Konstruktion und Untersuchung eines Datenbanksystems für Kaffeemaschinen-Logistik-Koordination

Stefan Klagge
Matrikel-Nr. 1443985
E-Mail: kosmo@tzi.de

Jan-C. Kranefeld
Matrikel-Nr. 1337718
E-Mail: jck@tzi.de

13.02.2004

Inhaltsverzeichnis

1 Einleitung

In der vorliegenden Arbeit soll ein Datenbank-System zur Abwicklung von Bestellungen für Kaffeemaschinen konstruiert werden. Dieses System ist völlig frei von uns erdacht und existiert real nicht.
Hierzu werden zunächst die Erfordernisse von Seiten der Anwender-Sicht in der Anwendungsbeschreibung erläutert, also beschrieben, welche Informationsobjekte ein solches Datenbanksystem umfassen können müsste.
Im nächsten Schritt müssen aus den so ermittelten Informationen sinnvolle Attribut-Tupel gebildet werden, was im Kapitel *Relationenbildung und Normalisierung* geschieht.
Aus der Beschreibung der daraus resultierenden Datenstruktur wird danach ein Konzept für das Datenbanksystem in Form eines *Entity-Relationship-Models* (ERM) dargestellt. Die verwendeten Sprachmittel werden zuvor erklärt.
Die Datenbank-Konzeption wird danach als objektrelationales Klassendiagramm mit Mitteln der *Unified Modeling Language* (UML) beschrieben. Hier werden neben Attributen und Klassenbeziehungen nun die Methoden eingeführt, die auf den jeweiligen Objekten ausgeführt können werden. Zu diesen werden dann SQL-Terme erstellt, die korrekte Operationen in Datenbanksystem-Anwendungen ermöglichen sollen.
Beim Datenbank-Entwurf wird nach dem sog. Drei-Ebenen-Konzept vorgegangen. Diese Ebenen werden dabei in eine „externe“, „konzeptionelle“und eine „interne“unterschieden.
Die *externe* Ebene betrachtet dabei den Blickwinkel aus Anwendersicht, also die fachlichen Anforderungen, die an das zu erstellende Datenbanksystem zu stellen sind. Es geht hier um den Informationsgehalt der Daten, die dargestellt werden sollen.
Die *konzeptionelle* Ebene ist ein Schema, nach welchem die zu speichernden Daten organisiert werden sollen. Außerdem werden hier die Beziehungen

definiert, die zwischen den einzelnen Objekten bestehen. Was auf konzeptioneller Ebene nicht definiert wird, kann später auf Benutzerebene auch nicht gespeichert oder abgerufen werden.
Die *interne* Ebene beschäftigt sich mit der physischen Speicherung der Daten, also z. B. die Netzwerk-Struktur des Unternehmens, in dem Datenbankserver und Clients definiert werden.
In der vorliegenden Arbeit soll nach Erörterung der (fiktiven) Anforderungsdefinition, die also der „externen Ebene“entspricht, der Schwerpunkt auf der „konzeptionellen Ebene“liegen. Mit der „internen Ebene“wollen wir uns hier jedoch nicht beschäftigen.

2 Anwendungsbeschreibung

Das fiktive System soll in der Lage sein, die Auslieferungs-Logistik für Kaffeemaschinen zu unterstützen. Die Kunden des Unternehmens bestellen dabei beliebig viele Kaffeemaschinen ihrer Wahl. Daraufhin soll von einem Mitarbeiter des Unternehmens ein Auftrag erfasst werden können, der genauere Angaben zur gewünschten Kaffeemaschine, sowie zu Lieferdatum, Lieferanschrift etc. enthält. Ein Kunde soll dabei immer eine Rechnungsanschrift, aber beliebig viele Lieferanschriften haben dürfen, weil z. B. Bäckereien viele Filialen haben können. Das Unternehmen organisiert die Auslieferung der Kaffeemaschinen über verschiedene Transportdienstleister. Diese Dienstleister können bei der Auftragserfassung frei gewählt werden.
Zunächst werden nun alle für den oben beschriebenen Geschäftsvorgang notwendigen Informationen und Aktionen erörtert und natürlichsprachlich geschildert werden.
Wir unterscheiden dabei in unternehmensnotwendige Informationen, also Daten, die im Unternehmen ohnehin vorliegen und zur Durchführung des Geschäftsprozesses der Kaffeemaschinen-Auslieferung benötigt würden, sowie in Erfassungs- und Zusammenstellungsfunktionen. Die hier aufgeführten Erfassungsfunktionen beziehen sich auf die Erfassung von Kaffeemaschinen, die Zusammenstellungsfunktionen auf den Auftragsversand an die Dienstleister und auf unternehmensinterne Auswertungen. Weiterhin werden noch Integritätsregeln erstellt, die die Speicherung von fehlerhaften Daten verhindern sollen.

