УДК 004.655+657/519.682.1

Тарек Юсеф Бади Биштави, Г.Н. Жолткевич, Ю.В. Соляник

Харьковский национальный университет им. В.Н. Каразина

ВЫПОЛНИМЫЕ СПЕЦИФИКАЦИИ В ПРОЕКТИРОВАНИИ БАЗ ДАННЫХ

Предложена постановка задачи реализации подхода к проектированию компонентов программного обеспечения, известного как a priori reasoning в применении к базам данных. Основная идея этого подхода заключается в явном использовании спецификации как инструкции к сборке компонента. Рассмотрены проблемы, возникающие на практике, преимущества возможной реализации идеи, подходы к синтезу (генерации) проектного решения. Перечислены задачи для дальнейшей работы.

проектирование программного обеспечения, выполнимые спецификации, верификация, базы данных, генерация кода, компонентные технологии, языки запросов

Введение

В статье рассматривается подход к проектированию компонентов программного обеспечения, известный как *a priori reasoning* в применении к базам данных. В [1, 2] предлагается классифицировать подходы к проектированию компонентов и их композиции как

– ***a posteriori*** – традиционный подход, при котором сначала собирается компонент, а затем проверяется на соответствие требованиям;

– ***a priori*** – подход, при котором из информации, заложенной в спецификации, извлекаются инструкции к сборке компонента.

Явное использование спецификации для построения решения предполагает наличие:

– формального языка спецификаций;

– методологии создания спецификации;

– критерия соответствия требованиям;

– процедуры трансляции (для полной или частичной автоматизации построения модели).

Цель этой работы – постановка задачи построения приемлемых на практике формального аппарата и программных средств поддержки проектирования баз данных на основе спецификаций требований к ним.

Работа построена следующим образом:

– в п. 1 дан обзор методологии проектирования для некоторых этапов жизненного цикла разработки баз данных;

– в п. 2 указаны некоторые проблемы, возникающие на практике;

– в п. 3 описаны идея компонентного подхода к проектированию баз данных и определяющая роль интерфейса;

– в п. 4 рассмотрены преимущества возможной реализации идеи с помощью декларативных языков программирования;

– в п. 5 рассматриваются подходы к синтезу (генерации) проектного решения;

– в п. 6 перечисляются задачи для дальнейшей работы.

1. Методология проектирования баз данных

Методологию проектирования баз данных [3], касающуюся затрагиваемых здесь задач, кратко можно резюмировать так:

– для основных категорий пользователей и направлений деятельности организации рассматриваются так называемые «пользовательские представления». Существуют т. н. «централизованный подход», при котором различные пользовательские представления объединяются общее единое представление, а также «метод интеграции представлений», когда требования к каждому пользовательскому представлению применяются для создания отдельной модели данных;

– после сбора и анализа фактов формируются требования к представлениям, содержащие перечень конкретных задач, реализуемых с использованием базы данных;

– на основе информации, имеющейся в спецификациях требований пользователей, проводится концептуальное проектирование базы данных – создание модели используемой в организации информации, не зависящей от любых физических аспектов ее представления;

– концептуальная модель уточняется и преобразуется в логическую модель данных, учитывающую особенности выбранной модели организации данных (например, реляционной), но без учета типа целевой СУБД;

– далее выбирается способ реализации разрабатываемой базы данных – физическая модель данных.

2. Проблемы, возникающие на практике

Сложность проектирования баз данных остается высокой. То же можно сказать о требованиях к профессиональному уровню и опыту работы участников проекта. Общеизвестные трудности формализации ограничивают возможности CASE-средств в автоматизации работы для большинства задач. Две

из многих причин [3], по которым цель проекта может быть не достигнута:

– отсутствие полной спецификации всех требований;

– недостаточная степень разделения общего глобального проекта на отдельные компоненты, поддающиеся полному контролю и управлению.

Вероятность неудачи настолько велика, что на практике разработчики часто отступают от «тяжеловесных» технологий, диктуемых стандартами, находя использование таких подходов, как *Extreme Programming* [4] «менее рискованным».

На практике с помощью диаграммного языка строится семантическая модель данных, которая затем в ручном или полуавтоматическом режиме преобразуется в целевую модель, чаще – реляционную. Выделим некоторые особенности.

1). Наиболее развитые средства проектирования, такие как UML & OCL [5] дают возможность обеспечить представления модели базы данных и требования к ним на разных уровнях абстракции. Несмотря на это, проектирование зачастую ориентировано на формализованное описание предметной области, а не на требования к определенному модулю (типичный *a posteriori*-подход).

