Relationales Datenbankmodell – Teil 2

Die Lösung besteht darin, diese Tabelle in drei Tabellen aufzuspalten:

Tabelle *Noten* Tabelle *SchülerInnen*

|  |  |  |
| --- | --- | --- |
| Schüler-nr | Fachnr | Note |
| 105 | 1 | 2- |
| 105 | 2 | 3+ |
| 105 | 3 | 1 |
| … |  |  |
| 213 | 1 | 1 |
| 213 | 2 | 3- |
| 213 | 3 | 5 |
| … |  |  |

|  |  |  |
| --- | --- | --- |
| Schüler-nr | Name | Adresse |
| … |  |  |
|  |  |  |
| 105 | Schmidt, Arabella | Gertigstraße 23, 22126 Hamburg |
| … |  |  |
| 213 | Wagner, Kai | Hofweg 5, 20035 Hamburg |
| … |  |  |

Tabelle *Fach*

|  |  |
| --- | --- |
| Fachnr | Fach |
| 1 | Mathe |
| 2 | Englisch |
| 3 | Deutsch |
| … |  |

Auf diese Weise müssen z.B. die persönlichen Schüler-Daten nicht für jedes Fach wiederholt werden – stattdessen wird dort jeweils die dem Schüler zugewiesene Schülernummer eingetragen.

Darüber hinaus sind nun die Daten **logisch sinnvoll** getrennt:
Angaben zur Person (und *nur* diese Angaben) stehen in der Tabelle *SchülerInnen,* Angaben zu den Noten (und *nur* diese Angaben) stehen in der Tabelle *Noten* und Angaben zu den Fächern (und *nur* diese Angaben) stehen in der Tabelle *Fach*.

Das Attribut *Schülernr* bildet also eine Verbindung, die **Relation**, zwischen den beiden Tabellen *SchülerInnen* und *Noten*, während das Attribut *Fachnr* die Tabellen *Noten* und *Fach* verbindet.

Das entsprechende **relationale Datenbankmodell** sieht folgendermaßen aus:

|  |
| --- |
| SchülerInnen |
| PK | Schülernr | int |
|  | Name | varchar(50) |
|  | Adresse | varchar(50) |

|  |
| --- |
| Noten |
| FK | Schülernr | int |
| FK | Fach | varchar(50) |
|  | Note | varchar(50) |

|  |
| --- |
| Fach |
| PK | Fachnr | int |
|  | Fach | varchar(50) |

Man erkennt, dass zwischen dem Attribut *Schülernr* in der Tabelle *Noten* und dem Attribut *Schülernr* in der Tabelle *SchülerInnen* eine Beziehung besteht, da zum einen der selbe Attributname verwendet wird und zum anderen eine Linie zwischen den entsprechenden Tabellenzeilen gezeichnet wurde.

Abkürzungen:

* PK: primary key = Primärschlüssel
* FK: foreign key = Fremdschlüssel
* int: integer
* varchar = variable character

**Aufgabe 2**

Recherchiere die unbekannten Begriffe und Zusammenhänge im obigen Modell. Beantworte dabei **unter anderem** auch folgende Fragen und notiere jeweils in eigenen Worten, was Du heraus gefunden hast.

* Wieso bestimmt gerade das Attribut *Schülernr* die Beziehung zwischen den Tabellen?
* Wieso stellt das Attribut *Schülernr* einmal einen Primär- und einmal einen Fremdschlüssel dar?
* Was ist ein Primärschlüssel? (Wozu dient er?)
* Wozu werden z.B. int und varchar verwendet? Was sind weitere Möglichkeiten?
* Wie ist eine Tabelle im relationalen Datenbankmodell aufgebaut?
* …

***Fortsetzung ERM***

**Identifizierung von Objekten: Primärschlüssel**

Einzelne Objekte der Miniwelt sind in aller Regel voneinander eindeutig unterscheid-bar. Diese Unterscheidbarkeit (Identifizierbarkeit) der Objekte ist über deren Attribute und Attributwerte möglich.

