OSTIS-2014
(Open Semantic Technologies for Intelligent Systems)

УДК 004.822:514

ПРАКТИЧЕСКИЕ АСПЕКТЫ ТЕСТИРОВАНИЯ РАСПРЕДЕЛЕННЫХ ПРИЛОЖЕНИЙ НА ОСНОВЕ СВЯЗАННЫХ ДАННЫХ

Солошич С.Н.*

* *Кременчугский национальный университет имени Михаила Остроградского, г. Кременчуг, Украина*

soloshich@gmail.com

В данной работе рассмотрены практические аспекты тестирования распределенных приложений, использующих триплесторы в качестве хранилищ связанных данных и предложен новый подход к тестированию подобных хранилищ, который позволяет выявлять проблемы, связанные с внутренним состоянием хранилища.
Ключевые слова: тестирование; распределенные приложения; связанные данные; SPARQL.

Введение

Целесообразность проведения тестирования удаленных открытых хранилищ связанных данных (англ. Triplestore) обусловлена необходимостью проверки работоспособности разрабатываемого пользовательского программного обеспечения (ПО) с этими хранилищами. Эта необходимость обусловлена тем, что хранилища от различных разработчиков программного обеспечения поддерживают разное подмножество запросов SPARQL (англ. SPARQL Protocol and RDF Query Language) и обладают различной функциональностью для обработки связанных данных. Уровень поддерживаемой функциональности хранилища различается от источника к источнику и определяет уровень поддерживаемой функциональности клиентского ПО, что зачастую ограничивает клиентское ПО в функциональности, делая недоступными функции доступа к связанным данным, например, по изменению этих данных, а именно функции SPARQL Update не доступны. Также важными проблемами в этом вопросе являются выявления противоречивости, полноты и избыточности связанных данных.

Тестирование хранилищ связанных данных также имеет важное значение для гарантирования высокого качества программного обеспечения, так как незамеченные ошибки могут привести к неисправимому искажению данных. Проблему тестирования хранилищ связанных данных в целом можно разбить на проблемы создания тестов, подготовку тестируемых данных и истолкование результатов испытаний. Среди трех проблем, проблема генерирования тестов напрямую влияет на эффективность тестирования. Традиционно, тестирование хранилищ связанных данных основано на проверке его способности выполнять набор предопределенных функций. Достижение элементарного (базового) уровня качества, стратегии тестирования методом черного ящика (англ. Black box) хранилищ связанных данных, является важным заданием, но использование метода белого ящика (англ. White box) необходимо для более тщательного ее тестирования. Несмотря на важность данной задачи, семантика операторов языка запросов к связанным данным SPARQL, встроенных в хранилища связанных данных, редко рассматривается в обычных методах тестирования «белого ящика». Мы предлагаем дополнить методы тестирования «белого ящика» с помощью использования семантики SPARQL. Данный подход заключается в преобразовании встроенных операторов SPARQL в процедуры в некоторых языках программирования общего назначения (далее – ЯПН) и тем самым генерации тестовых случаев с использованием традиционных методов тестирования «белого ящика». Дополнительные тестовые случаи, которые не охватываются в традиционном тестировании методом белого ящика, создаются для повышения эффективности тестирования пользовательского программного обеспечения хранилищ связанных данных.

Увеличение, как самой сложности хранилищ связанных данных, так и ожиданий относительно ее надежности способствовало возрастанию требований к методам тестирования программного обеспечения. Эффективное и действенное

тестирование программного обеспечения имеет решающее значение в цикле разработки программного обеспечения и технического обслуживания.

Тестирование хранилищ связанных данных имеет большое значение как на стадии разработки, так и в процессе работы, поскольку незамеченные ошибки в этих приложениях могут привести к неправильной модификации или случайному удалению важных данных. После того как данные ошибочно изменены, ошибка может распространяться и привести к увеличению разрушения данных, если она не отслежена. Так как не все транзакции могут быть развернуты, восстановления данных из резервных копий базы данных не сможет решить проблему. Хорошо известные правила функциональной зависимости в состоянии обеспечить целостность хранилищ связанных данных, но понесенные расходы, как правило, слишком велики для нетривиальных коммерческих хранилищ связанных данных [1].

