

*publisherId* verweist von *books* auf *publishers* (siehe Abbildung 5.3). Die in dieser Spalte gespeicherten Werte geben an, welches Buch in welchem Verlag erschienen ist.

Solange Sie keine Foreign-Key-Regel definieren (Details dazu folgen gleich), unterscheiden sich Foreign-Key-Spalten nicht von anderen Spalten. Egal, ob Sie Ihre Tabellen mit SQL und CREATE TABLE erzeugen oder eine grafische Benutzeroberfläche verwenden – es gibt kein Schlüsselwort, keine spezielle Syntaxform und keine Option, mit der Sie eine Fremdschlüsselspalte kennzeichnen.

Oder, anders formuliert: Sie als Anwender(in) eines DBMS wissen zwar, dass die von Ihnen definierte Spalte später einmal die IDs enthalten wird, die auf andere Tabellen verweisen. Das DBMS weiß dies aber nicht! Es kann weder kontrollieren, ob der von Ihnen gewählte Datentyp für eine Foreign-Key-Spalte sinnvoll ist, noch, ob die in der Spalte gespeicherten Werte korrekt sind.

### Master- und Detailtabellen

Manche Autoren bzw. Entwicklungsumgebungen verwenden bei 1:n-Beziehungen die Begriffe *Mastertabelle* und *Detailtabelle*. Dabei ist die Mastertabelle das Ziel der Verknüpfung. Die Detailtabelle hat eine Foreign-Key-Spalte mit den Verweisen auf die Mastertabelle. Im vorigen Beispiel wäre somit *publishers* die Mastertabelle, *books* die Detailtabelle.

Ich vermeide die Begriffe in diesem Buch. Zum einen finde ich die Nomenklatur verwirrend. Man kann nämlich ebenso schlüssig argumentieren, dass die Verlagsangabe eine Detailinformation eines Buchdatensatzes ist. Insofern stiften die Begriffe aus meiner Sicht eher Verwirrung, als dass sie Klarheit schaffen.

Zum anderen ist die Verwendung des Begriffs *Master* losgelöst vom Kontext im Englischen zunehmend verpönt, weil er an die Sklaverei erinnert. Beim Versionsverwaltungsprogramm Git wurde deswegen aus dem »Master-Branch« der »Main-Branch«.

## Referenzielle Integrität

Vielelleicht können Sie sich erinnern: In Abschnitt 2.4, »Datensicherheit und ACID«, habe ich den Begriff *Consistency* erläutert: Eine Datenbank gilt als *konsistent*, wenn sie diverse formale Regeln einhält. Unter anderem darf es in Primärschlüsselspalten keine Doppelgänger geben. Außerdem müssen alle Werte dem Datentyp und den sonstigen Attributen (NOT NULL, CHECK) der jeweiligen Spalte entsprechen.

In diesem Abschnitt geht es um eine weitere Regel: Sie besagt, dass Foreign-Key-Felder nie ins Leere führen dürfen. Der Begriff *referenzielle Integrität* bezieht sich darauf, dass Referenzen (also Verknüpfungen) stets korrekt sein müssen. Es darf nicht

vorkommen, dass ein Fremdschlüsselfeld einen Wert enthält, den es in der Primärschlüsselspalte der Zieltabelle gar nicht gibt (siehe Abbildung 5.4).

**Tabelle books**

<b>id</b>	<b>title</b>	<b>edition</b>	<b>year</b>	<b>publisherId</b>
1	An Introduction to Database ...	3	1981	1
2	The Relational Model ...	1	1990	267
3	Time and Relational Theory ...	2	2014	2
4	Database Design ...	2	2019	3
...				

