
Datenbanken und Informationssysteme

Material for CLIL included

Kurt Hillebrand

September 2014

3 Vom Konzeptionellen Datenmodell zur Implementierung in Tabellen

- Vorbemerkung: In diesem Kapitel können die Begriffe Datei - Satz - Feld analog zu den Begriffen Tabelle - Zeile - Spalte verwendet werden

3.1 Identifizierung von Entities

- Definitionsgemäß müssen Entities von einander unterscheidbar und damit identifizierbar sein.
 - Superschlüssel (Superkey, Oberschlüssel, Identifizierende Attributkombination): Menge von Attributen, bei denen sich je zwei Entities des Entity-Typs in mindestens einem Attributwert unterscheiden.
 - Schlüssel(kandidat) (Key, Candidate Key): Superschlüssel, der minimal ist (d.h. es kann kein Attribut weggelassen werden, ohne dass die identifizierende Eigenschaft verloren geht). Entweder hat ein Entity-Typ auf Grund der bisherigen Erhebungen mindestens einen Schlüssel oder ein eigener Schlüssel muss nun eingeführt werden (das muss immer möglich sein, da Entities definitionsgemäß voneinander unterscheidbar sein müssen)
 - NULL-Wert: Speziell definierter Wert, der bei jedem Attribut auftreten kann und bedeutet, dass bei diesem Entity keine Information über den Wert des Attributs vorhanden ist (Nicht zu verwechseln mit den Werten 0, " oder 'Null' eines Attributs).
 - Entity-Integrität: Bedingung, dass es für jeden Entity-Typ (mindestens) einen Schlüssel geben muss, in dessen Attributwerten kein NULL-Wert auftritt.
 - Einfacher Schlüssel: Schlüssel besteht aus einem Attribut
 - Zusammengesetzter Schlüssel (Concatenated Key): Schlüssel besteht aus mehreren Attributen
 - Teilschlüssel: Ein oder mehrere Attribute eines Zusammengesetzten Schlüssels
 - Primärschlüssel (Primary Key): Ein Schlüssel(kandidat) wird als Primärschlüssel definiert. Die anderen Schlüssel(kandidaten) werden als Alternate Keys bezeichnet.
 - Regeln für die Wahl des Primärschlüssels:
 - Der Primärschlüsselwert muss beim Hinzufügen eines Entities vollständig bekannt sein (Entity-Integrität)
 - Da der Primärschlüssel das Entity während der ganzen Lebenszeit identifiziert, darf sein Wert nicht geändert werden müssen (dadurch würde eigentlich ein neues Entity entstehen).
 - Der Primärschlüssel soll möglichst wenige Attribute umfassen, die Darstellung der Attributwerte soll möglichst kurz sein.
 - Die Attribute, die den Primärschlüssel darstellen, werden durch Unterstreichen kenntlich gemacht (bei Bedarf auch im Entity-Relationship-Diagramm):
Angestellter (Personalnummer, Name, Gehalt, Geburtsdatum)
 - Künstlicher Schlüssel (Surrogatschlüssel): Zur einfacheren Identifizierbarkeit von Entities wird oft auch ein zusätzliches, künstliches Attribut (meist fortlaufende Nummer) als Primärschlüssel eingeführt.
Beispiele: Personalnummer, Teilenummer
 - Der Künstliche Schlüssel muss beim Anlegen des Entities vom System vergeben werden können (in Ausnahmefällen auch Wahl durch den Benutzer); ein neuer Schlüssel soll daher einfach ermittelt werden können (z.B. fortlaufend nummerieren)
 - Beim Zugriff auf ein Entity braucht der Benutzer diesen (meist nicht sprechenden) Schlüssel nicht zu wissen, sondern muss über Attributwerte, die ihm bekannt sind, ein Entity auswählen können.
 - Vollständigkeitskontrolle ist bei einer fortlaufenden Nummerierung leicht möglich.
 - Alle Regeln für die Wahl des Primärschlüssels sind eingehalten.
 - Viele Datenbanksysteme sehen einen eigenen Datentyp für die automatische, fortlaufende (+1) Nummerierung vor (z.B. ZÄHLER, SERIAL, IDENTITY).
- Beispiel:
Änderung einer Abteilungsbezeichnung ist einfach möglich, wenn statt einer Abteilungsbezeichnung eine abstrakte Abteilungsnummer als Primärschlüssel verwendet wird.
- Beispiel: Chemische Elemente haben mehrere Schlüssel(kandidaten), nämlich Name (z.B. Eisen), Chemisches Zeichen (z.B. Fe), Ordnungszahl (z.B. 26), Atomgewicht (z.B. 55.847), etc.
Übung: Was würde man als Primärschlüssel wählen?

