Modul 100: Daten analysieren und strukturieren

Einführung in relationale Datenbanken

Diese Einführung ist relativ knapp und beschränkt sich auf diejenigen Aspekte von relationalen Datenbanken, die für dieses Modul wesentlich sind. Eine allgemeinere und ausführlichere Einführung in relationale Datenbanken erfolgt im Modul 104 im 2. Semester.

In diesem Dokument werden z.T. Begriffe verwendet, die Sie bereits aus vorangehenden Arbeitsblättern. Oder es kommen Begriffe und Definitionen vor, die später nochmals erklärt werden. Diese Wiederholung wird bewusst und mit Absicht gemacht: Einerseits ist es wichtig, dass Sie genau wissen, was die Begriffe bedeuten. Eine zweite, andere Definition kann für das Verständnis helfen. Andererseits werden die Begriffe hier ausschliesslich aus Sicht einer relationalen Datenbank betrachtet und nicht allgemein wie in andern Arbeitsblättern.

Was ist eine Datenbank?

Eine der wichtigsten Rechneranwendungen ist die Speicherung, Verwaltung und Manipulation beliebiger Informationen. Datenbanken erfüllen diesen Verwendungszweck am besten.

Der Begriff *Datenbank* bezeichnet zwei verschiedene Dinge: Zum einen die Datensammlung selbst, zum anderen das Programm, das diese Daten verwaltet. Bei den Daten handelt es sich um eine nach bestimmten Regeln strukturierte Ansammlung von Informationen zu verschiedenen Themengruppen, beispielsweise Kundendaten von Unternehmen, Diagnose- und Behandlungsinformationen von Ärzten oder die private CD-Sammlung eines Musikfans.

Das Anwendungsprogramm, mit dem diese Daten verwaltet werden können, enthält mehr oder weniger mächtige Funktionen zum Eingeben, Ändern, Suchen, Sortieren, Filtern und formatierten Ausgeben dieser Daten. Ein solches Programm wird als *Database Management System* (DBMS), also als Datenbankverwaltungssystem, bezeichnet.

Was ist eine relationale Datenbank?

Das relationale Datenbankmodell ist das am weitverbreitetste Datenmodell, welches in der Datenbankentwicklung als Standard genutzt wird.

Das relationale Datenbankmodell besteht aus drei wichtigen Bausteinen**:**

- Tabellen
- Attributen
- Beziehungen

Ein relationales Datenbankmodell ist eine Ansammlung von *Tabellen*, die miteinander verknüpft sind. Jede Zeile (auch Tupel genannt) in einer Tabelle ist ein Datensatz. Jedes Tupel besteht aus einer Reihe von Eigenschaften (Attributen), den Spalten der Tabelle.

Relation

Was bedeutet der Begriff Relation? Allgemein formuliert besteht eine Relation aus Attributen und Tupeln. In relationalen Datenbanken wird eine Relation durch eine Tabelle beschrieben. Einerseits ist also eine Tabelle eine Relation. Andererseits bezeichnet man

auch die Verknüpfung von Tabellen als Relation. Bei einer Beziehung zwischen zwei Tabellen stellen schliesslich ihre Elemente die Vereinigung der Elemente der beiden verknüpften Relationen dar.

Tabellen

Grundlegendes Konzept der relationalen Datenbank ist die *Tabelle*. Eine Tabelle stellt eine zweidimensionale Matrix aus Zeilen und Spalten dar. Während die Zeilen der Tabelle *Datensätze* repräsentieren, stellen die Spalten *Attribute* oder *Merkmale* dar.

Beispiel

Tabelle *Kunden*

KID	Name	Strasse	Hausnr	PLZ	Ort
1	Müller	Gantrischweg	55	3000	Bern
2	Becker	Albisstrasse	121	8000	Zürich
4	Wenger	Grüne Allee	30	3400	Burgdorf
5	Abegglen	Hopfenweg	12a	3000	Bern

Attribute

Die Tabelle *Kunden* weist eine Struktur auf. Ohne Struktur würde die Tabelle wie folgt aussehen:

Kunde
Müller, Gantrischweg 55, 3000 Bern
Becker, Albisstrasse 121, 8000 Zürich
Wenger, Grüne Allee 30, 3400 Burgdorf
Abegglen, Hopfenweg 12a, 3000 Bern

Ohne Struktur wäre es schwierig, die Daten einzufügen und auszuwerten. Weitere Beispiele von Datenbeständen ohne Struktur sind Bilder im JPG-Format oder Dokumente im Word-Format.