  

**Tabelle publishers**

<b>id</b>	<b>publisher</b>	<b>publCity</b>
1	Addison-Wesley	Reading
2	Morgan Kaufmann	Burlington
3	Apress	New York
...		
158	Rheinwerk Verlag	Bonn

Abbildung 5.4 Eine fehlerhafte Verknüpfung zwischen zwei Tabellen

Genau genommen besagt die Forderung nach Consistency, dass eine Datenbank, die vor dem Start einer Transaktion noch konsistent war, danach weiterhin konsistent sein muss. Es gibt eine Reihe von Datenbankoperationen, die zu einer Verletzung der referentiellen Integrität und somit zur Inkonsistenz führen können:

- ▶ Sie fügen einen neuen Datensatz ein, der auf einen gar nicht existierenden Datensatz in einer anderen Tabelle verweist.
- ▶ Sie löschen einen Datensatzes, auf den es einen Querverweis von einer anderen Tabelle gibt.
- ▶ Sie verändern die ID eines Datensatzes, auf den es einen Querverweis von einer anderen Tabelle gibt. (Ganz allgemein ist es absolut unüblich, die Primärschlüsselspalten existierender Datensätze zu verändern. Bei den meisten DBMS ist es aber nicht explizit verboten.)
- ▶ Sie verändern einen schon vorhandenen Querverweis, sodass er nun auf einen gar nicht existierenden Datensatz zeigt.

### Foreign-Key-Regeln (Foreign Key Constraints)

Durch die Definition von Foreign-Key-Regeln (oft verkürzt zu *FK-Regeln*, im Englischen *Foreign Key Constraints*) teilen Sie dem DBMS mit, welche Fremdschlüsselspalte auf welche Primärschlüsselspalte verweist. Außerdem legen Sie fest, wie das

DBMS auf etwaige Verstöße gegen die referenzielle Integrität reagieren soll. Der gängigste Weg besteht darin, einfach einen Fehler auszulösen und damit die Transaktion abzubrechen. Je nach Kontext kann es aber auch zweckmäßig sein, betreffende Foreign-Key-Felder auf NULL zu setzen oder die betreffenden Datensätze überhaupt zu löschen. Letzteres ist zugegebenermaßen ein radikaler Ansatz, um die Integrität der Daten sicherzustellen.

Die Festlegung der Foreign-Key-Regeln erfolgt wie üblich wahlweise in einer herstellerspezifischen Benutzeroberfläche zur Entwicklung der Datenbank oder durch SQL. Ich bleibe hier, wie schon in den vorigen Abschnitten, beim weitgehend plattformunabhängigen SQL-Code. Ein Beispiel dafür, wie Sie Tabellen samt Primär- und Fremdschlüsselpalten und Foreign-Key-Regeln in einer grafischen Benutzeroberfläche einrichten, folgt in [Abschnitt 5.3, »Foreign-Key-Beispiele«](#) (siehe insbesondere Abbildung 5.8).

Erfreulicherweise halten sich, was die FK-Regeln betrifft, viele DBMS weitgehend an die Syntax, die der SQL-Standard vorgibt. Demnach geben Sie den Verweis auf eine andere Tabelle wahlweise direkt bei der Deklaration der Spalte oder im Anschluss an die Spalte in Form einer Zusatzregel an. Die folgende Syntax ist zur besseren Lesbarkeit ein wenig vereinfacht. Beachten Sie, dass bei MySQL und MariaDB nur die zweite Variante funktioniert. Inlineregeln mit REFERENCES werden ohne Fehlermeldung einfach ignoriert.

```
-- Inlinedeklaration der FK-Regeln
-- (Vorsicht: wird von MySQL/MariaDB ignoriert)
fkcolname type attributes REFERENCES otherTable (pkcolname)
  [ON DELETE
   RESTRICT | CASCADE | SET NULL | NO ACTION | SET DEFAULT]
  [ON UPDATE
   RESTRICT | CASCADE | SET NULL | NO ACTION | SET DEFAULT]

-- oder separate FK-Regel nach der Deklaration der Spalten
[CONSTRAINT [constraintname]]
  FOREIGN KEY (fkcolname) REFERENCES otherTable (pkcolname)
  [ON DELETE
   RESTRICT | CASCADE | SET NULL | NO ACTION | SET DEFAULT]
  [ON UPDATE
   RESTRICT | CASCADE | SET NULL | NO ACTION | SET DEFAULT]
```

