Teil 9: Einführung in den logischen Entwurf

Literatur:

- Elmasri/Navathe:Fundamentals of Database Systems, 3. Auflage, 1999. Chapter 3, “Data Modeling Using the Entity-Relationship Model”
- Silberschatz/Korth/Sudarshan: Database System Concepts, 3. Auflage, Ch. 2, “Entity-Relationship Model”.
- Ramakrishnan: Database Management Systems, Mc-Graw Hill, 1998, Ch. 14, “Conceptual Design and the ER-Model”
- Kemper/Eickler: Datenbanksysteme, Ch. 2, Oldenbourg, 1997.
- Rauh/Stickel: Konzeptuelle Datenmodellierung, Teubner, 1997.
- Teorey: Database Modeling and Design, 3. Auflage, 1999.
- Barker: CASE*Method, Entity Relationship Modelling, Oracle/Addison-Wesley, 1990.
- Lipeck: Skript zur Vorlesung Datenbanksysteme, Univ. Hannover, 1996.

Lernziele

Nach diesem Kapitel sollten Sie folgendes können:

- Ein gegebenes Entity-Relationship-Diagramm in das relationale Modell übersetzen.

  D.h. ein äquivalentes relationales Datenbank-Schema konstruieren (einschließlich Schlüssel und Fremdschlüssel und ggf. weiteren IBen).
- Erklären welche Konstrukte (Kardinalitäten) nicht direkt übersetzt werden können.
- Typische ER-Strukturen (wie etwa viele-zu-viele-Beziehungen) in relationalen DB-Schemas wiedererkennen (etwas “Reverse Engineering”).

Inhalt

Übersicht

- Um ein relationales Schema zu entwickeln, entwirft man zunächst ein ER-Diagramm, und transformiert es dann in das relationale Modell, da das ER-Modell
  - ◇ eine bessere Dokumentation der Beziehung zwischen dem Schema und der realen Welt erlaubt,

    Z.B. Entity-Typen und Relationships unterscheidet.
  - ◇ eine nützliche graphische Notation hat,
  - ◇ Konstrukte wie Vererbung beinhalt, für die es keine Entsprechung im relationalen Modell gibt.

    Vererbung und andere nützliche Erweiterungen werden erst in der Vorlesung “Datenbanken II A: Datenbank-Entwurf” behandelt.

Logischer Entwurf: Ziel (1)

- Für ein gegebenes ER-Schema S_E soll ein relationales Schema S_R entwickelt werden, so dass es eine bijektive Abbildung τ zwischen den Zuständen für S_E und S_R gibt.

  D.h. für jeden möglichen DB-Zustand bezogen auf S_E gibt es genau einen Zustand bezogen auf S_R, und umgekehrt.

- Zustände, die im relationalen Schema möglich sind, aber keine Entsprechung bzgl. des ER-Schemas haben, müssen durch IBen ausgeschlossen werden.

  Z.B. können Relationships im ER-Modell nur zwischen bereits existierenden Entities bestehen. Im relationalen Modell müssen ungültige Referenzen durch Fremdschlüsselbedingungen ausgeschlossen werden.

Logischer Entwurf: Ziel (2)

