Datenbanksystem eines Abfallentsorgungsunternehmens mit dem Namen *Throw It Away*

Ausarbeitung im Bereich

Datenbanksysteme

Bremen, den 08. März 2009

Agnieszka Bugajewski
Utbremer Ring 203
28215 Bremen
E-Mail: aga@tzi.de
Matrikelnummer: 2028342

Rafael Bugajewski
Utbremer Ring 203
28215 Bremen
E-Mail: rab@tzi.de
Matrikelnummer: 1971058

Inhaltsverzeichnis

1. Einführung

Dieser Entwurf wurde im Rahmen des Kurses Datenbanksysteme im Wintersemester 2008/09 an der Universität Bremen ausgearbeitet.

Die Struktur des Dokuments richtet sich nach den Vorgaben von der offiziellen Webseite zur Vorlesung (siehe [1]) und beschreibt den Entwurf eines Datenbanksystem für ein fiktives mittelgroßes Unternehmen, das sich um die Müll- bzw. Abfallentsorgung kümmert.

Der Abschnitt Anwendungsbeschreibung erläutert die Grundlagen des Unternehmens und schildert Überlegungen, die im Rahmen der Ausarbeitung in den Entwurf eingeflossen sind. Es wird dabei auf unterschiedliche Modellierungsentscheidungen eingegangen und versucht die Vorteile, aber auch Einschränkungen des zugrunde liegenden Entwurfs zu präsentieren und dem Leser einen Ausblick beziehungsweise mögliche Evolutionsschritte darzubieten. Der Abschnitt ist von informeller Natur und soll einen Einstieg in weitere Abschnitte des Dokuments darstellen.

Der darauf folgende Abschnitt Konzeptioneller Entwurf zeigt diesen nicht nur auf, sondern erläutert einzelne Teile im Detail. Grundlage sowohl des Abschnitts als auch des Dokuments ist das in diesem Abschnitt präsentierte ER-Modell, welches im Laufe des Semesters entwickelt wurde und mehreren evolutionären Änderungen unterlag.

Anschließend werden Integritätsbedingungen aufgezeigt. Zu den formalen Definitionen werden jeweils informelle Erläuterungen präsentiert, die Entwurfsentscheidungen verdeutlichen sollen.

Im nächsten Abschnitt wird eine mögliche Realisierung des Modells in Form eines relationalen DB-Schemas aufgezeigt. Dabei wird hauptsächlich auf die einzelnen Entitätstypen und Beziehungen eingegangen. Bei der Erstellung wurde auf Standardverfahren zurückgegriffen und darauf geachtet, dass Redundanz und Anomalien verhindert werden, so wie es im Laufe der gesamten Veranstaltung gelehrt wurde.

In den letzten beiden Abschnitten des Dokuments werden sowohl die Sichten, als auch Standardanfragen mit SQL präsentiert.

2. Anwendungsbeschreibung

Es soll ein Datenbanksystem für ein Abfallentsorgungsunternehmen entwickelt werden, welches Probleme hat Ihre Fahrzeugkollone zu verwalten. Es handelt sich dabei um ein mittelgroßes Unternehmen mit dem Namen *Throw It Away*, das in verschiedenen Orten bundesweit für die Entsorgung von Abfall verantwortlich ist. Dabei werden sowohl Kleinstädte, als auch Metropolen, wie München oder Berlin bedient. Zu dem Hauptbereich des Unternehmens gehört die Entsorgung von Restmüll. Weitere Gebiete sind das Recycling von Papier und die korrekte Verarbeitung von Biomüll. Der Schwerpunkt des hier entwickelten Systems liegt auf der Entsorgung des Mülls. Das Unternehmen ist derartig groß, dass es in verschiedene Bereiche gegliedert ist. Nicht betrachtet werden bei der Entwicklung dieses Datenbanksystems Aspekte wie das vollständige Rechnungswesen, die eigentliche Wartung von Fahrzeugen oder die Kommunikation mit Geschäftskunden, die zum Beispiel als Müllabnehmer dienen. Ziel ist es durch das hier entwickelte System dem Kunden die Möglichkeit zu bieten alle Fahrzeuge und die damit in Zusammenhang stehende eigentliche Müllabfuhr zu verwalten und zu optimieren.

Im Unternehmen befinden sich selbstverständlich verschiedene Ressourcen. Die Hauptressourcen des Unternehmens sind Mitarbeiter, aber auch Fahrzeuge, die zum Vereinfachen und Beschleunigen der Müllentsorgung für Kunden dienen. Mitarbeiter, die in direktem Zusammenhang mit der Müllentsorgung stehen weisen entweder entsprechende Qualifikationen vor (Führerschein) oder können nur begleitend von einem anderen Mitarbeiter ihre Tätigkeit durchführen. Die verschiedenen Fahrzeuge werden von qualifizierten Mitarbeitern gefahren und unterwegs durch ihre Begleiter mit Müll gefüllt. Dabei gilt es zu beachten, dass ein Fahrzeug eine Tour bekommt, die für ihn angemessen ist und in der es den gesamten auftretenden Müll einsammeln kann. Des Weiteren muss es sich um ein speziell angepasstes Fahrzeug handeln. So wird zum Beispiel Papiermüll während der Fahrt von entsprechenden Fahrzeugtypen gepresst, damit es weniger Volumen hat und das Fahrzeug ein größeres Gebiet versorgen kann. Ein anderer wichtiger Aspekt in Zusammenhang mit Fahrzeugen ist die Instandhaltung. Ein Fahrzeug sollte regelmäßig gewartet werden, damit es im Einsatz zu keinerlei Problemen kommt. Reparaturen im Außendienst sind äußerst kostspielig und sollten daher nur im absoluten Ausnahmefall durchgeführt werden.

Fahrzeuge sammeln Abfälle bei Kunden ein, sie fahren also die entsprechenden Haushalte an, auf die später noch eingegangen wird. Der angesammelte Müll wird

anschließend zu der zuständigen Deponie gefahren. Hierbei ist es wichtig, dass der Müll in einer Deponie abgeliefert wird, die einerseits über genug Kapazitäten verfügt, andererseits aber auch nicht allzu weit entfernt ist, so dass lange Transportwege vermieden werden. Außerdem muss beachtet werden, dass Deponien nicht immer alle verschiedenen Sorten von Müll lagern können, da hierfür spezielle Gerätschaften notwendig sind. Aus diesem Grund kann ein Fahrzeug, das mit Papier beladen ist auch nur zu einer Deponie, die für die Lagerung dieser Abfallsorte geeignet ist.

Deponien liegen in verschiedenen Orten, die wiederum in Gebiete aufgeteilt sind. Der Ort, sowie sein Postleitzahlenbereich ist interessant, um Entscheidungen für die zukünftige Entwicklung des Unternehms zu treffen oder die Effizienz festzustellen (Wie viele Fahrzeuge sind für wie viele Einwohner zuständig? usw.) Sie sind dafür zuständig um eine logische Struktur zu bilden und sind nicht mit dem Wohnort einer Person oder Ähnlichem zu verwechseln.

Das Unternehmen ordnet Haushalte unterschiedlichen Gebieten zu. Ein Gebiet ist lediglich als eine geographisch nahe liegende Region zu verstehen. Gebiete sind für das Unternehmen relevant, da sie die Grundlage für die Organisation der gesamten Müllabfuhr bilden. Sie werden durch unterschiedliche Fahrzeuge versorgt und können sich auch überlappen. Ein Gebiet ist nicht im Sinne von Stadteil oder Ort zu verstehen, sondern eher als verschiedene Haushalte bei denen ein entsprechender Typ von Müll abgeholt werden muss. Es besitzt daher einen frei wählbaren Namen und - was für die Organisation wichtig ist - ein Anfangsdatum, sowie einen Tagesrhytmus in dem das entsprechende Gebiet von Fahrzeugen verschiedenen Typs befahren wird.

Die einzelnen Haushalte liegen in Gebieten und haben eine bestimmte Anzahl an Personen. An dieser richtet sich das Volumen und die Anzahl an verschiedenen Mülltonnen. Die Mülltonnen selber sind anhand einer Nummer eindeutig identifzierbar, so dass bestimmte Einschränkungen und Analysen durchgeführt werden können (Wie oft wurde diese Tonne entleert?) Tonnennummern werden beim Aufladen von Müll in ein Fahrzeug automatisch ausgelesen.

