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 IIA: 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.

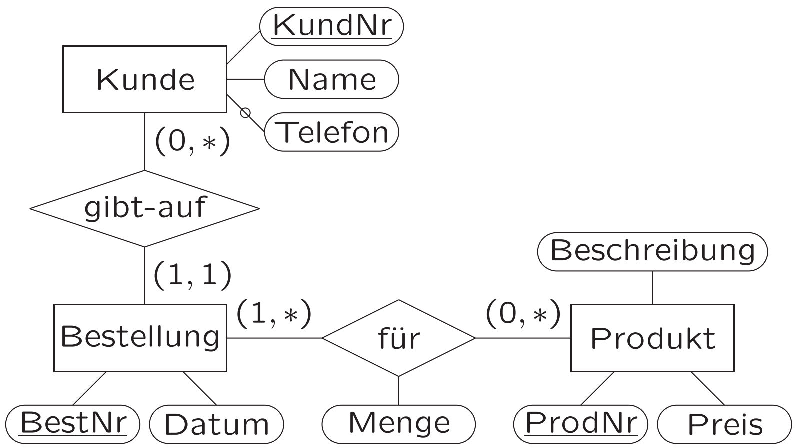
Logischer Entwurf: Ziel (2)

- 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



1. Schritt: Entity-Typen (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 1:1 in Spalten der Tabelle übersetzt.

  Falls das Attribut optional ist, erlaubt die Spalte Nullwerte.

- Primär-/Alternativschlüssel des Entity-Typs werden 1:1 in Schlüssel der Tabelle übersetzt.

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

1. Schritt: Entity-Typen (2)

Kunden

KundNr	Name	Telefon
10	Meier	624-9404
11	Müller	

Bestellungen

BestNr	Datum
200	23.11.2005
201	24.11.2005

Produkte

ProdNr	Beschreibung	Preis
1	Apfel	0.50
2	Kiwi	0.25
3	Orange	0.60

2. Schritt: 1:n Beziehungen (1)

- Hat eine Beziehung (ein Relationship) die maximale Kardinalität 1 auf einer Seite, so ist es eine eins-zu-viele (1:n) Beziehung.

  Hat es die maximale Kardinalität 1 auf beiden Seiten, so ist es eigentlich eine eins-zu-eins-Beziehung (siehe unten).

- Z.B. "gibt-auf" von "Kunde" zu "Bestellung".
- Dann wird der Schlüssel der "eins"-Seite (Kunde) als Spalte zur "viele"-Seite (Bestellung) zugefügt.
- Diese Spalte wird ein Fremdschlüssel, der die Zeile des entsprechenden Entities referenziert.

2. Schritt: 1:n Beziehungen (2)

- Ergebnis im Beispiel:

  `Bestellungen(`<u>`BestNr`</u>`, Datum, KundNr→Kunden)`

Bestellungen

BestNr	Datum	KundNr
200	23.11.2005	11
201	24.11.2005	11

Kunden

KundNr	Name	Telefon
10	Meier	624-9404
11	Müller	

2. Schritt: 1:n Beziehungen (3)

- Man darf die beiden Seiten nicht verwechseln, also z.B. nicht umgekehrt die Nummer der Bestellung als Fremdschlüssel in die Kundentabelle einfügen.
- Das würde nicht funktionieren, weil ein Kunde mehrere Bestellungen haben kann.

  Man wüßte dann nicht, welcher Wert in den Fremdschlüssel eingetragen werden soll (man brauchte ein mengenwertiges Attribut).
- Es muß daher immer die Seite mit der maximalen Kardinalität 1 den Fremdschlüssel bekommen.

  Dann gibt es nur ein Entity der anderen Seite, zu dem eine Beziehung besteht. Dessen Schlüsselwert wird im Fremdschlüssel eingetragen.

2. Schritt: 1:n Beziehungen (4)

- Ist die minimale Kardinalität 1 (wie im Beispiel), 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 minimale Kardinalität dagegen 0, so sind Nullwerte in der Fremdschlüsselspalte erlaubt.

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

2. Schritt: 1:n Beziehungen (5)

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

  Bestellungen(<u>BestNr</u>, Datum,
  aufgegeben_von→Kunden)
- 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.

