B. Mitschang et al. (Hrsg.): BTW 2017 – Workshopband,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2017 355

Modellierung von relationalen Datenbanken mit UML im Round-Trip-Engineering[1]

Björn Salgert[2], Thomas C. Rakow[3]

Abstract: Die Unified Modeling Language (UML) wird mithilfe eines Profils erweitert, um den Entwurf relationaler Datenbanken zu unterstützen. Elemente von UML wie Vererbung können zur Daten-bankmodellierung genutzt werden, und datenbankspezifische Elemente wie beispielweise Indizes lassen sich in UML modellieren. Es wird eine automatisierte Abbildung vom UML-Klassendiagramm zum Datenbankschema erklärt und das Round-Trip-Engineering zwischen UML und einem Datenbankschema dargestellt. Schließlich wird der Nutzen dieser Modellierungsmethode verdeutlicht.

Keywords: UML, Schema-Evolution, Objekt-relationale Abbildung, Round-Trip-Engineering

1 Einleitung

Beim Datenbankentwurf kommen verschiedene Diagramme zum Einsatz, wie das Entity-Relationship-Model oder das Relationendiagramm. Dabei wird das ER-Diagramm auf das Relationendiagramm abgebildet, danach wird aus dem Relationenmodell das SQL-Schema erstellt. Gebräuchliche Datenbankfunktionalitäten wie Indizes, Primärschlüssel oder Constraints lassen sich mit den verwendeten Diagrammen nicht abbilden. Durch die Verwendung mehrerer Diagramme (ER-Diagramm, Relationenmodell) entsteht bei Änderungsanforderungen der Aufwand, alle Diagramme aktualisieren zu müssen, damit die vorhandenen Diagramme nicht veralten. Eine Vereinfachung dieses Prozesses ist sinnvoll. Dazu bietet es sich an, zur Modellierung die in der Software-Entwicklung weit verbreitete Unified Modeling Language (UML) zu verwenden [BQ13; Fo07; OS13; RQ12].

Durch die Verwendung der UML werden die Datenbankmodelle einem breiteren Kreis von Entwicklern zugänglich gemacht. Dies verbessert u. a. die Kommunikation in Projekten. Der Ansatz dieser Arbeit ist es, die UML zum Entwickeln von Datenbankmodellen zu nutzen. Dadurch soll ermöglicht werden, ein UML-Klassendiagramm für die Anwendungslogik und die Datenbank zu verwenden, um so eine einfachere Anwendungsentwicklung zu ermöglichen. Durch eine Abstraktion zwischen dem UML-Diagramm und dem Datenbankmodell wird eine Unabhängigkeit von einem spezifischen Datenbankmanagmentsystem erreicht. Dadurch muss die Wahl für ein konkretes Datenbankmanagmentsystem erst spät in einem Projekt getroffen werden.

[1] Diese Arbeit basiert auf der Masterarbeit des Erstautors [Sa16]

[2] Hochschule Düsseldorf, Fachbereich Medien, Münsterstraße 156, 40476 Düsseldorf, bjoern.salgert@hs-duesseldorf.de

[3] Hochschule Düsseldorf, Fachbereich Medien, Münsterstraße 156, 40476 Düsseldorf, thomas.rakow@hs-duesseldorf.de

Änderungen an der Datenbank werden oft nicht in das einmal erstellte Diagramm des konzeptionellen Entwurfs übertragen. Dieses Problem wird durch ein sog. Round-Trip-Engineering gelöst. Dabei wird das Datenbankmodell mit der Datenbank synchron gehalten und Änderungen in der Datenbank spiegeln sich im Modell wider und umgekehrt. Darüber hinaus soll eine Kontrolle über den Abbildungsprozess des konzeptionellen Entwurfs zur Datenbank gewährleistet werden, da dieser abhängig von der Anwendung aus Varianten wählen muss (s. Vererbung). Diese Funktionalität wurde in einer Erweiterung des bestehenden UML-Tools UMLet umgesetzt [ATB03; TUT16].