Kunden bilden die für das Unternehmen kleinste Verwaltungseinheit. Sie gehören zu einem Haushalt und sind in Besitz verschiedener Mülltonnen. Jeder Kunde besitzt eine eindeutige Kundennummer und hat einen Kontostand, der über die Liquidität des Kundes eine Aussage macht. Der Kunde ist stets verantwortlicher Ansprechpartner (und nicht etwa der Haushalt).

3. Konzeptioneller Entwurf

3.1. ER-Diagramm

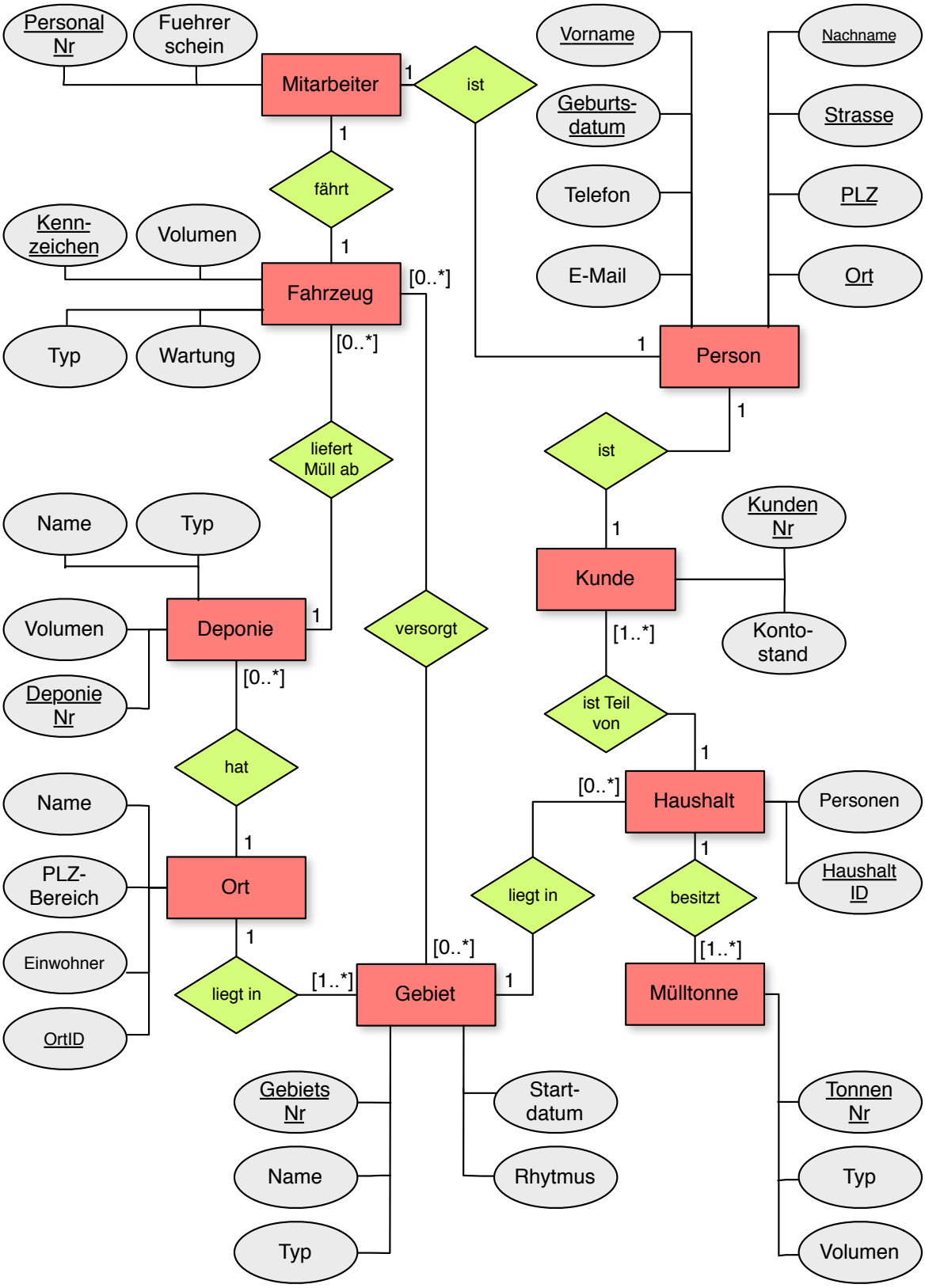

Abbildung 1: ER-Diagramm

3.2. Entitätstypen

Nachfolgend wird auf die einzelnen Entitätstypen eingangen. Dabei wird in der Erläuterung oft auf konkrete Entitäten und dessen Beziehungen zueinander verwiesen, um die Entitätstypen anschaulicher erläutern zu können. Es wird versucht an einigen Stellen einen detaillierteren Einblick als in der Anwendungsbeschreibung zu geben. Außerdem wird auf verschiedene Einschränkungen und Designentscheidungen eingegangen und aus welchen Gründen diese gegenüber Alternativen getroffen wurden.

PERSON (Vorname, Nachname, Geburtsdatum, Strasse, PLZ, Ort, Telefon, E-Mail)

Ein wichtiger Bestandteil des zugrunde liegenden Modells ist eine Person. Sie ist ein Entitätstyp, der durch den Vornamen, Nachnamen, Geburtsdatum, Straße, Postleitzahl und Ort eindeutig identiziert werden kann. Außerdem besitzt eine Person eine Telefonnummer und eine E-Mail-Adresse. Es macht Sinn die Adresse in Ihre Bestandteile aufzuteilen und separat abzuspeichern, da man auf dieser Grundlage einfacher auf den Daten arbeiten kann, um somit zum Beispiel statistische Auswertungen oder bestimmte Sichten auf den Daten durchführen zu können. Auch sind die Attribute, die eine Person eindeutig identifizieren in diesem Umfang sinnvoll. Die Wahrscheinlichkeit, dass zwei Personen den gleichen Vor- und Nachnamen haben, ist selbst in einer Kleinstadt relativ hoch. Es ist durch Zufall sogar möglich, dass zwei Personen mit dem Namen Sebastian Müller (als Beispiel) in dem gleichen Wolkenkratzer wohnen. Dass diese beiden Personen auch noch am gleichen Datum geboren sind, erscheint allerdings höchst unwahrscheinlich.

KUNDE (Vorname, Nachname, Geburtsdatum, Strasse, PLZ, Ort, Telefon, E-Mail, KundenNr, Kontostand)

Ein Kunde ist eine Person, daher vererbt dieser Entitätstyp alle oben genannten Eigenschaften. Außerdem besitzt ein Kunde eine eindeutig identifizierbare Kunden-Nummer, die zum Beispiel auf jeder Rechnung erwähnt wird oder bei Supportanfragen als Referenz genannt werden sollte. Des Weiteren besitzt jeder Kunde einen Kontostand. Dieser gibt Aussage darüber, ob sich ein Kunde in Zahlungsverzug befindet oder sogar ein Guthaben hat. Das komplette Rechnungswesen ist nicht Teil dieses Modell und der Schwerpunkt liegt ganz klar auf den logistischen Aspekten. In diesem Zusammenhang ist es allerdings wichtig, ob sich eine Person in Verzug befindet, da auf dieser Grundlage durch die Mitarbeiter beim Kunden vor Ort entsprechende Maßnahmen ergriffen werden können.

MITARBEITER (Vorname, Nachname, Geburtsdatum, Strasse, PLZ, Ort, Telefon, E-Mail, PersonalNr, Fuehrerschein)

Ein Mitarbeiter ist - genau so wie ein Kunde - eine Person. Mitarbeiter besitzen außerdem eine PersonalNr, die sie eindeutig identifiziert. Außerdem wird gespeichert, ob ein bestimmter Mitarbeiter einen Führerschein besitzt. Weist dieser entsprechende Qualifikationen vor, so ist er berechtigt ein Fahrzeug zu führen, andernfalls führt er nur Tätigkeiten als Begleiter eines Mitarbeiters mit Fahrerlaubnis durch. Mitarbeiter ohne Führerschein arbeiten demnach nie alleine im Außendienst.

