Smalltalk. В данной работе приводится ДОС языка АОП AspectTalk, которое заключается в задании тройки объектов (L, S, M).

Для каждого нетерминала X из грамматики G языка AspectTalk определяется язык L_X, грамматика которого получается из G заменой аксиомы на X. Множество всех языков L_X обозначается L и называется множеством *синтаксических областей*. Примерами синтаксических областей являются множество записей примитивных операций языка AspectTalk и сам язык AspectTalk.

Элементы множества S являются *доменами* — множествами с завершённым частичным порядком. Сумма доменов, декартово произведение доменов, а также множество отображений из домена в домен являются доменами; порядок на последних определяется с помощью порядков на первых. Допускаются рекурсивные определения доменов. Подробнее о доменах можно прочитать в [5]. Домены ДОС языка AspectTalk подбирались для отражения сущностей языка и включают, например, домен процедур — домен функций из домена состояний в домен состояний — и домен программ — домен функций из домена входных последовательностей в домен выходных последовательностей.

Множество функций M, отображающих из элементов L в элементы S, называется множеством *семантических отображений* и, фактически, задаёт интерпретацию языка: множество M включает функцию, сопоставляющую программам на AspectTalk элементы функционального домена.

ЛИТЕРАТУРА

1. *Стефанцов Д. А.* Реализация политик безопасности в компьютерных системах с помощью аспектно-ориентированного программирования // Прикладная дискретная математика. № 1(1). 2008. С. 94–100.
2. *Стефанцов Д. А.* Технология и инструментальная среда создания защищённых систем обработки информации // Прикладная дискретная математика. Приложение № 1. 2009. С. 55–56.
3. *Стефанцов Д. А., Крюкова А. Е.* Формальное доказательство семантической эквивалентности ядра языка АОП AspectTalk и языка ООП Smalltalk // Прикладная дискретная математика. Приложение № 3. 2010. С. 84–85.
4. *Tennent R. D.* Denotational semantics // Handbook of logic in computer science. Oxford, UK: Oxford University Press, 1994. V. 3. P. 169–322.
5. *Scott D. S.* Data types as lattices // Lecture Notes in Mathematics. 1975. V. 499. P. 579–651.

УДК 004.428

РАЗРАБОТКА И РЕАЛИЗАЦИЯ БИБЛИОТЕКИ ORM НА ЯЗЫКЕ C++[1]

Д. А. Стефанцов, Н. О. Ткаченко, Д. В. Чернов, Р. В. Шмакова

Распространённым способом хранения информации в компьютерных системах является использование реляционных баз данных (БД), при котором информация о сущностях и связях предметной области представляется в виде записей в таблицах. Каждому из полей таблицы соответствует имя и тип хранимых данных. Можно выделить два класса библиотек взаимодействия с системами управления БД (СУБД) для языков программирования (ЯП). Библиотеки первого класса устанавливают соответствие

[1]Работа выполнена в рамках реализации ФЦП «Научные и научно-педагогические кадры инновационной России» на 2009–2013 гг. (гос. контракт № П1010).

между данными, хранимыми в полях записей таблиц, и данными элементарных типов в соответствующем ЯП. Библиотеки второго класса характерны для объектно-ориентированных ЯП и устанавливают соответствие между записями БД и объектами в ЯП; при этом поля записей соответствуют член-данным объектов. Эти библиотеки, называемые также объектно-реляционными отображениями (или ORM — Object-Relational Mapping) [1], реализуются с помощью библиотек первого класса и удобнее в использовании, поскольку позволяют абстрагироваться от способа представления информации в БД и использовать только понятия соответствующего ЯП. Создана библиотека ORM для языка C++ [2], рабочее название которой — «C++ ORM on Templates» (COT).

Основной сложностью в реализации ORM является избавление пользователя от необходимости поддержания большого количества служебной информации, нужной для взаимодействия с СУБД. Во-первых, необходимо установить соответствие между таблицами БД и классами объектов в ЯП. При этом от пользователя ORM требуется задание имени класса, а также имен и типов член-данных его экземпляров. Запросы на вставку, удаление и изменение соответствующих записей в таблице БД генерируются ORM. Во-вторых, необходимо задавать запросы на выборку данных из таблиц БД, используя только член-функции классов (называемые в языке C++ *статическими*) и объектов и не прибегая к использованию языка запросов SQL [3]. Это означает необходимость генерации член-функций запросов на выборку данных, не существующих в момент реализации ORM. Традиционно оба перечисленных требования к ORM реализуются с помощью *рефлексии* [4] — способа метапрограммирования, заключающегося в доступе программы к информации о собственной структуре и возможности изменять эту структуру. Под *структурой программы* подразумевается совокупность типов данных и алгоритмов, используемых в ней.

Язык C++ не обладает встроенной возможностью рефлексии, поэтому реализациям ORM необходимо поддерживать метаданные (данные о данных) самостоятельно. Например, библиотека ODB [5] подразумевает использование директив типа `#pragma` в описании классов, соответствующих таблицам БД, и обработку этих описаний внешним транслятором, расширяющим структуру программы. Очевидны недостатки такого способа реализации ORM: наличие дополнительного транслятора, что затрудняет использование библиотеки на всех программно-аппаратных платформах, имеющих трансляторы с C++, а также невозможность автоматической проверки корректности программы до этапа трансляции. Другая библиотека, Wt::Dbo [6], требует поддержания метаданных от своих пользователей; кроме того, в член-функциях, осуществляющих запросы на выборку данных, необходимо использование частей строк запроса `SELECT` на языке SQL, что, во-первых, делает необходимым знание SQL пользователями Wt::Dbo и, во-вторых, может привести к появлению в программе SQL-инъекций [7].