Die Struktur besteht nun darin, dass wir *Attribute* oder *Merkmale* definieren: KID (= Kunden-ID oder Kundennummer), Name, Strasse, Hausnummer, Postleitzahl und Ort. Jedes Attribut hat einen bestimmten Wertebereich: Der *Name* kann z.B. maximal 50 Zeichen lang sein und alle druckbaren Zeichen aufnehmen. Die *PLZ* enthält immer eine vierstellige, ganze Zahl. Der Wertebereich muss auf jeden Fall eingehalten werden, z.B. können bei der Postleitzahl keine Buchstaben eingegeben werden.

Attributwerte

Die Attribute sind die Spalten der Tabellen. Jedes Attribut wird durch einen Namen und einen Wertebereich definiert. Attributwerte sind die effektiven Werte, die den Attributen in den Datensätzen zugewiesen werden. Z.B. ist *Ort* das Attribut und *Bern* ist der Attributwert von den Datensätzen 1 und 5.

Beziehungen

Eine Beziehung bedeutet, dass zwei Tabellen miteinander verknüpft werden. Die Verknüpfung (Beziehung) wird über den Primär- und Fremdschlüssel hergestellt: Ein Fremdschlüssel in einer Tabelle verweist auf den Primärschlüssel einer anderen Tabelle.

Wir ergänzen nun die Tabelle *Kunden* mit der Tabelle *Artikel* und stellen eine Beziehung zwischen den beiden Tabellen her. Dazu erweitern wir die Tabelle *Kunden* mit dem Attribut *FK_AID*. *FK_AID* ist ein Fremdschlüssel, der auf das Attribut *AID* der Primärtabelle *Artikel* verweist.

Im Moment kann somit jeder Kunde genau einen Artikel bestellen. In der Realität ist das natürlich nicht so, jeder Kunde kann beliebig viele Artikel bestellen. Aus didaktischen Gründen erarbeiten wir die "korrekte" Lösung jedoch stufenweise.

Tabelle *Artikel*

AID	Artikel	Preis
3	BenQ Monitor BL2405HT	179.00
5	Samsung SSD Portable T3 500 GB	209.00
8	HP Drucker Color LaserJet Pro MFP M377dw	369.00
9	Kingston DataTraveler microDuo 3C 128GB	58.00

Tabelle *Kunden*

KID	Name	Strasse	Hausnr	PLZ	Ort	FK_AID
1	Müller	Gantrischweg	55	3000	Bern	5
2	Becker	Albisstrasse	121	8000	Zürich	9
4	Wenger	Grüne Allee	30	3400	Burgdorf	5
5	Abegglen	Hopfenweg	12a	3000	Bern	3

Es gibt drei Arten von Beziehungen:

- Eine 1:1-Beziehung verknüpft einen Datensatz einer Tabelle mit genau einem Datensatz einer anderen Tabelle. Dies ist immer dann nützlich, wenn bestimmte Informationsaspekte nicht so oft benötigt werden wie andere Aspekte. Diese Beziehung kommt in der Praxis am wenigsten oft vor.
- Eine 1:n-Beziehung oder Eins-zu-viele-Beziehung verbindet einen Datensatz einer Tabelle mit beliebig vielen Datensätzen einer anderen Tabelle. Dies ist der häufigste Verknüpfungstyp. Detailinformationen über Werte, die in einer Spalte einer Tabelle oft vorkommen können, werden in einer separaten Tabelle erfasst. Bsp.: Jeder Kunde kann im Beispiel oben genau einen Artikel bestellen. Die Artikel werden in einer separaten Tabelle geführt, weil jeder Artikel von mehreren Kunden bestellt werden kann.
- Eine m:n-Beziehung, auch Viele-zu-viele-Beziehung genannt, kombiniert beliebig viele Vorkommen eines bestimmten Wertes mit beliebig vielen Vorkommen eines anderen.
  Bsp.: Jeder Kunde kann beliebig viele Artikel bestellen. Jeder Artikel kann von beliebig vielen Kunden bestellt werden. Das können wir nicht mehr mit den zwei Tabellen oben abbilden. Dazu braucht es eine dritte Tabelle, nämlich eine Zwischentabelle. Wir werden das gleich genauer anschauen.

