Tool Support for Model-Based Database Design with Event-B

Ahmed Al-Brashdi, Michael Butler, Abdolbaghi Rezazadeh, and Colin Snook

University of Southampton, Southampton, UK
{azab1g14,mjb,ra3,cfs}@ecs.soton.ac.uk

Abstract. UML-B provides a graphical notation for Event-B that enables formal development in a UML style. UB2DB is a tool that translates UML-B models to relational database implementations in SQL. The UB2DB tool is implemented as a plugin for Rodin, an extensible toolkit for Event-B. This paper presents the current version of UB2DB that translates the main components of UML-B class diagrams to SQL code. The generated SQL code defines a database and provides procedures that manipulate it. The UB2DB tool exploits the Eclipse Modeling Framework (EMF) to realise the required model transformation. The current tool provides the basis for a more comprehensive tool that will provide support for a broader range of UML-B features and support a variety of database components and constraints.

Keywords: Event-B, UML-B, Database design, Model-driven design

1 Introduction

The design of database systems is an important field in software engineering, and therefore it requires a verifiable and rigorous design approach. Event-B is a formal method for rigorous specification and verification of digital systems [1]. It is supported by an open platform called Rodin [2]. UML-B is a graphical notation for formal modeling in Event-B that is based on UML [10]. A tool, called iUML-B, is provided which supports building UML-B diagrams in Rodin and is integrated in an Event-B machine or context. The iUML-B tool is based on the Eclipse Modelling Framework (EMF) for Event-B [11]. The Eclipse Modeling Framework (EMF) is a framework for tools development which provides modeling and code generation facilities [12].

The UB2DB (UML-B to DataBase) tool enables developers to rigorously model their database or database intensive application in UML-B for verification and translate the model to SQL code. SQL is a relational database definition and manipulation language [7]. The UB2DB provides an automatic generation of SQL code from a model defined in UML-B in the Rodin platform. The generated SQL will create the relational database structure using *create* and *alter* commands and provide procedures that populate and manipulate the data.

Developers will model a database system in UML-B and verify it in the Rodin platform to detect any inconsistency or ambiguity. Then they will use UB2DB to translate the verified model to SQL code.

When modelling databases in UML-B, different refinement levels might be introduced so that the model of database can be built gradually. In the first abstraction level, an abstract view of the database is modelled and can be translated to SQL. In every refinement level, a more concrete view of the database is modelled which adds more details to the preceding one.

UB2DB translates directly from UML-B and not from Event-B. Since UML-B uses class diagrams, it is more aligned with the database as class diagrams are commonly used to describe databases. Developers will benefit from being able to generate Event-B from UML-B using the existing UML-B tool, allowing them to apply formal analysis to their UML-B models. While the UML-B class diagram clearly adds attributes to each class directly, Event-B list all variables together without clear distinction of each class's attribute. This makes it easier and more straightforward to translate from UML-B to database than from Event-B to database, and without compromising the model verification.

2 Translation process and approach

To translate from UML-B to a relational database, we designed our own meta-model for relational databases in EMF. Figure 1 shows a part of the defined meta-model. Each database is composed of tables, and each table may have many attributes. Each element has a name associated with it. Along with the name, attributes have types and constraints that specify if an attribute is *not null*, *unique* or has a *default* value.

The first step in UB2DB is to translate the EMF representation of UML-B to the EMF representation of the database such as translating a class in UML-B to a table, or a class diagram to a database. A second step is to generate the textual SQL code from the database EMF representation. For each UML-B model, there are two translations done, one to SQL by UB2DB, and another to Event-B by UML-B as in Figure 2. These two translations are separate from each other.

