**Trigger Based Assertion Tool in Software Development**

Sei Tanaka and Hiromi Kobayashi

**Abstract**—Assertions are attracting attention recently for preventing programs from incorrect behavior. OCL is a logical description language included in UML. Assertions, although not usually implemented in a database, are implemented in the application program and executed in the test phase. They are commented out in the operation phase for performance. However, faulty data are possible to receive in open distributed systems such as Web application systems. Therefore, assertions must be implemented in the database as well as in the application program and executed in the operation phase. When using assertions in commercial databases, an assertion statement must be expressed generally as triggers in SQL, now. In an earlier study, a transformation tool from OCL assertions to triggers was developed for the standpoint of database management. However, it seems not to totally consider an application development because only invariant constraint and basic logical expressions are dealt with. To overcome these problems of design by contract in software development, we developed a new transformation tool that is easy to use from GUI. This tool generates the trigger of PostgreSQL from an OCL assertion.

**Index Terms**—Assertion, OCL, safety, software development, trigger.

I. INTRODUCTION

Assertions are used recently for preventing software from incorrect behavior in software development. The assertion, i.e., precondition, postcondition, and invariant are desired not only in software specifications, but also in programs designed for the safe behavior of a system. A set of graphical notations, UML, is widely used in the description of software specifications. However, a logical description is necessary for the precise description of assertions in a specification. As a formal logical language based on first predicate logic, OCL was developed for describing logical specifications added to class diagrams or sequence diagrams as comments [1]-[3]. Assertions are generally implemented in an application program at client computers or application servers in Web services. However, assertions must be implemented in servers or database servers as well as application programs at client computers or application servers in open distributed systems such as Web applications in the Internet to ensure software safety. From another perspective, assertions are executed only in the test phase. They are commented out in the operation phase for performance. Application programs are not degraded, unlike hardware components, in terms of reliability. However, incorrect data are possible to receive in open distributed systems. Therefore, assertions must be implemented in a database as well as in an application program and must be executed in the operation phase in these systems.

When using assertions in the server, the description of assertions is definable as the CREATE ASSERTION statement in SQL92. However, assertions are not supported in commercial database servers in practical software development [4] because the description of SQL is apt to be made as simple as possible and conditions and constraints are implemented in the application program in conventional system developments. Therefore, such constraints must be expressed generally as triggers in SQL, now. Alternatively, CHECK statements at the table definition can be used only when constraints are simple. In addition, declarative statements such as CHECK are slower than trigger [5]. Consequently, some transformation tool is needed from OCL assertions in specification to triggers for the aid of software development. We previously proposed a transformation method from assertions to triggers and made a prototype tool [6]-[8]. Now, we developed a new tool that is easy to use from GUI for PostgreSQL. We add new expression types to adjust it to a course registration model because necessary expression types are dependent on application domains. Moreover, we utilize XMI (XML Metadata Interchange) [9], [10] to capable of use a class diagram from UML modeling tools such as ArgoUML [11] and Omondo [12]. We give an outline of this tool and show the result of transformation in this paper. Triggers differ slightly from SQL dialect of vendors, but our transformation framework is applicable to other SQLs.

II. RELATED WORK

Several studies have been conducted as transformation methods from constraints described using OCL to Java in application programs. Java code and Java Modeling Language (JML) [13] were combined for use in verification in studies [14], [15] as follows. Java code is generated from class diagrams. JML which describes comments of Java code is transformed from constraints represented using OCL. In another study, tools were developed in which assertions are generated automatically from Java code. Daikon [16]-[18] is a well-known tool. However, assertions are limited to those generated from Java code with a simple relation among attributes on Daikon. In earlier another study [19], [20], a tool called OCL2Trigger was developed. However, this tool aimed at database management. Basic logical expressions were used to preserve data integrity. The invariant constraint was only considered, and the study of precondition and...
postcondition were not conducted. However, some expression types are needed to add besides basic logical expressions in a software application. In addition, the precondition and postcondition are important to design by contract in software development. Moreover, when either of precondition or postcondition is needed, invariant is wasteful in operation because both precondition and postcondition are used for implementing invariant.

