Datenbanksystem-Vorlesung).

- Achtung: Manche Autoren verwenden bei der Min/Max-Notation eine Umkehrung der Kantenbeschriftung.

3.2.2 Modifizierte Chen-Notation (MC-Notation)

- MC-Notation ist eine Erweiterung der Chen-Notation:
  - Aussage "kein oder ein Element" mit dem Buchstaben c (choice, can)
  - Aussage "ein oder mehr Element(e)" mit dem Buchstaben m (must, multiple)

  angegeben wird.
  (Daher wird MC manchmal auch als Must-Can interpretiert.)

- **1:1** (1 zu 1): Jede Entität der ersten Entitätsmenge steht mit genau einer Entität der zweiten Entitätsmenge in Beziehung, und umgekehrt.

- **1:c** (1 zu (0 oder 1)): Jede Entität der ersten Entitätsmenge kann mit höchstens einer Entität der zweiten Entitätsmenge in Beziehung stehen. Jede Entitiät der zweiten Enititätsmenge steht mit genau einer Entität der ersten Entitätsmenge in Beziehung.

- **1:m** (1 zu (mindestens 1)): Jede Entität der ersten Entitätsmenge steht mit mindestens einer Entität der zweiten Entitätsmenge in Beziehung. Jede Entitiät der zweiten Enititätsmenge steht mit genau einer Entität der ersten Entitätsmenge in Beziehung.

- **1:mc** (1 zu (beliebig viele)): Jede Entität der ersten Entitätsmenge kann mit beliebig vielen Entitäten der zweiten Entitätsmenge in Beziehung stehen. Jede Entitiät der zweiten Enititätsmenge steht mit genau einer Entität der ersten Entitätsmenge in Beziehung.

- **c:c** ((1 oder 0) zu (0 oder 1); entspricht 1:1 in Chen-Notation): Jede Entität der ersten Entitätsmenge kann mit höchstens einer Entität der zweiten Entitätsmenge in Beziehung stehen, und umgekehrt.

- **c:m** ((0 oder 1) zu (mindestens 1)): Jede Entität der ersten Entitätsmenge steht mit mindestens einer Entität der zweiten Entitätsmenge in Beziehung. Jede Entitiät der zweiten Enititätsmenge kann mit höchstens einer Entität der ersten Entitätsmenge in Beziehung stehen.

- **c:mc** ((0 oder 1) zu (beliebig viele); entspricht 1:n in Chen-Notation): Jede Entität der ersten Entitätsmenge kann mit beliebig vielen Entitäten der zweiten Entitätsmenge in Beziehung stehen. Jede Entitiät der zweiten Enititätsmenge kann mit höchstens einer Entität der ersten Entitätsmenge in Beziehung stehen.

- **m:m** ((mindestens 1) zu (mindestens 1)): Jede Entität der ersten Entitätsmenge steht mit mindestens einer Entität der zweiten Entitätsmenge in Beziehung, und umgekehrt.

- **m:mc** ((mindestens 1) zu (beliebig viele)): Jede Entität der ersten Entitätsmenge kann mit beliebig vielen Entitäten der zweiten Entitätsmenge in Beziehung stehen. Jede Entitiät der zweiten Enititätsmenge steht mit mindestens einer Entität der ersten Entitätsmenge in Beziehung.

- **mc:mc** ((beliebig viele) zu (beliebig vielen)); entspricht m:n in Chen-Notation): Jede Entität der ersten Entitätsmenge kann mit beliebig vielen Entitäten der zweiten Entitätsmenge in Beziehung stehen, und umgekehrt.

3.2.3 Gegenüberstellung Min-Max-Notation, Chen-Notation und MC-Notation

Chen-Notation: Entität 1 ← 1:n → Entität 2
MC-Notation: Entität 1 ← c:mc → Entität 2
Min-Max-Notation: Entität 1 ← (0,1) ——— (0,*)→ Entität 2