Для тестирования хранилищ связанных данных должны быть разработаны как статические, так и динамические испытания. Статические мероприятия, такие как осмотр и проверка, позволяют определить соответствие прикладной программы и дизайна хранилищ связанных данных всем функциональным требованиям к данным и отсутствие конфликта между этими требованиями [2]. Кроме того, приложения должны ограничивать несанкционированный доступ [3], позволять одновременный доступ, и поддерживать восстановление данных после аппаратных или программных сбоев. Хотя различные исследования были проведены для определения статических методов тестирования проектирования хранилищ связанных данных, относительно мало внимания было уделено непосредственно динамическому тестированию хранилищ связанных данных. Предложен подход тестирования, который преобразует встроенные запросы SPARQL хранилищ связанных данных для процедур на языке программирования общего назначения. Целью преобразования является включение в семантику SPARQL таких запросов, которые позволят сгенерировать больше тестов и выявить недостатки, относящиеся к изменениям внутренних состояний распределенной базы данных. Подход позволяет сгенерировать тесты, которые функционируют в семантике GPL и SPARQL запросов, которые созданы с использованием традиционных методов тестирования «белого ящика» и, следовательно, помогает выявить больше ошибок, содержащихся в распределенных базах данных. В частности, он обнаруживает неисправности, которые произошли в некоторых специфических внутренних состояниях хранилищ связанных данных.

Хранилища связанных данных в целом можно разделить на две категории. Первая категория состоит из приложений, которые являются исключительно встроенными в SPARQL Update (далее – SPARUL). В этих приложениях запросы и транзакции указываются через SPARUL, а пользовательские интерфейсы определены с использованием элементов на языке запросов хранилища связанных данных. Другая категория включает в себя приложения, которые построены в SPARUL и ЯПН, таких как C++, C# и Java [4]. Запросы в SPARUL, мощном декларативном языке запросов, встраиваются в программы приложений, написанных на любом ЯПН, который называется “принимающим языком” [4]. В этих приложениях запросы и изменения данных записываются как встроенные операторы SPARQL. Другие функции, такие как взаимодействие с пользователями и отправка результатов в графическом интерфейсе пользователя, записывается посредством ЯПН. Задачей является тестирование этой категории приложений баз данных.

Тестирование приложений хранилищ связанных данных, отличается от тестирования структурированных программ. Исходные данные хранилищ связанных данных включают в себя как данные, вводимые пользователем, так и экземпляры объектов хранилищ связанных данных. Как указано выше в дополнение к проверке исхода с ожидаемым результатом, программисты или тестеры должны также проверить, является ли хранилище связанных данных согласованным и удовлетворяет ли исходным условиям в тестировании, которые предъявляться к хранилищам связанных данных. Проблему тестирования подобных хранилищ можно разделить на три подзадачи, как показано на рисунке 1. Это проблемы генерирования тестов, подготовки данных к тестам, выполнение тестов и проверка результата.

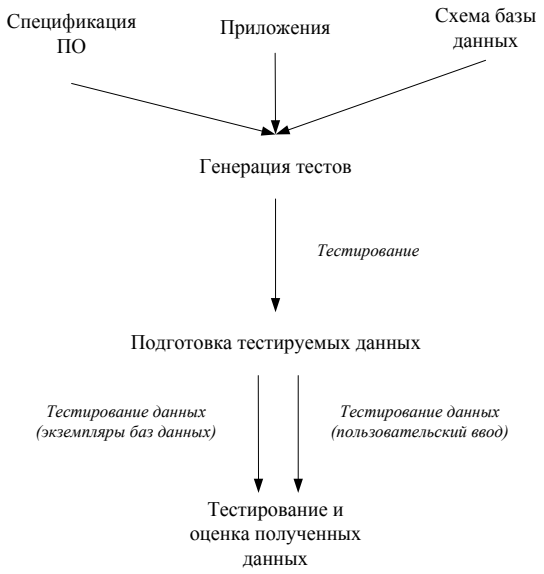

Рисунок 1 – Проблемы тестирования приложений хранилищ данных

Среди трех подзадач, проблема генерирования тестов нуждается в решении больше, чем остальные, поскольку эффективность тестирования

хранилища связанных данных непосредственно связана с созданием тестов. Как уже говорилось во введении, незамеченные недостатки в хранилищах связанных данных может привести к серьезному повреждению важных результирующих данных. Данная работа ориентируется на проблемы генерирования тестов. Традиционно методы тестирования «черного ящика» используются для создания функциональных тестов. Тестирование методами «белого ящика» требуется для достижения более высокого уровня качества и надежности программного обеспечения. В работе [5], Robbert и Maryanski предложил концептуальную модель подхода к генерации тестов, которые можно применить к хранилищам связанных данных.

1. Тестирование на основе методов черного ящика

В настоящее время большинство приложений хранилищ данных проверяются с использованием методов тестирования черного ящика, например, эквивалентное разделение, анализ граничных значений и причинно-следственные графические методы. В рамках этих методов тестовые сценарии получаются без посылания на построение прикладных программ. В соответствии с характеристиками программного обеспечения, тесты проводятся для проверки работоспособности независимо друг от друга. Это обычно используется для проверки соответствия характеристик программного обеспечения к требованиям приложения.