Recently, Dresden OCL2 toolkit was developed for transforming OCL into SQL table [21]. However, trigger is not used.

III. DESCRIPTION OF ASSERTION AND TRANSFORMATION

A. Outline of Transformation

The outline of our transformation method is presented in Fig. 1. It is transformed from assertions described using OCL with a class diagram to triggers. Templates and mapping are used to formalize this transformation. Using this transformation method, we developed a transformation tool using Java for PostgreSQL. In addition to triggers, functions must be defined in PostgreSQL. The description of triggers is not equivalent among DBMSs. However, our framework of transformation is useful by modifying templates and mapping when using other DBMSs. Therefore, a trigger and a function are generated from an assertion described using OCL with a class diagram of UML. The SQL tables are generated simultaneously from class diagrams as is well known in this field.

III. DESCRIPTION OF ASSERTION AND TRANSFORMATION

A. Outline of Transformation

The outline of our transformation method is presented in Fig. 1. It is transformed from assertions described using OCL with a class diagram to triggers. Templates and mapping are used to formalize this transformation. Using this transformation method, we developed a transformation tool using Java for PostgreSQL. In addition to triggers, functions must be defined in PostgreSQL. The description of triggers is not equivalent among DBMSs. However, our framework of transformation is useful by modifying templates and mapping when using other DBMSs. Therefore, a trigger and a function are generated from an assertion described using OCL with a class diagram of UML. The SQL tables are generated simultaneously from class diagrams as is well known in this field.

III. DESCRIPTION OF ASSERTION AND TRANSFORMATION

A. Outline of Transformation

The outline of our transformation method is presented in Fig. 1. It is transformed from assertions described using OCL with a class diagram to triggers. Templates and mapping are used to formalize this transformation. Using this transformation method, we developed a transformation tool using Java for PostgreSQL. In addition to triggers, functions must be defined in PostgreSQL. The description of triggers is not equivalent among DBMSs. However, our framework of transformation is useful by modifying templates and mapping when using other DBMSs. Therefore, a trigger and a function are generated from an assertion described using OCL with a class diagram of UML. The SQL tables are generated simultaneously from class diagrams as is well known in this field.

III. DESCRIPTION OF ASSERTION AND TRANSFORMATION

A. Outline of Transformation

The outline of our transformation method is presented in Fig. 1. It is transformed from assertions described using OCL with a class diagram to triggers. Templates and mapping are used to formalize this transformation. Using this transformation method, we developed a transformation tool using Java for PostgreSQL. In addition to triggers, functions must be defined in PostgreSQL. The description of triggers is not equivalent among DBMSs. However, our framework of transformation is useful by modifying templates and mapping when using other DBMSs. Therefore, a trigger and a function are generated from an assertion described using OCL with a class diagram of UML. The SQL tables are generated simultaneously from class diagrams as is well known in this field.

III. DESCRIPTION OF ASSERTION AND TRANSFORMATION

A. Outline of Transformation

The outline of our transformation method is presented in Fig. 1. It is transformed from assertions described using OCL with a class diagram to triggers. Templates and mapping are used to formalize this transformation. Using this transformation method, we developed a transformation tool using Java for PostgreSQL. In addition to triggers, functions must be defined in PostgreSQL. The description of triggers is not equivalent among DBMSs. However, our framework of transformation is useful by modifying templates and mapping when using other DBMSs. Therefore, a trigger and a function are generated from an assertion described using OCL with a class diagram of UML. The SQL tables are generated simultaneously from class diagrams as is well known in this field.