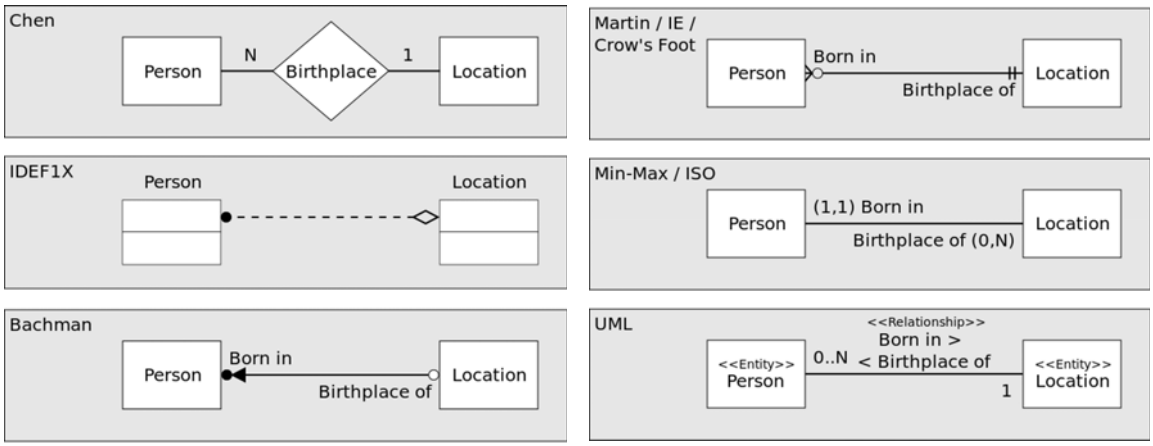
Entität 1	Chen-Notation	MC-Notation	Entität 2
(0,1)	1:1	c:c	(0,1)
(0,1)	1:n	c:mc	(0,*)
(0,1)	---	c:m	(1,*)
(0,*)	m:n	mc:mc	(0,*)
(1,1)	---	1:c	(0,1)
(1,1)	---	1:mc	(0,*)
(1,1)	---	1:1	(1,1)
(1,1)	---	1:m	(1,*)
(1,*)	---	m:mc	(0,*)
(1,*)	---	m:m	(1,*)

Tabelle 3-1: Vergleich der Notationen zur Kardinalität

3.2.4 Verschiedene Konventionen für Notation von Kardinalitäten

Achtung: In der Praxis / in Tools werden verschiedene Konventionen zur Notation von Kardinalitäten verwendet. Daher ist es oft wichtig, die im Diagramm benutzte Notation explizit zu benennen (z.B. Legende angeben).

Beispiele für Notationsformen:



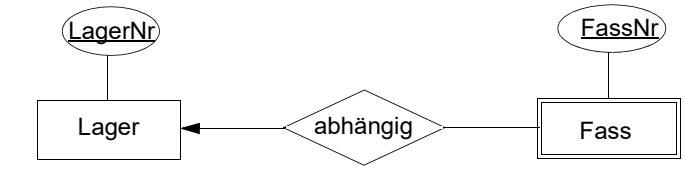
[Quelle: Wikipedia]

3.2.5 Weak Entity-Typ (Schwache Entity)

- Es gibt Situationen, in denen einem Entity-Typ **nicht genügend viele Attribute zur Bildung eines Schlüssels** zugewiesen werden können. Diese Entity-Typen bezeichnet man als Weak Entity-Typen bzw. schwache Entity-Typen.
- Ein Weak Entity kann nicht isoliert existieren, sondern ist stets über eine Existenzbedingung mit einem gewöhnlichen Entity verbunden.
- Die Schlüsselattribute eines Weak Entity-Typs werden durch den Schlüssel des übergeordneten Typs vervollständigt.
- Damit erbt der Weak Entity-Typ die Schlüsselattribute des übergeordneten Typs und erweitert diese um spezifische Schlüsselattribute.
- Die Beziehung eines Weak Entity-Typ ist verwandt zur IS-A-Beziehung, hat aber eine andere Semantik.
- Darstellung des Weak Entity-Typs durch doppelt umrandetes Rechteck.

- Beispiel:

Annahme: Die Fassnummern sind generell nicht eindeutig
jedoch sind die Fässer innerhalb eines Lagers eindeutig numeriert.



Lager(LagerNr, ...)

Fass(LagerNr, FassNr, ...)

3.2.6 Uminterpretierter Beziehungstyp

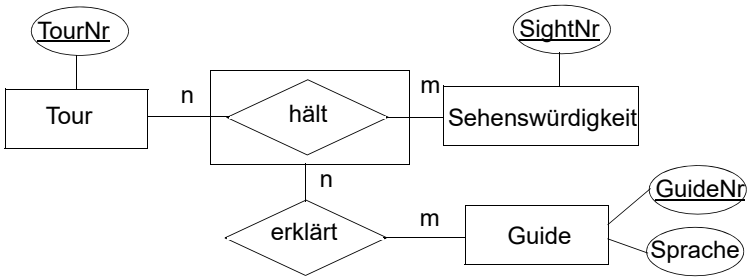
- Im herkömmlichen ER-Modell sind nur Beziehungs-Typen zwischen Entity-Typen möglich.

- In bestimmten Anwendungen kann es erforderlich werden, Beziehungen auch zwischen Beziehungstypen selbst zu modellieren.

