Relationale Datenbanken

Michael Dienert

16. September 2012

Vorbemerkung

In diesem Artikel wird ganz kurz erklärt, was relationale Datenbanken sind, wie sie entworfen werden und welche Regeln es dabei zu beachten gibt. Die wichtigsten Stichworte zu diesem Thema sind:

- Relation
- Entität
- Attribut
- Beziehung
- erste bis dritte Normalform: 1NF, 2NF, 3NF

Inhaltsverzeichnis

1 Ein paar Begriffe

Datenbank - eine Datenbank ist einfach nur eine Menge Daten, die verwaltet werden müssen. Wir beschränken uns natürlich auf elektronische Daten und damit ist eine Datenbank eine Ansammlung beliebig strukturierter Datensätze auf irgendeinem schreib- und lesbaren Speichermedium (Festplatte, Flash, RAM).

Relationale Datenbanken sind besondere Datenbanken, deren Aufbau weiter unten erklärt wird.

Datensätze - Datensätze sind eine Ansammlung von *mehreren* Daten. Die einzelnen Daten können *verschiedene Datentypen* besitzen (z.B. Text-Strings, Integer-Zahlen, Fliesskomma-Zahlen, usw.).

Da die einzelnen Daten verschiedene Datentypen haben dürfen, unterscheidet sich ein Datensatz von einem *Array*.

Andere Namen für Datensatz sind: *Record* und *Tupel*.

Datenbank-Management-System - ein Datenbank-Management-System ist eine Software, die die Daten in der Datenbank verwaltet und eine Schnittstelle zum Anwender bereitstellt. Meistens arbeiten die DBMS als Server, d.h. man kommuniziert über einen Port mit ihnen und benötigt dazu wiederum eine Client-Software.

RDBMS - die Abkürzung steht für Relational DataBase Management System, eben eine DB-Management-System für relational-organisierte Datenbanken.

Die wichtigsten RDBMS sind: Oracle, MySQL, MS SQL-Server, DB2 (IBM) und am allerwichtigsten, da in diesem Unterricht ausschliesslich verwendet: **PostgreSQL**.

2 Einführendes Beispiel

Um die wichtigsten Begriffe rund um relationale Datenbanken zu erklären, werden wir eine Datenbank für Musik-CDs entwerfen.

Wenn jemand ganz unvoreingenommen beginnt, seine CD-Sammlung in Form einer Datenbank zu katalogisieren, wird er vielleicht wie in Tabelle 1, Seite 4 gezeigt, beginnen. Auf den ersten Blick sind alle wichtigen Daten einer CD erfasst, und die Datensätze scheinen auch einigermassen strukturiert zu sein.

3 Erste Normalform, 1NF

Dieser Ansatz ist jedoch nicht sehr gut, was man zuerst bei der Eingabe der Datensätze und spätestens beim Arbeiten mit der Datenbank bemerken würde:

- Hat man z.B. 20 Alben der Rolling Stones im Schrank, müsste man bei der Datenerfassung 20 mal den Bandnamen *Rolling Stones* eintippen. Eine unnötige Tipperei, eigentlich müsste doch eine einmalige Eingabe reichen!

- In der Tabelle enthält jede Zeile am Ende eine Liste mit allen Titeln der jeweiligen CD. mit einer solchen Unterliste innerhalb einer Tabelle würden einige wichtige Datenbank-Funktionen *nicht funktionieren*. Z.B. kann man die Tabelle nach dem Bandnamen sortieren, man kann sie aber *nicht* nach den Musik-Titeln sortieren. Ausserdem wäre es unmöglich, die Spieldauer der einzelnen Titel in einer weiteren Spalte unterzubringen. Man könnte die Spieldauer höchstens mit in die Unterliste aufnehmen.

Um die Probleme mit der Unterliste zu umgehen, muss man beim Datenbankentwurf fogende Forderung erfüllen:

In einer Tabelle dürfen die Spalteneinträge nur *atomare* das heisst *unteilbare* Werte enthalten. Eine Tabelle, die diese Forderung erfüllt, ist in der **ersten Normalform**.

1NF = nur atomare Werte

Also, bringen wir die Tabelle in die erste Normalform. Tabelle 2 zeigt das Ergebnis.