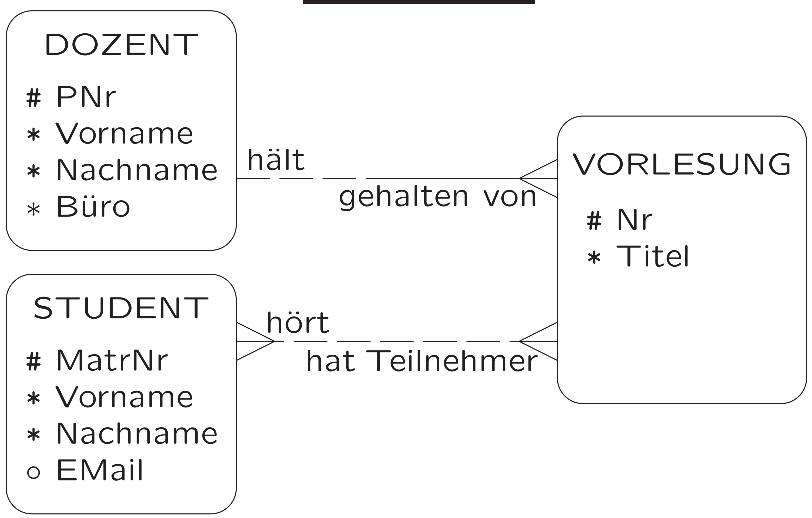
- Zusätzlich muss es möglich sein, Anfragen bezogen auf S_E in Anfragen bezogen auf S_R zu übersetzen, diese im relationalen System auszuwerten, und die Antwort zurückzuübersetzen.

- So kann dann die entworfene ER-Datenbank durch die tatsächlich implementierte relationale Datenbank simuliert werden.

  Jede Schema-Übersetzung muss auch die Transformation der einzelnen Schemaelemente erklären, so dass Anfragen (einfach) übersetzt werden können. Nur eine bijektive Abbildung der Zustände reicht nicht (man könnte ja alles in einer einzigen natürlichen Zahl codieren).

Inhalt

Beispiel



Entity-Typen, Attribute (1)

- Zuerst wird für jeden Entity-Typ eine Tabelle (mit gleichem Namen) erstellt.

  Alternativ kann man als Tabellennamen die Pluralform verwenden.

- Attribute des Entity-Typs werden in Spalten der Tabelle übersetzt.

  Falls das Attribut optional ist, erlaubt die Spalte Nullwerte.

- Primär-/Alternativschlüssel des Entity-Typs werden Schlüssel der Tabelle übersetzt.

  Falls der Entity-Typ keinen Primärschlüssel hat, wird ein künstlicher Schlüssel hinzugefügt.

Entity-Typen, Attribute (2)

DOZENTEN			
<u>PNr</u>	Vorname	Nachname	Büro
11	Paul	Molitor	217
12	Stefan	Brass	313

VORLESUNGEN	
<u>Nr</u>	Titel
1	OOP
2	DB I
3	TI

STUDENTEN			
<u>MatrNr</u>	Vorname	Nachname	EMail
101	Lisa	Weiss	lw@gmx.de
102	Michael	Grau	NULL

Eins-zu-viele Beziehungen (1)

- Hat eine Beziehung (Relationship) einen Krähenfuß nur auf einer Seite, so ist es eine eins-zu-viele (1:n) Beziehung. Beispiel:

- Die Beziehung wird übersetzt, indem der Schlüssel der "eins"-Seite (DOZENT) als Fremdschlüssel zur "viele"-Seite (VORLESUNG) hinzugefügt wird.

  Die Tabelle VORLESUNGEN wird also um die Spalte "PNr" (den Schlüssel der Tabelle DOZENTEN: "PersonalNr") erweitert.

Eins-zu-viele Beziehungen (2)

- Ergebnis im Beispiel:

  `VORLESUNGEN(`<u>`Nr`</u>`, Titel, PNr→DOZENTEN)`

DOZENTEN			
<u>PNr</u>	Vorname	Nachname	Büro
11	Paul	Molitor	217
12	Stefan	Brass	313

VORLESUNGEN		
<u>Nr</u>	Titel	PNr
1	OOP	12
2	DB I	12
3	TI	11

Eins-zu-viele Beziehungen (3)

- Man darf die beiden Seiten nicht verwechseln, also z.B. nicht umgekehrt die Nummer der Vorlesung als Fremdschlüssel in die Dozententabelle einfügen.

  Das geht nicht, weil ein Dozent mehrere Vorlesungen halten kann. Man wüßte dann nicht, welcher Wert in den Fremdschlüssel eingetragen werden soll (man brauchte ein mengenwertiges Attribut).