2.1 Unternehmensnotwendige Informationen

Es existieren im Unternehmen Daten über eigene Mitarbeiter, Kunden und Dienstleister, die für die Auftragserfassung relevant sind und daher gespeichert werden müssen:
Zu Mitarbeitern sollen der Vorname, der Nachname, die Anschrift, die Telefon-

Nummer, die E-Mail-Adresse und die Vollmacht des Mitarbeiters im Unternehmen festegehalten werden.
Zu Kunden sollen ein Name1 und ein Name2 gespeichert werden können, dabei soll bei natürlichen Personen unter Name1 der Nachname, unter Name2 der Vorname gespeichert werden; bei juristischen Personen soll die Firma unter Name1 eingetragen werden und, sofern ein Zeilenumbruch notwendig ist, soll der Rest der Firma unter Name2 eingetragen werden. Zu Kunden muss eine Rechnungsanschrift existieren, außerdem können beliebig viele Lieferanschriften existieren, die die Adressen von möglichen Filialen des Kunden aufnehmen können. Um den Kunden für Rückfragen etc. erreichen zu können, soll die Möglichkeit bestehen, eine Telefon-Nummer und eine E-Mail-Adresse zu speichern. Ausserdem wird zu jedem Kunden eine Gebietszuordnung gespeichert. Die Gebiete entsprechen dabei den Verkaufsgebieten, in denen der Kunde seinen Firmensitz hat. Außerdem wird zu jedem Kunden der Kontostand gespeichert, der angibt, wieviel Geld der Kunde dem Unternehmen noch schuldet und ein Kreditlimit, das die Bonität des Kunden angibt. Die Lieferanschriften des Kunden sollen lediglich die Adressdaten beinhalten.
Für die Dienstleister sollen neben Name1 und Name2, die in ihrer Funktion den selben Feldern wie bei Kunden entsprechen. Außerdem sind zu den Dienstleistern allerdings noch Zahlungs- und Lieferbedingungen vereinbart, die gespeichert werden müssen, sowie die Konditionen, zu denen die Auslieferung von Kaffeemaschinen erfolgt.
Weiterhin müssen die Daten der lieferbaren Kaffeemaschinen gespeichert werden, wobei jede Kaffeemaschine durch den Hersteller, die Typenbezeichnung und die Bestellnummer des Herstellers eindeutig gekennzeichnet ist. Jede Kaffeemaschine hat zudem einen Listpreis, der dem Kunden bei Bestellung belastet wird. Außerdem soll eine Beschreibung zu jeder Kaffeemaschine festgehalten werden.
Alle Kunden sind nach der Vertriebsstruktur im Unternehmen einem bestimmten Gebiet zugeordnet, und es ist von Interesse für das Unternehmen,

die Verkaufszahlen innerhalb aller Gebiete feststellen zu können. Weiterhin werden je nach Kunden-Zielgruppe verschiedene Kostenstellen mit den Kosten für die Auslieferung der Kaffeemaschinen belastet, wobei hier für das Unternehmen in periodischen Abständen von Interesse ist, welche Zielgruppe wieviel Kosten verursacht hat.

2.2 Erfassungsfunktionen

Es soll möglich sein, Aufträge zu erfassen und zu speichern. Im Auftrag sollen Angaben zum Liefertermin, zum Erfasser, zur Gebietszuordnung und zur Lieferanschrift des Kunden festgehalten werden. Weil der Auftrag an verschiedene Dienstleister erteilt werden kann, muss auch die Angabe zu diesem speicherbar sein.
Die zu erstellenden Aufträge sollen beliebig viele Positionen enthalten dürfen. Die Position beinhaltet Daten zur Kaffeemaschine, die ausgeliefert werden soll, sowie zur Anzahl der zu liefernden Geräte. Für den Bestellvorgang beim Dienstleister würde zwar die Angabe der Bestellnummer ausreichen, doch wären dann die aus den Auftragsdaten erzeugten Belege für die übrigen beteiligten Personen nicht lesbar. Deshalb sollen in den Positionen neben der Bestellnummer noch der Hersteller, die Typenbezeichnung und die Beschreibung der Kaffeemaschine aufgeführt werden.

2.3 Zusammenstellungs- und Auswertungsfunktionen

Kern der Anwendung soll es sein, erfasste Aufträge, die bislang nicht versendet oder storniert wurden, zum Versand zusammen zu stellen, um diese für die Erstellung von Liefer- und Rechnungsbelegen verfügbar zu machen.
Außerdem sollen periodisch die Absatzzahlen in den verschiedenen Verkaufsgebieten ermittelt werden können. Genauso sollen alle entstandenen Dienstleisterkosten entsprechend ihrer Zielgruppen-Zuordnung kumuliert werden können.

2.4 Integritätsregeln

Es können durch Bedienfehler der Datenbank Fehler entstehen, die es abzufangen gilt:
Als erstes Objekt beschreiben wir den Typ Person. Dieser Typ besitzt als Attribute Name1, Name2, Anschrift, Telefon, EMail. Dabei sind Name1 und Anschrift Pflichtfelder, die für jede Person angegeben werden müssen.
Die Attribute von Person werden an verschiedene andere Entities weitervererbt, die ihrerseits spezielle Personen darstellen: Kunde, Mitarbeiter, Dienstleister.
Zu Mitarbeitern muß die Personalnummer angegeben werden, da über diese aller Mitarbeiter eindeutig identifiziert werden. Außerdem wird zu Mitarbeitern die Vollmacht, die sie im Unternehmen haben, abgespeichert. Da jeder Mitarbeiter eine Vollmacht hat, ist dies ein Pflichtfeld.
Jeder Kunde besitzt eine Kundennummer. Mittels dieser Nummer ist der Kunde eindeutig zu identifizieren. Desweiteren darf die Gebietszuordnung nicht leer sein, weil sonst Auswertungen z. B. über den Verkauf von Maschinen in bestimmten Gebieten scheitern würden. Außerdem muss der Kontostand eines Kunden eine Zahl enthalten, da geklärt sein muß, ob dieser über das Kreditlimit hinaus geht.
Das Kreditlimit des Kunden muß ebenfalls gegeben sein, um die Bonität zu Kunden ermitteln zu können.
Die Dienstleister werden über die Dienstleisternummer identifiziert. Zudem dürfen die Konditionen, also den Preis für die Auslieferung einer Kaffeemaschine beim Dienstleister nicht leer sein, da diese vor der Auslieferung der Maschine immer geklärt sein müssen.
Ein Auftrag wird mittels der Auftragsnummer eindeutig identifiziert. Desweiteren müssen das Lieferdatum und die Kundendaten gegeben sein, die alle Angaben zum belieferten Kunden und der Lieferanschrift beinhalten.
Als letztes muß die Position immer im Auftrag enthalten sein, damit geklärt