FAHRZEUG (Kennzeichen, Typ, Volumen, Wartung)

Der Entitätstyp Fahrzeug definiert einen Müllabfuhrwagen, der zur gesamten Flotte des Unternehmens gehört und ordentlich beim Straßenverkehrsamt angemeldet wurde. Über das Kennzeichen lässt sich ein Fahrzeug eindeutig identifizieren. Des Weiteren wird gespeichert für welchen Typ von Abfällen das Fahrzeug geeignet ist (im Kontext relevant sind: Bio-, Papier- und Restmüll). Der Typ von Abfällen ist kein eigener Entitätstyp, da aufgrund der immer stärker wachsenden Bedeutung von informationstechnischen Systemen in der Müllentsorgung eher davon auszugehen ist, dass irgendwann nur noch ein Fahrzeugtyp den kompletten Müll transportieren wird und dieser anschließend vollautomatisch in einer Mülldeponie sortiert werden kann. Außerdem wird das Volumen des Fahrzeug gespeichert, so dass man in etwa abschätzen kann wie viele Fahrzeuge für eine Gegend benötigt werden.

DEPONIE (DeponieNr, Name, Typ, Volumen)

Der Entitätstyp Deponie modelliert eine Mülldeponie beliebiger Art. Jede Mülldeponie hat ihre eigene Nummer, über die sie referenziert werden kann. In einem Ort können mehrere Deponien existieren. Des Weiteren hat jede Deponie einen Namen, wie zum Beispiel “Recycling Center Findorff”. Sie kann - analog zu einem Fahrzeug - einen entsprechenden Typ von Müll entgegennehmen und weiterverarbeiten. Mülldeponien, die mehrere verschiedene Müllsorten verarbeiten können, sind in diesem Modell nicht vorhanden. Sollte dieser Fall auftreten, so ist es ohne Weiteres möglich sich die einzelnen Abteilungen als eigene Deponien vorzustellen, die in der Realität höchswahrscheinlich ihren eigenen Verwaltungsapparat besitzen. Das Volumen sagt darüber aus, wie viel Müll in einer Deponie gelagert werden kann. Somit kann man abschätzen wie viele Fahrzeuge mit einem entsprechenden Volumen Müll ablagern können, bevor die Deponie “überläuft”.

ORT (OrtID, Name, PLZ-Bereich, Einwohner)

Der Entitätstyp Ort dient dazu, um einerseits die Lokalisierung verschiedener Deponien vornehmen zu können, andererseits um eine einfachere und ortsbezogenere Aufteilung in verschiedene Gebiete zu ermöglichen. Er ist nicht zu verwechseln mit dem Ort, an dem eine Person wohnt. Bei diesem Entitätstyp ist nur ein bestimmter PLZ-Bereich zusammen mit dem Namen von Bedeutung (zum Beispiel 28). Außerdem wird eine Einwohnerzahl abgespeichert. Diese kann einerseits dazu dienen, um abzuschätzen wie viele Deponien oder Fahrzeuge notwendig sind, andererseits aber auch dazu, um in etwa eine Schätzung abzugeben, wie viele der Einwohner auch tatsächliche Kunden sind.

GEBIET (GebietsNr, Name, Typ, Startdatum, Rhytmus)

Orte alleine sind - vor allem wenn es sich um Großstädte handelt - zu granulär um zum Beispiel eine Planung über die Anzahl der einzusetzenden Fahrzeuge zu machen. Aus diesem Grund existiert der Entitätstyp Gebiet. Er hat nichts mit eigentlichen Stadtteilen gemeinsam, und kann unter Umständen sogar mehrere abdecken. Ein Gebiet ist schlussfolgernd lediglich eine für organisatorische Zwecke eingeführte Komponente, die eine künstliche Einteilung einer Gegend vornimmt. Gebiete können - vom Standpunkt einer Karte ausgesehen - problemlos überlappen. Somit ist es möglich für einen bestimmten geographischen Bereich zwei Gebiete zu definieren, wobei eines davon mit einer Restmüllabfuhr und das andere mit einer Papierabfuhr versorgt werden muss. Beide Prozesse können vollständig unabhängig voneinander durchgeführt werden. In geograpischen Bereichen, in denen die Haushalte selber dafür sorgen müssen, dass zum Beispiel der Papiermüll in ein naheliegendes Recycling-Center abgeliefert werden muss, ist diese Einteilung nicht sinnvoll. Das Modell ist in dieser Hinsicht flexibel und ermöglicht beide Interpretationen. Ein Gebiet hat einen frei wählbaren Namen, einen Typen, Startdatum und Rhytmus. Der Typ kann wie bereits bei einem Fahrzeug oder einer Deponie frei gewählt werden, wobei auch hier für die weitere Betrachtung ausschließlich die Typen Bio-, Papier- und Restmüll relevant sind. Das Startdatum legt fest, ab welchem Datum eine Abfuhr eines entsprechenden Types in einem Gebiet durchgeführt werden soll. Der Rhytmus legt fest, wie viele Tage zwischen einer Abfuhr liegen. Auf Grundlage dieser Daten kann relativ leicht festgestellt werden, wann eine nächste Abfuhr in einem Gebiet durchgeführt werden muss oder wann regulär eine Müllabfuhr stattgefunden hat.

HAUSHALT (HaushaltID, Personen)

Haushalt als Entitätstyp dient in der aktuellen Evolutionsstufe des Projekts dazu, um die Anzahl an Personen zuzuordnen. Alternativ hätte man die Personenanzahl direkt dem Kunden als Attribut zuordnen können, jedoch macht dies keinen Sinn - wie kann ein Kunde eine Personanzahl besitzen? Des Weiteren wurde versucht das Modell erweiterungsfähig zu machen. Denkbar wäre es also zum Beispiel dem Haushalt weitere Eigenschaften, die nicht unbedingt dem Kunden zuzuordnen sind - wie eine Art Maximalvolumen - zuzuweisen, das bestimmt wie viel Abfall man als Haushalt insgesamt produzieren darf. In Zeiten immer höher steigender Kosten und der Fokussierung auf Umweltschutz liegt dies sehr nahe.

MÜLLTONNE (TonnenNr, Typ, Volumen)

Eine Mülltonne wird einem Haushalt zugeordnet und besitzt eine eindeutige Nummer, so dass man eine Aussage über das Verhalten eines Kunden machen kann. Des Weiteren darf in eine konkrete Tonne nur ein bestimmter Typ von Müll, der über das entsprechende Attribut festgelegt wird. Die Tonne hat darüber hinaus ein Volumen. Je nach Haushaltsgröße können verschiedene Tonnengrößen und -typen zur Verfügung gestellt werden.

3.3. Beziehungen

Im Nachfolgenden werden die einzelnen Beziehungen textuell beschrieben. In den meisten Fällen wurde bereits in Zusammenhang mit den einzelnen Entitäten auf die Beziehungen eingegangen, so dass sich hier aus Gründen der Vollständigkeit nur kürzere Erläuterungen befinden.

fährt (MITARBEITER; FAHRZEUG)

Mitarbeiter fahren Fahrzeuge. Ein Mitarbeiter kann nur ein Fahrzeug fahren und ein Fahrzeug kann nur von einem Mitarbeiter gefahren werden.

liefertMüllAb (FAHRZEUG, DEPONIE)

Müll wird von Fahrzeugen in einer Deponie abgeliefert. Dadurch, dass ein Fahrzeug nur Müll eines entsprechenden Typs transportieren kann, wird auch nur Müll in genau einer Deponie abgeliefert. Das Abliefern von Teilladungen in unterschiedlichen Deponien macht an dieser Stelle wenig Sinn. Ein denkbarer Fall wäre, die Ausschöpfung der Kapazität einer Deponie, doch das System wird dafür entworfen, um solche Situationen zu verhindern. In einer Deponie hingegen können hingegen

beliebig viele Fahrzeuge den transportieren Abfall abliefern, so lange sie über genug Kapazitäten verfügt.

versorgt (FAHRZEUG, GEBIET)

Logisch definierte Gebiete werden von Fahrzeugen versorgt. Dabei kann ein Gebiet von beliebig vielen Fahrzeugen versorgt werden - je nachdem wie groß es ist, wie viele Haushalte sich dort befinden und viel Müll dadurch produziert wird. Auch der Typ des Abfalls spielt hierbei eine Rolle. Es ist zwar nicht angenehm, wenn ein Gebiet durch kein einziges Fahrzeug versorgt wird. Dies wird allerdings zugelassen, wenn Gebiete zum Beispiel neu erschlossen werden oder noch keine registrierten Haushalte in einem Gebiet liegen.