Primärschlüssel

Wie wir in der Tabelle *Kunden* sehen, wird jeder Datensatz mit einer eindeutigen Nummer identifiziert: 1 = Müller, 2 = Becker, 3 = Wenger. Deshalb wird der Primärschlüssel auch *Identifikationsmerkmal* genannt.

Eigenschaften:

- Jeder Datensatz hat als Primärschlüssel einen eindeutigen Wert, der nur 1x vorkommen darf.
- Der Primärschlüssel identifiziert den Datensatz.
- Wertebereich von Primärschlüsseln: Ganze, positive Zahlen.
- Für jede (!) Tabelle sollte ein Primärschlüssel definiert werden.
- Der Primärschlüssel ist nicht für den Endbenutzer, sondern für den Applikationsentwickler wichtig: Anhand des Schlüssels werden Datensätze identifiziert, mutiert oder gelöscht.
  Bsp.: Wenn wir den Müller löschen wollen, können wir nicht einfach sagen "lösche mir die Datensätze mit dem Namen *Müller*", weil möglicherweise mehrere Müller existieren. Der richtige Müller wird via Primärschlüssel identifiziert und gelöscht.
- Jeder Primärschlüsselwert darf nur 1x vorkommen. Jedoch ist es durchaus möglich, dass gewisse Werte nicht existieren. Z.B. fehlt in der Tabelle *Kunden* die Nr. 3.
- Die Verwaltung des Primärschlüssels überlässt man normalerweise dem DBMS. Für jeden neuen Datensatz wird automatisch der bisher höchste Wert ermittelt und um 1 erhöht. In SQLite heisst diese Eigenschaft AUTOINCREMENT.
- Es ist natürlich auch möglich, den Primärschlüssel manuell zu setzen. Dies bedeutet jedoch einen (unnötigen) Verwaltungsaufwand und man muss aufpassen, dass man nicht versucht, einen bereits existierenden Schlüssel einzufügen.

Fremdschlüssel

FK_AID in der Fremdtabelle *Kunden* ist ein Fremdschlüssel, der auf die Primärtabelle *Artikel* zeigt: *FK_AID* darf nur Werte enthalten, die im Primärschlüssel *AID* in der Tabelle *Artikel* vorkommen.

Eigenschaften:

- *FK_AID* zeigt auf den Primärschlüssel *AID* der Tabelle *Artikel*.
- *FK_AID* muss denselben Wertebereich haben wie *AID*.
- *FK_AID* darf nur Werte enthalten, die in *AID* der Tabelle *Artikel* vorhanden sind. Zudem ist ein leerer Wert (NULL) möglich.

Komplexe Beziehungen

Wir wollen unsere Datenbank nun erweitern: Jeder Kunde kann beliebig viele Artikel bestellen, jeder Artikel kann von beliebig vielen Kunden bestellt werden. Man könnte nun versucht sein, die Daten wie folgt einzugeben:

Tabelle *Kunden*

KID	Name	Strasse	Hausnr	PLZ	Ort	FK_AID
1	Müller	Gantrischweg	55	3000	Bern	5, 8
2	Becker	Albisstrasse	121	8000	Zürich	3, 5, 9

"Jeder Kunde kann beliebig viele Artikel bestellen, jeder Artikel kann von beliebig vielen Kunden bestellt werden" ist erfüllt, denn im Attribut *FK_AID* können beliebig viele Artikelnummern eingetragen werden.

Also ist alles ok? Leider nicht. Denn wir haben nun mehrere Werte im selben Datenfeld *FK_AID*, und damit ist eine wichtige Bedingung für relationale Datenbanken verletzt: Jedes Attribut muss einen atomaren Wertebereich haben. Denn Abfragen und Änderungen in der Datenbank werden erleichtert bzw. überhaupt erst möglich, wenn alle Attributwertebereiche atomar sind.