ist, welche Kaffeemaschine ausgeliefert wird. Die Position hat als Schlüssel die Auftragsreferenz und die Positionsnummer. Desweiteren darf die Anzahl und das Attribut Kaffeemaschine nicht leer sein, da geklärt sein muß welche und wie viele Kaffeemaschine(n) in einer Position enthalten sind.
Eine Kaffeemaschine wird immer durch die Schlüssel, Hersteller, Typ und Bestellnummer eindeutig identfiziert. Zu dem darf der Listenpreis nicht leer sein, die Summe aus Listenpreis und Kontostand das Kreditlimit des Kunden nicht übersteigen darf.

3 Relationenbildung und Normalisierung

Aus den Angaben in der Anwendungsbeschreibung müssen nun vor der Erstellung eines relationalen Datenbankmodells sinnvolle Relationen erstellt werden. Eine Relation ist dabei ein Tupel von Attributen, die im Zusammenhang mit einem Objekt verwendet werden. Anschaulich gesprochen handelt es sich dabei um eine Tabelle, deren Spalten die unterschiedlichen zusammengehörigen Tupel aufnehmen.
Für unser Beispiel wollen wir nur an der Relation `KUNDE` eine Normalisierung durchführen.

3.1 Datenkatalog, nicht normalisert

Es werden alle zu einem Objekt notwendigen Attribute erörtert. Dabei kann es zu Redundanzen kommen. Redundanzen sind dabei solche Daten, die unnötiger Weise mehrfach im Datenbanksystem vorkommen würden. Wir erörtern diesen Sachverhalt am Beispiel der Lieferanschriften der Relation `Kunde`:
Nach der Anwendungsbeschreibung wären die folgenden Felder zur Erfassung eines Kundendatensatzes notwendig:

- Name1
- Name2

- Anschrift
- Telefon
- EMail
- Gebietszuordnung
- Kontostand
- Kreditlimit
- Lieferstrasse
- LieferPLZ
- Lieferort

Hat ein Kunde nun mehrere Lieferanschriften, so würde eine Tabelle mit Beispieldaten, die nicht normalisiert ist, wie folgt aussehen:

Datentabelle `KUNDE` (nicht normalisiert):

Name1	Name2	...	Lieferstrasse	LiefPLZ	Lieferort
Frischback	Heinz Meier	...	Goethestr.	28999	Bremen
Frischback	Heinz Meier	...	Schillerstr.	28998	Bremen
Frischback	Heinz Meier	...	Herderstr.	27985	Achim
Cityback	Gerda Müller	...	Goethestr.	22999	Hamburg
Cityback	Gerda Müller	...	Lessingstr.	22999	Hamburg

Name1, Name2 u. s. w. wiederholen sich in dieser Form so oft, wie der Kunde Lieferanschriften hat. Dies kostet zum einen unnötig viel Speicherplatz, zum anderen werden bei späteren Anfragen auf dieser Tabelle die Relationentupel unnötig groß und die Anfragen damit sehr rechenintensiv.

3.2 Erste Normalform (1NF)

Vermieden wird die Wiederholung dadurch, daß die ursprüngliche Tabelle in zwei neue Attribut-Tupel zerlegt wird:

Tabelle `KUNDE`:

KundenNr	Name1	Name2	...
0001	Frischback	Heinz Meier	...
0002	Cityback	Gerda Müller	...

Tabelle `LIEFERANSCHRIFT`:

KundenNr	LiefNr	Lieferstrasse	LieferPLZ	Lieferort
0001	0001	Goethestr.	28999	Bremen
0001	0002	Schillerstr.	28998	Bremen
0001	0003	Herderstr.	27985	Achim
0002	0001	Goethestr.	22999	Hamburg
0002	0002	Lessingstr.	22999	Hamburg

In den zerlegten Tabellen gibt es keine Wiederholungen der Kundennamen mehr. Man spricht nun von Relationen in der ersten Normalform (1NF).
Um den Zusammenhang zwischen den Tupeln aus `KUNDE` und `LIEFERANSCHRIFT` auszudrücken, bedarf es nun des neuen Attributs `KundenNr`. Um eine Lieferanschrift zu einem Kunden eindeutig identifizieren zu können, führen wir ein weiteres Attribut `LiefNr` ein, was sich daher empfiehlt, da die übrigen Attribute nicht zwangsläufig eindeutig sein müssen. Diese beiden Attribute bilden nun zusammen den kombinierten Primärschlüssel für Lieferanschriften.