2). Полностью автоматизированная процедура трансляции логической модели базы данных в физическую модель не обеспечит полный набор ограничений на целостность данных, специфичный для конкретной реализации СУБД и дополнительных ограничений контекста. Так как изменить эту ситуацию в принципе невозможно, мы хотим только указать на трудоемкость этого этапа работы.

3). Перепроектирование модели при несоответствии требованиям или их изменениях обходятся дорого.

3. Интерфейс и схемы запросов

Конолли Т. и Бегг К. отмечают в [3]: «… любое приложение базы данных всегда полезно рассматривать с более широкой точки зрения – как разработку определенного компонента всей информационной системы организации в целом». И, далее: «Необходимо убедиться, что все функциональные возможности, предусмотренные в спецификациях требований пользователей, обеспечиваются пользовательским интерфейсом соответствующих приложений. Это относится как к проектированию прикладных программ доступа к информации в базе данных, так и к проектированию транзакций, т. е. проектированию методов доступа к базе данных».

Доступ к базе данных реализуется методами классов сервера приложений. Такие классы или их методы иногда называют «шлюзом».

В данной работе предлагается рассматривать БД как монолитный компонент программного обеспечения, вне зависимости от того, где размещен код, реализующий его интерфейс.

Напомним некоторые из классических требований к модулям в компонентных технологиях [1, 2, 6]:

– компоненты должны быть «черными ящиками», скрывающими от пользователя подробности реализации;

– интерфейс компонента должен предоставлять всю информацию, которая может понадобиться пользователю;

– более того, эта информация должна быть только тем, что может потребовать пользователь, - другими словами, ее должно быть необходимое и достаточное количество;

– интерфейс никогда не изменяется;

– спецификация компонента есть спецификация его интерфейса. Она должна содержать точное определение операций, контекстных зависимостей, и ничего больше.

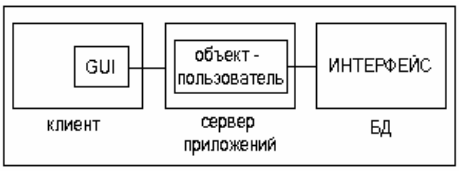

Рис. 1. Взаимодействие компонентов системы

Если рассмотреть базу данных как компонент программного обеспечения, то ее спецификация есть спецификация интерфейса компонента. Такой интерфейс можно формально задать конечным числом схем запросов на языках определения и манипулирования данными. Совокупность схем запросов может быть создана на основе спецификации требований к базе данных и рассматриваться как интерфейс компонента.

При конструировании модели базы данных (на любом уровне) эти схемы можно явно взять «за основу», как каркас формальной спецификации. Слово «каркас» используется здесь только потому, что для обеспечения целостности данных уровнем абстракции ниже, к ограничениям, задаваемым интерфейсом, должны быть добавлены ограничения контекста.

Тогда выполнение запроса, сформированного по одной из схем в клиентской части приложения (обычно, с помощью средств графического интерфейса), будет «корректным по построению» – невозможно будет сформировать «некорректный» запрос по одной из этих схем и отпадет необходимость в его верификации (построение, явно управляемое спецификацией).

Получение модели данных автоматически – одна из целей наших исследований. Спецификации, с помощью которых можно сгенерировать решение (или часть его), называют *выполнимыми*.

4. Возможная реализация

Процесс создания спецификации к модулю можно разбить на 2 этапа:

– получение формальной спецификации из спецификации на естественном языке;

– преобразование формальной спецификации в выполнимую.

Эти вопросы обсуждаются в [7, 9, 10].

Фундаментальная связь между программой P и ее спецификацией S есть корректность P по отношению к S. В логическом программировании эта связь особенно сильна [7, 8, 9, 12, 13], так как логика используется для описания как спецификации, так и программ. Логические программы могут рассматриваться как выполнимые спецификации для самих себя.

В [7] Р. Ковальски утверждает, что, когда спецификация *полностью* определяет отношения, которые должны быть вычислены, – не существует синтаксического различия между спецификацией и программой. Поэтому все отношения, определенные такими (полными) спецификациями, - выполнимы. Единственное различие между полной спецификацией и программой – эффективность выполнения. Стоит отметить, что существуют задачи, для которых корректность важнее скорости выполнения.

Логические языки программирования могут использоваться как языки определения и манипулирования данными. Проиллюстрируем выразительность формул «клауз Хорна» - формул первопорядковой логики в клаузальной форме, являющихся основой большинства языков логического программирования. Для пользователя языка, конструкции делятся на *правила* и *факты*.

Определение данных:

relation(x1, x2, …, xn).

relation(y1, y2, …, yn).