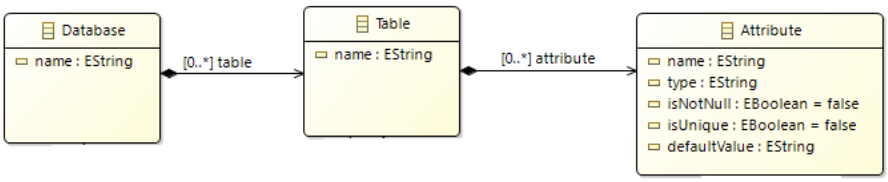

Fig. 1. Part of defined database meta-model

Each element in UML-B such as class diagram or class has a unique name. Classes may have associations between them that relate each class to another. UB2DB generates the SQL statements that create a database whose name is given by the class diagram name, and generates a table for each class in the

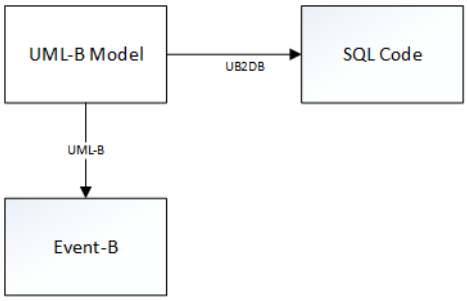

Fig. 2. Translation from UML-B model to Event-B and database

model. The associations between classes are translated to relations between tables. Each class attribute in the UML-B model will result in an attribute in the corresponding table. Each component such as association or attribute is translated into a separate statement in the generated SQL. This way, dealing with refinement will be easier as adding a new attribute for a class in a refined model will correspond to adding only one SQL statement that adds that attribute to the table instead of going through the whole process of creating a table again.

For each attribute in the model, there are some defined properties like total function, injective function and initial value. These properties are translated into constraints in the generated SQL. A class invariant might be added to constrain an attribute value such as $a \in 1..100$. Such an invariant is translated to a constraint in the generated SQL.

If an association between two classes is set as functional, it will be translated to an attribute of one of the tables. If the association is non functional (n:n association), it will be translated to a separate table with references to both classes/tables. An example of that is the association *member_pod* between *Member* class and *Pod* class in Figure 3. The UB2DB tool will generate a new table called *member_pod* with two references, one to *Member* table and another to *Pod* table. The following SQL statement is generated automatically by UB2DB which creates a table with the association name, *member_pod*. The table has two attributes: *member_id* and *pod_id* that reference *Member* and *Pod* tables.

```
CREATE TABLE member_pod(
   Member_id INT,
   Pod_id INT,
  PRIMARY KEY ( Member_id , Pod_id ),
  FOREIGN KEY (Member_id) REFERENCES Member(Member_id),
  FOREIGN KEY (Pod_id) REFERENCES Pod(Pod_id)
);
```

UML-B provides three kinds of events; *constructor*, *destructor* and *normal*. A constructor event should be selected for events that aim to create an instance of a class. Destructor is used for the opposite. For other operations, the normal

event is selected where it adds a guard automatically to check that the instance to select or update is an element of that class set. Each constructor event in UML-B is translated by UB2DB into a *procedure* with the *insert into table* statement in SQL. The procedure takes all class attributes and associations as parameters for the insertion. Destructor events are translated into procedure with the *delete from table* statement in SQL. Normal events are translated into procedures with an *update table* statement if the event has an override operator, or to a *select from* statement if the event does not have an action.

UB2DB generates SQL code in which class invariants are translated to constraints in the generated code. Event guards are also maintained by the translation to ensure correct implementations of the UML-B models. Also, the generated code ensures the atomicity of an event by translating all actions to one atomic transaction.

The UB2DB translation is implemented using a generic EMF translation plugin which is provided by University of Southampton. Translation and rules are contributed using the Eclipse extension mechanism. For each component in the database meta-model such as table or attribute, there is a rule defined by a Java class to translate or map UML-B to it. Each rule in UB2DB has *fire* and *dependencyOk* methods. The *fire* method does the mapping between UML-B elements to database elements. In the translation process, dependencies must be checked by the *dependencyOk* method before proceeding to the translation. A table is dependant on a database which means it cannot be generated before the database, and an attribute is dependant on a table. This also ensures an ordering of the translation of different components.