Одно из преимуществ методов тестирования черного ящика является то, что тестовые варианты могут быть получены независимо от программ на более ранней стадии цикла разработки программного обеспечения. Программисты могут проводить тесты во время разработки приложения. Таким образом, итоговое приложение, как правило, удовлетворяет большую часть сгенерированных тестов. Относительно небольшие усилия требуются для тестирования и отладки приложений. Еще одним преимуществом является то, что затраты на генерацию тестовых случаев с использованием методов «черный ящик» довольно низкие по сравнению с расходами, связанными с методами белого ящика. Таким образом, стоимость разработки может быть снижена. Кроме того, многие методы тестирования «черного ящика» уже хорошо апробированы. Тестеры могут выбрать наиболее подходящий метод для тестирования их приложений хранилищ данных. С другой стороны, без изучения программы приложения, невозможно знать, какая часть приложения проходит это тестирование. Практически, функциональные спецификации, как правило, указаны на естественном языке. Как отмечено в [2], использование естественного языка с целью определения ошибок в функциональной спецификации, таких как: дублирование, противоречивость и неполнота. Кроме того, большинство методов тестирования «черного ящика» нечувствительны к некоторым видам неисправностей. Общие виды неисправностей еще могут остаться незамеченными, даже если все тесты, определенные методом тестирования успешны. Типичным примером является своего рода неисправности, что происходят только в некоторых специфических последовательностях выполнения функций. Таким образом, необходимо искать методы для применения за пределами тестирования «черного ящика».

2. Традиционный подход к тестированию методами белого ящика

Подходы к тестированию методами «белого ящика» использующие покрытие операторов, условий, путей, функций, значений параметров могут быть использованы для тестирования некоторой части или применены ко всему хранилищу связанных данных для достижения более полного тестирования этого хранилища. Методы тестирования белого ящика позволяют изучить код подробно и убедится, что, по крайней мере, в определенной степени тестовое покрытие, такое как выполнение каждого оператора было достигнуто. Тем не менее, традиционные методы тестирования «белого ящика» имеют свои ограничения при тестировании хранилища связанных данных. Самое главное, они не полностью учитывают семантику запросов SPARQL, встроенных в прикладные программы распределенных приложений. Запросы SPARQL рассматриваются как «черные ящики». Как правило, на основе тестирования «белого ящика» генерируется только несколько тестов для тестирования встроенных SPARQL запросов. Следует отметить, что преднамеренно не генерируется никаких тестов, включающих семантику запроса SPARQL, которые отражают изменение внутреннего состояния хранилища связанных данных. Следовательно, можно утверждать, что традиционное тестирование методом «белого ящика» может пропустить тип неисправностей, связанных с внутренними изменениями в хранилище.

Чтобы удовлетворить условия этих методов для приложений хранилищ связанных данных, предлагается механизм, который позволяет сгенерировать дополнительные тесты из встроенных запросов SPARQL. Механизм, который показан на рисунке 2, преобразовывает операторы SPARQL хранилища связанных данных в операторы ЯПН. Затем стандартные методы «белого ящика» применяются к трансформированным запросам и другим операторам, записанным на «принимающем языке». Также задачей является включение семантики оператора SPARQL в генерацию теста. Другими словами, выполняется проверка правильность семантика запроса SPARQL в

сочетании с запросом языка программирования общего назначения.

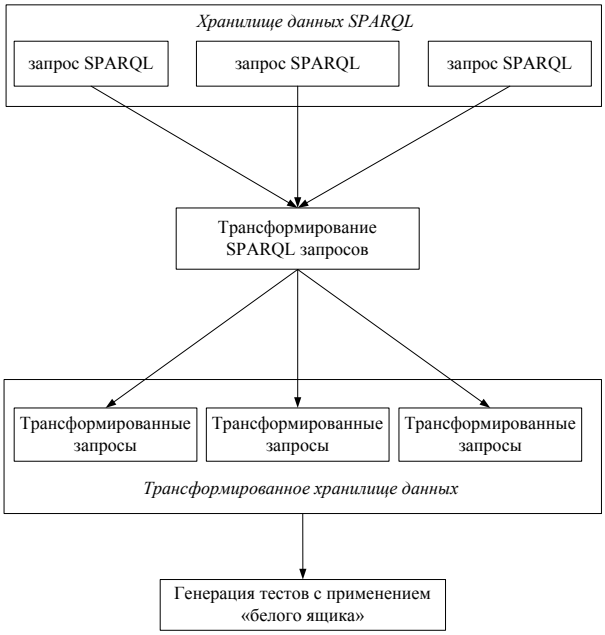

Рисунок 2 – Трансформация SPARQL запросов

Так же следует учитывать отсутствие возможности тестирования SPARQL операторов по отдельности, как «черные ящики», потому что функциональная спецификация не определена для запросов SPARQL. Кроме того, SPARQL не может быть достаточно эффективным, чтобы выполнить все функции хранилища связанных данных и операторы ЯПН, возможно, придется использовать для дополнения запросов SPARQL при выполнении некоторых функций.