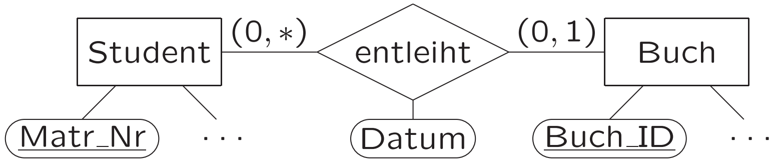
Namenskonflikte von Spalten

- 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.

  Dies passiert z.B., wenn sowohl der Schlüssel der Bestellungen, als auch der der Kunden einfach “`Nr`” oder “`ID`” heißen würde. Dann muß man bei Hinzufügung der Kundennummer zur Bestellungstabelle eine Umbennenung vornehmen. Wenn man Relationship-Namen als Fremdschlüssel-Namen verwendet, treten solche Probleme seltener auf.
- Selbstverständlich müssen solche Umbenennungen gut dokumentiert werden.

2. Schritt: Beziehungsattribute

- Beziehungen können Attribute haben, z.B.:



- Diese Attribute werden zusammen mit dem Fremdschlüssel für das Relationship gespeichert.

  `Bücher(`<u>`Buch_ID`</u>`, ..., Matr_Nr`^{o}`→Studenten, Datum`^{o}`)`

  "Matr_Nr" und "Datum" können Null sein, da nicht jedes Buch ausgeliehen wird, aber sie können nur zusammen Null oder nicht Null sein.
  Integritätsbedingung: ∀ `bücher B: B.matr_Nr is null` ↔ `B.datum is null`
  Solche IBen lassen sich als `CHECK`-Constraint in SQL formulieren.

2. Schritt: Eine Variante

- Eins-zu-viele-Beziehungen mit Kardinalität (0, 1) können in eine eigene Tabelle übersetzt werden.

  `entliehen_von(`<u>`Buch_ID`</u>`→Bücher,`
  `Matr_Nr→Studenten, Datum)`

- Die Schlüssel der beiden beteiligten Entities und die Beziehungsattribute werden in einer Tabelle gespeichert. Das Schlüsselattribut der Seite mit der (0, 1)-Kardinalität wird Schlüssel der Relation.

  Jedes Buch kann zu einer Zeit nur ein Mal ausgeliehen werden.

- Dies funktioniert mit der (1,1)-Kardinalität nicht.

3. Schritt: n:m Beziehungen (1)

- Eine Beziehung ist viele-zu-viele (n:m), wenn sie die maximale Kardinalität ∗ auf beiden Seiten hat (z.B. "für" im Beispiel).
- Viele-zu-viele-Beziehungen werden in eigene Tabellen übersetzt.
- Die Spalten der Tabelle sind die Schlüssel der teilnehmenden Entity-Typen.
  Sie bilden zusammen den Schlüssel dieser Tabelle.
- Diese Spalten sind zugleich auch Fremdschlüssel, die die Tabellen der Entity-Typen referenzieren.

3. Schritt: n:m Beziehungen (2)

- Beziehungsattribute werden als Spalten hinzugefügt. Sie sind nicht Teil des Schlüssels, z.B.

  `für(`<u>`BestNr`</u>`→Bestellungen, `<u>`ProdNr`</u>`→Produkte,`
  `Menge)`

- Man beachte, daß der Schlüssel von "für" wirklich aus "BestNr" und "ProdNr" bestehen muss.

  Da eine Bestellung mehrere Produkte beinhalten kann, kann "BestNr" allein nicht Schlüssel sein.
  Da das gleiche Produkt in verschiedenen Bestellungen auftreten kann, reicht auch "ProdNr" allein als Schlüssel nicht aus.

- Tabellen können umbenannt werden. Z.B. ist "`Bestell_Details`" ein besserer Name als "`für`".