3 Case study and evaluation

Two cases were built to study various components and relations of database systems and to help identify good practice in modelling databases in Event-B with levels of refinements. The first case study is a student enrollment and registration system while the other is a car sharing system. After having these two cases modelled in UML-B, we ran UB2DB on different abstraction levels to generate the database for them.

Starting from an abstract model, as in Figure 3 for the car sharing case study, where classes have associations but no attributes, the tool generates the tables with one attribute as a key for each table. Then the associations are added to the source tables as attributes that references the target table. As the same classes will appear in another refinement level, the *create table if not exists* command will create a table only if it does not already exist.

Further refinement of the model might include adding attributes to different classes as in Figure 4 for the student enrollment case study. Another refinement could add more detail to the model by introducing new classes and associate them to classes in the abstract model such as the *Booking* class for car sharing in Figure 5. Attributes and relations added in later refinements are translated to the *alter table* command so that we can build on the previous generated database

without rebuilding it. The *alter* command can be used to modify the structure of a table by adding, modifying or deleting attributes or relations.

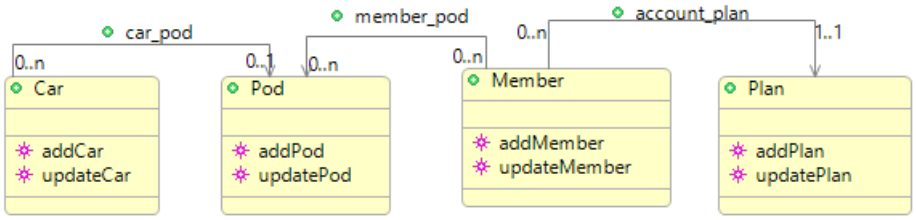

Fig. 3. Abstract model for car sharing case study

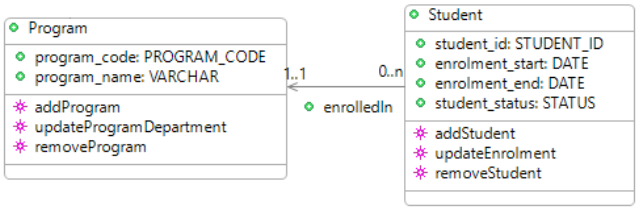

Fig. 4. Refinement by adding attributes in Program and Student classes

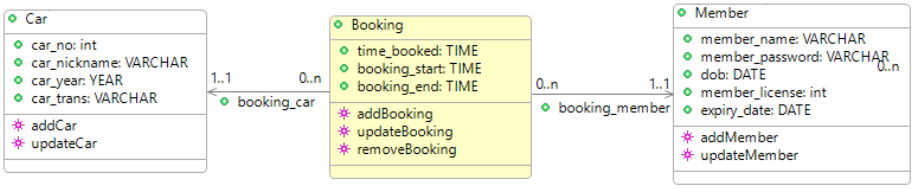

Fig. 5. New classes in refinement model

The generated SQL code was successfully imported in the database management system and all the supported database structures and constraints were successfully generated. This includes generating intermediate tables and assigning different constraints such as primary key, foreign key, not null, uniqueness, default value and basic check constraints. Events were translated to procedures for constructors, destructor or normal events. For any constructor event, the tool generated a procedure with a name as the event name and took the class

attributes and associations as parameters for the procedure. Destructor events were translated to procedures with one parameter corresponding to a key for the record to be deleted. Normal events were translated to either update or select procedures. However, UB2DB does not yet deal with complex queries in normal events in UML-B.

4 Related Work

Much work has been done in the area of formalizing databases or translating formal methods to database applications. Barros in [4] translates Z notation to relational databases with support for different operations and transactions. In [8], Khalafinejad and Mirian-Hosseinabadi present a method for translating Z notation to SQL and the Delphi programming language with no tool implementation.