3.3 Zweite Normalform (2NF)

Relationen in der zweiten Normalform liegen vor, wenn sich diese schon in der ersten Normalform befinden, und außerdem sämtliche Nichtschlüssel-Attribute vom ganzen Schlüssel, nicht nur von einem Teil des kombinierten Schlüssels abhängen. Dies trifft auf unser oben eingeführtes Beispiel bereits zu: `KUNDE` hat einen einfachen Primärschlüssel, nämlich `KundenNr`, `LIEFERANSCHRIFT` besitzt jedoch einen kombinierten Schlüssel, der aus den Schlüssel-Attributen `KundenNr` und `LiefNr` zusammengesetzt ist. Alle Nichtschlüssel-Attribute aus `LIEFERANSCHRIFT`, also `Lieferstrasse`, `LieferPLZ` und `Lieferort` sind jedoch abhängig vom ganzen Schlüssel.

3.4 Dritte Normalform (3NF)

Bei Relationen der dritten Normalform handelt es sich um solche, die bereits in der zweiten Normalform sind, es jedoch keine funktionalen Abhängigkeiten zwischen Nichtschlüssel-Attributen mehr gibt.
In unserem Beispiel besteht weiterhin eine funktionale Abhängigkeit zwischen den Attributen `Lieferstrasse`, `Lieferort` und `LieferPLZ`, weil eine Postleitzahl in Deutschland von Ort und Straße bedingt werden (zur Vereinfachung des Beispiels haben wir auf die Darstellung von Straße und Hausnummern in dieser Relation verzichtet).
Eine weitere Relationen-Zerlegung wäre daher nun möglich:

Tabelle `POSTLEITZAHLEN`:

Strasse	Ort	PLZ
Herderstr.	Achim	27985
Goethestr.	Bremen	28999
Schillerstr.	Bremen	28998
Goethestr.	Hamburg	22999
Lessingstr.	Hamburg	22999

Damit ließe sich nun eine Relation `LIEFERANSCHRIFT` ohne das bisherige Attribut `LieferPLZ` erstellen. Um dann die Postleitzahl zu einer Lieferanschrift ermitteln zu können, müsste diese über die Attribute `Lieferstrasse` und `Lieferort` mit dem kombinierten Schlüssel `Strasse` und `Ort` ermittelt werden.
Für unser Beispiel haben wir nicht alle Relationen in die dritte Normalform übertragen. Zwar würde eine Relation `Postleitzahlen` wwohl Speicherplatz sparen, allerdings müsste dann auch in Bezug auf die meisten der später durchzuführenden Anfragen stets wieder ein Verbund von z. B. `LIEFERANSCHRIFT` und `POSTLEITZAHLEN` vorgenommen werden, was stets Re-

chenleistung und damit Zeit in Anspruch nehmen würde. Außerdem besteht bei vollständiger Normalisierung aller Relationen immer wieder die Gefahr, daß spätere Verbünde nicht verlustlos sind, also Informationen unter Umständen sogar verloren gehen können.
In unserem Beispiel haben wir auch aus Gründen der Übersichtlichkeit auf vollständige Normalisierung verzichtet.

3.5 Die Relationen im verwendeten Beispiel:

Die folgenden Entities wurden für die vorliegende Arbeit aus der vorangegangenen Anwendungsbeschreibung entwickelt. Es sind in dieser textuellen Notation im einzelnen:

`PERSON(Name1, Name2, Anschrift, Telefon, EMail)`

`MITARBEITER(Personalnummer, Vollmacht)`

`KUNDE(KundenNr, Gebietszuordnung, Kontostand)`

`DIENSTLEISTER(DienstleisterNr, Lieferbedingung, Zahlungsbedingung, Kondition)`

`KAFFEEMASCHINE(Hersteller, Typ, Bestellnummer, Listpreis, Beschreibung)`

`AUFTRAGSKOPF(Auftragsnummer, Erfasser, Lieferdatum, ...)`

`POSITION(Auftragsreferenz, Positionsnummer,`
`Hersteller, Typ, Bestellnummer, Beschreibung, Listpreis)`

`LIEFERANSCHRIFT(KundenRef, LieferNr, Lieferstr, LieferPLZ, LieferOrt)`

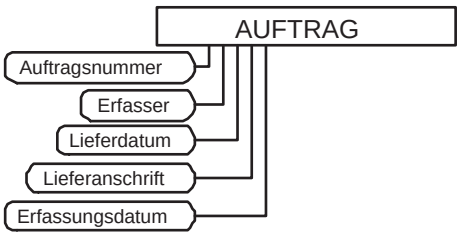
Dabei sind die Entity-Bezeichner in Großbuchstaben geschrieben. In den Klammern dahinter sind die zugehörigen Attribute angeführt, von denen die Schlüsselattribute unterstrichen sind.