Wir müssen eine neue, taugliche Lösung finden. Diese erhalten wir, indem wir eine neue Zwischentabelle erstellen. Die drei Tabellen sehen wie folgt aus:

Tabelle *Artikel*

AID	Artikel	Preis
3	BenQ Monitor BL2405HT	179.00
5	Samsung SSD Portable T3 500 GB	209.00
8	HP Drucker Color LaserJet Pro MFP M377dw	369.00
9	Kingston DataTraveler microDuo 3C 128GB	58.00

Tabelle *Kunden*

KID	Name	Strasse	Hausnr	PLZ	Ort
1	Müller	Gantrischweg	55	3000	Bern
2	Becker	Albisstrasse	121	8000	Zürich
4	Wenger	Grüne Allee	30	3400	Burgdorf
5	Abegglen	Hopfenweg	12a	3000	Bern

Tabelle *Bestellungen*

BID	FK_KID	FK_AID
1	1	5
2	1	3
3	1	9
4	4	8
5	4	3
6	4	5

Die Fremdtabelle *Bestellungen* stellt zwei Beziehungen her: Via *FK_KID* zur Primärtabelle *Kunden* und via *FK_AID* zur Primärtabelle *Artikel*.

Bedeutung der 6 Datensätze:

1. Müller hat die Samsung SSD bestellt
2. Müller hat den BenQ Monitor bestellt
3. Müller hat den Kingston USB Flashdrive bestellt
4. Wenger hat den HP Drucker bestellt
5. Wenger hat den BenQ Monitor bestellt
6. Wenger hat die Samsung SSD bestellt

Unser Bestellsystem ist schon recht gut, hat aber noch Mängel. Z.B. fehlt ein Bestelldatum, der Artikel kann nicht mehrfach bestellt werden und es können nicht mehrere Artikel zu einer Bestellung zusammengefasst werden. Für unsere Zwecke reicht die Datenstruktur aber.

Redundanz und Konsistenz

Redundanz

Redundanzen sind doppelte Informationen in einer Datenbank. Wenn Daten mehrfach gespeichert werden,

1. führt dies zu einem unnötigen Arbeitsaufwand, z. B. muss eine Änderung mehrfach vorgenommen werden.
2. wird Speicherplatz verschwendet.
3. können Mutationsanomalien auftreten, wenn nicht alle Daten an allen Stellen geändert werden.

Man spricht von einer redundanzfreien Datenbank, wenn alle doppelte Informationen entfernt werden können, ohne dass ein Informationsverlust stattfindet**.**

Beispiel 1 Redundanzen:

Vorname	Nachname	Vollname	Strasse	Hausnr	PLZ	Ort
Olaf	Müller	Olaf Müller	Gantrischweg	55	3000	Bern

In diesem Beispiel ist der *Vollname* redundant. Dieser kann aus *Vorname* und *Nachname* zusammengesetzt werden.

Beispiel 2 Redundanzen:

KID	Name	Strasse	Hausnr	PLZ	Ort
1	Müller	Gantrischweg	55	3000	Bern
2	Becker	Albisstrasse	121	8000	Zürich
4	Wenger	Grüne Allee	30	3400	Burgdorf
5	Abegglen	Hopfenweg	12a	3000	Bern

In diesem Beispiel sind *PLZ* und *Ort* redundant: *3000* und *Bern* kommen 2x vor. Was passiert, wenn der Ortsname geändert wird (was auch schon vorgekommen ist)? Der Name muss an zwei Stellen geändert werden, was früher oder später zu Inkonsistenzen führen wird (siehe nächstes Kapitel).

Beispiel 3 Redundanzen:

BID	FK_KID	FK_AID
1	1	5
2	1	3
3	1	9
4	4	8
5	4	3
6	4	5

Handelt es sich bei den doppelten Einträgen der Attribute *FK_KID* und *FK_AID* um Redundanzen? In der Literatur wird meist von den unvermeidbaren Redundanzen der Fremdschlüssel gesprochen. Tatsächlich kommt der Attributwert *1* im Attribut *FK_KID* mehrfach vor. Es handelt sich aber nicht um eine doppelte Information, die weggelassen werden könnte. Denn der Wert ist notwendig, um das Ereignis des Kaufes abzubilden.

Deshalb wollen wir nur dann von Redundanzen sprechen, wenn es sich um effektive Daten handelt. Also Daten aus der realen Welt, die der Benutzer verarbeiten will. Primär-

und Fremdschlüssel sind lediglich Hilfskonstrukte, um die Daten der realen Welt in einer relationalen Datenbank abbilden zu können.

Konsistenz

Die Konsistenz einer Datenbank ist dann gegeben, wenn die Daten frei von Widersprüchen sind. Bei den Beispielen 1 und 2 oben ist die Wahrscheinlichkeit gross, dass früher oder später Widersprüche auftreten, dass also die Daten, die identisch sein sollten, unterschiedlich sind.