- Beispiel: Attribute des Entity-Typs Angestellter

Personal#	Vorname	Nachname	Geburtsdatum	
x	x	x		Superschlüssel
x			x	Superschlüssel
	x	x	x	Schlüssel (eventuell nicht eindeutig)
x				Schlüssel / Primärschlüssel

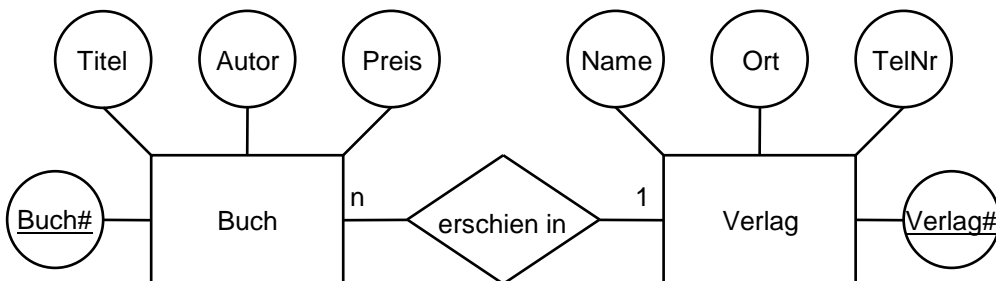
- Beispiel: Aus der Personalverwaltung

Möglicher Primärschlüssel	Vorteil	Nachteil
Personalnummer	kurz	(nicht sprechend)
Geburtsdatum+laufende Nummer		(nicht sprechend)
Familiennamen+laufende Nummer	(sprechend)	kann sich ändern (z.B. Heirat)
Sozialversicherungsnummer	muss ohnehin geführt werden	bei Eintritt eventuell (noch) nicht bekannt
Eintrittsdatum+laufende Nummer	Eintrittsdatum muss ohnehin geführt werden	kann sich bei Wiedereintritt ändern
Abteilungsnummer+laufende Nummer		kann sich ändern

- Beispiel: Kennzeichen als Primärschlüssel für ein Auto ist nicht gut geeignet, da ein Auto im Laufe der Zeit verschiedene Kennzeichen haben kann. Außerdem bringt die Verwendung von Wechselkennzeichen Probleme mit sich.
- Beispiel: Wie beurteilen Sie die Wahl von Telefonnummer als Primärschlüssel für eine Firma?

3.2 Redundanz und Fremdschlüssel

- Redundanz: Mehrmaliges Abspeichern desselben Sachverhalts.
- Ein Ziel des Datenentwurfs soll natürlich sein, Redundanz möglichst zu vermeiden ('One fact at one place')
- Beispiel 1):
Folgendes ER-Diagramm ist gegeben:



Buch (Buch#, Titel, Autor, Preis)

Verlag (Verlag#, Name, Ort, TelNr)

In einer Tabelle wird für jedes Buch eine Zeile abgespeichert. Die Information über den zugehörigen Verlag wird ebenfalls in dieser Zeile festgehalten.