CD-ID	Band Name	CD Title	Record Label	Songs
1	Little Feat	The Last Record Album	Warner Bros	Romance Dance All That You Dream ... Dont´t Bogart That Joint A Political Blues
2	The Rolling Stones	Get Yer Ya-Ya´s Out!	Decca	Jumpin´ Jack Flash Carol Stray Cat Blues Love In Vain ... Street Fighting Man
3	The Rolling Stones	Let It Bleed	Decca	Gimme Shelter Love in Vain Country Honk Live With Me Let It Bleed Midnight Rambler ...

Tabelle 1: Erster Versuch, eine CD-Datenbank zu entwerfen.

CD-ID	Band Name	CD Title	Record Label	Songs
1	Little Feat	The Last Record Album	Warner Bros	Romance Dance
1	Little Feat	The Last Record Album	Warner Bros	All That You Dream
1	Little Feat	The Last Record Album	Warner Bros	...
1	Little Feat	The Last Record Album	Warner Bros	Dont´ Bogart That Joint
1	Little Feat	The Last Record Album	Warner Bros	A Political Blues
2	The Rolling Stones	Get Yer Ya-Ya´s Out!	Decca	Jumpin´ Jack Flash
2	The Rolling Stones	Get Yer Ya-Ya´s Out!	Decca	Carol
2	The Rolling Stones	Get Yer Ya-Ya´s Out!	Decca	Stray Cat Blues
2	The Rolling Stones	Get Yer Ya-Ya´s Out!	Decca	Love In Vain
2	The Rolling Stones	Get Yer Ya-Ya´s Out!	Decca	...
2	The Rolling Stones	Get Yer Ya-Ya´s Out!	Decca	Street Fighting Man
3	The Rolling Stones	Let It Bleed	Decca	Gimme Shelter
3	The Rolling Stones	Let It Bleed	Decca	Love In Vain
3	The Rolling Stones	Let It Bleed	Decca	Country Honk
3	The Rolling Stones	Let It Bleed	Decca	Live With Me
3	The Rolling Stones	Let It Bleed	Decca	Let It Bleed
3	The Rolling Stones	Let It Bleed	Decca	Midnight Rambler
3	The Rolling Stones	Let It Bleed	Decca	...

Tabelle 2: Die Tabelle in der ersten Normalform.

4 Datenbankmodellierung

So, die Tabelle ist jetzt zwar in der ersten Normalform, aber das Problem mit dem mehrfachen Tippen gleicher Bandnamen ist jetzt noch viel schlimmer geworden. Sogar der CD-Titel taucht jetzt vielfach auf.

Um diese *Redundanzen* innerhalb der Tabelle zu vermeiden, muss man die Tabelle in mehrere Tabellen zerlegen, die miteinander durch sogenannte *Schlüssel* in Beziehung stehen.

Dieses Zerlegen in verschiedene Tabellen darf allerdings nicht nach Belieben erfolgen. Vielmehr muss man versuchen, die Wirklichkeit in ein *Datenbankmodell*, das aus sehr vielen Tabellen bestehen kann, abzubilden. Dieses Abbilden wird *Datenmodellierung* genannt. Das Modellieren einer Datenbank ist eine Kunst, weshalb Leute, die darauf spezialisiert sind, die absoluten Spitzenverdiener in der gesamten IT-Welt sind.

Bei so einem einfachen Zusammenhang wie in unserer CD-Tabelle, kommt man aber auch mit folgender Überlegung weiter:

- die Bandnamen werden durch Fussnoten ersetzt
- der CD-Titel und das zugehörige Label werden ebenfalls durch Fussnoten ersetzt
- die Fussnoten für Bandnamen und CD-Titel und Label werden in neuen Tabellen zusammengefasst.

Das sieht dann aus, wie in den drei Tabellen auf Seite 6.

Betrachtet man nun die Tabelle mit den CD-Titeln und den Labels, fällt sofort auf, dass auch hier nochmal das Fussnotenverfahren angewendet werden kann: auch das Plattenlabel gehört durch Fussnoten ersetzt und in eine extra Tabelle ausgelagert.

In der Praxis werden Datenbanken aber nicht mit dieser schrittweisen Methode entworfen, sondern es wird das sog. *Entity Relationship Modelling* angewendet, das schrittweise in den folgenden Kapiteln vorgestellt wird.