Beim vorgestellten Ansatz der Datenbankmodellierung wird ein weitverbreitetes und standardisiertes Modell für den kompletten Datenbankentwurf verwendet. Dadurch entfallen Modelltransformationen, und durch das Round-Trip-Engineering kann das Diagramm leicht über den kompletten Projektverlauf aktuell gehalten werden. Auch können direkte Änderungen an der Datenbankstruktur leicht ins Diagramm übernommen werden. Durch die Datenbankunabhängigkeit kann die Wahl eines konkreten DBMS im Laufe des Projektes getroffen werden und nicht bereits zu Beginn des Projektes. Durch die Nutzung der Möglichkeiten der UML sind die Modelle näher an der Anwendungsprogrammierung und werden von mehr Entwicklern direkt verstanden. Andere Ansätze für die objekt-relationale Abbildung wie objekt-relationale Mapper oder objektorientierte Datenbanken werden in dieser Arbeit nicht betrachtet.

In Abschnitt 2 werden bereits vorhandene Ansätze erläutert und evaluiert. Die Möglichkeiten, die die UML zur Datenbankmodellierung bildet, werden in Abschnitt 3 aufgezeigt. Die Erweiterungen für ein Round-Trip-Engineering werden in Abschnitt 4 beschrieben. Im darauffolgenden Abschnitt 5 wird die Softwareerweiterung von UMLet vorgestellt. Schließlich wird in Abschnitt 6 ein Fazit gezogen.

2 Verwandte Arbeiten

Die Arbeit „An interactive tool for UML class model evolution in database applications“ [MM13] verfolgt einen ähnlichen Ansatz wie die vorliegende Arbeit. Während die vorliegende Arbeit das UML-Diagramm auf ein Datenbankschema abbildet und aus diesem Schema dann das existierende Datenbankschema aktualisiert, wird in der Arbeit von Milovanovic und Milicev [MM13] das Datenbankschema aufgrund der Unterschiede des aktuellen UML-Diagramms mit dem vorherigen UML-Diagramm aktualisiert. Diese Methode kann daher allerdings keine Änderungen berücksichtigen, die in der Datenbank selbst vorgenommen wurden. Darüber hinaus erweitert die vorliegende Arbeit den Entwicklungsprozess zu einem Round-Trip-Engineering und berücksichtigt das Modellieren von datenbankspezifischen Elementen.

In „A Methodological Approach for Object-Relational Database Design using UML“ [MVC03] wird die Erweiterung der UML für die Modellierung von objekt-relationalen Datenbanken beschrieben. Als Datenbanksystem wird hier das Oracle RDBMS gewählt. Dieser Ansatz benötigt keine aufwendige Definition von Abbildungsregeln, da objekt-relationale Datenbanken objekt-orientierte Konzepte wie Vererbung beherrschen. Der

Nachteil dieser Methode ist jedoch, dass objekt-relationale Datenbanken sehr stark proprietär und weniger verbreitet sind als relationale Datenbanken. Die vorliegende Arbeit geht bei der Abbildung von UML noch einen Schritt weiter und bildet UML auf eine relationale Datenbank ab.

Der Artikel „A UML Profile for Data Modeling“ [Am03] beschreibt eine Erweiterung von UML mithilfe eines UML-Profils, das viele Aspekte relationaler Datenbanken beinhaltet. Es werden hier jedoch keine systematischen automatisierten Abbildungen beschrieben. Dies erschwert den Aufwand bei der Datenbankmodellierung, da die Abbildung des UML-Diagramms in eine Datenbank per Hand erfolgen muss.

Ein Teilaspekt des Round-Trip-Engineering ist das Reverse Engineering. Dies wird in der Arbeit „SQL2XMI: Reverse Engineering of UML-ER Diagrams from Relational Database Schemas“ [ACD08] beschrieben. Die Arbeit beschreibt, wie sich UML-Diagramme aus Datenbanken generieren lassen. Die vorliegende Arbeit berücksichtigt hingegen das komplette Round-Trip-Engineering. Auch werden in der Arbeit die relationalen Konstrukte nicht an die Elemente des UML-Klassendiagramms angepasst, sondern eins zu eins aus der Datenbank im UML-Diagramm dargestellt. Dadurch werden gebräuchliche UML-Elemente wie Assoziationen nicht unterstützt.