Laleau and Mammar in [9] present a tool that refines a UML specification into a B model and then to a database application. While their work is close to ours, they do not translate to update and read operations or deal with transaction management. Our work will provide extra features than theirs such as normalizations, design patterns, database security and translation to database views.

Davies et al. in [6] use a model-driven approach to automatically generate object-oriented databases with an extended version of B method and Object Constraint Language [14]. We are interested mainly in the relational model of database design.

Wang and Wahls in [13] developed a Rodin plug-in that translates Event-B to Java and JDBC code to create and query a database. While, to the best of our knowledge, this is the only work that translates Event-B to database applications, it has some limitations. The results in [5] identify major performance issues as well as the issues with preserving database integrity as in [3].

5 Conclusion and future work

UB2DB is a tool that translate UML-B models to relational databases by generating SQL statements that build the database and structure its tables and relations. The UML-B model is translated by the UML-B tool to Event-B for verification. UB2DB provides support to translate different components in UML-B model into code that can be easily imported in MySQL database and reserves the constraints such as *not null* and *unique*. It also provides support for events that create new instances of classes, delete an existing one, update its attributes or select from one or more classes.

In future, full support for events will be provided which might translate one event in UML-B to different statements in one procedures such as delete and insert when moving a record from one class to another. As the current implementation of the tool translate an abstract level without looking to preceding

abstraction level, the future plan is to make the tool look for the the specification of a model and all of its preceding abstractions. The tool will extend the support for class invariants. The future work includes looking at preserving normalization when modelling in UML-B and defining design patterns for database systems and supporting them by our tool.

References

1. Abrial, J.R.: Modeling in Event-B: system and software engineering. Cambridge University Press (2010)
2. Abrial, J.R., Butler, M., Hallerstede, S., Hoang, T.S., Mehta, F., Voisin, L.: Rodin: an open toolset for modelling and reasoning in Event-B. International journal on software tools for technology transfer 12(6), 447–466 (2010)
3. Al-Brashdi, A.: Translating Event-B to Database Application. Master's thesis, University of Southampton (2015)
4. Barros, R.S.M.: On the formal specification and derivation of relational database applications. Electronic Notes in Theoretical Computer Science 14, 3–29 (1998)
5. Catano, N., Wahls, T.: A case study on code generation of an ERP system from Event-B. In: Software Quality, Reliability and Security (QRS), 2015 IEEE International Conference on. pp. 183–188. IEEE (2015)
6. Davies, J., Welch, J., Cavarra, A., Crichton, E.: On the generation of object databases using Booster. In: Engineering of Complex Computer Systems, 2006. ICECCS 2006. 11th IEEE International Conference on. pp. 10–pp. IEEE (2006)
7. Garcia-Molina, H.: Database systems: the complete book. Pearson Education India (2008)
8. Khalafinejad, S., Mirian-Hosseinabadi, S.H.: Translation of Z specifications to executable code: Application to the database domain. Information and Software Technology 55(6), 1017–1044 (2013)
9. Mammar, A., Laleau, R.: UB2SQL: a tool for building database applications using UML and B formal method. Journal of Database Management 17(4), 70 (2006)
10. Snook, C., Butler, M.: UML-B and Event-B: An integration of languages and tools. In: Proceedings of the IASTED International Conference on Software Engineering. pp. 336–341. SE '08, ACTA Press, Anaheim, CA, USA (2008)
11. Snook, C., Fritz, F., Illisaov, A.: An EMF framework for Event-B. In: Workshop on Tool Building in Formal Methods - ABZ Conference, Orford, Canada (2010)
12. Steinberg, D., Budinsky, F., Merks, E., Paternostro, M.: EMF: eclipse modeling framework. Pearson Education (2008)
13. Wang, Q., Wahls, T.: Translating Event-B machines to database applications. In: Software Engineering and Formal Methods, pp. 265–270. Springer (2014)
14. Warmer, J., Kleppe, A.: The Object Constraint Language: Precise Modeling with UML. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (1999)