Um die Konsistenz der Daten sicherzustellen, stehen uns hauptsächlich folgende zwei Möglichkeiten zur Verfügung:

1. Wir stellen sicher, dass es keine Redundanzen gibt in der Datenbank. Dazu wird der Normalisierungsprozess angewendet. Unter Normalisierung versteht man die Aufteilung von Attributen in mehrere Tabellen mithilfe der Normalisierungsregeln und deren Normalformen.
   Die Normalisierung ist nicht Thema dieses, sondern des Folgemoduls 104. In diesem Modul ist das Datenbankschema vorgegeben oder Sie erarbeiten das Schema intuitiv mithilfe der Lehrperson und diesem Dokument.
2. Es werden sogenannte Integritätsbedingungen definiert. Die Integritätsbedingungen sollen dafür sorgen, dass keine unkorrekten Daten in die Datenbank gelangen. Eine Regel betrifft die Wertebereichsintegrität: Für jedes Attribut legen wir einen Wertebereich fest. Werte ausserhalb dieses Bereichs können nicht eingefügt werden. Eine andere Regel betrifft die referentielle Integrität: Wir definieren für einen Fremdschlüssel eine Einschränkung (CONSTRAINT), die besagt, dass nur gültige Werte eingefügt werden können. Gültige Werte sind Primärschlüsselwerte der Primärtabelle. Neben den gültigen Werten ist je nach Fall auch ein leerer Wert (NULL) möglich.
   Auch in diesem Fall gilt: Wir arbeiten zwar mit Primär- und Fremdschlüssel, in diesem Modul definieren wir aber keine CONSTRAINTs, d.h. wir können ungültige Werte im Fremdschlüssel erfassen. Die CONSTRAINTs wiederum sind Thema des Folgemoduls 104.

Um nun unsere Datenbank in den gewünschten Zustand ohne Redundanzen und frei von Inkonsistenzen zu bringen, müssen wir *PLZ* und *Ort* der Tabelle *Kunden* in eine weitere Tabelle auslagern.

Tabelle *Ort*

OID	PLZ	Ort
1	3000	Bern
2	3400	Burgdorf
3	8000	Zürich

Tabelle *Kunden*

KID	Name	Strasse	Hausnr	FK_OID
1	Müller	Gantrischweg	55	1
2	Becker	Albisstrasse	121	3
4	Wenger	Grüne Allee	30	2
5	Abegglen	Hopfenweg	12a	1

Die Fremdtabelle *Kunden* stellt via *FK_OID* eine Beziehung zur Primärtabelle *Ort* her.