Wenn sich die Foreign-Key-Regel auf die Primärschlüsselpalte bezieht, können Sie bei vielen DBMS (nicht bei MySQL) auf die Angabe des Spaltennamens verzichten:

```
-- fkcolname verweist auf die PK-Spalte von otherTable
fkcolname type attributes REFERENCES otherTable [options]
```

Bevor ich die Syntax erkläre, ein konkretes Beispiel (wobei das Schlüsselwort AUTO\_INCREMENT MySQL-spezifisch ist), das sich nochmals auf die Verknüpfung von Büchern und Verlagen bezieht (siehe Abbildung 5.4):

```
CREATE TABLE publishers(
    -- Primärschlüssel von publishers
    id          INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    publisher   VARCHAR(100) NOT NULL,
    publCity    VARCHAR(100))

CREATE TABLE books(
    -- Primärschlüssel von books
    id          INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    title       VARCHAR(100) NOT NULL,
    edition     INT DEFAULT 1,
    year        INT,
    -- Foreign Key, verweist auf publishers.id
    publisherId INT,
    -- dazugehörige Foreign-Key-Regel
    FOREIGN KEY (publisherId) REFERENCES publishers(id))
```

Die Foreign-Key-Regel in der letzten Zeile besagt, dass die Spalte *publisherId* der aktuellen Tabelle auf die Spalte *id* der Tabelle *publishers* verweist und dass das DBMS die referenzielle Integrität sicherstellen soll. Falls sich Primär- und Fremdschlüssel jeweils aus mehreren Spalten zusammensetzen, sieht die Syntax so aus:

```
FOREIGN KEY (fkcol1, fkcol2, fkcol3 ...)
REFERENCES otherTable (pkcol1, pkcol2, pkcol3 ...)
```

Bei diesem Beispiel ist die Spalte *publisherId* ohne NOT NULL deklariert. Es ist also erlaubt, ein Buch zu speichern, ohne einen Verlag anzugeben. Die Foreign-Key-Regel gilt somit nur, wenn ein Verlag angegeben wird. In diesem Fall muss *publisherId* mit einer ID aus der Tabelle *publishers* übereinstimmen.

Bei den meisten SQL-Dialektken können Sie die Schlüsselwörter FOREIGN KEY usw. direkt im Anschluss an die Spaltendeklaration angeben. Bei MySQL/MariaDB muss die Regel aber als eigenständiger Block angegeben werden, optional mit vorangestelltem CONSTRAINT [constraintname].

Beachten Sie, dass bei den Foreign-Key-Regeln die Reihenfolge der CREATE TABLE-Kommandos wichtig ist. Die meisten DBMS lösen einen Fehler aus, wenn Sie in einer FK-Regel auf eine noch gar nicht existierende Tabelle verweisen. Deswegen muss *publishers* vor *books* erzeugt werden. Bei Datenbanken, die aus vielen Tabellen bestehen, ist es einfacher, diese vorerst ohne FK-Regeln zu erzeugen und die FK-Regeln später mit ALTER TABLE hinzuzufügen.

## Reaktion auf die Verletzung der Foreign-Key-Regeln

Wie soll sich das DBMS verhalten, wenn es eine Verletzung der Foreign-Key-Regeln feststellt? Naheliegend ist es, einfach einen Fehler auszulösen. Damit scheitert die aktuelle Transaktion (daher die Bezeichnung RESTRICT für diese Variante), das Kommando wird nicht ausgeführt, und die Datenbank bleibt konsistent. Bei fast allen DBMS ist das das Standardverhalten. Bei Einfügeoperationen (INSERT) gibt es dazu auch keine sinnvollen Alternativen – bei der Veränderung oder beim Löschen eines Datensatzes sehr wohl. Diese Alternativen drücken sich durch die optionalen Syntaxbestandteile ON DELETE und ON UPDATE aus:

```
ON DELETE RESTRICT | CASCADE | SET NULL | NO ACTION | SET DEFAULT  
ON UPDATE RESTRICT | CASCADE | SET NULL | NO ACTION | SET DEFAULT
```

Beginnen wir mit Löschvorgängen: Stellen Sie sich vor, in unserer Bücherdatenbank gibt es drei Bücher, die im Verlag Addison-Wesley veröffentlicht wurden. Nun löschen Sie den Verlag aus der Datenbank (warum auch immer, das steht hier nicht zur Debatte). Das DBMS erkennt, dass drei Datensätze aus *books* auf den Verlag verweisen, und löst einen Fehler aus. Alternativ sind aber die folgenden Maßnahmen denkbar:

- ▶ Mit ON DELETE CASCADE löscht das DBMS auch gleich die drei bei Addison-Wesley erschienenen Bücher. Das ist zwar radikal, aber die Datenbank bleibt damit in einem konsistenten Zustand.