3 UML für den relationalen Datenbankentwurf

Für den Entwurf von Datenbankschemata mit UML eignet sich das Klassendiagramm. Bei diesem Diagrammtyp lassen sich die meisten Elemente für den Datenbankentwurf verwenden. Es lassen sich Klassen bidirektional auf Tabellen abbilden und auch die Assoziationen entsprechen im Wesentlichen den Schlüssel-Fremdschlüsselbeziehungen relationaler Datenbanken. UML-Diagramme eigenen sich sowohl für den konzeptionellen als auch den physischen Entwurf. Zu Beginn des Entwicklungsprozesses einer Anwendung können Klassen nur mit Attributen (auch ohne Datentypen) modelliert werden. Später können die Datentypen hinzugefügt sowie Indizes und weitere Bedingungen modelliert werden. Einige Tools verfügen über eine SQL-Generierung aus UML-Klassendiagramm, die jedoch die Modellierung vieler datenbankspezifischer Funktionalitäten wie Indizes nicht ermöglicht.

3.1 Erweiterung der UML

Die UML bietet vielfältige Möglichkeiten zur Modellierung von Anwendungen. Jedoch können die Möglichkeiten der UML nicht die Anforderungen jedes Projektes abdecken. Um die UML besser an die Anforderungen von Projekten anpassen zu können, sind in der UML Erweiterungsmöglichkeiten vorgesehen. Es werden zwei Erweiterungsmöglichkeiten unterschieden. Zum einen die leichtgewichtige Erweiterung durch UML-Profile und zum anderen die schwergewichtige Erweiterung durch Veränderungen des Metamodells. Durch das Metamodell wird die UML selbst definiert. Durch Veränderungen des Metamodells sind umfassende Veränderungen der UML möglich. Um die UML zur Datenbankmodellierung

zu verwenden, wird die UML mithilfe eines Profils erweitert. Dadurch bleiben die UML-Basiskonzepte erhalten und die UML-Notationen können weiterverwendet werden. Für Anwender entsteht daher kein zusätzlicher Lernaufwand, denn der Sinn der gemeinsamen Notationssprache bleibt erhalten. Natürlich ist die Erweiterung durch ein Profil einfacher als eine Anpassung des Metamodells.

Bei der Erweiterung der UML durch ein Profil wird die UML um neue Elemente ergänzt. Dies bedeutet, es kommen nur neue Elemente hinzu, bestehende Elemente der UML können mit einem Profil nicht verändert werden. Elemente, die mit einem Profil hinzugefügt werden können, sind hauptsächlich Stereotypen. Ein Profil selbst wird mithilfe eines UML-Profildiagramms definiert.

3.2 Modellierung datenbankspezifischer Eigenschaften

Ein Aspekt bei der Entwicklung eines Schemas sind Integritätsbedingen für einzelne Attribute (CHECK-Constraints in SQL). In UML lassen sich diese Bedingungen mithilfe der OCL [Fo07] modellieren. Viele datenbankspezifische Komponenten lassen sich nicht mit Hilfe der UML modellieren. Diese Elemente definieren wir in einem UML-Profil. Die Definition des Profils erfolgt mit folgendem UML-Profildiagramm in Abb. 1.

