Model-Based Database Design

Ahmed Al-Brashdi

Electronics and Computer Science
University of Southampton, UK
azab1g14@soton.ac.uk

Abstract. Many databases contain critical resources and data upon which important decisions rely and therefore their design requires a rigorous, verifiable approach. Formal methods are mathematical and verifiable tools to specify and prove software specifications. While formal methods have been applied to database systems, there is a lack of a comprehensive, tool-supported methodology for formal development of practical database systems. This research tries to bridge the gap between the two areas by introducing an approach to rigorously design databases using the Event-B formal method. It follows a case study methodology and models this case gradually in a UML-like notation called UML-B which is translated to Event-B and verified in the Rodin platform. This PhD work presents general guidelines for modelling database applications in Event-B and proving their consistency, integrity and security considering different types of classes and relations. It provides translation method and tool to transform proved graphical models to efficient database systems.

Keywords: Formal methods, UML-B, Database design, Code generation

1 Introduction

The design of database systems is one of the most important disciplines of software engineering development as it carries critical resources and data upon which important decisions rely. While designing a small database application might be simple and straightforward, current enterprises depend on very large and complex database systems. For every increase in complexity, the possibility of inconsistency and errors increases as well. One way of addressing this issue is to construct database applications using a rigorous design methodology. Using *formal methods*, a developer can specify complex systems and use formal verification to detect ambiguities and inconsistencies.

Unverified database design might lead to critical consequences. Inconsistent or corrupt data in patient databases might result in a patient getting a wrong prescription which could harm his/her life. If an enterprise derives its decisions from a non-accurate database, it may result in a loss of money. This highlights the question of how to guarantee database consistency and integrity in the early stages of development.

To address these issues, we propose the development of a mathematically provable approach for designing and creating databases. The research aims to develop a method and a tool to help developers rigorously design the structure of their database application and create the methods for manipulating and maintaining its data. It defines guidelines for modelling databases using a UML-like graphical notation called UML-B [16] in the Rodin platform [2] and use it to prove the consistency of the model specifications in Event-B [1]. A tool, called iUML-B, is used which supports building diagrams in UML-B and is integrated in an Event-B machine or context. A verified database implementation code will be generated with respect to consistency, integrity and security.

2 Related work

Barros in [6] translates Z specifications to relational databases. The work covers different CRUD operations as well as transactions, sorting aggregations and other database components. The work translates the textual specification in Z and there is no graphical model of the representation in which the process should start with.

In [13], Khalafinejad and Mirian-Hosseinabadi present a method that translates Z notation to SQL and Delphi programming language by extending the Delphi library to support Z structures. While the work provides a formal basis to build a database prototype, it does not implement a tool for the translation. Only the formal definition of translation function is presented in the paper without the implementation of these functions.

Schlatte and Aichernig in [15] present a method to create a VDM formal model for relational databases. The method transfers an entity-relation diagram to VDM-SL constructs along with additional constraints. The method includes formalizing database structure, SQL data types, database operations and runtime checking using triggers.

Mammar and Laleau in [14] present a tool that refines a UML specification into a B model and then to a database application. The work helps a modeller to design a UML diagram and then translate that model to a B specification and on to Java and SQL code. Only the first normal form is guaranteed in Laleau and Mammar's work and other normal forms are up to the UML designer to satisfy.

Davies et al. in [11] explore how to formalize object database specifications and constraints using Booster notation [10]. They use a model-driven approach to automatically generate object-oriented databases. An extended version of B method and Object Constraint Language (OCL) [20] are used to implement the object database in which methods are implemented as functions in C language.

Wang and Wahls in [19] 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 an Event-B to database applications, it has some limitations. The results shown in [9] identify major performance issues as well as the issues with preserving database integrity as shown in [3].

In [4], Bahmani et al. present an automated method for relational database normalization that generates 2NF, 3NF and BCNF. The methods used to automate the normalization process depends on scanning an existing data in a table and determining the dependencies between its attributes. However, the focus of our research is to guarantee the normalization when designing the database structure and before populating any data on the databases.