CD-ID	Band Name - Nr	CD Title & Record Label - Nr	Songs
1	(1)	(a)	Romance Dance
1	(1)	(a)	All That You Dream
1	(1)	(a)	...
1	(1)	(a)	Dont´ Bogart That Joint
1	(1)	(a)	A Political Blues
2	(2)	(b)	Jumpin´ Jack Flash
2	(2)	(b)	Carol
2	(2)	(b)	Stray Cat Blues
2	(2)	(b)	Love In Vain
2	(2)	(b)	...
2	(2)	(b)	Street Fighting Man
3	(2)	(c)	Gimme Shelter
3	(2)	(c)	Love In Vain
3	(2)	(c)	Country Honk
3	(2)	(c)	Live With Me
3	(2)	(c)	Let It Bleed
3	(2)	(c)	Midnight Rambler
3	(2)	(c)	...

Band Name - Nr	Band Name
1	Little Feat
2	The Rolling Stones

CD-Title/Label - Nr	CD-Title & Label
a	The Last Record Album & Warner Bros
b	Get Yer Ya-Ya´s Out! & Decca
c	Let It Bleed & Decca

Tabelle 3: Auslagerung der mehrfach vorkommenden Datenwerte.

5 Einige Datenbankbegriffe

Folgende Begriffe sind bei der Modellierung wichtig:

5.1 Entität und Attribut

Der wichtigste Begriff in der Datenbankwelt ist die *Entität*, auf englisch *entity*.

> Eine *Entität* ist ist ein Objekt, über das Informationen gesammelt werden müssen. Die Informationen werden dabei mit Hilfe von *Attributen* beschrieben. Ein Attribut beschreibt dabei Informationen *exakt einer* Entität.

In unserem Beispiel ist die CD eine Entität, während der CD-Titel eines ihrer Attribute ist. Weitere Beispiele für Entitäten bei z.B. einer Produktdatenbank sind: Artikel, Lieferanten, Bestellungen, Kunden, ... Die Attribute von *Kunde* sind: *Kunden-NR, Kundenname, Kundenanschrift, ...*

> Entitäten werden immer im Singular aufgeführt!

Also *Kunde*, nicht Kunden.
Ist das Datenmodell fertig, hat man für jede Entität in der Regel eine Tabelle. Also:

> Entität = Tabellenname

Die Attribute sind dann wiederum nichts anderes, als die Spalten der Tabelle:

> Attribut = Spalte

5.2 Relation

Während der Begriff Entität ein bisschen schwierig ist, ist der Begriff *Relation* ganz einfach:

> Relation = Tabelle

Also Achtung! Man darf *Relation* nicht mit Beziehung verwechseln! Der Begriff der Beziehung taucht beim Datenmodellieren nämlich auch noch auf.

5.3 Schlüssel und Tupel

Noch zwei wichtige Begriffe:

> Tupel = Zeilen einer Tabelle

Alle Tupel müssen eindeutig identifizierbar sein. Deshalb bekommen alle Tupel einen *Primärschlüssel* z.B. eine Nummer. Schlüssel wird im Englischen mit *ID* abgekürzt. Ich werde diese Abkürzung ab hier ebenfalls verwenden.
Für die Schlüssel müssen einige Bedingungen gelten:

- Sie müssen eindeutig sein
- Sie dürfen *nicht NULL* sein
- Sie müssen unveränderlich sein

Beim Datenbankentwurf ist es nicht zwingend notwendig, einen zusätzlichen, numerischen Schlüssel hinzuzufügen. Oftmals enthalten die Datensätze von sich aus Attribute, die die Schlüsselbedingung erfüllen. Da man so wenig wie möglich Spalten (=Attribute) in einer Tabelle haben möchte, fügt man dann nicht einen zusätzlichen, numerischen Schlüssel hinzu.

Beispiel:

Eine Datenbank enthält eine Ländertabelle:

```
isoKennung Land
---------------------
GA         Gabon
GM         Gambia
GE         Georgia
DE         Germany
GH         Ghana
GI         Gibraltar
GR         Greece
```

Hier kann man direkt den ISO-3166-Ländercode als Primärschlüssel nehmen.

Es ist auch erlaubt und wird gelegentlich angewendet, *mehrere* Attribute zusammenzufassen, so dass ihre *Kombination eindeutig* und damit zum Primärschlüssel wird.