- Bei 1:n Beziehungen ist die Relation ja eigentlich eine Funktion, und der Fremdschlüssel verweist in Richtung dieser Funktion.

  Dann gibt es nur ein Entity der anderen Seite, zu dem eine Beziehung besteht. Dessen Schlüsselwert wird im Fremdschlüssel eingetragen. Ggf. hilft es, den Krähenfuß als Pfeilspitze (aber am Ende) zu sehen.

Eins-zu-viele Beziehungen (4)

- Ist die Linie auf der Seite mit dem Fremdschlüssel durchgezogen (minimale Kardinalität 1), so sind keine Nullwerte in der Fremdschlüsselspalte erlaubt (d.h. man deklariert sie "`NOT NULL`").

  Man beachte, dass das so nur für eins-zu-viele (und eins-zu-eins) Beziehungen funktioniert. Bei viele-zu-viele Beziehungen sind Nullwerte in den Fremdschlüsseln immer ausgeschlossen, aber trotzdem kann man die minimale Kardinalität 1 nicht sicherstellen. Siehe unten.

- Ist die Linie dagegen gestrichelt (minimale Kardinalität 0), so sind Nullwerte erlaubt.

  Der Wert in der Fremdschlüssel-Spalte ist Null für die Entities, die nicht an der Beziehung teilnehmen.

Eins-zu-viele Beziehungen (5)

- Alternativ kann man den Beziehungsnamen (in der richtigen Leserichtung) als Fremdschlüssel-Namen verwenden, z.B.:

  `VORLESUNGEN(`<u>`Nr`</u>`, Titel, gehalten_von→DOZENTEN)`

- So wird die Bedeutung der Beziehung besser im relationalen Schema dokumentiert.
- Andererseits ist es aber nützlich, wenn man durch gleich benannten Spalten sofort sieht, über welche Spalten Tabellen verknüpft werden können.

Namenskonflikte von Spalten

- Natürlich müssen alle Spalten einer Tabelle eindeutige Namen haben.
- Hat ein hinzugefügter Fremdschlüssel den gleichen Namen wie eine vorhandene Spalte, muss mindestens eine der beiden Spalten umbenannt werden.

  Das wäre im Beispiel passiert, wenn man auch der Schlüssel der Dozenten einfach "Nr" genannt hätte (ebenso bei einem Schlüssel "ID" für alle Tabellen). Wenn man Relationship-Namen als Fremdschlüssel-Namen verwendet, treten solche Probleme seltener auf.

- Selbstverständlich müssen solche Umbenennungen gut dokumentiert werden.

Zusammenges. Fremdschlüssel

- Ein zusammengesetzter Fremdschlüssel wird verwendet, um eine Tabelle mit einem zusammengesetzten Schlüssel zu referenzieren.
  Vorlesung(<u>Nr</u>, Titel,
  (Vorname, Nachname) $\rightarrow$ DOZENTEN)

  Bei Minimum-Kardinalität 0 (gestrichelte Linie bei VORLESUNG) könnten "Vorname" und "Nachname" Null sein, aber nur zusammen.

Viele-zu-viele Beziehungen (1)

- Eine Beziehung ist viele-zu-viele (n:m), wenn auf beiden Seiten ein Krähenfuß ist:

- Viele-zu-viele-Beziehungen werden in eigene Tabellen übersetzt. Die Spalten dieser Tabelle sind die Schlüssel der teilnehmenden Entity-Typen.
- Sie bilden zusammen den Schlüssel der Tabelle.
- Diese Spalten sind zugleich auch Fremdschlüssel, die die Tabellen der Entity-Typen referenzieren.