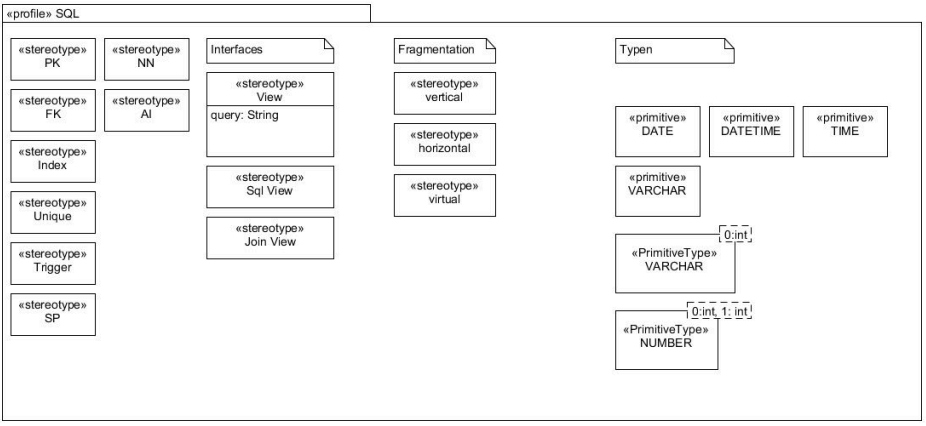

Abb. 1: UML-Profil für die Datenbankentwicklung

Im Profil sind zu meist Stereotypen für Datenbankkonstrukte, wie `NOT NULL`, `INDEX`, `PK`, `FK`, `VIEW` definiert. Darüber hinaus enthält das Profil die aus SQL bekannten Datentypen, wie z. B. `VARCHAR`. Das Profil erweitert die bisherigen Arbeiten, die UML-Profile zur Datenbank Modellierung entwickelt haben [Go03; Sp01] um Abbildungsmöglichkeiten für die Vererbung.

3.3 Vererbung

Aus dem erweitertem ER-Diagramm (EER) sind drei Strategien für die Abbildung der Vererbung bzw. ist-ein-Beziehung bekannt. Diese Strategien sind die vertikale, horizontale und virtuelle Fragmentierung [Fa07].

Eine virtuelle Fragmentierung bietet sich an, wenn die Unterklassen wenige oder kaum eigene Attribute besitzen. In diesem Fall werden alle Attribute der Unterklasse in die Oberklasse verschoben und es wird ein zusätzliches Attribut für den Typ eingeführt.

Eine horizontale Fragmentierung bietet sich an, wenn die Oberklasse abstrakt ist, siehe Abb. 2. Dieser Fall ist analog zu einer ist-ein-Beziehung mit totaler Teilnahme.

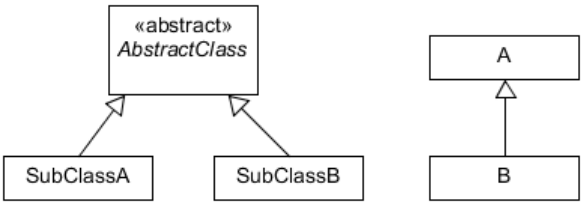

Abb. 2: Vererbung mit abstrakter Klasse

Da die Oberklasse in diesem Fall nicht abgebildet wird, müssen Beziehungen, die sich auf die Oberklasse beziehen, in allen Unterklassen berücksichtigt werden.

Eine vertikale Fragmentierung bleibt bei der Abbildung einer Vererbung sehr nah am UML-Diagramm, da alle beteiligten Klassen als Tabellen abgebildet werden. Die Tabellen der Unterklassen erhalten den Primärschlüssel der Oberklasse als Primär- und als Fremdschlüssel.

Wie im EER-Diagramm beinhaltet auch die UML einige Attribute, die die Teilnahme an der Generalisierungsbeziehung regeln. Diese Attribute sind: *complete* (total) oder *incomplete* (partiell) und *disjoint* (disjunkt) oder *overlapping* (überlappend). In Klammern sind die Entsprechungen des EER-Diagramms auf Deutsch angegeben [RQ12]. Bei einer überlappenden Teilnahme an einer Generalisierung sollte der Entwickler jedoch beachten, dass diese in Java (oder anderen OO-Sprachen) mit den Sprachkonstrukten selbst nicht umsetzbar sind. Welche Art der Vererbung bzw. Fragmentierung verwendet werden soll, kann der Modellierer per Stereotyp angeben («vertical», «horizontal» oder «virtual»), default-mäßig wird vertikal fragmentiert (s. oben).