6 Entity Relationship Modelling

Um bei der Datenmodellierung nicht jedesmal riesige Tabellen schreiben zu müssen, wurde eine Schreibweise eingeführt, bei der nur die Entität und ihre Attribute in einem Rechteck dargestellt werden.

Unser Beispiel der CD-Datenbank sieht in dieser Form so aus:

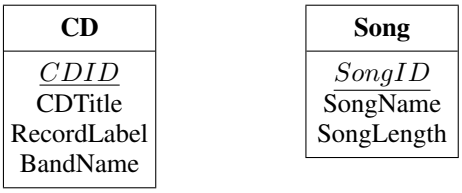

Abbildung 1: Entitäten-Blockdiagramm

Wieso wurde aus der ursprünglichen Tabelle *CD* gerade die Tabelle *Song* ausgegliedert?

Laut Definition ist eine Entität ein Objekt, über das man Informationen sammeln möchte. *Song* ist so ein Objekt: wichtige Informationen über einen Song sind natürlich der Name und die Spieldauer. *Song* ist somit eine eigene Entität und nicht nur Attribut von *CD*. Als Entität wird es in der Datenbank zu einer neuen Tabelle.

Bis jetzt berücksichtigt diese Darstellung die Beziehungen zwischen den beiden Tabellen noch nicht. Die Beziehungen zwischen *Song* und *CD* sind doch so:

- Eine CD enthält einen oder mehrere Songs.
- Ein Song ist immer *nur einmal* auf einer CD.

Hierbei ignorieren wir, dass eine Aufnahme auf mehreren verschiedenen CD´s veröffentlicht sein könnte und dass auf einer CD eine Reprise vorkommt.

Im Blockdiagramm werden diese Beziehungen so dargestellt:

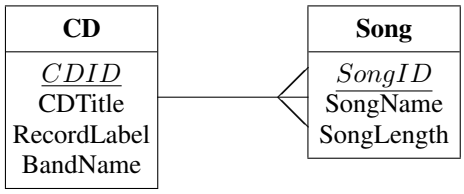

Abbildung 2: Entity-Relationship-Blockdiagramm

Diese Darstellung wird als *Krähenfuss-Notation* bezeichnet. Es ist eine *eins-zu-viele* Beziehung.

Habe ich nun die Tabelle *Song* ausgegliedert, sind die meisten Redundanzen (mehrfaches Auftauchen gleicher Werte) verschwunden. Geblieben ist aber immer noch, dass die Rolling Stones zweimal auftauchen und auch deren damaliges Label *Decca* kommt zweimal vor. Es sieht also so aus, als ob sich aus der Tabelle *CD* noch mehr Tabellen ausgliedern lassen:

Da es auch CD´s von Solo-Musikern gibt, ersetzen wir *Band* ab jetzt durch *Artist*.

Wir gliedern also die Tabelle *Artist* aus. Aber in welcher Beziehung steht diese Tabelle zum Rest?

Ein Künstler schreibt einen oder mehrere Songs, aber jede spezielle Aufnahme wurde nur von einem Künstler geschrieben. Das ergibt eine eins-zu-viele Beziehung zwischen *Artist* und *Song*

Damit sieht das Entity-Relationship-Diagramm jetzt so aus:

Welche Beziehung besteht zwischen *Artist* und *CD*? Ein Künstler erscheint auf einer oder mehreren CD´s und auf einer CD wiederum veröffentlichen einer oder mehrere Künster ihre Werke. Das ergibt somit eine *viele-zu-viele* Beziehung.

Derartige viele-zu-viele Beziehungen *müssen* immer in zwei eins-zu-viele Beziehungen aufgelöst werden. Erreicht wird dies durch eine *Verbindungsentität*, die zwischen den beiden ursprünglichen Entitäten steht.

In unserem Fall ist also *Song* die Verbindungsentität.