Buch+Verlag

<u>Buch#</u>	Titel	Autor	Preis	Verlag#	Name	Ort	TelNr
17	999 Zaubertränke	Miraculix	150	5	Styria	Graz	+43 316 70630
23	Römer für Anfänger	Asterix	130	3	Oldenburg	Wien	+43 1 727258
48	Gallische Superdiät	Obelix	120	5	Styria	Graz	+43 316 70630
70	Selbstverteidigung	Asterix	240	7	Manz	Wien	+43 1 533171
151	Handbuch des Druiden	Miraculix	350	5	Styria	Graz	+43 316 70630
258	Tarnen und Täuschen	Asterix	130	5	Styria	Graz	+43 316 70630

Redundanz: Die Tatsache, dass ein Verlag mit einer bestimmten Nummer den entsprechenden Namen hat, am entsprechenden Ort ist und die entsprechende Telefonnummer hat, wird mehrmals (nämlich bei jedem Buch, das der Verlag herausgebracht hat) abgespeichert.

- Beispiel 2):
In einem Bestellsystem werden pro Bestellung ein Kopf (Bestellnummer, Lieferant, Gesamtwert der Bestellung,...) und die Bestellpositionen (Artikel, Menge, Preis,...) abgespeichert.
Redundanz: Die Tatsache, dass die Bestellung einen bestimmten Wert hat, ist zweimal festgehalten (nämlich als Gesamtwert der Bestellung abgespeichert und als Summe der Positionswerte errechenbar).
- Hauptprobleme bei redundanten Daten:
 - Speicherplatzverbrauch
 - Änderungsanomalie (Update Anomaly / Dependency): Tatsache, dass bei Führung redundanter Daten die Änderung einer Tatsache an mehreren Stellen vorgenommen werden muss.
Beispiel 1): Änderung der Telefonnummer des Styria-Verlages
 - Zeitverbrauch: Auf Grund der Änderungsanomalie.
 - Inkonsistenz (Inconsistency) der Daten: Die Daten sind in einem, in sich logisch falschen, Zustand; zu einem Sachverhalt liegen widersprüchliche Informationen vor.
Dateninkonsistenz kann auf Grund der Änderungsanomalie auftreten.
Beispiel 1): Änderung des Ortes beim Styria-Verlag. Aus verschiedenen Gründen (Programmierfehler bis Hardwarefehler) wird der Ort nicht in allen betroffenen Sätzen geändert.
 - Einfüge- und Löschanomalien (Insertion, Deletion Anomaly / Dependency): Vor dem Einfügen des ersten und nach dem Löschen der letzten Zeile, die die (potentiell) redundante Information enthält, ist diese Information nicht vorhanden.
Beispiel 1): Bevor in einem Verlag das erste Buch erscheint, existieren die Daten über den Verlag noch nirgends; nachdem ein Buch aus dem Sortiment genommen wird (z.B. Buch# 70), sind die Daten über den Verlag verloren (Umgehung mit fiktiven Buchnummern wäre in diesem Fall möglich, diese sind aber bei der Verarbeitung der Buchinformationen immer zu berücksichtigen).
 - Zugang zu den redundanten Informationen ist umständlich und zeitaufwendig.
Beispiel 1): Liste aller Verlage.
- Lösung zu Beispiel 1):
In einer Tabelle 'Verlag' wird für jeden Verlag eine Zeile abgespeichert. In einer Tabelle 'Buch' wird für jedes Buch eine Zeile abgespeichert. Um festhalten zu können, in welchem Verlag das Buch erschien, wird in dieser Tabelle noch eine Spalte definiert in dem jeweils der Primärschlüsselwert des Verlages abgespeichert wird.

Buch

Buch#	Titel	Autor	Preis	Verlag#
17	999 Zaubertränke	Miraculix	150	5
23	Römer für Anfänger	Asterix	130	3
48	Gallische Superdiät	Obelix	120	5
70	Selbstverteidigung	Asterix	240	7
151	Handbuch des Druiden	Miraculix	350	5
258	Tarnen und Täuschen	Asterix	130	5

Verlag

Verlag#	Name	Ort	TelNr
3	Oldenburg	Wien	+43 1 727258
5	Styria	Graz	+43 316 70630
7	Manz	Wien	+43 1 533171