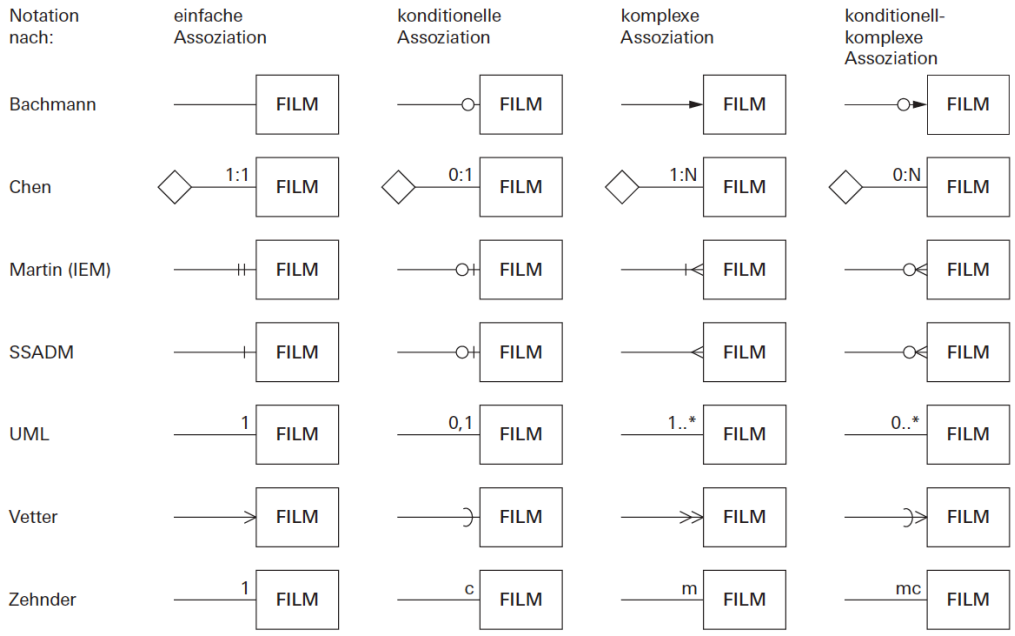
Entity-Relationship-Modell

Bisher haben wir unser Datenbankmodell schrittweise und intuitiv hergeleitet. Zum Schluss wollen wir noch das erarbeitete Modell aufzeichnen. Dazu wird üblicherweise das Entity-Relationship-Modell ERM verwendet. Dieses dient dazu, im Rahmen eines Softwareprojektes einen Ausschnitt aus der realen Welt zu beschreiben. Das ER-Modell besteht aus einem Diagramm und einer Beschreibung der darin verwendeten Elemente. Im Rahmen dieses Moduls ist nur das Diagramm relevant, dieses wird Entity-Relationship-Diagramm ERD genannt.

In der Softwareentwicklung wird natürlich systematisch und nicht intuitiv vorgegangen, aufgrund der Anforderungen des Kunden werden drei aufeinanderfolgende Datenbankmodelle erstellt:

1. Konzeptionelles Datenmodell
2. Logisches Datenmodell
3. Physikalisches Datenmodell

Verbreitete Notationen für ERD



Wir verwenden im Modul 100 die Zehnder- oder die Martin-Notation (auch *Krähenfussnotation* genannt).

In der Zehnder-Notation bedeutet:

1 = genau 1
c = 0 oder 1
m = 1 oder mehrere
mc = 0, 1 oder mehrere

Unterschiedliche Begriffe

Die Begriffe, die in einem relationalen DBMS verwendet werden, sind unterschiedlich zu den Begriffen des Entity-Relatioinship-Modells. Nachfolgend eine Übersicht über die Begriffe, ihre Bedeutung und alternative Bezeichnungen.

Begriffe rund um die Tabelle:

Tabelle	Relationale Datenbank	Entity-Relationship-Modell
Tabellendefinition	Relationstyp, Relationsschema	Entitätstyp
Spalte	Attribut	Attribut
Zeile	Tupel, Datensatz	Entität
Inhalt	Relation	Entitätsmenge
Zelle	Attributwert	Attributwert
Beziehung	Beziehung, Relation	Beziehung, Relation

Synonyme:

Begriff	Synonym
Tabelle	Relation, Entitätstyp, Entitätsmenge
Tabellenzeile	Tupel, Datensatz, Entität
Tabellenspalte	Attribut, Merkmal
Wertebereich	Domäne
Relation	Tabelle, Beziehung
Beziehung	Relation, Verknüpfung (von Tabellen)

Konzeptionelles Datenmodell

Das konzeptionelle Modell bietet den höchsten Abstraktionsgrad und stellt eine globale, unternehmensweite Sicht auf die in der Datenbank zu verwaltenden Daten dar. Das konzeptionelle Modell wird aufgrund der Anforderungen des Kunden erstellt und dient zur Überprüfung mit dem Kunden, ob an alles gedacht wurde. Aufgrund des Übersichtscharakters des konzeptionellen Modells enthält es keine Details.

Hinweis: Der Kunde, der einen Artikel bestellt, ist nicht zu verwechseln mit dem Kunden, für den wir die Datenbank erstellen. Dieser Kunde ist der Auftraggeber und Benutzer der Datenbank, der seinerseits seine Kunden eingibt.

Wie Sie sehen, wird die Zwischentabelle *Bestellungen* nicht dargestellt. Den Auftraggeber interessiert diese Tabelle nicht. Ihn interessiert nur, dass seine Kunden seine Artikel bestellen können.

Beziehung *Kunden – Artikel* ist mc:m: Jeder Kunde kann einen oder mehrere Artikel bestellen (es ist nur Kunde, wer mind. einen Artikel bestellt). Jeder Artikel wird von keinem, einem oder mehreren Kunden bestellt.