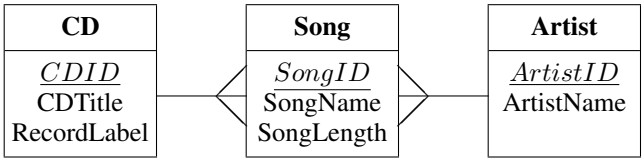

Abbildung 3: Entity-Relationship-Blockdiagramm

Das jetzige Modell ist immer noch nicht frei von Redundanzen. Wir müssen noch das Record-Label auslagern. Ein Label gibt ja viele CD´s heraus, so dass die Labels zwangsläufig mehrfach vorkommen würden, wenn man nicht eine eigene Entität *RecordLabel* schafft:

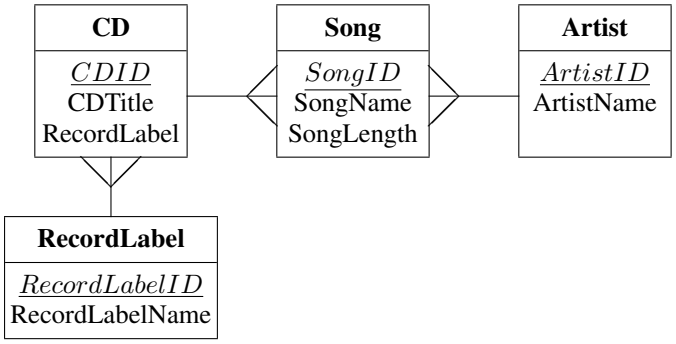

Abbildung 4: CD-Datenbank in der zweiten Normalform

7 Zweite Normalform, 2NF

Die Tabellen unserer Beispieldatenbank sind mittlerweile in der *zweiten Normalform*.

> Die 2NF kann nur verletzt werden, wenn der Primärschlüssel aus mehr als einem Attribut zusammengesetzt ist

Die Tabelle 2 auf Seite 4 ist *nicht* in der 2NF, wo ist also der zusammengesetzte Schlüssel?

In dieser Tabelle ist erst die Kombination aus `'CD-ID'` und `'Songs'` *eindeutig* und bildet somit einen zusammengesetzten Primärschlüssel (vorausgesetzt, auf einer CD gibt es nicht zwei Stücke mit dem gleichen Songnamen).

Eine solche minimale Menge von Attributen, die einen Datensatz eindeutig identifiziert wird *candidate key* genannt. Eine Tabelle kann mehrere solcher *candidate keys* haben. Der *candidate key* der aus den wenigsten Attributen besteht wird dann zum *primary key* ernannt.

Der candidate key von Tabelle 2 ist also die Menge `{CD-ID, Songs}`

Nun kann man die Bedingung für die 2NF formulieren:

> Hat eine Tabelle einen zusammengesetzten Schlüssel (candidate key) müssen die Attribute von diesem Schlüssel als Gesamtheit (whole key) abhängen. Hat eine Tabelle mehrere candidate keys muss das für alle candidate keys gelten.

In unserer ersten Tabelle (Tab. 2, S. 4) hängt das Attribut `'CD-Title'` eindeutig von `'CD-ID'` ab, nicht aber von `'Songs'` und damit natürlich auch nicht von der Kombination `{CD-ID, Songs}`. Die Tabelle ist also sicher nicht in der 3NF.

8 Normalisierung durch Entity Relationship Modelling

> Hat man durch Entity Relationship Modelling ein Datenbankmodell entwickelt, dass nur noch 1-1 und 1-n Beziehungen enthält, sind die zugehörigen Tabellen (Relationen) sicher mindestens in der 2NF.

9 Dritte Normalform

> Die dritte Normalform verlangt, dass es zwischen den Attributen keine Abhängigkeiten geben darf: die Attribute dürfen einzig und allein nur vom Primärschlüssel abhängen (nothing but the key).

Beispiel: Wir erweitern die Tabelle *RecordLabel* um das Land, in dem sich die Plattenirma befindet und dessen ISO-Länderkennung.

Diese beiden Attribute hängen jedoch voneinander ab: Deutschland hat die Kennung **de**. Zieht eine Firma von Deutschland nach London, müsste man *beide* Attribute ändern:

$$\text{Deutschland} \rightarrow \text{England}$$
$$\text{de} \rightarrow \text{uk}$$

Verhindern kann man dies -wie sollte es auch anders sein- wieder durch Auslagern einer eigenen Entität *Country*, mit den Attributen *CountryName* und *CountryCode*

10 Merksatz für die 1NF, 2NF und 3NF

> the key (1NF), the whole key (2NF) and nothing but the key (3NF)