Viele-zu-viele Beziehungen (2)

- Im Beispiel enthält die Beziehungs-Tabelle jeweils Paare aus Matrikelnummer und Vorlesungs-Nr:

  HÖRT(<u>MatrNr</u>$\rightarrow$STUDENTEN, <u>Nr</u>$\rightarrow$VORLESUNGEN)

  Die Semantik eines Relationship-Typs ist eine Menge von Paaren aus Entities der beteiligten Entity-Typen. Dem entspricht eine Menge von Paaren aus Schlüsselwerten der korrespondierenden Tabellen.

- Der Schlüssel muss beide Attribute enthalten:
  - ◇ Da ein Student mehrere Vorlesungen hören kann, ist "MatrNr" alleine nicht ausreichend.
  - ◇ Da eine Vorlesung von vielen Studenten gehört wird, ist auch Vorlesungs-Nr allein kein Schlüssel.

Viele-zu-viele Beziehungen (3)

Beispiel-Zustand:

HÖRT

<u>MatrNr</u>	<u>Nr</u>
101	2
102	2
102	3

STUDENTEN

<u>MatrNr</u>	Vorname	Nachname	EMail
101	Lisa	Weiss	lw@gmx.de
102	Michael	Grau	NULL

VORLESUNGEN

<u>Nr</u>	Titel	PNr
1	OOP	11
2	DB I	12
3	TI	11

Viele-zu-viele Beziehungen (4)

Namen der Beziehungs-Tabellen:

- Die Namen der Beziehungen sind teilweise nicht optimal als Namen der Beziehungs-Tabellen.
- Die Tabellen können umbenannt werden, das muss aber ausreichend dokumentiert werden.

  Es muss klar sein, welche Tabelle das Relationship implementiert.
- Im Beispiel könnte man die Tabelle statt "HÖRT" etwa "VORLESUNGS_TEILNAHME" nennen.

  Oder nur "TEILNAME". Die Umbenennung ist aber Geschmackssache, da "HÖRT" die Bedeutung schon recht klar macht. Hat man vorher dagegen als Beziehungs-Namen z.B. "für", "von" oder "hat" gewählt, ist eine Umbenennung jetzt unbedingt nötig.

Viele-zu-viele Beziehungen (5)

Minimale Kardinalität:

- Bei dieser Übersetzung sind Vorlesungen ohne Studenten möglich (im Beispiel OOP) und Studenten, die keine Vorlesung hören.
- In der Beziehungstabelle "HÖRT" sind Nullwerte ausgeschlossen (weil Primärschlüssel-Spalten).
- Aber nicht alle Studenten bzw. Vorlesungen müssen von einem Fremdschlüssel referenziert werden.

  Fremdschlüssel müssen immer auf existierende Zeilen zeigen, aber es ist nicht verlangt, das jede Zeile auch "getroffen" wird.

Viele-zu-viele Beziehungen (6)

Minimale Kardinalität, Forts.:

- Z.B. könnte man bei dem obigen Schema einige Vorlesungen und Studenten eintragen, aber die Bezeihungstabelle "HÖRT" zunächst leer lassen.

  Extremfälle wie leere Tabellen können öfters helfen, die Korrektheit eines Schemas oder auch einer Anfrage zu prüfen.
- D.h. das relationale Schema implementiert korrekt die gestrichelte Linie (Minimalkardinalität 0, optionale Teilnahme) auf beiden Seiten.

Viele-zu-viele Beziehungen (7)

Problem bei minimaler Kardinalität 1:

- Die minimale Kardinalität 1 (durchgezogene Linie, verpflichtende Teilnahme) kann man dagagen bei viele-zu-viele Beziehungen nicht direkt übersetzen.
- Immerhin ist die minimale Kardinalität 0 allgemeiner: Diese Übersetzung erlaubt auch alle Zustände, die bei Minimalkardinalität 1 möglich sind.
- Daher wählt man sie, und schließt die unzulässigen DB-Zustände mit einer Integritätsbedingung aus.