3.4 Modellierung eines Indexes

Indizes sind für die Leistungsfähigkeit von Datenbanksystemen sehr wichtig. In einer Tabelle können mehrere Indizes definiert werden. Daher reicht es nicht aus, einen «index»-Stereotyp zu definieren, bei dem alle Attribute, die mit dem Stereotyp ausgezeichnet sind, zu einem

Index zusammengefasst werden. Aus diesem Grund wird der Stereotyp «index» für Klassen definiert. Die Semantik dieses Stereotyps bedeutet, dass alle Attribute der Klasse, die mit dem Stereotyp «index» ausgezeichnet sind, einen Index bilden. Der Index wird für die Klasse erstellt auf die die «index»-Klasse mit einer Assoziation verweist. Abb. 3 verdeutlicht diese Semantik man Beispiel eines Indizes für das TPC-H Schema:

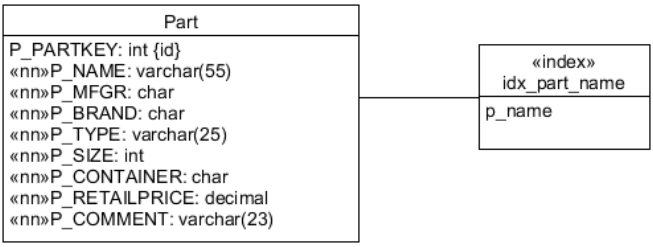

Abb. 3: Indexdefinition in UML am Beispiel TPC-H

4 Round-Trip-Engineering

Wichtig für einen effizienten Entwicklungsprozess ist das Automatisieren von Aufgaben. Aus diesem Grund wurde Umlet mit einem Round-Trip-Engineering ausgestattet und zu Umlet+RTE. Das ermöglicht es, aus dem für das Relationenschema verwendete UML-Klassendiagramm die Datenbank automatisch zu erstellen bzw. eine ältere Version der Datenbank zu aktualisieren. Auch lässt sich so aus einer Datenbank ein UML-Diagramm erstellen. Bei der Aktualisierung einer Datenbank wird die Datenbank aktualisiert und zusätzlich wird ein SQL-Script erstellt, das die Datenbank ebenfalls aktualisieren kann. Dadurch können Produktupdates einfach realisiert werden. Die Generierung der Skripte findet unter Berücksichtigung des vorhandenen Schemas statt. Durch die automatisierte Abbildung von z. B. Vererbungsstrukturen lassen sich die Möglichkeiten der UML weiterhin zur Modellierung verwenden.

4.1 Software Architektur

Im Gegensatz zu einem Round-Trip-Engineering zwischen dem UML-Klassendiagramm und beispielsweise dem Java-Quellcode gibt es beim Round-Trip-Engineering zu einer Datenbank einige Besonderheiten. Zum einen müssen verschiedene DBMS mit unterschiedlichen SQL-Dialekten berücksichtigt werden. Zum anderen gibt es drei Komponenten beim Round-Trip-Engineering: das Diagramm, die Datenbank und das SQL-Schema. Das folgende UML-Verteilungsdiagramm in Abb. 4 skizziert das Round-Trip-Engineering für die Datenbankmodellierung.

Die Datenbank wird im Diagramm durch den Knoten DatabaseLayer repräsentiert. Der DatabaseLayer enthält die Komponente DatabaseConnection. Diese Komponente benutzt

