

UML-Diagramme: Typische Fehler/Entity Relationship 1:1 (Klassennotation)

Wechseln zu:[Navigation](#), [Suche](#)

Dieser Artikel erfüllt die [GlossarWiki-Qualitätsanforderungen](#) nur teilweise:

| | | | | |
|---|--|--|---------------------------------|--|
| Korrektheit: 4 (größtenteils überprüft) | Umfang: 4 (unwichtige Fakten fehlen) | Quellenangaben: 3 (wichtige Quellen vorhanden) | Quellenarten: 3 (gut) | Konformität: 5 (ausgezeichnet) |
|---|--|--|---------------------------------|--|

Übersicht

| | |
|--------------------------------|---|
| Use-Cases-Diagramme: | Akteure und Aktionen , Include und Extend , Vererbung , Systemgrenzen |
| Entity-Relationship-Diagramme: | 1:1-Beziehungen , 1:n-Beziehungen , m:n-Beziehungen (Klassennotation) |
| | 1:1-Beziehungen , 1:n-Beziehungen , m:n-Beziehungen (Krähenfußnotation) |

Falsch

Prinzipiell sind zwischen zwei Tabellen (Entities und Relationships) alle Arten von Beziehungen möglich. Allerdings machen gerade Anfänger häufig Fehler bei der Wahl der Vielfachheiten (1; 0,1; 0..* ...). Die „fehlerhaften“ Beispiele können in bestimmten Situation korrekt sein. In den allermeisten Fällen liegt aber ein Fehler vor.

1:1-Beziehungen

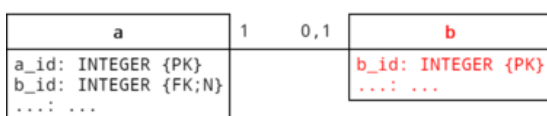


Fast korrekt! (Formal müsste man noch zwei Foreign Keys von a nach b und umgekehrt definieren.)

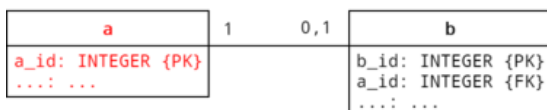
Aber: Eine Tabelle genügt, sofern es sich um echte Tabellen handelt!

1:1-Beziehungen zwischen zwei Views oder einer View und einer normalen Entity-Tabelle kommen dagegen häufiger vor.

1:0,1-Beziehungen

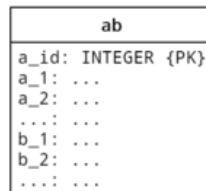


Schlecht: Es ist besser, den Foreign Key bei b einzufügen, da dessen Foreign-Key-Attribut nicht nullable ist und daher immer einen Wert beinhalten muss.



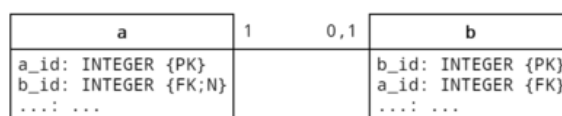
Besser, aber noch nicht ganz korrekt, da nicht verhindert werden kann, dass einem a-Tupel mehr als ein b-Tupel zugeordnet wird.

Richtig



Bei echten 1:1-Beziehungen reicht eine Tabelle aus. Vorteile: Man benötigt nur ein Identifikator-Attribut und spart sich bei den zugehörigen Abfragen einen Join.

Meistens ist die angegebene 1:1-Beziehung jedoch einfach falsch, sondern es ist eine andere Beziehung (1:n, m:n o. Ä.) gemeint.



Dieses Diagramm ist (fast) korrekt. In SQL muss allerdings einer der beiden Foreign Keys nachträglich mittels ALTER TABLE eingefügt werden, da ein Verweis auf eine noch nicht definierte Tabelle nicht erlaubt ist.



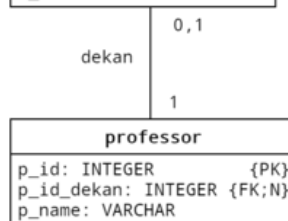
```
{CHECK:
  b_id = (SELECT a.b_i FROM a WHERE a.a_id = a_id)
}
```

Nun ist das Diagramm vollkommen korrekt, da sichergestellt ist, dass ein a-Tupel und ein b-Tupel jeweils gegenseitig aufeinander verweisen. Allerdings werden derart komplexe Check Constraints nicht von allen DBMS unterstützt.

Beispiel

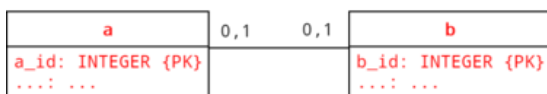


Jede Fakultät hat einen Dekan, jeder Professor ist Dekan von höchstens einer Fakultät.



```
{CHECK:
  p_id =
  (SELECT f.p_dekan_id
   FROM fakultaet f
   WHERE f.f_id =
         f_id_dekan
  )
}
```

0,1:0,1-Beziehung



Hier fehlen die Foreign Keys.



Analog zur 1:0,1-Beziehung benötigt man zwei Foreign Keys (und evtl. auch noch eine komplexe Check Constraint). In diesem Fall müssen allerdings beide Foreign Keys nullable sein.

1 Quellen

Kowarschick (MMDB): Wolfgang Kowarschick; Vorlesung „Multimedia-Datenbanksysteme“; Hochschule: [Hochschule Augsburg](#); Adresse: [Augsburg](#); [Web-Link](#); 2016; Quellengüte: 3 (Vorlesung)

Kowarschick (MMDB-Skript): Wolfgang Kowarschick; Vorlesung Multimedia-Datenbanksysteme – Sommersemester 2018; Hochschule: [Hochschule Augsburg](#); Adresse: [Augsburg](#); [Web-Link](#); 2018; Quellengüte: 4 (Skript)

[StarUML](#)

2 Siehe auch

[Unified Modeling Language](#)

Kategorie:

[UML](#)

Diese Seite wurde zuletzt am 14. Juni 2019 um 13:45 Uhr bearbeitet.

Inhalt verfügbar unter [CC BY-SA 4.0](#).