Viele-zu-viele Beziehungen (8)

Problem bei minimaler Kardinalität 1, Forts.:

- Beispiel: Vorlesungen müssen Teilnehmer haben:

- Zusätzliche Integritätsbedingung im Tupelkalkül:

$$\forall\ \texttt{VORLESUNG V}\colon\ \exists\ \texttt{HÖRT H}\colon\ \texttt{V.Nr} = \texttt{H.Nr}$$

- Die obige Formel sieht wie eine Fremdschlüsselbedingung aus, ist aber keine, weil das referenzierte Attribut hier nicht Schlüssel ist.

  Sie ist ein allgemeine Integritätsbedingung im relationalen Modell.

Viele-zu-viele Beziehungen (9)

Problem bei minimaler Kardinalität 1, Forts.:

- Man kann diese Bedingung später in den Anwendungsprogrammen überprüfen.

  Sie kann nur in der `CREATE TABLE` -Anweisung nicht deklarativ festgelegt werden. Im konkreten Beispiel ist es aber schlecht, dass man eine Vorlesung dann gleich mit ihrem ersten Teilnehmer einfügen muss. Besseres Beispiel: Gruppen-Abgaben für Hausaufgaben sind von mindestens einem Studenten.

- SQL-Anfrage zur Prüfung der Daten:

```
SELECT 'Vorlesung ohne Hörer: ' || V.Nr
FROM   VORLESUNG V
WHERE  NOT EXISTS(SELECT * FROM HÖRT H
                  WHERE  H.Nr = V.Nr)
```


Min. Kardinalität bei 1:n

- Auf der "viele"-Seite (mit Krähenfuß) kann man beide Fälle behandeln:
  - ◇ Minimalkardinalität 1 (durchgezogene Linie): "`NOT NULL` für den Fremdschlüssel.
  - ◇ Minimalkardinalität 0 (gestrichelte Linie): Nullwerte werden für den Fremdschlüssel erlaubt.
- Auf der "eins"-Seite hat man dagegen das gleiche Problem wie bei viel-zu-viele-Beziehungen.

  Man kann nicht mit Standard-Constraints erreichen, dass jede Zeile von einem Fremdschlüsselwert referenziert ist. Nur die Minimalkardinalität 0 ist direkt übersetzbar, für die verpflichtende Teilnahme braucht man eine zusätzliche Integritätsbedingung.

Alternative für 1:n

- Man kann auch Eins-zu-viele-Beziehungen als eigene Tabelle übersetzen, dann aber nur noch mit Minimalkardinalität 0 auf beiden Seiten.

  Es lohnt sich also nur, wenn die Minimalkardinalität auf der "viele"-Seite (mit Krähenfuß) 0 ist. Wenn tatsächlich nur wenige Entities an der Beziehung teilnehmen, ist es möglicherweise interessant.

- Im Unterschied zur viele-zu-viele Beziehung besteht der Schlüssel nur aus dem der "viele"-Seite:

$$\texttt{HÄLT(PNr}\rightarrow\texttt{DOZENTEN, }\underline{\texttt{Nr}}\rightarrow\texttt{VORLESUNGEN)}$$

Inhalt

Schwache Entities (1)

- Wird ein schwaches Entity übersetzt, müssen die Schlüsselattribute des Master-Entities als Fremdschlüssel und Teil des Schlüssels zugefügt werden:

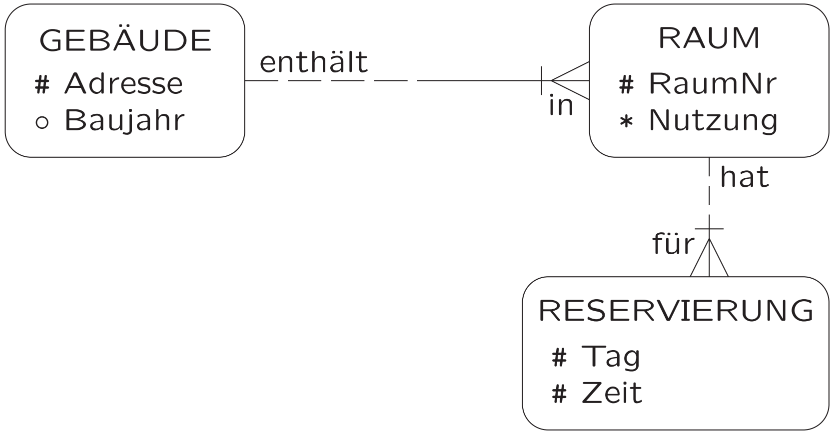
  RAUM(Adresse → GEBÄUDE, RaumNr, . . .)