3. Schritt: n:m Beziehungen (3)

für		
BestNr	ProdNr	Menge
200	1	1
200	2	1
201	1	5

Bestellungen		
BestNr	Datum	KundNr
200	23.11.2005	11
201	24.11.2005	11

Produkte		
ProdNr	Beschreibung	Preis
1	Apfel	0.50
2	Kiwi	0.25
3	Orange	0.60

3. Schritt: n:m Beziehungen (4)

- Eine minimale Kardinalität $\neq 0$ (z.B. 1) kann bei einer viele-zu-viele-Beziehung nicht durch Standard-IBen des relationalen Modells sichergestellt werden.
- Obige Übersetzung stellt nicht sicher, daß jede Bestellung mindestens ein Produkt enthält.
- Man braucht noch folgende Integritäts-Bedingung:

  $\forall$ **bestellungen** B: $\exists$ **für** F: `B.bestnr` $=$ `F.bestnr`

  Zu jeder Bestellung gibt es einen Eintrag in "`für`" (und damit auch in `Produkt` wegen des Fremdschlüssels).

3. Schritt: n:m Beziehungen (5)

- 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.
- Man kann diese Bedingung später in den Anwendungsprogrammen überprüfen.

  Sie kann nur in der `CREATE TABLE`-Anweisung nicht deklarativ festgelegt werden.

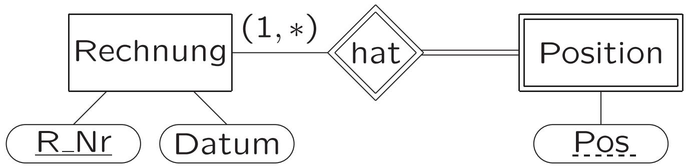
Zusammenges. Fremdschlüssel

- Ein zusammengesetzter Fremdschlüssel wird verwendet, um eine Tabelle mit einem zusammengesetzten Schlüssel zu referenzieren.

  Vorlesung(<u>Nr</u>, Titel, (VName, NName) $\rightarrow$ Dozent)
- Wäre die Minimum-Kardinalität 0, könnten "VName" und "NName" Null sein, aber nur zusammen.

Inhalt

Schwache Entities (1)



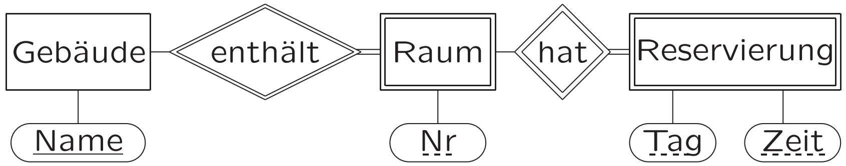
- Wird ein schwaches Entity übersetzt, müssen die Schlüsselattribute des Master-Entities als Fremdschlüssel und Teil des Schlüssels zugefügt werden:

  Position(R_Nr → Rechnung, Pos, . . .)

- Das implementiert automatisch die Beziehung.

  Eine solche Beziehung muss in Schritt 2 ignoriert werden. Es ist sinnvoll, hier "DELETE CASCADES" für den Fremdschlüssel zu spezifizieren. Beachte: Im relationalen Modell kein gestricheltes Unterstreichen.

Schwache Entities (2)

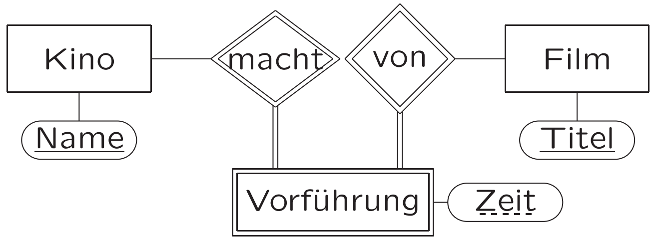


- Bei Hierarchien von schwachen Entities erben alle untergeordneten Entities die Schlüsselattribute:

  Gebäude(Name)
  Räume(Name → Gebäude, Nr)
  Reservierungen((Name, Nr) → Räume, Tag, Zeit)

- Muss Name in Reservierungen als Fremdschlüssel deklariert werden, der Gebäude referenziert?

Schwache Entities (3)



- Assoziation Entities erben Schlüsselattribute von mehr als einer Quelle:

  Kinos(Name)
  Filme(Titel)
  Vorführungen(Name → Kinos, Titel → Filme, Zeit)

Inhalt

Schritt 4: Eins-zu-Eins-Bez. (1)

- Eine Beziehung ist eins-zu-eins, wenn sie die maximale Kardinalität 1 auf beiden Seiten hat.
- Die Übersetzung ist zunächst die gleiche wie für eins-zu-viele-Beziehungen.