Таким образом, проблему тестирования хранилищ связанных данных распределенных приложений декомпозируем на проблемы генерирования тестов, подготовки данных исследования, выполнения тестов и итогового контроля, что согласуется с положениями функционального тестирования, которое используется для генерации тестов для тестирования приложений баз данных, в то время как запросы SPARQL, встроенные в прикладные программы конкретно не рассматривается в тестировании методом «белого ящика».

Проведенное исследование подходов к генерации тестов для хранилищ связанных данных выявило, что все они имеют слабые места, но объединение разных методов позволяет их устранить.

Заключение

В данной работе рассмотрены практические аспекты тестирования хранилищ связанных данных и предложен новый подход к проведению тестирования, который преобразует запросы SPARQL в процедуры на языке программирования общего назначения и применяет традиционные методы «белого ящика» для процедур и принимающих запросов, используемых для генерации тестов. Полученные в результате тесты помогут выявить недостатки, связанные с внутренним состоянием хранилища связанных данных.

В дальнейшем будет проведено больше эмпирических исследований для повышения эффективности предложенного подхода к тестированию.

Библиографический список

[Гергор, 2007] Гергор, Х. В. Бобби. Шаблоны интеграции корпоративных приложений / Х. Гергор, В. Бобби. // Пер. с англ. – М.: ООО «И.Д. Вильямс», 2007. – 672 с.

[Громюк, 2011] Громюк Б.П. Аналіз та оцінка скриптів SQL / Б.П. Громюк // Восточно-Европейский журнал передовых технологий, 2011, 6/2 (54), С. 22-25.

[Малишевський, 2009] Малишевський О.Г. Пріоритезація тестів як метод швидкого виявлення серйозних помилок / О.Г. Малишевський // Комп'ютерні технології, 2009, Вип. 93, С. 100-101.

[Фаулер, 2007] Фаулер, М. Архитектура корпоративных программных приложения / М. Фаулер // Пер. с англ. – М.: Издательский дом "Вильямс", 2007. – 544 с.

[DuCbarme, 2011] DuCbarme, B. Learning SPARQL / B. DuCbarme // O'Reilly, 2011. – 256c. [Фаулер, 2007] Фаулер, М. Архитектура корпоративных программных приложения / М. Фаулер // Пер. с англ. – М.: Издательский дом "Вильямс", 2007. – 544 с.

[DuCbarme, 2011] DuCbarme, B. Learning SPARQL / B. DuCbarme // O'Reilly, 2011. – 256c.

[Harinath, 2010] Harinath, S. Microsoft SPARQL Server Analysis Services 2008 with MDX / S. Harinath, R. Zare. – «Sams», 2010. – 1072 с.

[Kaner, 2001] Kaner, C. Testing computer software / C. Kaner, J. Falk, H. Nguyen. – M: DiaSoft, 2001. – 538 с.

[Robbert, 1991] Robbert M.A. Automated Test Plan Generator for Database ApplicationSystems, Proceedings of the 1991 ACM SIGSMALL/ M.A. Robbert, F.J. Maryanski // PC Symposium on Small Systems, ACM Press, 1991, p.100-106.

[Майер-Шенбергер В., 2014] Майер-Шенбергер В. Большие данные. Революция, которая изменит то, как мы живем, работаем и мыслим / Майер-Шенбергер В., Кукьер К. – М.: Манн, Иванов и Фербер, 2014. – 240 с.:ил.

[Гудсон, 2013] Гудсон Дж. Практическое руководство по доступу к данным / Д. Гудсон, Р.Стюарт. – СПб.:БВХ-Петербург, 2013. – 304с.:ил.

[Палкин, 2013] Палкин Н.Б. Бизнес-аналитика от данных к знаниям, 2-е издание / Палкин Н.Б., Орешков В.И. – СПб.: Питер, 2013. – 704 с.:ил.

PRACTICAL ASPECTS TESTING DISTRIBUTED APPLICATIONS BASED ON THE RELATED DATA

Soloshych S.N. *

**Kremenchuk Mykhailo Ostrohraskyi National University, Kremenchuk, Ukraine*

soloshich@gmail.com

This thesis overviews the practical aspects of distributed applications testing based on the linked data, a new approach to the testing of distributed databases is proposed, that can identify problems associated with the internal state of a distributed database.