4 Konzeptionelle Datenbankstruktur

4.1 Begriffserklärungen zum Entity-Relationship Modell

Das „Entity-Relationship-Model“ist ein Meta-Datenmodell um Datenbanksysteme abstrakt darzustellen. Zur Darstellung als ERM kann man sich die künftige Struktur verdeutlichen, ohne sich schon auf eines der klassischen Datenbankmodelle festlegen zu müssen.
Grundsätzlich werden im ERM alle Objekte der realen Welt als Entities dargestellt und die Zusammenhänge zwischen diesen als Relationen vermerkt. Eine genaue Notation ist jedoch, anders als z. B. in der Unified Modeling Language (UML), nicht verbindlich definiert. Vielmehr ist das ERM lt. Scheer häufigeren Änderungen unterworfen. Wir wollen daher zunächst die in der vorliegenden Arbeit verwendete Notation näher erläutern:

4.1.1 Entity mit Attributen



In unserem Beispiel wird das Objekt Auftrag als Entity beschrieben. Diese wird mittels eines Rechtecks dargestellt. An der Entity *Auftrag* sind mit Hilfe einer Kante die *Attribute* verbunden. Die Attribute sind Eigenschaften dieser Objekte. Sie werden mittels eines Rechteckes (jedoch mit abgerundeten Kanten) dargestellt. In unserem Beispiel ist das Objekt Auftrag mit dem Attribut Auftragsnummer verbunden. Dieses Attribut stellt gleichzeitig das Schlüsselattribut dar, weshalb es unterstrichen ist. Sofern mehrere Attribute gemeinsam einen kombinierten Schlüssel bilden, so sind all diejenigen unterstrichen.

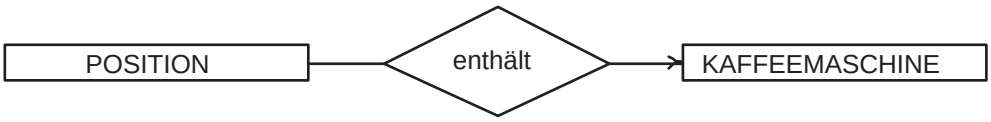
4.1.2 Relationen

Relationen sind Beziehungen zwischen den einzelnen Entitys. (Nicht zu verwechseln mit Relationen im Zusammenhang mit relationalen Datenbankmodellen!)

In unserem Beispiel ist die Entity Kunde mit der Entity Auftrag verbuden. In diesem Fall ist die Relation zwischen den Entitys erteilt. Das beudetet, ein Kunde erteilt einen oder mehere Aufträge. Diese Relationen werden immer durch eine Raute dargestellt.

4.1.3 Funktionale Beziehungen

Wenn eine Entity immer maximal zu einer weiteren Entity in Relation steht, spricht man auch von einer funktionalen Beziehung.
In unserem Beispiel ist zu jeder Position eine Kaffeemaschinentyp zugeordnet.

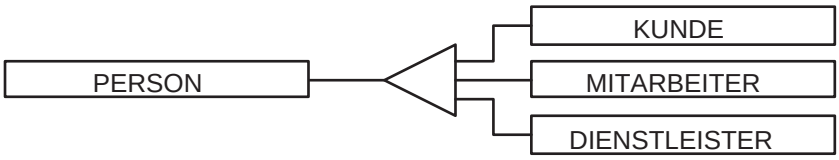


4.1.4 Generalisierung/Spezialisierung

Bei der Generalisierung werden ähnliche Objekttypen zu einem übergreifenden Objekttyp zusammengefasst. Eigenschaften, die den Ausgangsobjekten gemein sind, werden dabei auf den generalisierten Objekttyp übertragen, so daß sie nur noch durch davon abweichende Attribute beschrieben werden müssen. Der Vorgang der Generalisierung kann auch in der umgekehrten Form, der Spezialisierung, auftreten, indem ein Gattungsbegriff in Teilbereiche zerlegt wird. Aus diesem Grunde werden die Verbindungslinien zwischen den Oberbegriffen und den Teilbegriffen nicht gerichtet gezeichnet. Bei der

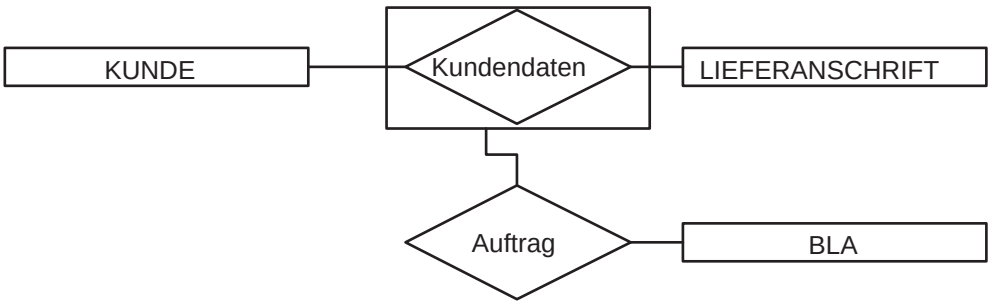
Spezialisierung werden Eigenschaften des generalisierten Objekttyps auf die spezialisierten Objekttypen vererbt. Diese können daneben auch eigene Attribute besitzen und auch ererbte abändern.

In unserem ERM wird die Entity *Person* in verschiedene Entitys (Kunde, Mitarbeiter, Dienstleister, Ansprechpartner) aufgespalten beziehungsweise spezialisert. Dabei werden die Attribute der Person an die einzelenen Entitys weitervererbt.