Aber es wird ein zusätzlicher Schlüssel konstruiert, siehe unten.

Schritt 4: Eins-zu-Eins-Bez. (2)

- In diesem Beispiel ist es besser, den Schlüssel von Angestellter in die Tabelle Abteilung zu übernehmen, als umgekehrt, da die Abteilung die Kardinalität (1,1) hat:

  Abteilung(<u>AbName</u>, . . . , Leiter → Angestellter)

- 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. Schlüssel mit Nullwerten sind problematisch. Außerdem wäre dann ein allgemeiner Constraint erforderlich, um zu sichern, daß jede Abteilung einen Leiter hat.

Schritt 4: Eins-zu-Eins-Bez. (3)

- "Leiter" ist nun auch Schlüssel für die Tabelle "Abteilung" (!), da ein Angestellter maximal Leiter einer Abteilung sein kann.
- "Leiter" ist nur ein Alternativschlüssel, nicht Teil des Primärschlüssels.
- Das sichert die maximale Kardinalität 1 auf der Angestellten-Seite.

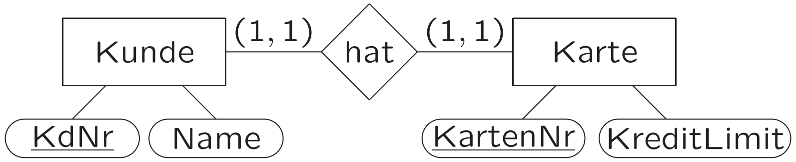
Schritt 4: Eins-zu-Eins-Bez. (4)

- 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?

Schritt 4: Eins-zu-Eins-Bez. (5)



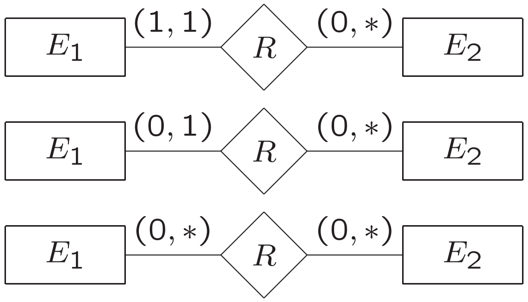
- Um die minimale Kardinalität 1 auf beiden Seiten zu gewährleisten, müssen die Tabellen zu einer Tabelle zusammengefasst werden.

  KundeKarte(KdNr, Name, <u>KartenNr</u>, KreditLimit)
- KdNr und KartenNr sind beides Schlüssel. Einer wird als Primärschlüssel ausgewählt, der andere ist Alternativschlüssel.

Inhalt

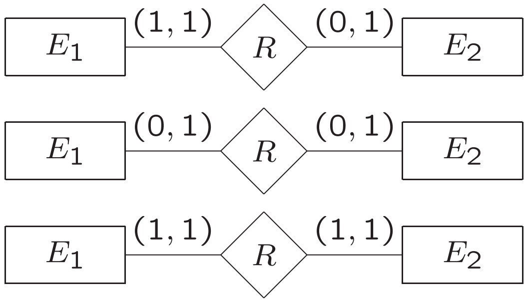
Einschränkungen (1)

- 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:



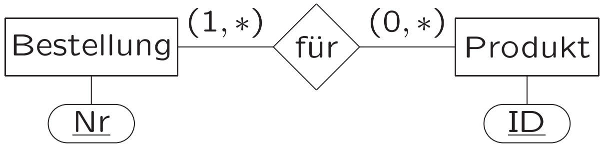
Einschränkungen (3)

- 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.

Einschränkungen (4)

- Speziell ist die Kardinalität $(1, *)$ manchmal sinnvoll, z.B. sollte jede Bestellung mindestens ein Produkt umfassen:



- Die Minimumkardinalität 1 kann in den aktuellen DBMS nicht deklarativ gesichert werden.

  Man verwendet stattdessen die gleiche Übersetzung wie für $(0, *)$ und spezifiziert zusätzlich einen allgemeinen Constraint.

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).