- Das implementiert automatisch die Beziehung.

  Eine solche Beziehung darf dann natürlich nicht nochmal getrennt übersetzt werden. Ggf. "`DELETE CASCADES`" für den Fremdschlüssel.

Schwache Entities (2)

Mehrstufige Hierarchie:



Schwache Entities (3)

- Bei Hierarchien von schwachen Entities erben alle untergeordneten Entities die Schlüsselattribute:

  GEBÄUDE(Adresse, Baujahr°)
  RÄUME(Adresse → GEBÄUDE, RaumNr,
  Nutzung)
  RESERVIERUNGEN((Adresse, RaumNr) → RÄUME,
  Tag, Zeit)

- Aufgabe: Muss "Adresse" in "RESERVIERUNGEN" zusätzlich als Fremdschlüssel deklariert werden, der "GEBÄUDE" direkt referenziert?

Schwache Entities (4)

Übersetzungs-Reihenfolge:

- Während sonst die Reihenfolge der Übersetzung der Entity-Typen egal ist, muss bei schwachen Entities
  - ◇ zuerst der übergeordnete Entity-Typ übersetzt sein (so dass sein Schlüssel bekannt ist),
  - ◇ bevor dann der untergeordnete Entity-Typ übersetzt werden kann.
- Dies erklärt auch, warum keine Zyklen in schwachen Entity-Beziehungen möglich sind.

Schwache Entities (5)

Übersetzungs-Reihenfolge, Forts.:

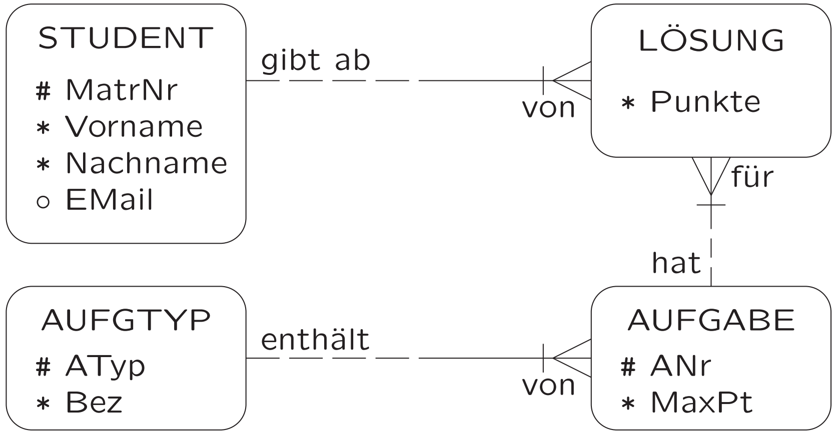
- Normale Beziehungen werden anschließend übersetzt (nachdem alle Entity-Typen übersetzt sind). Die Reihenfolge ist wieder beliebig.

  Dies gilt auch für normale Relationships, an denen schwache Entity-Typen beteiligt sind.
- Nachdem das Schema übersetzt ist, wird man versuchen, die "`CREATE TABLE`"-Anweisungen so anzuordnen, das Fremdschlüssel nur auf vorher angelegte Tabellen verweisen.

  Geht nicht immer. Notfalls "`ALTER TABLE`" / "`CREATE SCHEMA`", s. Kap. 10.

