

4 Die Datenbank Kuchenbestellung

In diesem Kapitel werde ich die Theorie aus Kapitel 2 *Die Datenbank Buchausleihe* an Hand einer weiteren Datenbank *Kuchenbestellung* vertiefen. Da du bereits die Grundzüge von SQL-Abfragen gelernt hast, wirst du hier auch schwierigere Abfragen an diese Datenbank stellen können.

4.1 Kuchenbestellung (Teil 1)

Vor den Weihnachtsferien spendierte Jonas der Klasse zwei Kuchen. Da diese Kuchen so gut geschmeckt haben, wollen nun einige Mitschüler bei ihm Kuchen bestellen. Jonas willigte ein und schon nach kurzer Zeit hat er folgende Bestellung:

<u>Kuchenbestellung bei Jonas</u>			
Robert Drewitz	2x Schneeseiten	a' 3,00 €	}
	für 13.3.2003		
	3x Marmorkuchen	a' 2,00 €	
	für 2.4.2003		}
1x Apfelschnitten	a' 3,50 €		
	für 21.3.2003		15,50 €
Florian Haas	2x Marmorkuchen	a' 2,00 €	}
	für 17.3.2003		
	1x Apfelschnitten	a' 3,50 €	
	für 3.4.2003		}
Denise Pfeiler	2x Frankfurt Kranz	a' 5,00 €	}
	für 29.3.2003		
	1x Kirschbrot	a' 4,00 €	
	für 5.4.2003		}

Abbildung 4.1: Kuchenbestellung „Kuchen01“

Schnell sieht Jonas ein, dass er ohne eine Datenbank den Überblick über die Bestellungen verliert. Mit seinem Wissen über relationale Datenbanken fällt es ihm auch nicht schwer, die Kuchenbestellungen zu organisieren.

4.1.1 Informationsstruktur

Die Informationsstruktur dieser Bestellung ist recht einfach. Es existieren ein paar Kunden, die Kuchen bestellen. Jonas hat offensichtlich seinen Kunden mitgeteilt, dass er an einem Tag nur eine Art von Kuchen liefern kann, aber davon beliebig viele. Er bittet seine Kunden, eine Bestellung sofort zu bezahlen, da er ja Ausgaben zum Kaufen der Zutaten hat. Außerdem kann er sich sicher sein, dass die Kunden den Kuchen abholen, wenn sie diesen im Voraus bezahlt haben.

In der Abbildung 4.1 erkennst du, dass zwei Klassen *Kunde* und *Gebäck* erstellt werden müssen:



Abbildung 4.2: Die beiden Klassen *Kunde* und *Gebäck*

Übung 4.1:

Jonas benötigt in seiner Bestellung zusätzlich noch die Informationen *Datum* und *Anzahl*. Überlege, wo diese untergebracht werden können.

4.1.2 Klassendiagramm

Wie diese beiden Klassen in Beziehung zueinander stehen, wird in einem Klassendiagramm dargestellt.

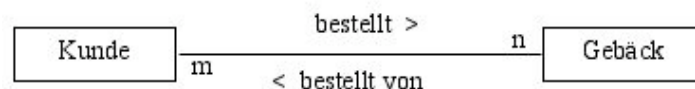


Abbildung 4.3: Das Klassendiagramm

Zwischen den Klassen *Kunde* und *Gebäck* besteht eine n:m-Beziehung.: Jedes Objekt der Klasse *Kunde* kann ein oder mehrere (**m**) Gebäcke bestellen.

Jedes Objekt der Klasse *Gebäck* kann von einem oder mehreren (**n**) Kunden bestellt werden.

Solche n:m-Beziehungen lassen sich nicht direkt in einer relationalen Datenbank implementieren. Vielmehr muss diese durch Definition einer neuen Klasse, beispielsweise *Bestellung*, in zwei 1:m-Beziehungen übergeführt werden.

In dieser Klasse *Bestellung* müssen dann die noch fehlenden Attribute *Datum* und *Anzahl* aufgenommen werden.

4.1.3 Implementierung

Im Prinzip ist jetzt der theoretische Teil Datenbankentwicklung abgeschlossen. Das relationale Datenmodell kann nun mit Hilfe des Werkzeugs ACCESS implementiert werden.