- Dabei wird ein Beziehungs-Typ, der mit anderen Entity-Typen (bzw. Beziehungs-Typen) in Beziehung gesetzt werden soll, uminterpretiert. D.h. der Beziehungs-Typ wird in der Folge wie ein Entity-Typ behandelt.

- Zur grafischen Notation wird ein Rechteck über die Raute des Beziehungstyps geschlossen.
  Kanten, die zur Raute gehen, markieren eine Beteiligung am Beziehungstyp.
  Kanten, die zum Rechteck gehen, markieren eine Beteiligung des Uminterpretierten Beziehungstyps an einer anderen Beziehung

❒ Beispiel:

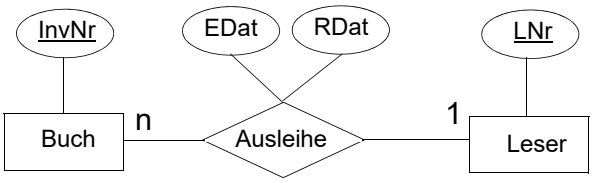


3.2.7 Historische Daten

❒ Für sehr viele Anwendungen ist es entscheidend, ob historische Daten gespeichert werden müssen oder nicht.

❒ Beispielsweise ist es entscheidend, ob bei einer Bibliothek die Daten der Leihe über zurückgegebene Bücher erhalten bleiben müssen.

❒ Betrachten wir die Ausleihrelation, die bislang in diesen Unterlagen verwendet wurde und ergänzen wir sie um das Entlehndatum:



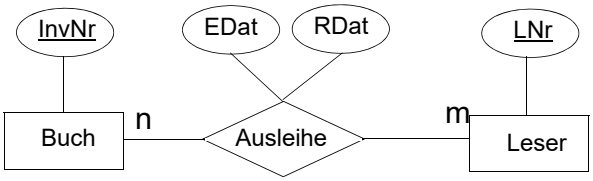
- Beispielsammlung für Ausleihe für den Fall, dass nur aktuelle Ausleihen gespeichert werden:

InvNr	LNr	EDat	RDat
1024	125	12.03.2007	12.05.2007
3700	125	12.03.2007	12.05.2007
5045	170	28.02.2007	28.04.2007
8345	131	09.03.2007	09.05.2007

- Beispielsammlung für Ausleihe für den Fall, dass auch historische Ausleihen gespeichert werden sollen:

InvNr	LNr	EDat	RDat
1024	125	12.03.2007	12.05.2007
3700	125	12.03.2007	12.05.2007
7830	125	01.08.2005	01.10.2005
5540	182	04.01.2007	04.03.2007

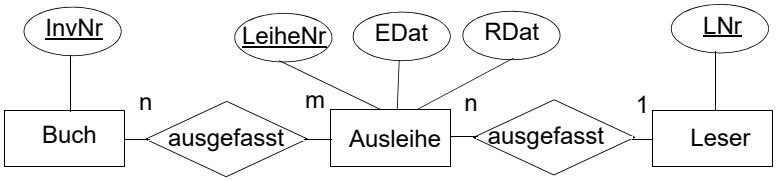
- Für die Speicherung historischer Ausleihen ist eine n:m-Beziehung nötig



- Beispielsammlung für Ausleihe für den Fall, dass ein Leser ein Buch öfters ausborgen kann:

InvNr	LNr	EDat	RDat
1024	125	12.03.2007	12.05.2007
3700	125	12.03.2007	12.05.2007
3700	125	12.01.2007	12.03.2007
7830	125	01.08.2005	01.10.2005

- Um ein wiederholtes Ausleihen eines Buchs zu speichern, wird die Ausleihe zur Entität, die mit Leser und Buch in Beziehung steht:



Literatur und Quellen:

- Peter Pin-Shan Chen: *The Entity-Relationship Model - Toward a Unified View of Data.* In: ACM Transactions on Database Systems, 1/1/1976, ACM-Press, ISSN 0362-5915, S. 9-36
  - Download über Uni Wien Rechner/VPN: http://doi.acm.org/10.1145/320434.320440
- J.M. Smith, D.C.P. Smith: *Database Abstractions: Aggregation and Generalization*, ACM Transactions on Database Systems, Vol. 2, No. 2 (1977), S. 105-133
  - Download über Uni Wien Rechner/VPN: http://doi.acm.org/10.1145/320544.320546
- J. M. Smith and D. C. P. Smith: *Database Abstraction: Aggregation*, Communications of the ACM, Vol. 20, Nr. 6, pp. 405-413, June 1977
  - Download über Uni Wien Rechner/VPN: http://doi.acm.org/10.1145/359605.359620

- Tool zur Vertiefung: BEE-UP Modelling Tool: http://austria.omilab.org/psm/content/bee-up/download?view=download