Bartino and Sandhu in [7] discuss *Discretionary Access Control (DAC)* for relational databases which includes two access control models; *content-based* and *role-based*. The *Mandatory Access Control (MAC)* regulate the access between subjects and objects. They also discuss the access control for object and XML databases. In [12], Dollinger highlighted the two models, DAC and MAC and compares the second against Clark and Wilson Model. Both papers emphasis the importance of database security in research and practice.

3 Expected Contributions

At the end of this PhD research, we expect to deliver a comprehensive method and a supporting tool to translate UML-B models to database systems. The emphasis of this work will be on relational database systems but we plan to extend in future to support other models of database systems such as NoSQL. The current work provides a method and a tool to model and translate different components of relational databases a long with their constraints. It will translate the constraints that are supported by Structured Query Language (SQL) such as unique, default or not null values.

Contribution 1: To define general guidelines on how to model database systems in UML-B and prove them in Rodin platform. A set of case studies are being modelled to deduce the guidelines which could be applicable to a wide class of database systems. These guidelines will help the modeller to specify different constraints and invariants that satisfy the system requirements.

Contribution 2: To define translation rules from UML-B and Event-B models to database systems. These rules will be used by a tool that generates the database implementation code from the model.

Contribution 3: To incorporate access control and privacy properties in our database design method based on a security-sensitive case study like Internet banking. Our approach will follow access roles and policies strategies to ensure an authorized access or modification of the data.

Contribution 4: To automate the database normalization process by letting the modeller specifies the functional dependencies between attributes. The research aims to study the possibility of automating the normalization of a model to BCNF using these functional dependencies.

Contribution 5: To provide a library of different data types and operation on them using the Theory feature [8] in Roding platform. By using these data types like *String*, *Date*, *Binary object* or *Time*, a modeller can specify events that operate on these data types such as string substitution.

Contribution 6: To build a tool that supports the above contributions and translate the UML-B model to database system. It will generate the code that creates the database structure and manipulates its contents.

Contribution 7: To extend the method and the tool to support distributed database transactions with multiple simultaneous users. The research should make use of formal methods patterns represented in the literature to model distributed transactions. These patterns, like *timeout*, *cancel* and *failure* patterns in [5], might address the issue of distributed transactions of multiple users in databases.

4 Work so far

Sine October 2015 where the PhD started, we have achieved the following:

Contribution 1: We identified general guidelines on how to model database system in UML-B with different classes and associations. The guidelines were concluded by modelling a student enrollment and registration system followed by a second case study of car sharing system. This enabled us to identify the common features in both systems and derive the guidelines. The guidelines will be extended to suggest design patterns for relational database modelling. The design patterns should be concluded by examining and modelling different case studies in UML-B. It also includes how to specify different operations on database systems when modelling them in UML-B and Event-B.

Contribution 2: So far, we have defined twenty rules to map UML-B models to database implementations and constraints. The rules specify how to translate or map UML-B components to relational database components. It also defines how to translate Event-B invariants such as total or injective functions to SQL constraints. The rules will be extended while developing and evaluating the tool in Contribution 6.

Contribution 3: We started to work on modelling database security on an Internet banking system by introducing access rules and policies to database objects. These rules defines who can access the data and what operations can they perform on it.

Contribution 6: We developed a working prototype of the tool that generates SQL code from the model defined in UML-B. Part of the tool development was defining a meta-model of relational database and use it for model to model transformation. The relational database meta-model is defined using Eclipse Modeling Framework (EMF) [18]. Figure 1 shows part of our database meta-model used for the tool where database is composed of tables, table may have references to other tables and table may have different attributes. The tool takes the EMF model of UML-B [17] and maps it to our relational database model. The tool then generates the SQL code from the database EMF model. While the tool development is a continuous process that incorporates other contributions throughout this PhD work, the initial version of the tool provides the following:

- Generates the database and its tables with their associations from any UML-B class diagram model as SQL statements.
- Translates each class attribute in UML-B diagram to an attribute to the corresponding table.
- Translates functional associations between classes to referential integrity between two tables. If the association, C, is not a function but a relation, say $A \leftrightarrow B$, the modeller is encouraged to model it as a separate class, C, and provide associations from it to both classes, A and B. However, the tool may provide an automated refinement of relational associations to associate classes in future development.
- Translates typing invariants of total functions for associations and attributes in UML-B models to *not null* constraints, injective functions to *unique* constraints and initial values to *default* constraints in SQL.
- Translates some events in UML-B to corresponding database operations such as *insert*, *select*, *update* or *delete*. Each event is translated to a stored procedure in SQL. Constructor events in UML-B, for example, are translated to procedures that takes all class attributes and associations as parameters and execute an *insert into* statement with these parameters values. Procedures can be used to check for some guards in the run-time to ensure user supplies correct data.