B. OCL Assertion Template

OCL is a logical language developed for specifying assertions in UML. Assertions represented using OCL are added to a class diagram as text style comments. We use types and attributes defined in class diagrams representing assertions. Fig. 2 presents templates of an OCL assertion. In this figure, assert_type is denoted as pre, post, and inv, respectively representing precondition, postcondition, and invariant. expression represents an assertion expression in OCL. Necessary expression types are regarded as different from application domains. However, five types represented as from expr_type1 to expr_type5 are observed in the course registration system at our university up to the present, where expr_type4 and expr_type5 do not belong to the basic logical expression[6]-[8]. attri_name at expressions corresponds with attr_name, denoting a name of attribute in a class. Comparison operators are described in operator1 and operator2. Constant values are written in value1 and value2. With regard to expression types, expr_type1 restricts the attribute values of a class. expr_type2 restricts the number of instances generated from a class using size(), which is the retrieved number of instances. The collection operation is used for retrieving a set of instances. expr_type3 restricts the number of instances satisfying the condition using size() and select(). A set of instances satisfying the conditional expression is retrieved using select(). expr_type4 avoids duplicated registration of an attribute using isUnique(). expr_type5 restricts an attribute on conditions in excludes() or includes(). Logical operators such as and or is useful in these templates, but are omitted from Fig. 2 for simplicity.

C. Trigger Template

A trigger is a statement that is executed automatically as a side-effect of a modification to tables of a database. This modification is INSERT, UPDATE, or DELETE. A name of a table where the trigger is caused, the conditions under which the trigger is to be executed, the actions to be taken when the trigger executes, and the trigger timing are described in the statement of a trigger in general. However, when using a trigger of PostgreSQL, a statement of function in which the transaction is described is necessary. The templates of a trigger and function are shown, respectively, in Fig. 3(a) and 3(b). Arbitrary names can be taken in trigger_name, function_name, and cursor_name. manipulation_type is DELETE, UPDATE, or INSERT representing modification operation of SQL, in Fig. 3(a). trigger_type denotes the timing when the trigger occurs; it is written BEFORE or AFTER, which respectively mean that the trigger occurs before or after the SQL is executed. When assertions are preconditions and postconditions, BEFORE and AFTER are used, respectively. When the assertion is invariant, both BEFORE and AFTER are used as a quasi-action. Then the corresponding trigger is made. In the template of function, function_name() corresponds to that in the template of the trigger. var_type denotes a variable of the type in the function, cursor_name denotes the cursor name taken as an arbitrary name. key is an attribute (or a set of attributes) given from a class diagram. In addition, table_name is given from class_name, and variable1 and variable2 are given from those of the class diagram. Furthermore, operator_cur and operator_if are described, respectively, as comparison operators, and numeric1 and numeric2 are written as constant numbers.
We developed a user friendly GUI tool by which a trigger and a function are transformed from an OCL assertion and a class diagram using Java. This tool is constructed from Class Diagram Parser, Class-Table Converter, OCL Assertion Editor, OCL Syntax Checker, and OCL-Trigger Translator (Mapper) for improving the formerly developed tool [6]-[8].

The transformation procedure is the following, as depicted in Fig. 4. A class diagram is inserted through a UML modeling tool into Class Diagram Parser using XMI. It is parsed using SAX and converted to tables using Class-Table Converter. An OCL assertion is created using OCL Assertion Editor or inserted using a text file now. However, it will be inserted through a UML modeling tool after the OCL is formalized as XMI sufficiently in the future. Next, an OCL assertion is parsed using OCL Syntax Checker. Subsequently, the OCL assertion is transformed to a trigger and function using attributes above tables in OCL-Trigger Translator. A part of GUI of OCL-Trigger Translator tool is presented in Fig. 5. An OCL assertion is inserted as a text file through pushing Open text button or described on the left side text box in accordance with the template of OCL. Trigger name and its function name are written into the lower text boxes. After the center Generate button is pushed, the syntax check is performed. Next, a pair of trigger and function is created on the right side text box. Finally, it is saved as a text file by pushing the Save button.

Fig. 5. Snapshot of OCL-trigger translator.

Fig. 3. Template of trigger and its function.