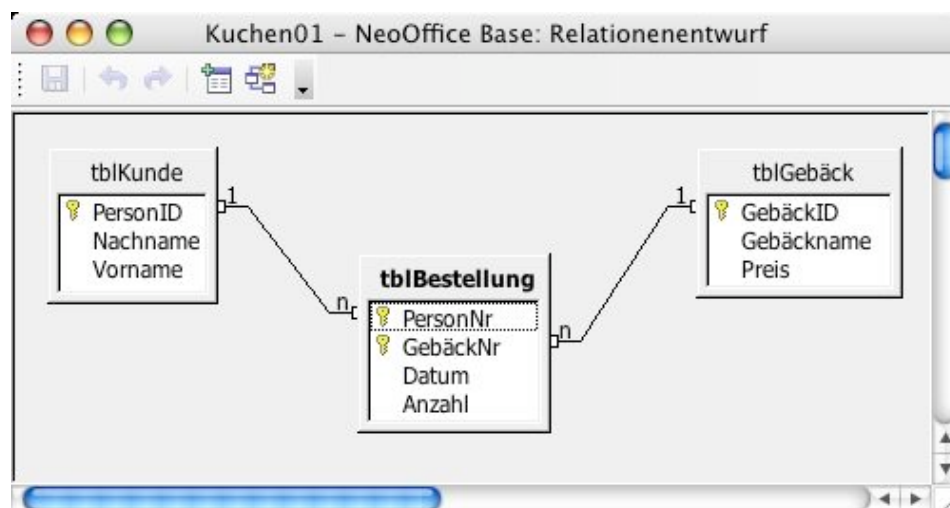


Abbildung 4.4: Die Klassen von *Kuchen01* im Fenster „Relationenentwurf“

4.1.4 Übungsphase

Übung 4.2:

Implementiere die Kuchenbestellung (Teil 1) in Star/NeoOffice. Bezeichne die Datenbank als *Kuchen01*.

Übung 4.3:

Erstelle folgende Abfragen in SQL:

- Welches Gebäck hat Drewnick bestellt?
- Wie hoch ist der Gesamtpreis von Drewnick?
- Von wie vielen Personen wurde Marmorkuchen bestellt?
- Wie oft wurde insgesamt Marmorkuchen bestellt?

- e) Welche Bestellungen müssen im den Monat März ausgeliefert werden?
- f) Welche Bestellungen müssen im [Monat] ausgeliefert werden?
(Parameter-Abfrage)
(In Star/NeoOffice ist mir diese Abfrage noch nicht gelungen)

4.2 Kuchenbestellung (Teil 2)

Übung 4.4:

Jonas muss die Kuchenbestellungen ausliefern. Dazu benötigt er von seinen Kunden die Telefonnummern und die Adressen. Dazu erweitert er die bestehende Datenbank aus Kapitel 4.1 zur Datenbank *Kuchen02*.

- a) Erstelle die Klassen mit allen Attributen und ein Klassendiagramm für die Datenbank *Kuchen02*.

Jonas startet eine Werbeaktion. Dazu teilt er sein Gebäck in die Kategorien Torte, Kuchen und Sonstiges. Auf die Torten gibt er einen Rabatt von 10%, auf Kuchen 15% und auf Sonstiges 20%.

- b) Erweitere das unter a) erstellte Klassendiagramm.
- c) Implementiere dein relationales Datenmodell in Star/NeoOffice.

Nun erstelle folgende Abfragen und überprüfe die gelieferten Ergebnisse mit Hilfe eines Taschenrechners:

- d) Welchen Preis muss Drewnick für die drei Marmorkuchen bezahlen?
- e) Welchen Gesamtpreis muss Drewnick bezahlen?
- f) An welche Kundenadressen müssen die Marmorkuchen geliefert werden?
- g) Weitere Abfragen

4.3 Kuchenbestellung (Teil 3)

Kurze Zeit später bestellen Robert und Denise weitere Gebäcke. Da Jonas jedoch keinen Laptop hat, muss er diese Bestellungen zunächst noch zu den restlichen hinschreiben. Sein Bestellschein sieht nun folgendermaßen aus:

<u>Kundenbestellung bei Jonas</u>			
Robert Denise	2x Streuselkuchen für 13.3.2003	a' 3,00 €	} 15,50 €
	3x Marmorkuchen für 2.4.2003	a' 2,00 €	
	1x Apfelschnitten für 21.3.2003	a' 3,50 €	
Florian Maas	2x Marmorkuchen für 17.3.2003	a' 2,00 €	} 7,50 €
	1x Apfelschnitten für 3.4.2003	a' 3,50 €	
Denise Pfeiler	2x Frankfurt-Käse für 29.3.2003	a' 5,00 €	} 14,00 €
	1x Kieselbrot für 5.4.2003	a' 4,00 €	
Robert Denise	2x Frankfurt-Käse für 10.4.2003	a' 5,00 €	10,00 €
Denise Pfeiler	1x Apfelschnitten für 10.4.2003	a' 3,50 €	} 3,50 €

Abbildung 4.5: Kuchenbestellung „Kuchen03“

Zu Hause angekommen, trägt er diese neuen Bestellungen in seine Datenbank *Kuchen01* ein.