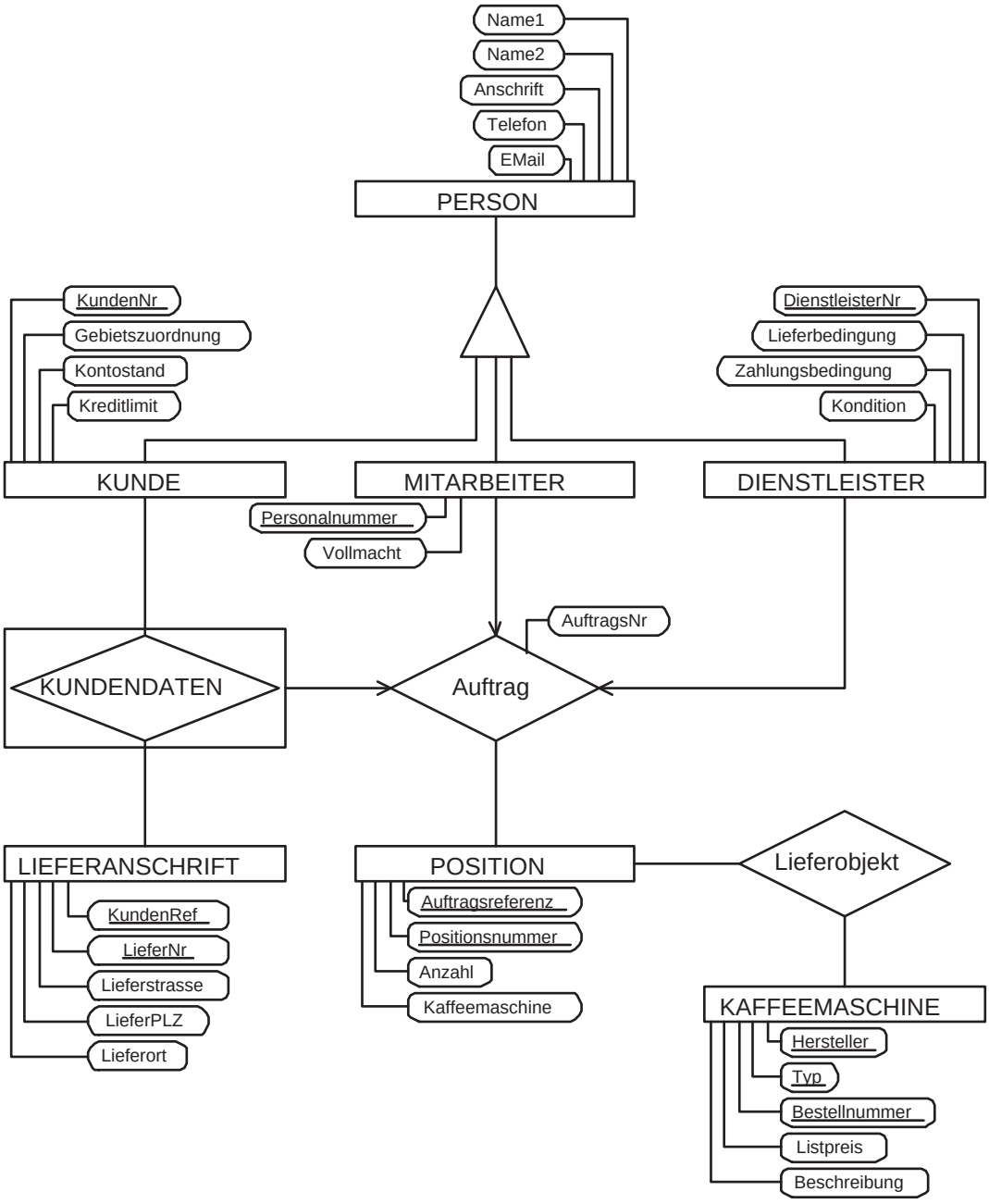
4.1.5 Uminterpretation einer Relation

In unserem ERM kommen Relationen vor, die Ihrerseits wieder ein Entity-Typ sind (vgl. Scheer, 1998).



In diesem Beispiel sieht man, daß die Kunden- und Lieferanschriftentity eine Relation bilden, die Ihrerseits eine Entity in Bezug auf die Relation Auftrag darstellt. Anschaulich gesagt ist hierunter zu verstehen, daß die Attribute des Verbundes von KUNDE und LIEFERANSCHRIFT in Beziehung zum Auftrag stehen.

4.2 ERM-Darstellung der Anwendung Kaffeemaschinen-Logistik-Koordination:



4.3 UML-Darstellung der Anwendung Kaffeemaschinen-Logistik-Koordination:

Im folgenden wird das Datenbankmodell noch einmal in Form eines UML-Diagramms dargestellt werden.
Zunächst wird die Klasse `Person` eingeführt, die eine abstrakte Klasse ist. Praktisch heißt dies, daß von dieser Klasse niemals ein Instanz erstellt wird. Die in der Klasse definierten Attribute dienen lediglich der Vererbung an die Klassen Kunde, Mitarbeiter und Dienstleister. Dies wird durch das UML-Vererbungs-Symbol im Diagramm dargestellt. Die drei Klassen, die die Eigenschaften der abstrakten Klasse `Person` erben, erhalten weitere Attribute, die den speziellen realen Personen zugeordnet sind. Weiterhin wird eine Klasse `Lieferanschrift` definiert. Die Klassen `Kunde` und `Lieferanschrift` enthalten zusammen alle Attribute, die die `Kundendaten` ausmachen, wobei in den `Kundendaten` jeweils ein Objekt vom Typ `Kunde` mindestens eine bis beliebig viele Lieferanschriften hat. Daß die Kundendaten in der beschriebenen Form zusammen gesetzt sein *müssen*, wird im Diagramm durch die Kompositions-Symbole verdeutlicht.
Ein `Auftrag` ist wiederum zwingend aus einer Instanz vom Typ `Kundendaten` und mindestens einer bis beliebig vielen Instanzen vom Typ `Position` zusammen gesetzt, wie schon vorher als Komposition, weil es ohne Kundendaten und ohne Positionen keinen Auftrag gäbe.
Außerdem soll der Auftrag Angaben zum `Mitarbeiter`, der in diesem Zusammenhang als Erfasser auftritt, und zum `Dienstleister`, der den Auftrag ausführt, aufweisen. Diese sind, weniger streng, mit dem Aggregations-Symbol verknüpft, da sie nicht zwingend zum Entstehen eines Auftrags nötig sind.
Eine Instanz vom Typ `Position` muß notwendiger Weise eine Kaffeemaschine enthalten, und zwar genau eine. Soll eine Weitere Kaffeemaschine mit diesem Auftrag ausgeliefert werden, so müsste hierfür dann eine weitere Po-