В данной работе используется метапрограммирование на шаблонах в C++, позволяющее расширять структуру программы на этапе её трансляции. Современные трансляторы имеют необходимые алгоритмы оптимизации программ, что делает использование метапрограммирования на шаблонах возможным не только в качестве эксперимента. Популярные библиотеки Boost [8] являются примером использования этого способа написания программ на C++.

В языке C++ возможно рекурсивное описание типов с помощью конструкций `typedef` и `typename`. Библиотека COT использует для хранения информации о типах параметров и результатах выполнения запросов односвязные списки классов, построенные этим способом.

Для описания *моделей* — классов в ЯП, соответствующих таблицам в БД, — в COT используются макросы `BEGIN_MODEL`, `END_MODEL` и `FIELD` (см. строки 1–4 листинга 1). Данные макросы заменяются препроцессором на описание соответствующего класса, которое включает определения: 1) член-данных его экземляров; 2) классов, содержащих информацию об этих член-данных и их связях с полями таблицы в БД; 3) статических член-функций, выполняющих запросы к СУБД, и другую служебную информацию.

Запрос на выборку данных приведён в строках 6–8 листинга 1 — это статическая член-функция `filter` класса модели, принимающая переменное количество параметров. Шаблонные параметры заключены в угловые скобки и представляют собой связанные булевыми функциями условия выборки — шаблонные классы сравнения `Lt`, `Gt` или `Eq`, осуществляющие проверку отношения «меньше», «больше» или «равно» соответственно. Параметром шаблонных классов сравнения является имя поля модели; значения, с которыми сравниваются эти поля, передаются в член-функцию `filter` в качестве параметров в круглых скобках (при этом первым должно быть указано количество сравнений).

```
1 BEGIN_MODEL(Author)
2     FIELD(name, StringValue<256>)
3     FIELD(age, IntValue)
4 END_MODEL
5
6 authors = Author::filter<
7         And< Lt<Author::_age_>, Eq<Author::_name_> >
8     >::with(2, 40, "John Smith");
```

Листинг 1. Пример описания модели и запроса на выборку данных в COT

Предоставляемые библиотекой COT средства разрабатывались по аналогии с ORM из библиотеки Django [9] для ЯП Python [10] — одной из самых известных реализаций ORM. Описание модели и запроса на выборку данных, реализованные с помощью Django ORM, аналогичные представленным на листинге 1, можно найти в строках 1–3 и 5–6 листинга 2 соответственно.

```
1 class Author(models.Model):
2     name = models.CharField(max_length=256)
3     age = models.IntegerField()
4
5 authors = \
6     Author.objects.filter(age__lt=40, name="John Smith")
```

Листинг 2. Пример описания модели и запроса на выборку данных в Django ORM

Заметим, что в Django ORM условия выборки объединяются логической связкой «и», при этом использование связок «или» и «не» затруднено. В COT шаблонные классы `And`, `Or` и `Not` равноправны.

Для связи с СУБД использован метод подготовленных запросов, позволяющий единожды произвести синтаксический разбор строк на языке SQL и впоследствии только подставлять параметры при выполнении запроса. Это делает невозможным изменение синтаксической структуры запроса после его подготовки, что исключает SQL-инъекции. Для передачи результатов выполнения запросов из процедур использованы

указатели из библиотеки Boost, отслеживающие количество ссылок на объекты для автоматического управления памятью.

Работа библиотеки изучалась с помощью инструментов GNU gprof [11] и Valgrind [12]. Результаты анализа показали, что библиотека COT в среднем не уступает в производительности библиотеке ODB. При реализации судейской системы для проведения игр по защите информации CTF [13] COT превысила по быстродействию библиотеку ODB не менее чем на 9 %. Использование инструмента Valgrind позволило избавиться от утечек памяти.

ЛИТЕРАТУРА

1. *Ambler S. W.* Mapping Objects to Relational Databases: O/R Mapping In Detail / Ambysoft Inc. 2010. `http://www.agiledata.org/essays/mappingObjects.html`
2. *Страуструп Б.* Дизайн и эволюция языка C++. Объектно-ориентированный язык программирования. М.: ДМК Пресс, 2000. 448 с.
3. *Musteata B. and Lesser R.* Standard SQL Relational Database Language Guide and Reference. TLM, Inc., 1988. 275 p.
4. *Sobel J. M. and Friedman D. P* An Introduction to Reflection-Oriented Programming / Computer Science Department, Indiana University, USA. 1996. 20 p. `http://www.cs.indiana.edu/hyplan/jsobel/rop.ps.gz`
5. ODB: C++ Object-Relational Mapping (ORM) / Code Synthesis Tools CC. 2011. `http://www.codesynthesis.com/products/odb/`
6. *Deforche K.* Wt::Dbo Tutorial. 2010. `http://www.webtoolkit.eu/wt/doc/tutorial/dbo/tutorial.html`
7. SQL Injection / OWASP — The Open Web Application Security Project. 2011. `https://www.owasp.org/index.php/SQL_Injection`
8. *Dawes B., Abrahams D., and Rivera R.* Boost C++ Libraries. 2011. `http://www.boost.org/`
9. Django / Django Software Foundation. 2011. `http://www.djangoproject.com/`
10. The Python Tutorial / Python Software Foundation. 2011. `http://docs.python.org/tutorial/`
11. GNU gprof / Free Software Foundation Inc. 2009. `http://sourceware.org/binutils/docs/gprof/index.html`
12. Valgrind / Valgrind^{TM} Developers. 2011. `http://valgrind.org/`
13. *Ткаченко Н. О., Чернов Д. В.* Разработка и реализация сервера игры CTF // Прикладная дискретная математика. Приложение. 2010. № 3. С. 62–64.