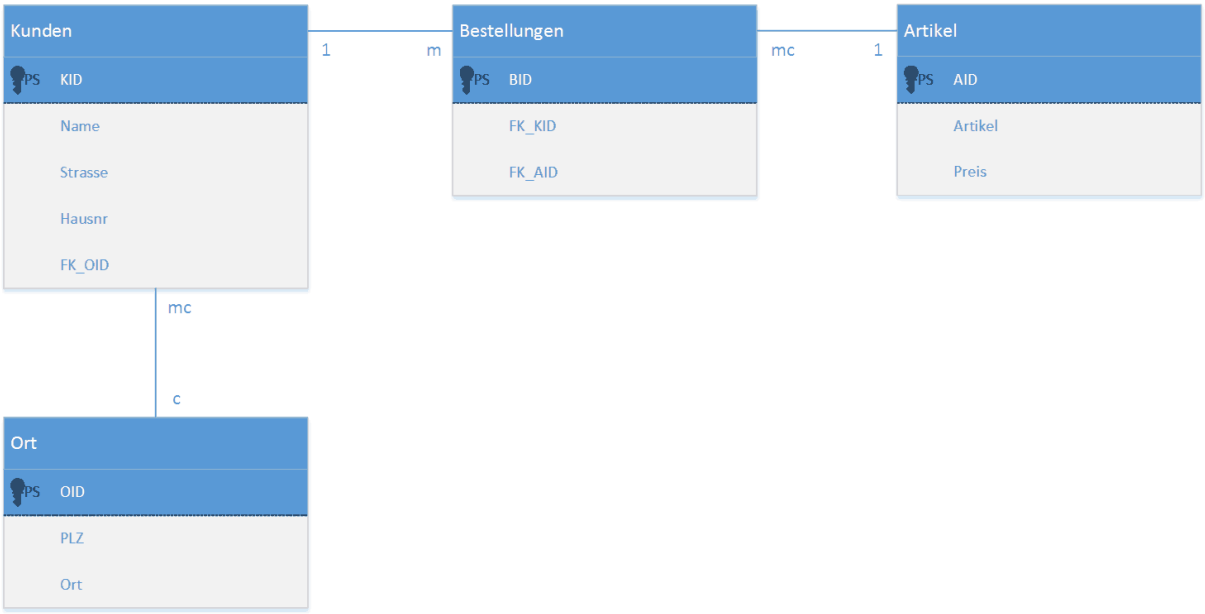
Beziehung *Kunden – Ort* ist mc:c: Jedem Kunden ist kein oder ein Ort zugewiesen (er wohnt in diesem Ort, evtl. ist der Ort unbekannt). In jedem Ort wohnen kein, ein oder mehrere Kunden.

Im Modul 100 arbeiten wir vorwiegend mit dem konzeptionellen Datenmodell.

Logisches Datenmodell

Das logische Modell wird aus dem konzeptionellen heraus entwickelt und stellt das Datenmodell so dar, wie es vom DBMS gesehen wird. Das Modell ist je nach verwendetem DBMS unterschiedlich: Das logische Datenmodell einer relationalen Datenbank ist also verschieden von demjenigen einer objektorientierten, hierarchischen oder Netzwerk-Datenbank.

Der wichtigste Punkt, den Sie bei der Überführung Ihres Datenmodells in ein logisches Modell beachten müssen: Es mit einem relationalen DBMS nicht möglich, direkt m:n-Beziehungen aufzulösen. Daher müssen Sie für alle m:n-Beziehungen eine zusätzliche Tabelle einführen, die die m:n-Beziehung in zwei 1:m-Beziehungen umwandelt (das ist in diesem Fall die Tabelle *Bestellungen*).



Das logische Datenmodell wird Thema im Modul 104 sein.

Physikalisches Datenmodell

Das physikalische Modell ist das Modell mit der geringsten Abstraktion, da es festlegt, wie die Daten auf den Datenträgern gespeichert werden. Bei relationalen Datenbanken tritt das physikalische Modell etwas in den Hintergrund, da der Designer in der Regel nicht weiss, wie genau die Daten in den vom DBMS verwalteten Dateien organisiert sind. Nichtsdestotrotz kann man bei relationalen Datenbankanwendungen die Performance

optimieren, indem man sich z.B. Gedanken darüber macht, welche Datenbankdateien auf welchen Festplatten gespeichert werden, wie die Datenbank den Hauptspeicher verwendet oder wie Indices physikalisch organisiert werden.