– занесение информации в базу данных с помощью отношения *relation* (факты).

relation(X1, X2, …, Xn) :- X1>0, …, Xn>0. – правило, с помощью которого задаются ограничения на переменные формулы.

Транзакция извлечения:

relation(x1, .., Xk, ..xn). – схема запроса. Одна или более переменных в верхнем регистре выделяют набор термов в формуле. С точки зрения пользователя – это объекты или свойства объектов, информацию о которых требуется получить. Результатом запроса будут все конкретизации **Xk** в модели **M**, при которых формула ***relation(x1, .., Xk, ..xn)*** истинна.

Возможность использовать рекурсию позволяет строить сложные запросы. Отметим, что сведения о реализации, такие как структура, невидимы для пользователя. На этой идее И. Братко в [8] основывает подходы к абстрагированию данных с помощью языка Prolog. Программист сосредоточивает внимание на объектах и отношениях между ними.

Хорошим средством для реализации идеи п. 3 является язык Datalog [11, 12] – логический язык программирования, позволяющий выражать сложные запросы к базе данных. Синтаксически Datalog является подмножеством языка Prolog, но имеет ряд отличий, связанных с ориентацией на работу с реляционной базой данных. В частности, встроенная проверка формулы на «безопасность» сразу снимает много вопросов, касающихся построения корректных схем. Другое важное отличие Datalog от Prolog – нечувствительность к порядку правил и фактов. Для пользователей это означает устранение зависимости декларативной семантики от процедурной семантики. Простота языка и значительная часть мощи, унаследованной от языка Prolog, делает Datalog декларативным средством спецификации, определения и манипулирования данными. Для задач быстрого прототипирования средства Prolog/Datalog, по всей видимости, не имеют равных.

Существуют [14, 15] реализации трансляции Datalog → SQL (с оптимизирующими преобразованиями).

При изменениях требований в циклических схемах разработки ограничения на целостность данных, «распространяются» от спецификации в модель. Как подмечено в [16], модульность убывает «от аксиом к теоремам». Использование логического языка одновременно в качестве языков спецификации и запросов имеет здесь преимущество.

5. Возможные методы синтеза решения

Трансляция спецификации в модель базы данных является целью данной работы. Для этого потребуется формализовать задачу, определить требования к языкам и критерий соответствия требованиям.

Логические языки программирования позволяют также получать решение задачи как трассировку пути достижения цели. Однако такой подход приведёт к большому перебору вариантов при поиске доказательства. Поэтому реализации автоматического метода доказательства используют эвристики. Вообще привлечение эвристических методов поиска решений, таких как поиск по критерию, деревья решений, генетическое программирование, нейронные сети, уже практикуется в проектировании [17, 18].

6. Направления исследований

В первую очередь дальнейшая работа будет направлена на

– формализацию постановки задачи;

– определение требований к языкам спецификации и запросов;

– определение критерия соответствия модуля требованиям;

– исследование возможной степени автоматизации задач.

В работах [19 – 21] построен формальный аппарат для моделирования данных на основе алгебраических структур. В результате построена математическая модель данных. Эта модель имеет два

уровня – структурный и семантический. Развит подход к проверке корректности моделей данных. Были приведены и доказаны алгоритмы проверки корректности концептуальной модели предметной области. Это дает возможность программной реализации заданного метода в форме CASE-средства концептуального моделирования.

Мы ставим задачу исследования возможности трансляции моделей, представленных формализмами из [19 – 21] в языки определения данных и обратно.

Список литературы

1. Lau K.K. and Ornaghi M.. A formal approach to software component specification // In D. Giannakopoulou, G. Leavens, and M. Sitaraman, editors, Proceedings of Specification and Verification of Component-based Systems Workshop at OOPSLA. – 2001.

2. Lau K.K. Component certification and system prediction: Is there a role for formality? // I. Crnkovic, H. Schmidt, J. Stafford, and K. Wallnau, editors, Proceedings of the Fourth ICSE Workshop on Component-based Software Engineering,. IEEE Computer Society Press. – 2001. –-P. 80-83.

3. Конолли Т., Бегг К. Базы данных: проектирование, реализация и сопровождение – М: Вильямс, 2003. – 1440 с.

4. Ауэр К., Миллер Р. Экстремальное программирование: постановка процесса. С первых шагов и до победного конца. – С.-Пб.: Питер, 2004. – 368 с.

5. Арлоу Д., Нейштадт А. UML 2 и Унифицированный процесс: практический объектно-ориентированный анализ и проектирование, 2-е издание. – С.-Пб.: Символ, 2007. – 624 с.

6. Роджерсон Д. Основы COM. – М.: Русская редакция Channel Trading Ltd, 1997. – 228 с.