besitzt (HAUSHALT, MÜLLTONNE)

Jeder Haushalt besitzt mindestens eine Mülltonne und eine Mülltonne kann nur exakt zu einem Haushalt gehören. Die Zuordnung einer Mülltonne geschieht deswegen zu einem Haushalt und nicht zu einem Kunden, weil aufgrund der Haushaltsgröße das zugeordnete Volumen ermittelt wird.

liegtIn (GEBIET, ORT)

Ein Gebiet liegt in einem bestimmten Ort und ein Ort kann in mehrere Gebiete aufgeteilt sein, er muss aber zumindest genau ein logisches Gebiet beinhalten, weil diese Form von Entität grundlegend für die Planung ist.

liegtIn (HAUSHALT, GEBIET)

Auch Haushalte liegen in Gebieten. Für Haushalte gelten die gleichen Beziehungsregeln wie für Gebiete und Orte. Diese Form der Beziehung verdeutlicht noch einmal den zentralen Aspekt eines logischen Gebietes und abstrahiert die Planung von konkreten geographischen Lagen.

hat (DEPONIE, ORT)

Jede Deponie hat einen Ort zu dem sie gehört. Orte müssen keine Deponien haben. Man stelle sich vor, dass eine Deponie in einer Kleinstadt existiert, die von zahlreichen Dörfern umgeben ist. Hierbei ist die Stadt dafür verantwortlich, dass in dem entsprechenden Landkreis der Müll entsorgt wird, sie muss allerdings nicht in jedes Dorf mit eigener Postleitzahl eine eigene Deponie errichten.

istTeilVon (KUNDE, HAUSHALT)

Jeder Kunde ist Teil eines entsprechenden Haushalts und ein Haushalt muss mindestens einen Kunden besitzen. Im anderen Falle macht das Speichern eines Haushalts keinen Sinn. Die Bedeutung eines Haushalts wurde bereits an mehreren Stellen im Dokument erläutert.

ist (PERSON, KUNDE)

Jeder Kunde ist auch eine Person. Eine Person ist somit eine Verallgemeinerung eines Kunden.

ist (PERSON, MITARBEITER)

Jeder Mitarbeiter ist auch eine Person. Eine Person ist somit eine Verallgemeinerung eines Mitarbeiters.

4. Integritätsbedingungen

An dieser Stelle werden die Integritätsbedingungen des zugrunde liegenden Datenmodells dargestellt. Hierbei wird eine formale Notation verwendet, die durch eine informelle Beschreibung erläutert wird. Durch die Integritätsbedingungen wird sichergestellt, dass keine inkonsistenten oder ungültigen Daten gespeichert werden, die zu einem unvorhergesehenen Zustand führen.

Weitere Integritätsbedingungen, die an dieser Stelle nicht aufgeführt werden, sind unter Umständen den nachfolgenden Kapiteln zu entnehmen.

Die Menge der natürlichen Zahlen $\mathbb{N}$ beinhaltet laut unserer Definition 1 als die kleinste natürliche Zahl. Durch diese Annehmlichkeit ist es in einigen Fällen nicht notwendig zusätzliche Bedingungen zu erstellen, die Wertebereiche x mit der Eigenschaft $x > 0$ festlegen. Somit wird eine höhere Übersicht gewährleistet.

KUNDE

1. $\forall$ K1, K2 $\in$ KUNDE (K1.KundenNr = K2.KundenNr $\Rightarrow$ K1 = K2)

Kunden dürfen nicht die gleiche Kundennummer haben, da Sie ansonsten nicht mehr eindeutig identifizierbar sind.

2. $\forall$ K $\in$ KUNDE (K.KundenNr = x | x $\neq \varnothing$)

Jeder Kunde muss eine Kundennummer besitzen.

3. $\forall$ K1, K2 $\in$ KUNDE (K1.Vorname = K2.Vorname $\wedge$ K1.Nachname = K2.Nachname $\wedge$ K1.Geburtsdatum = K2.Geburtsdatum $\wedge$ K1.Strasse $\wedge$ K2.Strasse $\wedge$ K1.PLZ = K2.PLZ $\wedge$ K1.Ort = K2.Ort $\Rightarrow$ K1 = K2)

Kunden dürfen darüber hinaus nicht den gleichen Vornamen, Nachnamen, Geburtsdatum, Strasse, PLZ und Ort haben. Dies spiegelt die personenbezogenen eindeutigen Daten wider, die ein Mensch im wirklichen Umfeld aufweist.

4. $\forall$ K $\in$ KUNDE (K.Vorname = x $\wedge$ K.Nachname = y $\wedge$ K.Geburtsdatum = z $\wedge$ K.Strasse = i $\wedge$ K.PLZ = j $\wedge$ K.Ort = k | x, y, z, i, j, k $\neq \varnothing$)

Jeder Kunde muss einen Vornamen, Nachnamen, Geburtsdatum, Strasse, PLZ und Ort haben.

5. $\forall$ K $\in$ KUNDE $\exists$ H $\in$ HAUSHALT (K.HaushaltID = H.HaushaltID)

Jeder Kunde besitzt einen Haushalt, dem er zugeordnet werden kann. Somit wird sicher gegangen, dass keine leeren Haushalte existieren, die folgerichtig keinen Müll produzieren und entsprechend irrelevant sind.

MITARBEITER

1. $\forall$ M1, M2 $\in$ MITARBEITER (M1.PersonalNr = M2.PersonalNr $\Rightarrow$ M1 = M2)

Mitarbeiter dürfen nicht die gleiche Personalnummer haben, da Sie ansonsten nicht mehr eindeutig identifizierbar sind.

2. $\forall$ M $\in$ MITARBEITER (K.PersonalNr = x | x $\neq \varnothing$)

Jeder Mitarbeiter muss eine Personalnummer besitzen.

3. $\forall$ M1, M2 $\in$ MITARBEITER (M1.Vorname = M2.Vorname $\wedge$ M1.Nachname = M2.Nachname $\wedge$ M1.Geburtsdatum = M2.Geburtsdatum $\wedge$ M1.Strasse $\wedge$ M2.Strasse $\wedge$ M1.PLZ = M2.PLZ $\wedge$ M1.Ort = M2.Ort $\Rightarrow$ M1 = M2)

Mitarbeiter dürfen darüber hinaus nicht den gleichen Vornamen, Nachnamen, Geburtsdatum, Strasse, PLZ und Ort haben. Dies spiegelt die personenbezogenen eindeutigen Daten wider, die ein Mensch im wirklichen Umfeld aufweist.

4. $\forall$ M $\in$ MITARBEITER (M.Vorname = x $\wedge$ M.Nachname = y $\wedge$ M.Geburtsdatum = z $\wedge$ M.Strasse = i $\wedge$ M.PLZ = j $\wedge$ M.Ort = k | x, y, z, i, j, k $\neq \varnothing$)

Jeder Mitarbeiter muss einen Vornamen, Nachnamen, Geburtsdatum, Strasse, PLZ und Ort haben.

5. $\forall$ M $\in$ MITARBEITER $\forall$ F $\in$ FAHRZEUG (M.PersonalNr = F.PersonalNr $\Rightarrow$ M.Fuehrerschein = ja)

Es dürfen nur Mitarbeiter mit Fahrberechtigung einem Fahrzeug zugeordnet werden. Unqualifizierte Mitarbeiter dürfen keine Fahrzeuge fahren.

6. $\forall$ M $\in$ MITARBEITER (M.Fuehrerschein = x | x = {ja, nein})

Jeder Mitarbeiter hat entweder einen oder keinen Führerschein.

FAHRZEUG

1. $\forall$ F1, F2 $\in$ FAHRZEUG (F1.Kennzeichen = F2.Kennzeichen $\Rightarrow$ F1 = F2)

Fahrzeuge dürfen nicht über das gleiche amtliche Kennzeichen verfügen, da dies gegen die aktuelle Gesetzgebung verstößt.

2. $\forall$ F $\in$ FAHRZEUG (F.Volumen = x $\wedge$ F.Typ = y $\wedge$ F.Wartung = z | x, y, z $\neq \varnothing$)

Jedes Fahrzeug muss ein bestimmtes Volumen haben und es muss zu jeder Zeit feststellbar sein, welchen Abfalltyp das Fahrzeug transportieren kann und wann die nächste Wartung stattfinden soll.