Schwache Entities (6)

Association-Entities:



Schwache Entities (7)

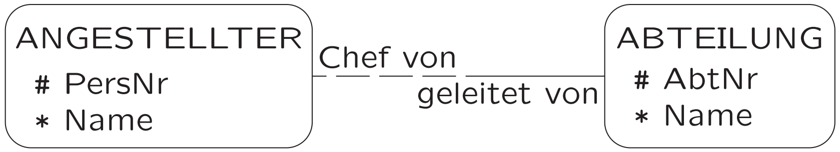
Association-Entities, Forts.:

- Assoziation Entity-Typen (im Beispiel "LÖSUNG") erben Schlüsselattribute von mehr als einer Quelle:

  STUDENTEN(<u>MatrNr</u>, Vorname, Nachname, EMail°)
  AUFGTYPEN(<u>ATyp</u>, Bez)
  AUFGABEN(<u>ATyp</u> → AUFGTYPEN, <u>ANr</u>, MaxPt)
  LÖSUNGEN(<u>MatrNr</u> → STUDENTEN, (<u>ATyp</u>,<u>ANr</u>) → AUFGABEN, Punkte)
- Dies würde einer viele-zu-viele-Beziehung mit einem Attribut "Punkte" entsprechen.

Inhalt

Eins-zu-eins-Beziehungen (1)



- Eine Beziehung ist eins-zu-eins, wenn sie auf keiner Seite einen Krähenfuß hat.
- Die Übersetzung ist zunächst die gleiche wie für eins-zu-viele-Beziehungen.

  Aber es wird ein zusätzlicher Schlüssel konstruiert, siehe unten.

Eins-zu-eins-Beziehungen (2)

- In diesem Beispiel ist es besser, den Schlüssel von ANGESTELLTER in die Tabelle ABTEILUNGEN zu übernehmen, als umgekehrt, da auf der Abteilungsseite die Minimalkardinalität 1 ist:

  ABTEILUNGEN(<u>AbtNr</u>, . . . ,
  Leiter → ANGESTELLTE)
- So werden Nullwerte vermieden, und die Minimumkardinalität 1 gewährleistet.

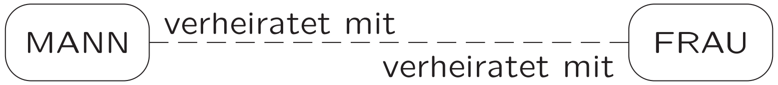
  Würde man den Namen der Abteilung in die Angestellten-Tabelle aufnehmen, so könnte er Null sein. Das ist problematisch für den zusätzlichen Schlüssel, der aus der 1:n eine 1:1-Beziehung macht (s.u.). Außerdem IB nötig für "jede Abteilung hat Leiter" (Min. Kard. 1).

Eins-zu-eins-Beziehungen (3)

- Bisher ist das die Übersetzung für 1:n-Beziehungen.

  Sind allgemeiner als 1:1, und lassen auch alle 1:1-Zustände zu.
- Weil es eine 1:1-Beziehung ist, ist "Leiter" ist nun auch Schlüssel für die Tabelle "ABTEILUNGEN":
  - ◇ Ein Angestellter kann maximal Leiter einer Abteilung sein.
  - ◇ Es gibt keine zwei Zeilen mit gleicher PersNr in Spalte "Leiter".
- "Leiter" ist nur ein Alternativschlüssel, nicht Teil des Primärschlüssels.

Eins-zu-eins-Beziehungen (4)



- Der Schlüssel einer der beiden Tabellen wird als (optionaler) Fremdschlüssel in die andere Tabelle übernommen.

  Es wäre aber falsch, beides zu tun (Redundanz).

- Oder man bildet eine eigene Tabelle (für die Bez.):

  Heirat(MName → Mann, FName → Frau)

- Übung: Was ist der/die Schlüssel?