Namensgebung

Im Modul 100 wollen wir folgende Regeln für die Namen von Tabellen und Attributen anwenden:

- Tabellen sollen einen sprechenden Namen haben.
- Zwischentabellen für m:n-Beziehungen haben entweder einen sprechenden Namen wie *Bestellungen* im Beispiel oben. Alternativ dazu können die beiden Tabellen mit Unterstrich verwendet werden: *Kunden_Artikel*.
- Der Name von Primärschlüsseln wird aus dem ersten Buchstaben des Tabellennamens und *ID* zusammengesetzt, z.B. *KID* für die Tabelle *Kunden*.
- Der Name von Fremdschlüsseln wird zusammengesetzt aus *FK_* (für Foreign Key) und dem Namen des referenzierten Primärschlüssels, z.B. *FK_OID* für den Fremdschlüssel, der auf den Primärschlüssel *OID* der Tabelle *Ort* verweist.
- Alle übrigen Attribute sollen sprechende Namen haben, z.B. *Strasse* für die Strasse des Kunden.

Repetitorium AB100-06:

Aufgabe 1

Begriffe:

A	Attributwert	**B**	Redundanz
C	Relation	**D**	Beziehung
E	Inkonsistenz	**F**	Attribut
G	Primärschlüssel	**H**	Fremdschlüssel
I	Wertebereich	**J**	Integritätsbedingung
K	Tabelle	**L**	Komplexe Beziehung

Welcher der oben aufgeführten Begriffe passt jeweils am besten zu den unten aufgeführten Beschreibungen? Mehrfachnennungen sind möglich (mehrere Begriffe passen zu einer Beschreibung bzw. ein Begriff passt zu mehreren Beschreibungen).

Beschreibung:

Muss denselben Wertebereich haben wie das referenzierte Attribut.
A☐ B☐ C☐ D☐ E☐ F☐ G☐ H☐ I☐ J☐ K☐ L☐

Kann auftreten, wenn es Redundanzen gibt in einer Datenbank.
A☐ B☐ C☐ D☐ E☐ F☐ G☐ H☐ I☐ J☐ K☐ L☐

Regeln, die sicherstellen, dass keine Inkonsistenzen entstehen können.
A☐ B☐ C☐ D☐ E☐ F☐ G☐ H☐ I☐ J☐ K☐ L☐

Besteht aus Attributen und Datensätzen.
A☐ B☐ C☐ D☐ E☐ F☐ G☐ H☐ I☐ J☐ K☐ L☐

Eine Viele-zu-viele-Beziehung (m:n).
A☐ B☐ C☐ D☐ E☐ F☐ G☐ H☐ I☐ J☐ K☐ L☐

Doppelte Daten, die ohne Informationsverlust weggelassen werden können.
A☐ B☐ C☐ D☐ E☐ F☐ G☐ H☐ I☐ J☐ K☐ L☐

Die Menge der möglichen Werte, die ein Attribut annehmen kann.
A☐ B☐ C☐ D☐ E☐ F☐ G☐ H☐ I☐ J☐ K☐ L☐

Attribut, das keine doppelten Werte enthalten darf.
A☐ B☐ C☐ D☐ E☐ F☐ G☐ H☐ I☐ J☐ K☐ L☐

Eine Verknüpfung von zwei Tabellen.
A☐ B☐ C☐ D☐ E☐ F☐ G☐ H☐ I☐ J☐ K☐ L☐

Definition der Spalte einer Tabelle.
A☐ B☐ C☐ D☐ E☐ F☐ G☐ H☐ I☐ J☐ K☐ L☐

Einzelner Datenwert eines Datensatzes.
A☐ B☐ C☐ D☐ E☐ F☐ G☐ H☐ I☐ J☐ K☐ L☐

Kommt in einer korrekt konzipierten Datenbank nicht vor.
A☐ B☐ C☐ D☐ E☐ F☐ G☐ H☐ I☐ J☐ K☐ L☐

Aufgabe 2 (Zusatzaufgabe)

Wir wollen unser Bestellsystem perfektionieren: Jeder Kunde soll mehrere Bestellungen mit Datum tätigen können. Pro Auftrag kann er beliebig viele Artikel bestellen, d.h. jeder Auftrag stellt genau ein Bestellereignis dar.

Erweitern Sie die bestehende Datenbank und entwerfen Sie ein taugliches konzeptionelles Datenmodell.

Weitere Herausforderung: Entwickeln Sie aus dem konzeptionellen das logische Datenmodell.