- Fremdschlüssel (Foreign Key, External Key): Spalte (oder Spalten) einer Tabelle, die in einer anderen Tabelle den Primärschlüssel bildet (oder bilden).
Beispiel 1): Die Spalte 'Verlag#' in der Tabelle 'Buch' ist ein Fremdschlüssel (weil diese Spalte in der Tabelle 'Verlag' Primärschlüssel ist).
- Detailtabelle (Childtable): Tabelle, die den Fremdschlüssel enthält (bei 1:n-Bezeichnung auf der n-Seite)
- Mastertabelle (Parenttable): Tabelle, die nicht den Fremdschlüssel enthält (bei 1:n-Bezeichnung auf der 1-Seite)
- Referentielle Integrität (Referential Integrity): Bedingung, dass jeder Fremdschlüsselwert als Primärschlüsselwert existieren muss oder der NULL-Wert ist.
- Maßnahmen, um die Referentielle Integrität sicherzustellen:
 - Bei jeder Aufnahme oder Änderung eines Fremdschlüsselwerts muss überprüft werden, ob dieser Wert als Primärschlüsselwert existiert.
Beispiel 1): Wenn der Wert der Spalte 'Verlag#' in der Tabelle 'Buch' aufgenommen oder geändert wird, muss überprüft werden, ob dieser Wert in der Spalte 'Verlag#' der Tabelle 'Verlag' existiert.
 - Beim Löschen einer Zeile aus der Tabelle, wo der Fremdschlüsselwert als Primärschlüsselwert existiert, muss überprüft werden, ob der Primärschlüsselwert nirgends als Fremdschlüsselwert auftritt.
Beispiel 1): Beim Löschen einer Zeile aus der Tabelle 'Verlag' muss überprüft werden, ob der Wert der Spalte 'Verlag#' in keiner Zeile der Tabelle 'Buch' als Wert der Spalte 'Verlag#' auftritt.

- Bemerkung zu Beispiel 2):
Führen des Gesamtwertes der Bestellung im Bestellkopf kann sinnvoll sein, wenn
 - die Gesamtwerte der Bestellungen oft gelesen, jedoch vergleichsweise selten geändert werden und
 - sichergestellt ist, dass es zu keinen Inkonsistenzen kommt (Transaktionen, Angabe der Integritätsbedingung bei der Definition der Datenbank, Triggers).
 Der zusätzliche Speicherplatzbedarf für den Gesamtwert ist in diesem Fall vertretbar.

3.3 Implementierung von ER-Diagrammen in Tabellen

- Überführung des Konzeptionellen Datenmodells in das Relationale Datenmodell
- Entity-Typ: Für jeden Entity-Typ E wird eine Tabelle T angelegt
 - Für jedes Attribut wird eine entsprechende Spalte definiert
 - Jedes Entity wird durch eine Zeile repräsentiert
 - Die Attributwerte werden in den entsprechenden Spalten der Zeilen (Zellen) als Inhalt abgelegt
- 1:1-Beziehungs-Typ:
 - T1 und T2 seien die beiden Tabellen, die die Entity-Typen darstellen, die am Beziehungs-Typ beteiligt sind
 - Die Tabelle T1 wird um Spalten erweitert, in denen der Primärschlüssel der Tabelle T2 als Fremdschlüssel geführt wird
 - Wenn das Entity, das durch die Zeile repräsentiert wird, an keiner Beziehung des Beziehungs-Typs beteiligt ist, enthält der Fremdschlüssel den NULL-Wert
 - Wenn ein Entity-Typ vollständig am Beziehungs-Typ teilnimmt, soll die entsprechende Tabelle als T1 gewählt werden
 - Da es sich um einen 1:1-Beziehungs-Typ handelt, muss sichergestellt werden, dass im Fremdschlüssel jeder Primärschlüsselwert höchstens einmal auftritt (abgesehen von NULL-Werten)
 - Hinweis: Wenn beide Entity-Typen vollständig am Beziehungs-Typ teilnehmen und sie nicht an anderen Beziehungs-Typen beteiligt sind, könnten beide Entity-Typen samt Beziehungs-Typ auch in einer gemeinsamen Tabelle dargestellt werden (die beiden Entities, die in Beziehung stehen, kommen jeweils in eine Zeile). Dieser Fall tritt jedoch sehr selten auf.
 - Beispiele:

Abteilung (<u>Abteilungs#</u> , ..., Personal# des Leiters)	Personal# des Leiters muss eindeutig sein
Mann (<u>Mann#</u> , ..., Frau# der Gattin)	Frau# der Gattin muss (abgesehen von NULL-Werten) eindeutig sein
Person (<u>Personen#</u> , ..., Personen# des Gatten)	Personen# des Gatten muss (abgesehen von NULL-Werten) eindeutig sein
- 1:n-Beziehungs-Typ:
 - T1 sei die Tabelle, die den Entity-Typ auf der n-Seite, T2 die Tabelle, die den Entity-Typ auf der 1-Seite des Beziehungs-Typs, darstellt
 - Die Tabelle T1 wird um Spalten erweitert, in denen der Primärschlüssel der Tabelle T2 als Fremdschlüssel geführt wird
 - Wenn das Entity, das durch die Zeile repräsentiert wird, an keiner Beziehung des Beziehungs-Typs beteiligt ist, enthält der Fremdschlüssel den NULL-Wert
 - Beispiele:

Angestellter (<u>Personal#</u> , ..., Abteilungs# zu der er gehört)
Projekt (<u>Projekt#</u> , ..., Personal# des Leiters)
Angestellter (<u>Personal#</u> , ..., Personal# des Vorgesetzten)
- m:n-Beziehungs-Typ:
 - T1 und T2 seien die beiden Tabellen, die die Entity-Typen darstellen, die am Beziehungs-Typ beteiligt sind
 - Es wird eine eigene Tabelle T erstellt, in der die Primärschlüssel der Tabellen T1 und T2 als Fremdschlüssel geführt werden - Assoziative Tabelle (Junction Table)
 - Der Primärschlüssel der Tabelle T ist ein zusammengesetzter Schlüssel, der aus den Spalten beider Fremdschlüssel besteht
 - Für jedes Attribut des Beziehungs-Typs wird die Tabelle T um eine Spalte erweitert
In diesem Fall muss (bei mehreren Beziehungen zwischen demselben Entity-Paar) der Primärschlüssel um ein oder mehrere Attribute (des Beziehungs-Typs) erweitert werden (so dass Eindeutigkeit erreicht wird) oder überhaupt ein künstlicher Schlüssel eingeführt werden (eher bei eigenem Entity-Typ)
 - Hinweis: Es könnten auch 1:1- und 1:n-Beziehungs-Typen auf diese Weise, in einer eigenen Tabelle, abgebildet werden. Dies ist im Besonderen dann sinnvoll, wenn sehr wenige Beziehungen zwischen den Entities existieren und daher die Spalten mit den Fremdschlüsseln sehr oft den NULL-Wert enthalten oder wenn NULL-Werte prinzipiell nicht verwedet werden sollen.
Es ist dann unbedingt sicherzustellen, dass diese Tabelle tatsächlich einen 1:1- oder 1:n-Beziehungstyp darstellt (Eindeutigkeit des / der entsprechenden Fremdschlüssels)