Eins-zu-eins-Beziehungen (5)

- Um die minimale Kardinalität 1 auf beiden Seiten zu gewährleisten, müssen die Tabellen zu einer Tabelle zusammengefasst werden.

  KundeKarte(KdNr, Name, <u>KartenNr</u>, KreditLimit)

- KdNr und KartenNr sind beides Schlüssel.

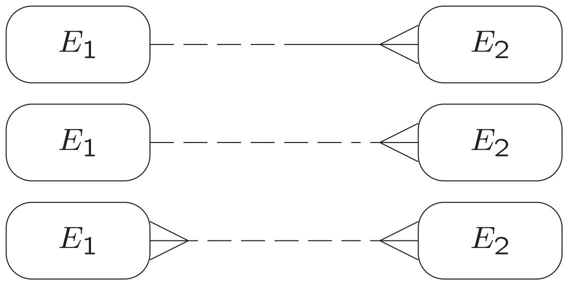
  Einer wird Primärschlüssel, der andere Alternativschlüssel.

Inhalt

1. Ziele des logischen Entwurfs
2. Grundlegende ER-Konstrukte
3. Schwache Entities
4. Eins-zu-Eins-Beziehungen
5. Letzte Schritte, Einschränkungen

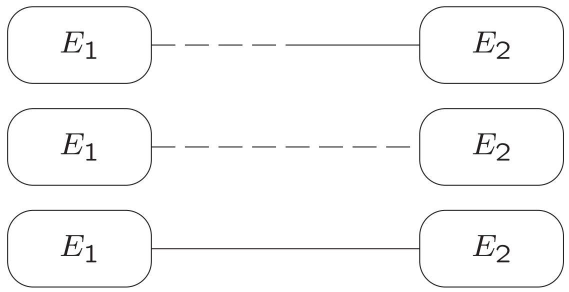
Einschränkungen (1)

- Folgende Kardinalitäten können mit den oben genannten Methoden übersetzt werden (wobei nur die Standard-Constraints des relationalen Modells verwendet werden):



Einschränkungen (2)

- Zusätzlich können alle Arten von eins-zu-eins-Beziehungen behandelt werden:



Einschränkungen (3)

- Trifft auf eine Beziehung keiner dieser Fälle zu, ist eine allgemeine Integritätsbedingung nötig, die auf eine der folgenden Arten implementiert wird:
  - ◇ Überprüfungen in Anwendungsprogrammen, die zur Einfügung von Daten verwendet werden.
  - ◇ Überwachung über Trigger.

    Trigger sind in der DB gespeicherte Prozeduren, die automatisch, z.B. für jedes eingefügte/modifizierte Tupel, ausgeführt werden.
  - ◇ Updates nur über "stored procedures" in der DB.
  - ◇ SQL-Anfragen, die von Zeit zu Zeit ausgeführt werden, um IB-Verletzungen zu suchen.

Schritt 5: Überprüfung (1)

- Zum Schluss werden die erstellten Tabellen überprüft, um festzustellen, ob sie Sinn machen.
- Z.B. kann man die Tabelle mit einigen Beispielzeilen füllen.
- Ist ein korrektes ER-Schema korrekt in das relationale Modell übersetzt, so erhält man ein korrektes relationales Schema.
- Trotzdem kann die manuelle Übersetzung Fehler mit sich bringen, und das ER-Schema kann versteckte Fehler enthalten.

Schritt 5: Überprüfung (2)

- Manchmal sind Tabellen redundant und können gelöscht werden.
- Man sollte ein letztes Mal über die Umbenennung von Tabellen und Attributen nachdenken.
- Wenn zwei Tabellen den gleichen Schlüssel haben, sollte man überlegen sie zu verschmelzen (das bedeutet aber nicht, dass man es immer tun muss!).
- Außerdem überprüft man die erstellten Tabellen auf relationale Normalform (z.B. 3NF, BCNF, 4NF) (vgl. Kapitel 11).