*Die minimale Kombination aller Attribute, durch deren Wert ein bestimmtes Objekt eindeutig identifiziert wird, heißt* ***Primärschlüssel****. Ein Primärschlüssel, der sich aus mehreren Attributen zusammensetzt, wird* ***zusammengesetzter Primärschlüssel*** *genannt. Ein Attribut eines zusammengesetzten Primärschlüssels wird als* ***Teilschlüssel*** *bezeichnet.*

Oft gibt es keine Kombination von Attributen, die sich als Primärschlüssel eignet. Beim Objekttyp Schüler würden die Attribute 'Name', 'Vorname' und 'Geburtsdatum' zusammen immer noch nicht ausreichen, um mehrdeutige Werte auszuschließen. In solchen Fällen muss man ein neues Attribut als *künstlichen Primärschlüssel* einführen. Das Attribut Schüler# (# steht für Nummer) wäre ein solcher künstlicher Schlüssel.

Künstliche Primärschlüssel sollten so kurz wie möglich gewählt werden, um die Hand-habbarkeit und Übersichtlichkeit zu erhöhen. Klassifizierende Primärschlüssel, z.B. erster Buchstabe des Nachnamens + erster Buchstabe des Vornamens + Geburtsjahr sind zu vermeiden, da ein korrekt definiertes Attribut nicht weiter unterteilbar sein soll.

**Beziehungen (Relationen)**

In der Miniwelt gibt es nicht nur Objekte, sondern diese Objekte stehen auch in viel-fältigen Abhängigkeiten und Beziehungen zueinander.



Im klassischen ER-Modell werden Entitäts-Typen durch Rechtecke, Attribute durch Kreise/Ellipsen und Beziehungen durch Rauten dargestellt. Hervorzuheben ist, dass auch eine Beziehung Attribute besitzen kann (s.u.).

Bei diesen Beziehungen kann man vereinfacht drei Typen unterscheiden:

**1:1 Beziehungen**

Hier steht ein Objekt mit genau einem Objekt in Beziehung. Ein typisches Beispiel hierfür wäre die Beziehung 'verheiratet'. Hier steht ein Element des Typs Mensch mit einem Objekt des Typs Mensch in Beziehung.

**1:N Beziehung**

Ein Beispiel für eine derartige Beziehung wäre „Tutor“. Ein Element des Typs Lehrer kann mehrere Elemente des Typs Schüler als Tutand haben. Aber ein Schüler kann nur einen Tutor haben.

**M:N Beziehung**

Ein Schüler wird von mehreren (M) Lehrern unterrichtet. Ein Lehrer unterrichtet aber auch mehrere(N) Schüler. Die Beziehung 'unterrichtet' ist also eine M:N Beziehung.

Beschreiben wir unsere Miniwelt als eine Menge von Objekten, zwischen denen beliebige Beziehungen bestehen, dann erhalten wir ein Netzwerk, in dem die Objekte durch unterschiedliche Beziehungen miteinander verknüpft sind. Kennt eine Datenbeschreibungssprache Objekte (Entities) und beliebige Beziehungen (SETS), dann können solche Netzwerke mit ihr beschrieben werden. Sie führt zu

**Netzwerkdatenbanken**:



Lässt man zur Beschreibung eines Systems nur hierarchische (1:N) Beziehungen neben den Objekten zu, dann kommt man zu einer **hierarchischen Datenbank**:



Auf dem theoretischen Sektor recht aktuell ist das **objektorientierte Datenmodell**. Hier wird versucht die Elemente über eine Klassenstruktur objektorientiert zu modellieren. Bisher befinden sich derartige Systeme noch weitgehend im Entwicklungs- bzw. Erprobungsstadium, besitzen also noch keine praktische Bedeutung.

Natürlich kann man auch Beziehungen als Objekte betrachten. Bei diesem Ansatz spricht man dann allgemeiner von Relationen statt von Objekttypen bzw. Beziehungstypen. Ein solches Datenmodell besteht dann aus einer Reihe von Tabellen (oder Relationen). Auf diese Art kommt man zu einer relationalen Datenbank.

**Das relationale Datenmodell**

Der Übergang vom ER-Model zum relationalen Datenbankmodell (RDM) ist ein erneuter Modellierungsschritt, bei dem aber die Komplexität anwächst, da für die konkrete Implementierung der Datenbank weitere Informationen notwendig sind.

Im relationalen Datenmodell werden alle Informationen (Objekte und Beziehungen) in Form von zweidimensionalen Tabellen (Relationen) niedergelegt.

Jede Relation besitzt folgende grundlegende Merkmale:

* Die Tabelle ist zweidimensional
* Die Spalten der Tabelle entsprechen den Attributen. Die Anordnung der Spalten innerhalb der Tabelle ist beliebig.
* Die Zeilen der Tabelle entsprechen den Objekten. Die Anordnung der Zeilen in innerhalb der Tabelle ist beliebig.