Contribution 4,5 and 7: To be done.

In summary, this PhD work will contribute a rigorous method and tool for modelling relational database applications and automatically generate the code for the database structure and manipulation. With extensive evaluation and experiments, database integrity, consistency, security and privacy will be addressed in the research. A further step is to incorporate distributed databases into this research and prove their dependability.

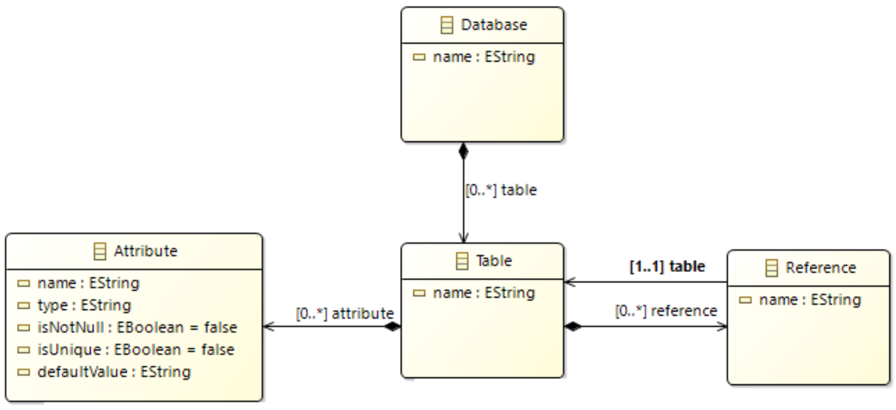

Fig. 1. Part of defined database meta-model

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. Bahmani, A.H., Naghibzadeh, M., Bahmani, B.: Automatic database normalization and primary key generation. In: Electrical and Computer Engineering, 2008. CCECE 2008. Canadian Conference on. pp. 000011–000016. IEEE (2008)
5. Ball, E., Butler, M.: Event-B patterns for specifying fault-tolerance in multi-agent interaction. In: Methods, models and tools for fault tolerance, pp. 104–129. Springer (2009)
6. Barros, R.S.M.: On the formal specification and derivation of relational database applications. Electronic Notes in Theoretical Computer Science 14, 3–29 (1998)
7. Bertino, E., Sandhu, R.: Database security-concepts, approaches, and challenges. Dependable and Secure Computing, IEEE Transactions on 2(1), 2–19 (2005)
8. Butler, M., Maamria, I.: Practical theory extension in Event-B. In: Theories of Programming and Formal Methods, pp. 67–81. Springer (2013)
9. 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)
10. Davies, J., Crichton, C., Crichton, E., Neilson, D., Sørensen, I.H.: Formality, evolution, and model-driven software engineering. Electronic Notes in Theoretical Computer Science 130, 39–55 (2005)
11. 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)
12. Dollinger, R.: Database security models-a case study. In: 8th international conference on intelligant engineering systems. pp. 313–318. IEEE (2004)
13. 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)
14. Mammar, A., Laleau, R.: From a B formal specification to an executable code: application to the relational database domain. Information and Software Technology 48(4), 253–279 (2006)
15. Schlatte, R., Aichernig, B.K.: Database development of a work-flow planning and tracking system using VDM-SL. In: Workshop Materials: VDM in Practice (1999)
16. 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)
17. 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)
18. Steinberg, D., Budinsky, F., Merks, E., Paternostro, M.: EMF: Eclipse Modeling Framework. Pearson Education (2008)
19. Wang, Q., Wahls, T.: Translating Event-B machines to database applications. In: Software Engineering and Formal Methods, pp. 265–270. Springer (2014)
20. Warmer, J., Kleppe, A.: The Object Constraint Language: Precise Modeling with UML. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (1999)