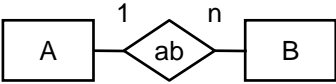
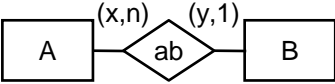
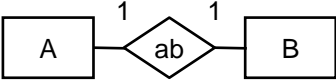
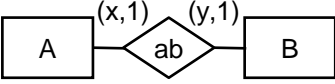
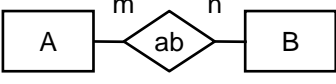
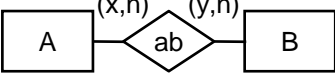
- Beispiele:
 - arbeitet an (Personal#, Projekt#, %-Zeitanteil)
 - benötigt (Projekt#, Bauteil#)
 - liefert (Bauteil#, Lieferanten#)
 - kauft (Kunden#, Artikel#, Stück)
 - hat besucht (Personen#, Schul#, Monate, Kosten)
 - ist Komponente von (Bauteil# der Komponente, Bauteil# des Zusammenbaus, Stück)
 - kauft (Kunden#, Artikel#, Datum, Stück, Rabatt) Annahme: An einem Tag bei gleichem Kunden und Artikel nur ein Rabatt
- Mehrwertiger Beziehungs-Typ:
 - Analog m:n-Beziehungs-Typ wird eine eigene Tabelle T erstellt, in der alle Primärschlüssel der Tabellen, die die am Beziehungs-Typ beteiligten Entity-Typen darstellen, als Fremdschlüssel geführt werden
 - Der Primärschlüssel der Tabelle T ist ein zusammengesetzter Schlüssel, der aus den Spalten aller Fremdschlüssel besteht
 - Für jedes Attribut des Beziehungs-Typs wird die Tabelle T um eine Spalte erweitert
In diesem Fall muss (bei mehreren Beziehungen zwischen demselben Entity-Tupel) der Primärschlüssel um ein oder mehrere Attribute (des Beziehungs-Typs) erweitert werden (so dass Eindeutigkeit erreicht wird) oder überhaupt ein künstlicher Schlüssel eingeführt werden (eher bei eigenem Entity-Typ)
 - Beispiele:
 - unterrichtet (Klassen#, Lehrer#, Gegenstands#, Stundenanzahl)
 - liefert (Bauteil#, Lieferanten#, Projekt#, Datum, Menge) Annahme: keine Teillieferungen an einem Tag
- Vollständiges Beispiel (siehe 2.3.3):

Angestellter (<u>Personal#</u> , sonstige Attribute von Angestellter, <u>Abteilungs#</u>) Abteilung (<u>Abteilungs#</u> , sonstige Attribute von Abteilung, <u>Personal#</u>) Projekt (<u>Projekt#</u> , sonstige Attribute von Projekt, <u>Personal#</u>) Bauteil (<u>Bauteil#</u> , sonstige Attribute von Bauteil) Lieferant (<u>Lieferanten#</u> , sonstige Attribute von Lieferant) arbeitet an (<u>Personal#</u> , <u>Projekt#</u>) benötigt (<u>Projekt#</u> , <u>Bauteil#</u>) liefert (<u>Bauteil#</u> , <u>Lieferanten#</u>) Bedingungen: <ul style="list-style-type: none"> - Wert von <u>Abteilungs#</u> in Angestellter darf nirgends NULL sein (min=1) - Wert von <u>Personal#</u> in Abteilung darf nirgends NULL sein (min=1) und derselbe Wert darf nur einmal auftreten (1:1-Beziehungs-Typ) - Wert von <u>Personal#</u> in Projekt darf nirgends NULL sein (min=1) und derselbe Wert darf höchstens 3 mal auftreten (max=3) - usw. 	Beziehungs-Typ: gehört zu Beziehungs-Typ: leitet Beziehungs-Typ: verantwortet
---	---