3. $\forall$ F $\in$ FAHRZEUG $\exists$ D $\in$ DEPONIE (F.DeponieNr = D.DeponieNr)

Jedes Fahrzeug liefert Müll in einer Deponie ab, die auch tatsächlich existiert.

4. $\forall$ F $\in$ FAHRZEUG (F.Volumen = x | x $\in \mathbb{N}$)

Das Volumen eines Fahrzeugs muss eine natürliche Zahl sein.

Darüber hinaus kann (muss aber nicht) festgelegt werden, dass ein Fahrzeug nur Müll in einer Deponie abliefern sollte, die mit dem entsprechenden übereinstimmt und dass ein Fahrzeug nur ein Gebiet bedienen sollte, dessen Typ übereinstimmt.

DEPONIE

1. $\forall$ D1, D2 $\in$ DEPONIE (D1.DeponieNr = D2.DeponieNr $\Rightarrow$ D1 = D2)

Jede Deponie ist über ihre Deponienummer eindeutig identifizierbar.

2. $\forall$ D $\in$ DEPONIE (D.Typ = x $\wedge$ D.Volumen = y | x, y $\neq \varnothing$)

Jede Deponie muss einen Typen sowie ein maximales Volumen (Kapazität) haben. Der Name ist freiwillig, da Deponien über ihre Nummern bereits eindeutig identifiziert werden können. Er dient der besseren Interpretation der Daten und kann zum Beispiel “Mülldeponie Bremen Nord” lauten.

3. $\forall$ D $\in$ DEPONIE $\exists$ O $\in$ ORT (D.OrtID = O.OrtID)

Jede Deponie besitzt einen Ort, dem sie zugeordnet werden kann. Es würde wenig Sinn machen Deponien zuzulassen, die keinerlei geographische Lage besitzen.

4. $\forall$ D $\in$ DEPONIE (D.Volumen = x | x $\in \mathbb{N}$)

Das Volumen einer Deponie muss eine natürliche Zahl sein.

ORT

1. $\forall$ O1, O2 $\in$ ORT (O1.OrtID = O2.OrtID $\Rightarrow$ O1 = O2)

Orte dürfen nicht die gleiche eindeutige Kennzeichnung haben, da Sie ansonsten nicht mehr korrekt identifizierbar sind. Die OrtID kann dieses nur unter der hier genannten Integritätsbedingung gewährleisten.

2. $\forall$ O $\in$ ORT (O.Name = x $\wedge$ O.PLZ-Bereich = y $\wedge$ O.Einwohner = z | x, y, z $\neq \varnothing$)

Jeder Ort muss einen gültigen Namen, Bereich und die Anzahl der Einwohner besitzen. Orte ohne diese Attribute gibt es in der Realität nicht, daher werden sie mit derartigen Einschränkungen auf das Modell abgebildet.

3. $\forall$ O $\in$ ORT (O.Einwohner = x | x $\in \mathbb{N}$)

Die Einwohneranzahl eines Ortes ist eine natürliche Zahl.

GEBIET

1. $\forall$ G1, G2 $\in$ GEBIET (G1.GebietsNr = G2.GebietsNr $\Rightarrow$ G1 = G2)

Gebiete dürfen nicht die gleiche Gebietsnummer aufweisen, da Sie ansonsten nicht mehr eindeutig identifizierbar sind.

2. $\forall$ G $\in$ GEBIET (G.GebietsNr = x $\wedge$ G.Typ = y $\wedge$ G.Startdatum = z $\wedge$ G.Rhytmus = i | x, y, z, i $\neq \varnothing$)

Jedes Gebiet muss eine Gebietsnummer, einen Typ, Startdatum und Rhytmus haben. Somit wird davon ausgegangen, dass lediglich Gebiete mit einem konkreten Mehrwert und Nutzen für das Unternehmen im Datenbestand gepflegt werden. Gebiete, die nicht von Fahrzeugen versorgt werden sollten nicht im Datenbestand auftauchen.

3. $\forall$ G $\in$ GEBIET $\exists$ O $\in$ ORT (G.OrtID = O.OrtID)

Jedes Gebiet liegt in exakt einem Ort.

4. $\forall$ G $\in$ GEBIET (G.Rhytmus = x | x $\in \mathbb{N}$)

Der Rhytmus eines Gebietes ist eine natürliche Zahl. Er repräsentiert den Tagesrhytmus der Versorgung eines Gebietes durch entsprechende Fahrzeuge.

HAUSHALT

1. $\forall$ H1, H2 $\in$ HAUSHALT (H1.HaushaltID = H2.HaushaltID $\Rightarrow$ H1 = H2)

Jeder Haushalt hat eine Haushalts-Identifikationsnummer, so dass er eindeutig identifiziert werden kann.

2. $\forall$ H $\in$ HAUSHALT (H.HaushaltID = x $\wedge$ H.Personen = y | x, y $\neq \varnothing$)

Jeder Haushalt benötigt eine HaushaltID und eine Anzahl an Personen. Haushalte ohne Personen sind für das Unternehmen irrelevant, da sie keinerlei Abfälle produzieren.

3. $\forall$ H $\in$ HAUSHALT (H.Personen = x | x $\in \mathbb{N}$)

Die Anzahl der Personen in einem Haushalt ist eine natürliche Zahl. Halbe Personen machen genau so wenig Sinn, wie eine zu genaue Spezifizierung des Volumens einer Deponie oder Tonne (z.B. 60,15 Liter).

4. $\forall$ H $\in$ HAUSHALT $\exists$ G $\in$ GEBIET (H.GebietsNr = G.GebietsNr)

Jeder Haushalt liegt in exakt einem Gebiet.

MÜLLTONNE

1. $\forall$ M1, M2 $\in$ MÜLLTONNE (M1.TonnenNr = M2.TonnenNr $\Rightarrow$ M1 = M2)

Jede Mülltonne besitzt eine global eindeutig identifizierbare Tonnennummer.

2. $\forall$ M $\in$ MÜLLTONNE (M.TonnenNr = x $\wedge$ M.Typ = y $\wedge$ M.Volumen = z | x, y, z $\neq \varnothing$)

Jede Mülltonne muss eine Tonnenummer, einen Typ und ein Volumen besitzen.

3. $\forall$ M $\in$ MÜLLTONNE (M.Volumen = x | x $\in \mathbb{N}$)

Das Volumen einer Mülltonne wird in Litern als natürliche Zahl angegeben. Eine Mülltonne muss mindestens ein Volumen von einem Liter haben. Das Volumen

einzelner Tonnen ist in der Realität so hoch, dass es keinen Sinn macht Nachkommastellen zu berücksichtigen.

4. $\forall$ M $\in$ MÜLLTONNE $\forall$ H $\in$ HAUSHALT (M.HaushaltID = H.HaushaltID)

Jede Mülltonne ist eindeutig zu einem Haushalt zuzuordnen.

5. Realisierung als relationales DB-Schema

In diesem Abschnitt wird das bisher erstellte Modell in eine Ansammlung von Relationen umgewandelt. Hierbei werden zuerst die Entitätstypen und anschließend die Beziehungen übersetzt.

Die meisten Entitätstypen können direkt als Relationen übernommen werden. Dabei werden die Attribute der Entitätstypen auf die Relationen abgebildet. Bei Relationen, die über eine 1:1 oder 1:n Beziehung miteinander verknüpft sind, wird zusätzlich bei der entsprechenden Relation ein Fremdschlüssel angelegt und die eigentliche Beziehung kann entfallen. n:m Beziehungen werden in eine eigenständige Relation übersetzt.

5.1. Übersetzte Entitätstypen

KUNDE (Vorname, Nachname, Geburtsdatum, Strasse, PLZ, Ort, Telefon, E-Mail, KundenNr, Kontostand, HaushaltID)

Der Entitätstyp KUNDE kann übernommen werden. Die Kundennummer dient in diesem Fall als Primärschlüssel, da sie eine kleinere Wertemenge bildet, als die eindeutigen Personeninformationen zusammen. Die Beziehung zwischen KUNDE und HAUSHALT wird als Fremdschlüssel modelliert.