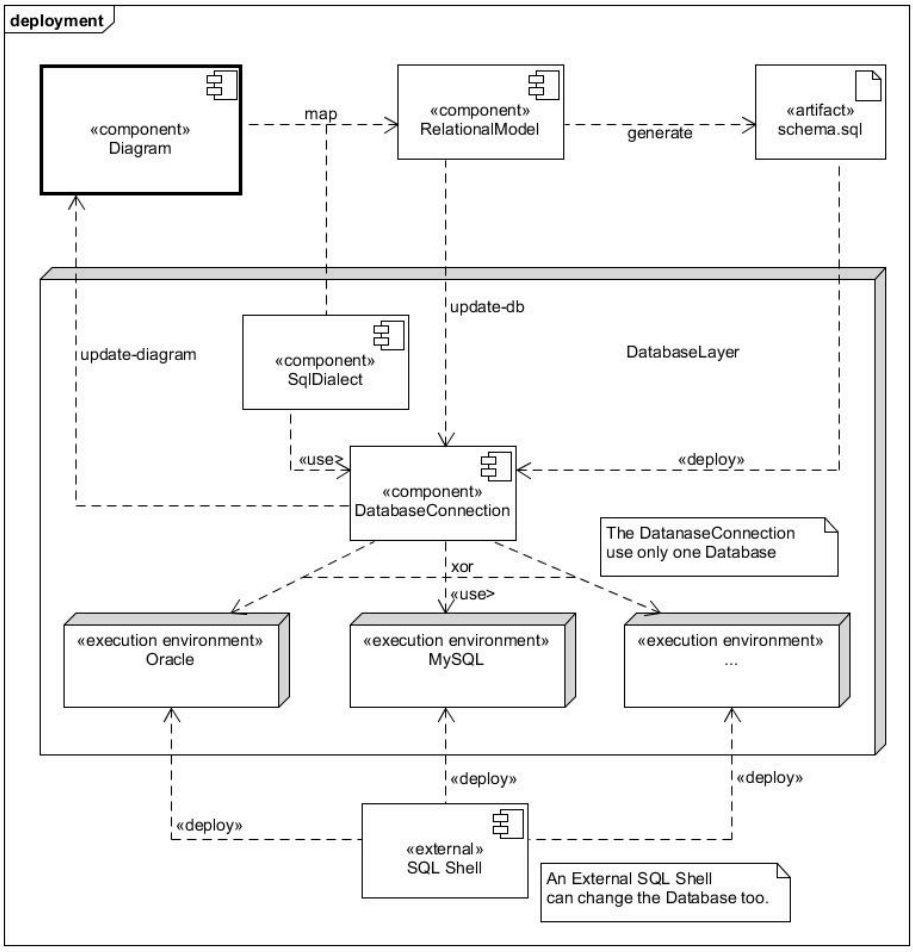

Abb. 4: Skizze Round-Trip-Engineering

genau eine Datenbank («use»). Durch diese Konstruktion wird die Unabhängigkeit von einem einzelnen DBMS umgesetzt.

Die Komponente Diagram repräsentiert das UML-Klassendiagramm zur Datenbankmodellierung. Diese Komponente bildet das Modell auf ein Relationendiagramm (RelationalModel) ab. Aus dem Relationendiagramm wird mithilfe des SqlDialect des verwendeten DBMS das SQL-Schema (schema.sql) generiert. Das SQL-Schema kann eine komplette Datenbank erzeugen. Mit der Datei schema.sql kann z. B. das SQL-Schema auf eine Produktionsumgebung übertragen werden. Durch die automatische Generierung des SQL-Schemas steht immer eine aktuelle und ausführbare SQL-Datei mit dem Schema zur Verfügung.

In dem Modell sind auch externe Änderungen der Datenbank vorgesehen. Diese Möglichkeit wird durch die Komponente SQL Shell symbolisiert. Die externen Änderungen an der Datenbank werden auch im Diagramm übernommen. Wichtig ist, dass die Komponente DatabaseConnection nur eine Datenbank verwenden kann («use»), da es nicht möglich ist, dass die Datenbank die Anwendung bei Änderungen am Schema informiert.

Das Schema (schema.sql) kann über die DatabaseConnection auf dem Datenbankserver ausgeführt werden («deploy»). Änderungen an der Datenbank werden von der Komponente DatabaseConnection erkannt und ins Modell bzw. Diagramm übertragen (update-diagram). Bei dem Update des Diagramms werden die relationalen Strukturen der Datenbank auf die Elemente des UML-Diagramms abgebildet. Umgekehrt werden Änderungen am Diagramm in der Datenbank umgesetzt (update-db). Bei diesem Schritt wird das UML-Diagramm zuerst in ein Relationenmodell umgewandelt. Nach diesem Zwischenschritt wird die Datenbank über die Database-Connection aktualisiert.