7. Kowalski, R. The Relationship between Logic Programming and Logic Specification // In Phil. Trans. R. Soc. Lond. A. – 1984. – Vol 312. – P. 345-361.

8. Братко И. Алгоритмы искусственного интеллекта на языке PROLOG, 3-е издание. – М.,С.-Пб, К. Вильямс , 2004. – 640 с.

9. Lau K.-K. and Ornaghi M. Forms of logic specifications: A preliminary study. // In J. Gallagher, editor, Proc. OPSTR'96 , LNCS 1207, Springer-Verlag. – 1997. – P. 295-312.

10. Deville, Y., Lau, K., Logic Program Synthesis, // Journal of Logic Programming. – 1994. – P. 321-350.

11. Herv´e Gallaire and Jack Minker. Logic and Databases. – Plenum Press, New York, 1978. – 460 p.

12. Чери С., Готлоб Г, Танка Л. Логическое программирование и базы данных. – М.: Мир, 1992. – 352 с.

13. Lau K.-K. and Ornaghi M. The relationship between logic programs and specifications: The subset example revisited. J. of Logic Programming 30(3):239--257, March 1997.

14. Elnar Hajiyev, Mathieu Verbaere, Oege de Moor. CodeQuest: Scalable Source Code Queries with Datalog // Dave Thomas (editor), 20th European Conference on Object-Oriented Conference, Nantes, France, July 2006. – P. 2-27.

15. Kemal Koymen. A datalog interface for SQL (abstract). // CSC '90: Proceedings of the 1990 ACM annual conference on Cooperation, page 422, New York, NY, USA, 1990. ACM Press.

16. Хофштадтер Д. Гедель, Эшер, Бах: эта бесконечная гирлянда. – Самара: Издательский Дом «Бахрах-М», 2001. – 752 с.

17. Джонс Дж. К., Методы проектирования. – М.: Мир, 1986. – 326 с.

18. Li, Hao and Brinkley, James F and Gennari, John H. Semi-automatic Database Design for Neuroscience Experiment Management Systems // Proceedings, MedInfo. San Francisco, CA. – 2004. – P. 1717

19. Жолткевич Г.Н., Семенова Т.В. К проблеме формализации концептуального моделирования информационных систем // Вісник Харківського національного університету. Серія «Математичне моделювання. Інформаційні технології. Автоматизовані системи управління». – 2003. – № 605. – С. 33-42.

20. Семенова Т.В. Морфизмы полусхем и их приложения // Вісник Харківського національного університету. Серія «Математичне моделювання. Інформаційні технології. Автоматизовані системи управління». – 2005. – № 703. – С. 198-206.

21. Жолткевич Г.Н., Федорченко К.А. Об одном классе концептуальных моделей // Вестник НТУ «ХПИ»: Сборник научных трудов. Тематический выпуск «Системный анализ, управление и информацион технологии». – Х.: НТУ «ХПИ». – 2006. – № 19. – С. 51-56.

Поступила в редколлегию 28.02.2008

Рецензент: д-р тезн. наук, проф. О.Е. Федорович, Национальный аэрокосмический университет им. Н.Е. Жуковского «ХАИ», Харьков.

ЗДІЙСНИМІ СПЕЦИФІКАЦІЇ В ПРОЕКТУВАННІ БАЗ ДАНИХ

Тарек Юсеф Баді Біштаві, Жолткевич Г.М., Соляник Ю.В.

Запропонована постановка задачі реалізації підходу до проектування компонентів програмного забезпечення, відомого як a priori reasoning в застосуванні до баз даних. Основна ідея цього підходу полягає в явному використанні специфікації як інструкції до збірки компоненту. Розглянуті проблеми, що виникають на практиці, переваги можливої реалізації ідеї, підходи до синтезу (генерації) проектного рішення. Перераховані завдання для подальшої роботи.

Ключові слова: *проектування програмного забезпечення, здійснимі специфікації, верифікація, бази даних, генерація коду, компонентні технології, мови запитів.*

EXECUTABLE SPECIFICATIONS ARE IN PLANNING OF DATABASES

Tarek Yusef Badi Bishtavi, Zholtkevich G.N., Solyanik Yu.V.

Raising of task of realization of approach is offered to planning of components of software, known as and priori reasoning in application to the bases of information. The basic idea of this approach consists in the obvious use of specification as instruction to assembling of component. Problems, arising up in practice, advantages of possible realization of idea, approaches to the synthesis (generations) of project decision, are considered. Tasks are transferred for further work.

Keywords: *planning of software, executable specifications, verification, databases, code generation, technologies of components, languages of queries.*