MITARBEITER (Vorname, Nachname, Geburtsdatum, Strasse, PLZ, Ort, Telefon, E-Mail, PersonalNr, Fuehrerschein)

Analog zum Entitätstyp KUNDE kann auch der Entitätstyp MITARBEITER übernommen werden. Des Weiteren wird die Personalnummer als Primärschlüssel aus den bereits oben genannten Gründen verwendet.

FAHRZEUG (Kennzeichen, Typ, Volumen, Wartung, PersonalNr, DeponieNr)

Die Relation FAHRZEUG kann vollständig von dem Entitätstyp übernommen werden. Die Beziehung zwischen FAHRZEUG und MITARBEITER wird als Fremdschlüssel modelliert. Analog dazu wird die Beziehung zu einer DEPONIE modelliert.

DEPONIE (DeponieNr, Name, Typ, Volumen, OrtID)

Die Relation DEPONIE kann vollständig von dem Entitätstyp übernommen werden. Die Beziehung zwischen DEPONIE und ORT wird als Fremdschlüssel modelliert.

ORT (OrtID, Name, PLZ-Bereich, Einwohner)

Die Relation ORT kann vollständig von dem Entitätstyp übernommen werden.

GEBIET (GebietsNr, Name, Typ, Startdatum, Rhytmus, OrtID)

Die Relation GEBIET kann vollständig von dem Entitätstyp übernommen werden. Die Beziehung zwischen GEBIET und ORT wird als Fremdschlüssel modelliert.

HAUSHALT (HaushaltID, Personen, GebietsNr)

Die Relation HAUSHALT kann vollständig von dem Entitätstyp übernommen werden. Die Beziehung zwischen HAUSHALT und GEBIET wird als Fremdschlüssel modelliert.

MÜLLTONNE (TonnenNr, Typ, Volumen, HaushaltID)

Die Relation MÜLLTONNE kann vollständig von dem Entitätstyp übernommen werden. Die Beziehung zwischen MÜLLTONNE und HAUSHALT wird als Fremdschlüssel modelliert.

5.2. Übersetzte Beziehungen

FAHRZEUGVERSORGTGEBIET (FahrzeugID, GebietsNr)

Die Beziehung versorgt (FAHRZEUG, GEBIET) wurde in eine neue Relation FAHRZEUGVERSORGTGEBIET übersetzt, dessen Primärschlüssel sowohl die eindeutige FahrzeugID, als auch die GebietsNr sind. Auf diese Weise kann die bereits im ER-Diagramm modellierte n:m Beziehung umgesetzt werden.

Entfallene Beziehungen

Aufgrund der bereits erfolgten Modellierung verschiedener Beziehungen als Fremdschlüssel, können folgende Beziehungen entfallen:

1. fährt (MITARBEITER; FAHRZEUG)
2. liefertMüllAb (FAHRZEUG, DEPONIE)
3. besitzt (HAUSHALT, MÜLLTONNE)
4. liegtIn (GEBIET, ORT)
5. liegtIn (HAUSHALT, GEBIET)
6. hat (DEPONIE, ORT)
7. istTeilVon (KUNDE, HAUSHALT)

Folgende Beziehungen entfallen, da sie eine ist-Beziehung darstellen, die lediglich zu Zwecken der abstrakten Modellierung von Vererbung dient:

1. ist (PERSON, KUNDE)
2. ist (PERSON, MITARBEITER)

6. Standardoperationen in SQL (und QUEL)

In diesem Kapitel wird das zuvor definierte relationale DB-Schema am konkreten Beispiel mit Standardanfragen in SQL verarbeitet. Anschließend werden im letzten Unterabschnitt einige Beispiele für die Sprache QUEL aufgezeigt.

6.1. Erzeugen von Tabellen

Im Nachfolgenden werden alle oben erwähnten Relationen mit Hilfe von SQL erzeugt. Neben dem Namen der einzelnen Relationen werden alle Attribute, sowie Primär- und Fremdschlüssel angegeben. Attribute besitzen einen bestimmten Datentypen und einen Wertebereich, der hinter ihrem Namen angegeben wird. Außerdem befinden sich Integritätsbedingungen hinter dem Datentypen eines Attributs. Fremdschlüssel referenzieren ein Attribut einer anderen Relation, die beide in diesem Zusammenhang syntaktisch notiert werden müssen.

KUNDE

```
CREATE TABLE Kunde (
     KundenNr int(11) NOT NULL,
     Kontostand int(11) NOT NULL,
     Vorname varchar(50) NOT NULL,
     Nachname varchar(50) NOT NULL,
     Geburtsdatum date NOT NULL,
     Strasse varchar(50) NOT NULL,
     PLZ varchar(10) NOT NULL,
     Ort varchar(50) NOT NULL,
     Telefon varchar(20) default NULL,
     E-Mail varchar(50) default NULL,
     HaushaltID int(11) NOT NULL,
     PRIMARY KEY (KundenNr),
     FOREIGN KEY (HaushaltID) ⏎
          REFERENCES Haushalt (HaushaltID)
)
```

MITARBEITER

```
CREATE TABLE Mitarbeiter (
     PersonalNr int(11) NOT NULL,
     Fuehrerschein enum('ja','nein') NOT NULL,
     Vorname varchar(50) NOT NULL,
     Nachname varchar(50) NOT NULL,
     Geburtsdatum date NOT NULL,
```

```
     Strasse varchar(50) NOT NULL,
     PLZ varchar(10) NOT NULL,
     Ort varchar(50) NOT NULL,
     Telefon varchar(20) NOT NULL,
     E-Mail varchar(50) NOT NULL,
     PRIMARY KEY (PersonalNr)
)
```

FAHRZEUG

```
CREATE TABLE Fahrzeug (
     Kennzeichen varchar(10) NOT NULL,
     Volumen int(11) NOT NULL,
     Typ varchar(20) NOT NULL,
     Wartung date NOT NULL,
     PersonalNr int(11) NOT NULL,
     DeponieNr int(11) NOT NULL,
     PRIMARY KEY (Kennzeichen),
     FOREIGN KEY (PersonalNr) ⏎
          REFERENCES Mitarbeiter (PersonalNr)
     FOREIGN KEY (DeponieNr) ⏎
          REFERENCES Deponie (DeponieNr)
)
```

DEPONIE

```
CREATE TABLE Deponie (
     DeponieNr int(11) NOT NULL,
     Name varchar(50) default NULL,
     Typ varchar(20) NOT NULL,
     Volumen int(11) NOT NULL,
     OrtID int(11) NOT NULL,
     PRIMARY KEY  (DeponieNr),
     FOREIGN KEY (OrtID) REFERENCES Ort (OrtID)
)
```

ORT

```
CREATE TABLE Ort (
     OrtID int(11) NOT NULL,
     Name varchar(50) NOT NULL,
     PLZ-Bereich varchar(10) NOT NULL,
```

```
     Einwohner int(11) NOT NULL,
     PRIMARY KEY (OrtID)
)
```

GEBIET

```
CREATE TABLE Gebiet (
     GebietsNr int(11) NOT NULL,
     Name varchar(50) default NULL,
     Typ varchar(20) NOT NULL,
     Startdatum date NOT NULL,
     Rhytmus int(11) NOT NULL,
     OrtID int(11) NOT NULL,
     PRIMARY KEY (GebietsNr),
     FOREIGN KEY (OrtID) REFERENCES Ort (OrtID)
)
```

HAUSHALT

```
CREATE TABLE Haushalt (
     HaushaltID int(11) NOT NULL,
     Personen tinyint(4) NOT NULL,
     GebietsNr int(11) NOT NULL,
     PRIMARY KEY (HaushaltID),
     FOREIGN KEY (GebietsNr) ⏎
          REFERENCES Gebiet (GebietsNr)
)
```

MÜLLTONNE

```
CREATE TABLE Muelltonne (
     TonnenNr int(11) NOT NULL,
     Typ varchar(20) NOT NULL,
     Volumen int(11) NOT NULL,
     HaushaltID int(11) NOT NULL,
     PRIMARY KEY (TonnenNr),
     FOREIGN KEY (HaushaltID) ⏎
          REFERENCES Haushalt (HaushaltID)
)
```

FAHRZEUGVERSORGTGEBIET

```
CREATE TABLE FahrzeugVersorgtGebiet (
  Kennzeichen varchar(10) NOT NULL,
  GebietsNr int(11) NOT NULL,
  PRIMARY KEY (Kennzeichen,GebietsNr)
)
```