Ein wesentlicher Aspekt bei dem beschriebenen Round-Trip-Engineering sind die Daten. Neben der Notation für Klassen bietet die UML auch eine Notation für Objekte. Dadurch können mit der UML auch Datensätze bzw. Instanzen modelliert werden. Es ist sinnvoll einen initialen Datenbestand für die Produktivumgebung zu modellieren (zu diesem initialen Datenbestand kann z. B. ein erster Benutzer gehören). Der Schritt, alle Datensätze aus der Entwicklungsdatenbank ins Diagramm zu übernehmen, kann zu Problemen führen. Problematisch kann in diesem Zusammenhang sein, dass das Diagramm nach einem Testen der Anwendung mit Datensätzen überfüllt ist. Für den Umgang mit Datensätzen sollten Optionen geschaffen werden.

Das angedachte Round-Trip-Engineering weist einige Besonderheiten auf. Es findet nicht zwischen einer SQL-Datei und dem Klassendiagramm statt, sondern zwischen dem Klassendiagramm und der Datenbank. Bei der Abbildung vom Diagramm zur Datenbank im Round-Trip-Engineering wird mittels einer Modell-Transformation ein Relationenmodell erstellt, das schließlich mit der Datenbank abgeglichen wird.

5 Umsetzung der Software

Das Werkzeug zur Datenbankmodellierung wurde als Erweiterung +RTE der unter der GNU-GP-Lizenz verfügbaren Software UMLet umgesetzt [ATB03; TUT16]. Das Tool wurde wegen seiner einfachen Bedienung, seiner Verfügbarkeit im Quellcode und seiner Erweiterbarkeit ausgewählt [Sa16]. Der größte Vorteil bei dieser Vorgehensweise ist, dass viele Funktionen in der Software schon vorhanden sind. Damit ist für den eigentlichen Diagramm-Editor keine Arbeit mehr nötig und es bleibt mehr Zeit für die Implementierung des Round-Trip-Engineering übrig. Sichtbar wurde UMLet durch +RTE kaum verändert. Es gibt im Wesentlichen nur zwei zusätzliche Button: einen Button zur Aktualisierung des Diagramms aus Änderungen in der Datenbank und einen weiteren Button, um die Änderungen des Diagramms in die Datenbank zu übertragen. Bei einer Änderung der Datenbank wird zusätzlich eine SQL-Datei mit dem kompletten Schema angelegt. Der Nutzer kann wählen, ob er die Datenbank direkt aktualisieren will oder nur das SQL-Skript mit den UPDATE- und ALTER-Anweisungen erhalten will. Darüber hinaus gibt es eine zusätzliche Palette mit vorgefertigten Elementen für den Datenbankentwurf (z. B. Klasse mit Primärschlüssel, Index, etc.).

Bei der Evaluierung stellt sich die Frage: „Helfen UML-Diagramme in der Datenbankentwicklung bzw. Modellierung?“ [BQ13]. Das Tool wurde bislang einmal im Praktikum für Datenbanksysteme 2 im Studiengang Medieninformatik eingesetzt. Trotz Problemen mit der Stabilität des verwendeten Tools UMLet+RTE konnte ein Verständnis für die Modellierung in UML entwickelt und die Notwendigkeit der Konsistenz zwischen Modell und Datenbank vermittelt werden.

6 Fazit und Ausblick

Diese Arbeit beschreibt die Modellierung von Datenbankschemata mittels des UML-Klassendiagramms und der standard-konformen Erweiterung durch ein UML-Profile. Der Einsatzbereich der UML wird auf die Datenbankmodellierung ausgedehnt und durchgehend für den konzeptionellen sowie den physischen Datenbankentwurf genutzt. Änderungen erfolgen durch das Round-Trip-Engineering konsistent in Modell und Datenbanksystem.