3.4 Zusammenfassung der Vorgangsweise

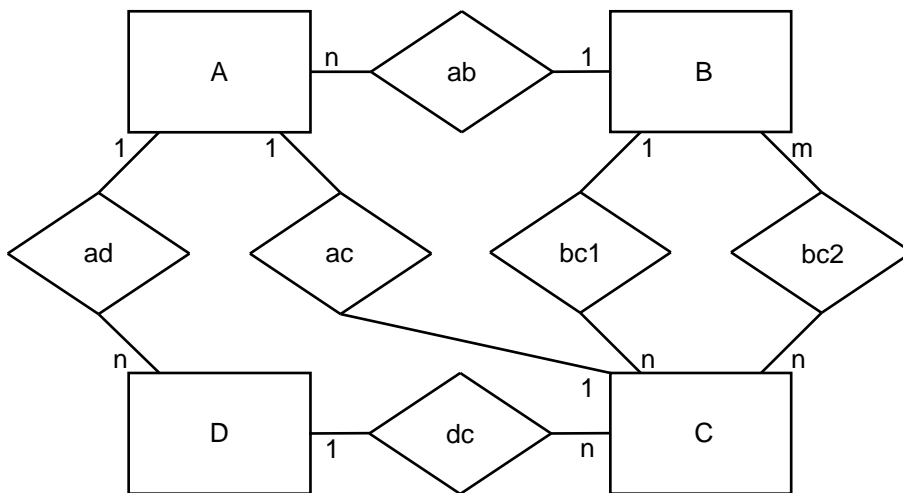
- Bestimme die Entity-Typen
- Bestimme die Beziehungs-Typen
- Bestimme die Komplexitätsgrade der Beziehungs-Typen (Art oder (min,max))
- Bestimme die Attribute von Entity-Typen und Beziehungs-Typen samt ihrer Wertebereiche (Domains)
- Bestimme die Primärschlüsseln
 - geeignete Kombinationen aus bestehenden Attributen
 - Künstliche Schlüssel
- Bestimme eventuelle zusätzliche, anwendungsspezifische Integritätsbedingungen (z.B. ein Angestellter ist für höchstens 3 Projekte verantwortlich)
- Wenn das System im Relationalen Datenmodell implementiert wird, so leite die benötigten Tabellen mit ihren Spalten (Zeilen Aufbau) ab
- Wenn das System mit Dateien implementiert wird, so leite die benötigten Dateien mit ihren Feldern (Satz Aufbau) ab

3.5 Übersicht: Vom ER-Diagramm zu den Tabellen

Beziehungskardinalität	(min,max)-Notation
 <p>A (<u>A#</u>, ...) B (<u>B#</u>, ..., A# > A)</p>	 <p>Tabellenstruktur siehe links</p> <p>x=0 -</p> <p>x=1 Jeder PK-Wert von A.A# muss als FK-Wert in B.A# mindestens einmal vorkommen, daher: insert A → (insert B oder update B.A#) delete B oder update B.A# → FK-Wert noch einmal vorhanden? (Zeile mit PK-Wert löschen???)</p> <p>y=0 FK-Wert darf NULL sein</p> <p>y=1 FK-Wert darf nicht NULL sein</p>
 <p>A (<u>A#</u>, ...) B (<u>B#</u>, ..., A# > A)</p> <p>oder</p> <p>B (<u>B#</u>, ...) A (<u>A#</u>, ..., B# > B)</p> <p>FK-Wert muss (abgesehen von NULL-Werten) eindeutig sein = 'UNIQUE except NULL'</p>	 <p>x=0 & y=0 Tabellenstruktur siehe links Wo kommen weniger NULL-Werte zustande?</p> <p>x=0 & y=1 A (<u>A#</u>, ...) B (<u>B#</u>, ..., A# > A) A# NOT NULL und UNIQUE</p> <p>x=1 & y=0 B (<u>B#</u>, ...) A (<u>A#</u>, ..., B# > B) B# NOT NULL und UNIQUE</p> <p>x=1 & y=1 Sehr selten, Designfehler?, Nur ein ET?</p>
 <p>A (<u>A#</u>, ...) B (<u>B#</u>, ...) AB (<u>A#</u> > A, <u>B#</u> > B)</p> <p>Beziehungsattribute zusätzlich in AB, dann PK von AB beachten</p> <p>Höherwertige BT analog</p>	 <p>Tabellenstruktur siehe links</p> <p>x/y=0 -</p> <p>x/y=1 Jeder PK-Wert von A / B muss als FK-Wert in AB mindestens einmal vorkommen, daher: insert A / B → insert AB delete AB → FK-Wert noch einmal vorhanden? (Zeile mit PK-Wert löschen???)</p>

- Beispiel

ERD:



Tabellen:

B (B#, ...)
 A (A#, ..., B# > B)
 D (D#, ..., A# > A)
 C (C#, ..., D# > D, B# > B, A# > A / UNIQUE except NULL)
 BC(B# > B, C# > C)

Die Reihenfolge der Tabellen-Definitionen ist zu beachten.

Für den 1:1-BT wurde A# als FK in C verwendet, andernfalls gibt es Probleme bei der Definitions-Reihenfolge.