Hinweis:

Verwende als Grundlage für diese Übungseinheit die Datenbank *Kuchen01*. Erstelle also nun eine Datenbank *Kuchen03* und importiere aus *Kuchen01* alle drei Tabellen.

Übung 4.5:

Erstelle die Datenbank *Kuchen03* und füge die beiden neuen Bestellungen hinzu.

Erstelle folgende Abfragen:

- a) Welches Gebäck hat Drewnick bestellt?
- b) Wie hoch ist der Gesamtpreis von Drewnick?

Du hast nun folgende Probleme:

Aus der Abfrage a) kannst du nicht mehr erkennen, welche Kuchen Drewnick in seiner ersten und welche Kuchen er in seiner zweiten Bestellung beauftragt hat.

Aus der Abfrage b) kannst du nur den Gesamtbetrag aus Drewnicks beiden Bestellungen ermitteln. Er hat jedoch seine erste Bestellung bereits gezahlt und möchte also nun den Betrag der zweiten Bestellung abfragen.

Eine Lösung dieser Probleme wäre die Löschung des ersten Auftrags. Was passiert aber dann, wenn Drewnick die zweite Bestellung vor dem Auslieferungstermin des Marmorkuchens aufgibt. Eine Löschung der ersten Bestellung würde dann unweigerlich zum Verlust dieser Daten führen. Dies ist natürlich nicht sinnvoll.

Für die Lösung dieser Probleme solltest du noch einmal die Informationsstruktur betrachten, die diesem neuen Problem zu Grunde liegt:



Abbildung 4.6: Informationsstruktur von Kuchenbestellung *Kuchen03*

Diese Informationsstruktur wird nun in das Klassendiagramm übergeführt.

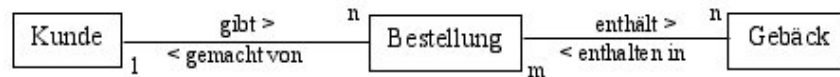


Abbildung 4.7: Klassendiagramm von *Kuchen03*

Jedes Objekt der Klasse *Kunde* kann ein oder mehrere (**m**) Bestellungen aufgeben.

Jedes Objekt der Klasse *Bestellung* ist von genau einem (**1**) Kunden in Auftrag gegeben worden.

Jedes Objekt der Klasse *Bestellung* kann ein oder mehrere (**m**) Gebäcke enthalten.

Jedes Objekt der Klasse *Gebäck* kann in einem oder mehreren (**m**) Bestellungen enthalten sein.

Vergleiche diese neue Version mit der ersten Datenbank *Kuchen01*:

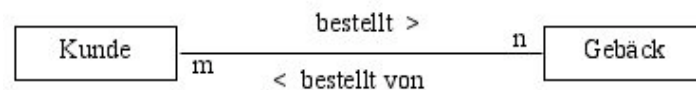


Abbildung 4.8: Das Klassendiagramm von *Kuchen01*

Kunde und Gebäck stehen in einer n:m-Beziehung zueinander. In einer relationalen Datenbank müssen solche Beziehungen durch zwei 1:m-Beziehungen aufgelöst werden.

Neu ist nun, dass diese Auflösung in unserem Fall jedoch eine 1:m-Beziehung zwischen *Kunde* und *Bestellung* und eine weitere n:m-Beziehung zwischen *Bestellung* und *Gebäck* ergibt. Diese wird in einem nächsten Schritt mit Hilfe einer weiteren Klasse *Enthalten* wiederum zerlegt.

Übung 4.6:

Implementiere das oben erstellte relationale Datenmodell in Star/NeoOffice als *Kuchen04*. Übertrage den neuen Bestellschein „Kuchenbestellung bei Jonas“ in die Datenbank *Kuchen04*.

Erstelle nun folgende Abfragen:

- Welches Gebäck hat Drownick bestellt?
- Welche Einzelpreise sind in der Bestellung 0?
- Welchen Gesamtpreis hat die Bestellung 0?
- Welche Einzelpreise sind in einer bestimmten Bestellung? (Parameter-Abfrage)
- Welchen Gesamtpreis hat eine bestimmte Bestellung?

- (Parameter-Abfrage)
- f) Welche Kunden haben Marmorkuchen bestellt?

4.4 Kuchenbestellung (Teil 4)

Weiteres Vorgehen:

Nun könnte man die Datenbanken *Kuchen02* und *Kuchen04* zusammenfassen. Hierbei kann man zusätzlich ein Auftragsdatum, ein Rechnungsdatum und den Zahlungseingang berücksichtigen.