sition angelegt werden. Dies ist wiederum durch das Kompositions-Symbol und die eins zu eins-Kardinalität deutlich gemacht.
Jeder `Auftrag` ist zunächst wie oben beschrieben zusammengesetzt aus Attributen anderer Klassen. Er hat aber zudem noch eigene Attribute: Die Auftragsnummer ist ein Long-Integer-Wert, der jeden Auftrag eindeutig identifizierbar macht, der Liefertermin ein Date-Wert, der das mit dem Kunden vereinbarte Datum der Kaffeemaschinen-Anlieferung enthält.
Wir haben im folgenden Diagramm der Übersichtlichkeit halber auf die grafische Notation von „Constraints“, also Zusicherungen, verzichtet. (Kardinalitäten werden hier allerdings nicht in OCL, sondern nur grafisch an den Assoziationen dargestellt.) Es sollen exemplarisch einige in der Anwendungsbeschreibung schon natürlichsprachlich genannten semantischen Integritätsbedingungen hier in der „Object Constraint Language“(OCL) aufgeführt werden.
Als erstes Beispiel wollen wir prüfen, ob zu allen Personen zumindest der Name1 und die Anschrift existiert:

```
context Person inv: self.Name1 -> forAll (not null).
```

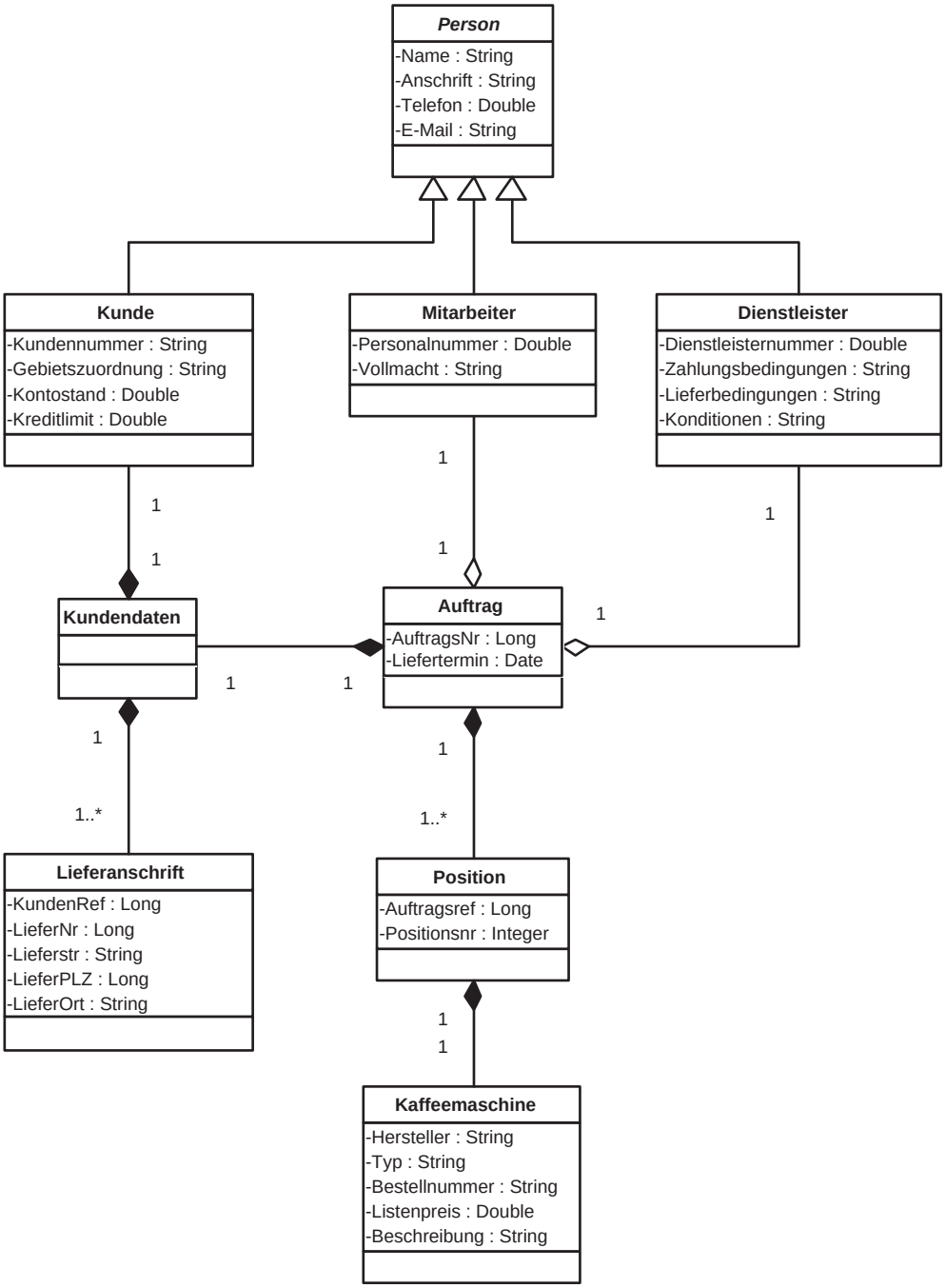
Dies bedeutet nun, daß im Zusammenhang mit dem Objekt `Person` kein Element der Datenmenge des Attributs `Name1` den Null-Wert enthalten darf. Die Abfrage für das Fehlen der Anschrift würde analog hierzu gebildet.