Bei dem Thema dieser Arbeit gibt es viele Anknüpfungspunkte für künftige Entwicklungen. Als Erstes wäre eine Erweiterung des Round-Trip-Engineerings auf eine objekt-orientierte Sprache wie Java sinnvoll. Dadurch bekäme das Round-Trip-Engineering neben dem UML-Diagramm und der Datenbank einen weiteren Teilnehmer. Darüber hinaus sollte auch der XMI-Standard in der Anwendung Verwendung finden. Durch die Unterstützung des XMI-Standards könnten die UML-Diagramme auch in anderen UML-Tools genutzt werden. Auch wird die Anwendungsentwicklung weiter unterstützt, indem Formulare automatisch aus dem Modell erstellt werden können. Hierfür müssen die Eigenschaften der Klassen mit entsprechenden Annotationen versehen werden und Templates für die Formulare erstellt werden. Schließlich können auch NoSQL-Datenbanken, berücksichtigt werden. So ließe sich das UML-Klassendiagramm auch auf eine dokumentbasierte Datenbank wie z. B. MongoDB abbilden. Auch ließe sich neben dem SQL-Schema ebenfalls ein XML-Schema generieren. Dies würde den Datenaustausch mit anderen Anwendungen vereinfachen.

Literatur

[ACD08] Alalfi, M. H.; Cordy, J. R.; Dean, T. R.: SQL2XMI: Reverse Engineering of UML-ER Diagrams from Relational Database Schemas. In: 15th Working Conference on Reverse Engineering. Institute of Electrical und Electronics Engineers (IEEE), S. 187–191, Okt. 2008.

[Am03] Ambler, S. W.: A UML Profile for Data Modeling, 2003, URL: http://www.agiledata.org/essays/umlDataModelingProfile.html, Stand: 20. 10. 2016.

[ATB03] Auer, M.; Tschurtschenthaler, T.; Biffl, S.: A flyweight UML modelling tool for software development in heterogeneous environments. In: Proceedings 29th Euromicro Conference. Institute of Electrical und Electronics Engineers (IEEE), 2003.

[BQ13] Byrne, B.; Qureshi, S.: UML Class Diagram or Entity Relationship Diagram: An Object Relational Impedance Mismatch. In: Proceedings of 6th Int. Conf. of Education, Research, and Innovation. 2013.

[Fa07] Faeskorn-Woyke, H.; Bertelsmeier, B.; Riemer, P.; Bauer, E.: Datenbanksysteme. Pearson Studium, 2007.

[Fo07] Forbrig, P.: Objektorientierte Softwareentwicklung mit UML. Hanser Fachbuchverlag, 2007.

[Go03] Gornik, D.: UML data modeling profile, IBM Rational Software Whitepaper TP 162 05/02, 2003.

[MM13] Milovanovic, V.; Milicev, D.: An interactive tool for UML class model evolution in database applications. Software & Systems Modeling 14/3, S. 1273–1295, Sep. 2013.

[MVC03] Marcos, E.; Vela, B.; Cavero, J. M.: A Methodological Approach for Object-Relational Database Design using UML. Software and Systems Modeling 2/1, S. 59–72, März 2003.

[OS13] Oestereich, B.; Scheithauer, A.: Analyse und Design mit der UML 2.5. Gruyter, de Oldenbourg, 2013.

[RQ12] Rupp, C.; Queins, S.: UML 2 glasklar. Hanser Fachbuchverlag, 2012.

[Sa16] Salgert, B.: Modellierung von Datenbanken mit UML im Round-Trip-Engineering, Masterarbeit, Hochschule Düsseldorf, 2016.

[Sp01] Sparks, G.: Database modelling in UML, Sparx Systems whitepaper, 2001.

[TUT16] The UMLet Team: UMLet 14.2 – Free UML Tool for Fast UML Diagrams, 2016, URL: http://www.umlet.com, Stand: 20. 10. 2016.