6.2. Modifizieren der Tabellenstruktur

Löschen und Umbennen von Tabellen

Tabellen können komplett aus der gesamten Datenbank gelöscht werden. Das Löschen der gesamten Tabelle Haushalt würde zum Beispiel mit folgendem Befehl durchgeführt werden:

```
DROP TABLE Haushalt;
```

Eine Tabelle kann man mit folgender Syntax umbennen:

```
ALTER TABLE Haushalt RENAME Wohnung;
```

Modifizieren von Attributen in Tabellen

Tabellenstrukturen können nach ihrem Erstellen mit Hilfe von SQL verändert werden. Eine oft anzutretende Operation, die mit der Evolution eines Datenbanksystementwurfs Verwendung findet, ist das Hinzufügen oder Löschen von Attributen. Zum Beispiel kann man der Tabelle Haushalt ein Attribut hinzufügen, dass ein maximales Abholvolumen festlegt:

```
ALTER TABLE Haushalt ADD MaxVolumen int(11);
```

Dieses Attribut kann man mit folgendem Befehl wieder löschen:

```
ALTER TABLE Haushalt DROP MaxVolumen;
```

Es gibt auch eine Möglichkeit nachträglich den Datentyp eines Attributs zu verändern. Als Beispiel wird an dieser Stelle die Personenanzahl modifiziert:

```
ALTER TABLE Haushalt MODIFY Personen int(11);
```

Außer diesen Operationen ist es möglich Attribute umzubennen. Folgendes Beispiel ändert den Namen des Attributs Personen in AnzahlPersonen:

```
ALTER TABLE Haushalt CHANGE Personen AnzahlPersonen int(11);
```

6.3. Modifizieren der vorhandenen Daten

In diesem Abschnitt finden Grundoperationen Erwähnung, die von SQL unterstützt werden, wenn man nicht die eigentliche Struktur, sondern gespeicherte Daten löschen, ändern oder einfügen möchte.

Löschen von Daten

Zum Löschen aller Mülltonnen aus der Tabelle Muelltonne kann folgender Befehl werden werden:

```
DELETE FROM Muelltonne;
```

Man möchte nur in bestimmten Situationen eine komplette Tabelle leeren, daher kann mit folgendem Befehl nur eine bestimmte Mülltonne aus der Tabelle entfernt werden:

```
DELETE FROM Muelltonne WHERE TonnenNr = 1234;
```

Alternativ könnte man alle Mülltonnen löschen, deren Typ Papier ist, da diese Sorte von Abfall zum Beispiel zusammen mit dem Restmüll weggeschmissen werden kann:

```
DELETE FROM Muelltonne WHERE Typ = ‘Papier’;
```

Aktualisieren von Daten

Zum Aktualisieren von Daten in einer Tabelle kann der **UPDATE** Befehl verwendet werden. Als Beispiel sollen alle Tonnen auf den Typ Restmüll gesetzt werden, da es nur noch eine Deponie gibt, die vollautomatisch für die Sortierung des jeweiligen Abfalltyps verantwortlich ist; die Haushalte dürfen somit in jede Tonne alles hinein schmeißen:

```
UPDATE Muelltonne SET Typ = ‘Restmuell’;
```

Oft möchte man jedoch nicht dermaßen gravierende Änderungen am Datenbestand durchführen. Um zum Beispiel die Mülltonnen des Haushalts mit der Nummer 9876 auf ein Volumen von 60 Litern zu setzen, würde man folgenden Befehl verwenden:

```
UPDATE Muelltonne SET Volumen = 60 WHERE HaushaltID = 9876;
```

Einfügen von Daten

Das Hinzufügen von Daten wird mit Hilfe des **INSERT INTO** Befehls durchgeführt. Als Beispiel soll an dieser Stelle ein neuer Kunde hinzugefügt werden:

```
INSERT INTO Kunde VALUES (
     123,
     0,
     'Hans',
     'Wurst',
     '1975-05-03',
     'Neue Landstraße 20',
     'D-56789',
     'Entenhausen',
     '0865-43211234',
     'hanswurst@fleischerei.de',
     9876
);
```

Mit dieser Syntax muss man allerdings die Reihenfolge der Attribute in der entsprechenden Tabelle kennen. Daher gibt es folgende übersichtlichere Möglichkeit neue Daten einzufügen:

```
INSERT INTO Kunde SET
     KundenNr = 123
     Kontostand = 0,
     Vorname = 'Hans',
     Nachname = 'Wurst',
     Geburtsdatum = '1975-05-03',
     Strasse = 'Neue Landstraße 20',
     PLZ = 'D-56789',
     Ort = 'Entenhausen',
     Telefon = '0865-43211234',
     E-Mail = 'hanswurst@fleischerei.de',
     HaushaltID = 9876;
```

Bei dieser Form der Syntax hat man ein eindeutige Zuordnung der konkreten Daten zu den Attributsnamen und darf die Attributreihenfolge sogar vertauschen. Es gibt noch eine weitere Form der INSERT INTO Syntax, die Attributsnamen und konkrete Daten auseinander hält, auf die hier aber nicht weiter eingegangen werden soll.

6.4. Standardanfragen

Nachdem grundlegende Operationen auf Tabellen und ihren Daten gezeigt wurden, sollen nun Standardanfragen in SQL vorgestellt werden. Anfragen werden mit dem **SELECT FROM** Befehl realisiert, der durch einen **WHERE** Teil erweitert werden kann, um eingeschränkte Ergebnismengen zu erhalten. Nachfolgend werden einige Beispiele auf Grundlage des hier entworfenen Datenbanksystems dargelegt, die sowohl mit Hilfe der SQL Syntax, als auch in informeller Form erläutert werden.

Alle Kunden

Der nachfolgende einfache Befehl wählt alle Kunden aus der entsprechenden Tabelle aus.

```
SELECT * FROM Kunde;
```

Alle Kunden mit negativem Kontostand

Der WHERE Teil ermöglicht es Bedingungen für die Anfrage festzulegen. Hier werden zum Beispiel alle Kunden mit negativem Kontostand abgefragt, um ihnen beispielsweise eine Mahnung zu schicken.

```
SELECT * FROM Kunde WHERE Kontostand < 0;
```

Alle Kunden nach Geburtsdatum sortiert

Die Ergebnismenge kann auch sortiert werden. So bekommt man bei dieser Anfrage als Resultat alle Kunden, die nach ihrem Geburtsdatum sortiert sind.

```
SELECT * FROM Kunde ORDER BY Geburtsdatum;
```

Anzahl Kunden in verschiedenen Städten

Mit Hilfe der **GROUP BY** Klausel kann man die Ergebnismenge nach einem bestimmten Kriterium gruppieren. So werden hier zum Beispiel die Städte gruppiert. Des Weiteren wird die **COUNT** Klausel verwendet, die es ermöglicht die Anzahl der

Resultate zu zählen. Als Ergebnis erhält man die Anzahl der Kunden in allen gespeicherten Städten.

SELECT Ort, **COUNT(*) FROM** Kunde **GROUP BY** Ort;

Weitere häufige Funktionen, die zusammen mit der **GROUP BY** Klausel verwendet werden, sind:

1. **MIN()**: Liefert das Minimum eines Attributs (für Zahlen oder Zeichenketten)
2. **MAX()**: Liefert das Maximum eines Attributs (für Zahlen oder Zeichenketten)
3. **SUM()**: Liefert die Summe eines entsprechenden Attributs, sofern es eine Zahl ist
4. **AVG()**: Liefert den Durchschnitt analog zu **SUM()**

Älteste Kunden in verschiedenen Städten

Analog zu dem vorherigen Beispiel und unter Einbezug der vorgestellten Funktionen kann man den jeweils ältesten Kunden aus allen Städten auswählen, um ihm aus Promotionszwecken einen Einjahres-Gutschein über freie Müllentsorgung zu schenken.

```
SELECT Ort, MIN(Geburtsdatum) FROM Kunde GROUP BY Ort;
```

Anzahl ältester Kunden in verschiedenen kundenreichen Städten

Mit Hilfe der **HAVING** Klausel kann man weitere Bedingungen festlegen. So ist es zum Beispiel möglich mit folgendem Befehl jeweils den ältesten Kunden auszuwählen, der in einer kundenreichen Stadt wohnt, um somit Kosten beim Marketing einzusparen. Mund-zu-Mund Propaganda ist am günstigsten und somit kann man davon ausgehen, dass möglichst viele Kunden das Unternehmen weiterempfehlen. Ältere Leute sind vertrauenswürdiger, da sie über mehr Lebenserfahrung verfügen. Kundenreiche Städte werden dadurch definiert, dass in ihnen mehr als 500.000 Kunden wohnen.

```
SELECT Ort, COUNT(*), MIN(Geburtsdatum) FROM Kunde
     GROUP BY Ort HAVING COUNT(*) > 500000;
```

Zehn kundenreichste Städte

Ergebnismengen kann man durch die **LIMIT** Klausel in ihrer Anzahl einschränken. Folgendes Beispiel zeigt die zehn kundenreichsten Städte.

```
SELECT Ort, COUNT(*) AS Anzahl FROM Kunde
     GROUP BY Ort ORDER BY Anzahl LIMIT 10;
```

Zusätzlich wurde in diesem Beispiel ein Alias festgelegt und die Kombination verschiedener Klauseln durchgeführt.