Es liegt auf der Hand, dass die Option ON DELETE CASCADE nur mit Vorsicht eingesetzt werden sollte. In der Praxis ist die Variante aber durchaus nicht so unattraktiv, wie es vielleicht den Anschein hat. Das manuelle Löschen von Datensätzen aus verknüpften Tabellen mit FK-Regeln ist ausgesprochen mühsam. Wenn Sie als Administrator wissen, was Sie tun, kann ON DELETE CASCADE eine Menge Mühe sparen. Umgekehrt kann das unbedachte Löschen eines Datensatzes zu einer wahren Kettenreaktion führen – also Vorsicht!

- ▶ Bei ON DELETE SET NULL setzt das DBMS bei den betreffenden drei Büchern die Spalte *publisherId* auf NULL. Damit bleiben die Bücher in der Datenbank; allerdings ist ihnen jetzt kein Verlag mehr zugeordnet.
- ▶ ON DELETE SET DEFAULT bewirkt, dass der für die Spalte *publisherId* festgelegte Defaultwert eingestellt wird. Das setzt natürlich voraus, dass es so einen Defaultwert gibt, was beim obigen Beispiel nicht der Fall ist. Es wäre aber denkbar, dass Sie in der Tabelle *publishers* einen Datensatz mit dem Text *Verlag unbekannt* einfügen. Die ID dieses Datensatzes könnten Sie als Defaultwert festlegen, also:

```
CREATE TABLE books(...,  
    publisherId INT DEFAULT 1234 REFERENCES publishers(id))
```

Ganz ähnlich sieht es bei UPDATE-Kommandos aus. Nehmen wir an, ein Datenbankkommando versucht, die ID von Addison-Wesley zu ändern. Außer dem Abbruch der Transaktion gibt es die folgenden Möglichkeiten:

- ▶ Bei ON UPDATE CASCADE werden die *publisherId*-Werte entsprechend angepasst. Die Datensätze der Bücher verweisen somit weiterhin auf den gleichen Verlag.
- ▶ ON UPDATE SET NULL bewirkt, dass bei den betroffenen Büchern *publisherId* auf NULL gesetzt wird.
- ▶ Mit ON UPDATE SET DEFAULT stellt das DBMS die *publisherId*-Felder der betroffenen Datensätze auf den für die Spalte gültigen Defaultwert.

#### »RESTRICT« versus »NO ACTION«

Damit verbleiben nur noch die beiden Optionen RESTRICT und NO ACTION zu besprechen. Beide lösen einen Fehler aus, bei RESTRICT sofort, NO ACTION dagegen erst beim Ende der Transaktion. (Den Unterschied erkläre ich im folgenden Abschnitt.)

Der SQL-Standard schlägt NO ACTION als Defaultverhalten vor, die meisten DBMS außer MySQL halten sich daran.

### Konsistenz, aber nicht sofort (Deferred Constraints)

Wann werden Foreign-Key-Regeln überprüft, wann führen sie gegebenenfalls zu einem Fehler? Die naheliegende Antwort lautet: sofort, das heißt, sobald das Kommando vom DBMS ausgeführt wird.

Es gibt aber eine zweite Denkweise: Vielleicht ist das Kommando, das die referenzielle Integrität verletzt, nur eines von mehreren Kommandos in einer Transaktion. Dann ist denkbar, dass die weiteren Kommandos die Verletzung der Integrität wieder beheben. Insofern wäre es sinnvoll, den Test der Integrität erst am Ende der Transaktion durchzuführen und auch den Fehler erst zu diesem Zeitpunkt auszulösen.