IV. TOOL

Fig. 4. Activity of our tool.

Fig. 6. Tables of course registration model.

V. TRANSFORMATION RESULT

Fig. 6 is the set of tables for extracting server side assertions that are transformed from a class diagram of a course registration model. This model is extracted from the course registration system of our university. The conversion from a class diagram to tables is well known; therefore, it is omitted herein. A Student table has attributes studentId, studentName, and semester. A Course table has attributes courseId, courseName, minSemester, i.e., the minimum number of semesters a course can be taken, and period.
(week[s] and hour[s]). A Register table has studentId, courseId, and period. AcqCourseRec (Acquired Course Record) table has studentId, courseId, and semester that was taken.

Examples of OCL assertion are shown as follows.
1) \[ expr_{type1} \] The constraint of attribute[s]
context Student
pre: Student.semester \leq 16
The semester of Student is less than or equal to 16.
2) \[ expr_{type2} \] The constraint of the number of attributes
context Student
inv: Student -> size() \leq 50
The number of Student is less than or equal to 50.
3) \[ expr_{type3} \] The constraint of the number of attributes
context Register
pre: Register -> select(Register.courseId = 100)
-> size() \leq 50
Course with courseId=100 can be registered less than or equal to 50 persons.
4) \[ expr_{type4} \] The prevention of duplication
context Course
pre: Course -> isUnique (Course.courseName)
The same name cannot be used as a courseName.
5) \[ expr_{type5} \] The constraint of attribute[s]
context Register
pre: Register.courseId = 101
-> includes(AcqCourseRec.courseId = 100)
The Course having courseId=101 requires one to take courseId=100 beforehand.

According to the timing (before or after) described in a trigger, if a constraint is satisfied, a table is modified (insert, update, or, delete) correctly. Otherwise, a table is not modified and this transaction is aborted, now. But, abort actions are needed to be considered in the future work because such actions are dependent on a business rule.

VI. CONCLUSION
A closed system is presumed in a conventional system development. Therefore, software designers are aware of where each application program module is connected. Consequently, assertions are usually implemented in application programs while the description of SQL is conducted as simply as possible. In addition, assertions are executed only in the test phase. In contrast, in open distributed systems, especially on the Internet, software designers do not estimate where each application program module is connected and incorrect data are possible to receive. Therefore, assertions must be implemented in database servers as well as application program at client computers or application servers and executed in the operation phase to ensure software safety.

This paper presents an outline of our new tool for implementing assertions such as precondition, postcondition, and invariant using the trigger of PostgreSQL and shows the result of this transformation. Transformation is executed from an OCL assertion and a class diagram to a trigger and a function using templates. XMI format is used to capable of use a class diagram from UML modeling tools. Triggers differ among DBMSs; however, the idea of using templates and mapping can be applicable. The description styles of assertions are five types, which are found as typical types in the course registration system in our university, now. Necessary assertion types are dependent on the application domain. Even if the ASSERTION statement is supported in commercial databases in the future, trigger is still useful because of its various expressive power and superior performance compared with declarative statements. Additional work must be conducted to implement other description styles of assertions if they are found.

REFERENCES


Sei Tanaka received the B.Eng. and M.Eng. degrees from Tokai University, Japan, in 2012 and 2014, respectively. His research interests include software specification methods, formal method, and database applications. He is now with Information System Engineering, Co., Ltd, Tokyo.

Hirom Kobayashi received the Dr. Eng. degree in systems engineering from Shinshu University, Japan, in 1993. He previously worked for several software companies. He joined the Department of Management Engineering at Tokai University in 1993 and joined the Department of Information Media Technology in 2001. He is currently a professor in the Department of Information Media Technology. He served as an associate editor in chief for the Journal of the Institute of Electronics, Information and Communication Engineers in Japan. His research interests include software specification methods, database applications, mobile phone systems, dependable computer systems, and distributed computer systems. He is a member of the IEEE Computer Society and the ACM.