Kunden mit unbekannter Schreibweise des Nachnamens wiederfinden

Man möchte einen Kunden wiederfinden, dessen Vorname man kennt, aber nicht mehr weiß wie der Nachname geschrieben wurde (Schmidt, Smith, Smitt, Schmitt, ...). Mit folgender Anfrage bekommt man als Ergebnis alle Kunden mit dem Vornamen Thomas und einem Nachnamen, der die Zeichenkette “mi” enthält.

```
SELECT * FROM Kunde
     WHERE Vorname = ‘Thomas’ AND Nachname LIKE ‘%mi%’;
```

Durch die Eingrenzung der Ergebnismenge ist es nun viel einfacher möglich den entsprechenden Kunden wiederzufinden. Das %-Zeichen dient als Platzhalter für beliebige Zeichen in der Zeichenkette; er kann an beliebiger Stelle und beliebig oft in der **LIKE** Klausel auftreten.

Kunden mit einem Haushalt von mehr als 5 Personen

An dieser Stelle werden zwei Tabellen über das Attribut HaushaltID miteinander verknüpft. Sie werden des weiteren mit einem Alias versehen, so dass es lesbarer ist, auf welche Attribute man genau zugreift.

```
SELECT K.KundenNr, K.Vorname, K.Nachname, H.Personen
     FROM Kunde AS K, Haushalt AS H
     WHERE K.HaushaltID = H.HaushaltID AND H.Personen > 5;
```

Kunden, die eine Papiertonne besitzen

Ähnlich zu der vorherigen Anfrage werden in diesem Beispiel Tabellen verknüpft, wobei es hier drei anstatt zwei sind.

```
SELECT K.KundenNr, K.Vorname, K.Nachname
     FROM Kunde AS K, Haushalt AS H, Muelltonne AS M
     WHERE K.HaushaltID = H.HaushaltID
     AND H.HaushaltID = M.HaushaltID
     AND M.Typ = ‘Papiertonne’;
```

Mitarbeiter mit Führerschein und dem dazugehörigen Fahrzeug

Es sollen alle Mitarbeiter angezeigt werden, die einen Führerschein besitzen. Außerdem ist es erwünscht das dazugehörige Kennzeichen und den Fahrzeugtyp anzuzeigen.

```
SELECT M.PersonalNr, M.Vorname, M.Nachname, F.Kennzeichen,
F.Typ
     FROM Mitarbeiter AS M, Fahrzeug AS F
     WHERE M.PersonalNr = F.PersonalNr
     AND M.Fuehrerschein = ‘ja’;
```

Kunden mit Haushalten, die in einem bestimmten Gebiet liegen

Nachfolgend werden alle Kunden ausgewählt, dessen Haushalte in einem Gebiet liegen, das mit der Zeichenkette hausen endet. Zusätzlich werden die Ergebnisse nach dem Nachnamen des Kunden sortiert.

```
SELECT K.KundenNr, K.Vorname, K.Nachname
     FROM Kunde AS K, Haushalt AS H, Gebiet AS G
     WHERE K.HaushaltID = H.HaushaltID
     AND H.GebietsNr = G.GebietsNr
     AND G.Name LIKE ‘%hausen’
     ORDER BY K.Nachname;
```

Weitere Anfragen

Auf Grundlage der in diesem Abschnitt genannten Palette an verschiedenartigen Anfragen kann man auf relativ einfache Weise - hauptsächlich durch Ersetzen der Attribute und Tabellen, aber auch durch die Kombination miteinander - alle anderen für das entworfene Datenbanksystem relevanten Anfragen erzeugen.

4.1. Beispiele für Standardanfragen in QUEL

Analog zu Anfragen in SQL sollen an dieser Stelle einige Beispielanfragen in QUEL formuliert werden.

Alle Kunden

Ähnlich zu einer SQL-Anfrage können alle Kunden abgefragt werden. In diesem Beispiel werden Kundennummer, Vor- und Nachname aller Kunden ausgegeben. Durch **`range of`** und **`retrieve`** wird ein ähnlicher Effekt erzielt wie durch **`SELECT FROM`**.

```
range of k is Kunde
retrieve (k.KundenNr, k.Vorname, k.Nachname)
```

Alle Kunden mit negativem Kontostand

Ähnlich zu dem vorherigen Beispiel werden alle Kunden mit negativem Kontostand abgefragt.

```
range of k is Kunde
retrieve (k.KundenNr, k.Vorname, k.Nachname)
where (k.Kontostand < 0)
```

Mitarbeiter mit Führerschein und dem dazugehörigen Fahrzeug

Im Nachfolgenden wird die - bereits im SQL-Teil erläuterte - Anfrage gestellt, dessen Resultat alle Mitarbeiter mit einem Führerschein und dem dazugehörigen Fahrzeug darstellt.

```
range of m is Mitarbeiter
range of f is Fahrzeug
retrieve (m.PersonalNr, m.Vorname, m.Nachname, f.Kennzeichen,
f.Typ)
where (m.PersonalNr = f.PersonalNr) and (m.Fuehrerschein =
‘ja’)
```

5. Sichten

Sichten spiegeln virtuelle Tabellen in einem Datenbanksystem wider. Sie werden eingesetzt, um einen bestimmten Blick auf vorliegende Daten zu erlangen (zum Beispiel aus der Perspektive einer bestimmten Anwendergruppe), können aber auch für die Vereinfachung von Standardanfragen verwendet werden, da sie effektiv eine SQL-Anfrage widerspiegeln. Einen genaueren Einblick in Sichten, insbesondere in die damit im Zusammenhang stehenden Ebenen einer Architektur bietet das veranstaltungsbegleitende Material auf das an dieser Stelle verwiesen wird (siehe [1]).

Ein Beispiel für eine Sicht ist die verbundene Ansicht einer Deponie zusammen mit dem dazugehörigen Ort und den Mitarbeitern, dessen Fahrzeug Müll abliefert. Mit dieser Sicht kann man sich einen relativ guten Überblick über die vorhandenen Ressourcen verschaffen. Außerdem kann diese Sicht gut für Standardanfragen dienen, die zum Beispiel ein Gebiet mit einschließen. Das Verwenden einer Sicht erhöht hierbei die Übersichtlichkeit, da in der angesprochenen Anfrage nicht mehr so viele Tabellen miteinander verbunden werden müssten.

```
CREATE VIEW Ressourcenueberblick AS
SELECT O.Name, O.PLZ-Bereich, D.Name, D.Typ, D.Volumen,
F.Kennzeichen, F.Wartung, M.PersonalNr, M.Vorname, M.Nachname
FROM Ort AS O, Deponie AS D, Fahrzeug AS F, Mitarbeiter AS M
WHERE O.OrtId = D.OrtId
AND D.DeponieNr = F.DeponieNr
AND F.PersonalNr = M.PersonalNr
AND M.Fuehrerschein = ‘ja’
AND F.Typ = D.Typ
```

6. Literatur

[1] Martin Gogolla; Begleitmaterial zur Veranstaltung Datenbanksysteme im WS 2008/09: http://db.informatik.uni-bremen.de/teaching/courses/ws2008_dbs/.

[2] Andreas Heuer, Gunter Saake; Datenbanken: Konzepte und Sprachen, 2 Auflage; Bonn: mitp-Verlag, 2000.