Ein anderes Beispiel für eine Integritätsbedingung ist, daß der Liefertermin eines Auftrags in der Zukunft liegen muss:

```
context Auftrag inv: self.Liefertermin -> forAll (Liefertermin >
Today)
```

Für *Today* wird dabei davon ausgegangen, daß es eine Variable ist, die das aktuelle Tagesdatum enthält.

Nun die Darstellung unserer Beispieldatenbank im UML:



5 Standard Query Language (SQL)

Bei SQL handelt es sich um eine Anfragesprache für relationale Datenbanksysteme. SQL wird dabei als „Deskriptiv“bezeichnet, da in wohldefinierter Form angegeben wird, welche Daten durch den entsprechenden Term ermittelt werden sollen.

5.1 Sichten in SQL

Als erstes erstellen wir hier einen View über die erfaßten Aufträge des jeweiligen Tages. Wir gehen davon aus, daß der heutige Tag der 15. Januar des Jahres 2004 ist. Dabei werden alle Attribute des Auftrages mit in diesen View mit hinein übernommen.

```
create view Auftraege_Heute as
  select * from Auftrag
    where date = '15-Jan-04';
```

Im zweiten View sollen alle Positionen zusammengefasst werden, die am heutigen Tag (15. Januar 2004) ausgeliefert wurden.

```
create view Geliefert_Heute as
  select Position, Lieferdatum from Auftrag
    where Lieferdatum = '15-Jan-2004';
```

In diesem View sollen alle Kaffeemaschinen und die Anzahl der gelieferten Maschinen eines Herstellers aufgelistet werden.

```
create view Geliefert_Jahr as
  select Kaffeemaschine, sum(Anzahl) from Position
    where Kaffeemaschine.Typ = 'Bremer';
```

5.2 Standardabfragen in SQL

In der ersten Abfrage werden alle Kaffeemaschinen eines Herstellers aufgelistet.

```
select * from Kaffeemaschine
  where Hersteller = 'Bremer';
```

Die nächste Abfrage listet alle Kaffeemaschinen auf. Diese werde aber nach Herstellern sortiert.

```
select * from Kaffeemaschine
  order by Hersteller;
```

Als nächstes wird ein neue Kaffeemaschine in die Datenbank eingepflegt.

```
insert into Kaffeemaschine values ('Bremer', 'Ultra', '0815', '499',
'Blau metallic lackiert');
```

Mit dieser Abfragen werden alle Kunden heraus gefiltert, die an das Unternehmen noch Zahlungen zu leisten haben.

```
select * from Kunde
  where Kontostand < 0;
```

In der folgenden Abfrage werden die Namen der Kunden, ihre dazugehörige Kundennummer und die Auftragsnummern, die etwas bestellt haben aufgelistet. Dabei ist es notwendig die Tabellen der Kunden und die Tabelle der Aufträge mit einander zu vereinigen.

```
select Kundennummer, Name from Kunde
  where Kundennummer = 4711
union
select Kundennummer, Auftragsnummer from Auftrag
  where Kundennummer = 4711;
```

In der folgenden Abfrage soll die im Unternehmen billigste lieferbare Kaffeemaschine ausgegeben werden. Mittels dieser Anfrage kann einem Kunden das günstigste Angebot gemacht werden.

```
select Hersteller, Typ, Bestellnummer, Preis from Kaffeemaschine K
  where K.Preis = (select min(Preis) from Kaffeemaschine);
```

Im folgenden soll eine veraltete Kaffeemaschine, mit der Bestellnummer, wieder aus der Datenbank entfernt werden. Dies geschieht mit Hilfe des DELETE Befehls. Das Statement sieht im folgenden so aus:

```
delete from Kaffeemaschine
  where Bestellnummer = 2543;
```

6 Literaturverzeichnis

Bei unserer Recherche für diese Hausarbeit haben wir folgende Literatur verwendet:

Erler, Dr. Thomas, „Das Einsteigerseminar UML“, 1. Auflage 2000, bhv Verlag Kaarst

Gogolla, Dr. Martin, „Introduction to the Unified Modeling Language (UML)“, Uni-Bremen FB Informatik 2002

Gogolla, Dr. Martin, Skript zur Vorlesung 'Datenbanksysteme', Bremen 2003

Misgeld, Wolfgang D., „SQL Einstieg und Anwendung“, 4. Auflage 2001, Hansa Verlag München

Scheer, August W., „Wirtschaftsinformatik – Referenzmodelle für Geschäftsprozesse, Studienausgabe“, 2. Auflage 1998, Springer Verlag München