Die Sinnhaftigkeit einer verzögerten Constraint-Verarbeitung wird oft mit zyklischen Foreign-Key-Regeln zwischen zwei oder mehr Tabellen begründet. In so einem Fall (den man in der Praxis tunlichst zu vermeiden sucht) steht man vor einem Henne-Ei-Problem: Das Einfügen eines neuen Datensatz in Tabelle A ohne einen dazugehörigen Datensatz in Tabelle B ist unmöglich, bevor der Datensatz in Tabelle B existiert – und umgekehrt. Abhilfe schafft es, beide Einfügekommandos in einer Transaktion zu bündeln und mit der Auswertung der FK-Regeln bis zum Ende der Transaktion zu warten.

Der SQL-Standard sieht *Deferred Constraints* (also die verzögerte Auswertung von Regeln) vor. Diese Möglichkeit wird aber bei weitem nicht von allen DBMS unterstützt. Musterschüler sind diesbezüglich PostgreSQL und Oracle.

## Implementierungsspezifische Details

- ▶ Microsoft SQL Server unterstützt das Schlüsselwort RESTRICT nicht.
- ▶ Bei MySQL und MariaDB müssen Foreign-Key-Regeln als separate Constraints deklariert werden (also nicht direkt im Rahmen der Spaltendeklaration). NO ACTION funktioniert wie RESTRICT. FK-Fehler werden daher sofort ausgelöst, nicht erst am Ende der Transaktion. Die Auswertung von Foreign-Key-Regeln erfolgt grundsätzlich sofort.
- ▶ Oracle unterstützt nicht alle Constraint-Varianten. Insbesondere fehlt ON UPDATE CASCADE.
- ▶ Bei Oracle und PostgreSQL können Sie den Zeitpunkt, zu dem FK-Regeln ausgewertet werden, mit den zusätzlichen Constraint-Optionen DEFERRABLE bzw. NOT DEFERRABLE sowie mit INITIALLY DEFERRED oder INITIALLY IMMEDIATE beeinflussen.
- ▶ Bei Oracle und PostgreSQL können Sie mit SET CONSTRAINTS DEFERRED|IMMEDIATE festlegen, ob Constraints sofort oder erst beim Ende der Transaktion überprüft werden. Diese Option betrifft auch FK-Regeln, sofern dort die Variante NO ACTION verwendet wird.
- ▶ SQLite akzeptiert zwar Foreign-Key-Regeln in CREATE und ALTER TABLE, allerdings muss ihre Einhaltung beim Verbindungsaufbau des Clients mit PRAGMA foreign\_key = ON explizit aktiviert werden. Die Syntaxvarianten ON UPDATE ... bzw. ON DELETE ... werden nicht unterstützt.

## FK-Regeln vorübergehend deaktivieren

Die meisten DBMS bieten eine Möglichkeit, Foreign-Key-Regeln vorübergehend zu deaktivieren. Dafür gibt es zwei Gründe:

- ▶ Die Kontrolle der FK-Regeln kostet Zeit. Wenn Sie sehr viele Datensätze auf einmal importieren oder aus einem Backup einspielen möchten, ist es schneller, die Kontrolle der Regeln erst zum Schluss durchzuführen. Das gilt insbesondere dann, wenn zu erwarten ist, dass es ohnedies keine Probleme geben wird.
- ▶ Der Import vieler Datensätze bei aktiven FK-Regeln setzt voraus, dass die Datensätze in der richtigen Reihenfolge eingefügt werden (im Beispiel dieses Abschnitts also zuerst die Verlage und dann die Bücher). Bei der Deaktivierung der FK-Regeln spielt die Reihenfolge dagegen keine Rolle – entscheidend ist nur, dass die FK-Regeln zum Schluss erfüllt sind.

Guter Rat ist teuer, wenn beim späteren Versuch, die FK-Regeln zu aktivieren, Probleme auftreten. Wie Sie mit SELECT-Kommandos nach Datensätzen suchen, die FK-Regeln verletzen, zeige ich Ihnen in Abschnitt 13.2, »Subqueries«.