Таблица 3

Матрица оптимальной программы обслуживания

Величина интервала ТО	Номер средства комплекса жизнеобеспечения здания				
	Средство 1	Средство 2	Средство 3	Средство 4	Средство 5
10	0	0	1	0	0
20	0	0	0	0	0
30	0	0	0	0	0
40	1	0	0	0	0
50	0	1	0	1	0
60	0	0	0	0	1

ской эффективности КСЖ имеют значения: C_{max} = 1938,64 усл. ед.; C_{min} = –489356,17 усл. ед.

Сравнительные результаты расчетов для всех трех последовательно решенных в соответствии с описанным алгоритмом задач в относительной и натуральных шкалах измерений двухкритериальной процедуры синтеза оптимальной программы обслуживания КСЖ приведены в табл. 2.

Оптимальная матрица $\boldsymbol{\delta}_3^*$, определяющая программу обслуживания средств КСЖ, представлена в табл. 3.

Таким образом, в статье приведено решение задачи синтеза программы обслуживания по фактическому техническому состоянию средств КСЖ зданий, оптимальной по двум критериям. В качестве исходных данных для моделирования расчетов использованы показатели, которые нетрудно получить при эксплуатации объектов строительства. Это существенно повышает прикладной характер полученных выводов. Результаты решения задачи могут применяться в структурах, выполняющих функции по организации и осуществлению технической эксплуатации зданий и сооружений.

СПИСОК ЛИТЕРАТУРЫ

1. Надежность в технике. Термины и определения [Текст]/ГОСТ 27.002-89.

2. Экономика строительства [Текст]/Под ред. И.С. Степанова. –М.: Юрайт, 1997.–С. 416.

3. **Зеленцов, В.А.** Надежность, живучесть и техническое обслуживание сетей связи [Текст]/В.А. Зеленцов, А.А. Гагин. –Л.: МО, 1991.–С. 169.

4. **Рогонский, В.А.** Эксплуатационная надежность зданий и сооружений [Текст]/В.А. Рогонский, А.И. Кострич, В.Ф. Шеряков [и др.]. –СПб.: ОАО Изд-во «Стройиздат СПб», 2004.– С. 272.

5. **Герцбах, И.В.** Модели профилактики (Теоретические основы планирования профилактических работ) [Текст]/ И.В. Герцбах. –М.: Сов. радио, 1969. –С. 216.

6. Менеджмент риска. Метод структурной схемы надежности [Текст]/ГОСТ Р 51901.14-2005.

7. **Юдин, Д.Б.** Вычислительные методы теории принятия решений [Текст] / Д.Б. Юдин. –М.: Наука, 1989. –320 с.

УДК: 004.4'244, 004.4'236

Е.В. Филичев, С.М. Устинов

РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЙ НА ОСНОВЕ МОДЕЛЕЙ ВЫСОКОГО УРОВНЯ АБСТРАКЦИИ

Веб-приложения занимают большую часть рынка программного обеспечения (ПО). С развитием веб-технологий функции, раньше реализуемые лишь с использованием ПО, устанавливаемого на компьютеры клиента, становятся доступны и реализуемы с помощью веб-приложений, которые предъявляют значительно меньшие требования к оборудованию ра-

бочего места и не нуждаются в установке. Многие организации полностью отказываются от «толстых» клиентов, переходя на веб-интерфейсы, что позволяет значительно сократить расходы на сопровождение программного обеспечения и повысить эффективность работы. Задача администрирования целого парка компьютеров с установленным ПО сводится к администрированию одного веб-сервера со всеми приложениями. Естественным образом повышаются требования к самим веб-приложениям, возрастает их сложность и, как следствие, сложность их разработки. Вместе с тем, актуальность приобретает задача модификации веб-приложения, наращивания функциональных возможностей, изменения бизнес-логики для того, чтобы удовлетворять постоянно изменяющимся требованиям.

Один из наиболее эффективных способов снижения трудоемкости разработки приложений вообще и веб-приложений в частности – использование моделей высокого уровня абстракции и механизмов генерации программного кода приложения по этим моделям. Следует отметить, что данная задача является нетривиальной и до сих пор нет ее универсального решения. Существует несколько подходов для упрощения этой задачи, среди которых можно выделить использование метамоделирования как наиболее перспективный. В данной статье предлагается один из способов использования моделей высокого уровня абстракции для разработки веб-приложений.

Развитие техник моделирования

Первые попытки сократить дистанцию между моделью и кодом целевой системы были предприняты еще в 1970-х гг., после формулирования П. Ченом в 1976 г. [1] подхода сущностей-отношений. Несмотря на некоторые недостатки в выразительных возможностях, подход можно признать успешным, т. к. результатом его появления стало последующее создание инструментальных систем, способных генерировать программный код по моделям, среди которых можно отметить RISE, ErWin, DeZign и др. Подход сущностей отношений показал, что при использовании достаточно формализованной модели задача генерации программного кода по ней является выполнимой.

Затем последовала техника IDEF1x, обобщившая опыт, полученный при разработке таких техник моделирования, как IDEF1, сущностей-отношений, реляционного подхода Э. Кодда, модели Объектов-Ролей Д. Найсена [2]. В IDEF1X была введена поддержка моделирования логических типов данных с использованием классификационной структуры или конструкции обобщения/специализации, отношения категоризации, представляющих собой взаимно исключающие подмножества общей сущности, а также понятие ключа. Благодаря всему этому, IDEF1X на момент появления являлась самой передовой техникой для определения логического дизайна баз данных и приложений и физического дизайна реализации базы данных, кроме того, она получила поддержку в различных программных продуктах (ER/Studio, ERwin и т. д.). Поддержка выражалась в возможности генерировать скрипты DDL для создания базы данных по диаграммам IDEF1X. Если бы уровень абстракции модели был выше, то, возможно, такого успеха техника IDEF1X не сумела бы достичь.

Следующим важным этапом развития моделирования можно считать создание группы объектного моделирования OMG и разработку универсального языка моделирования UML. UML объединил в себе идеи, использованные при разработке техник OOSE (Object-Oriented Software Engineering) Буча и Якобсона [3], техники Д. Рамбо [4], Шлера и Меллора [5]. Результатом сотрудничества Г. Буча из корпорации Rational Software, И. Якобсона из Objectory и Д. Рамбо из General Electric в 1997 г. стал UML 1.0, хорошо определенный, выразительный, мощный, применимый в широком спектре проблем и областей язык моделирования. Он был предложен для стандартизации группе Объектного Моделирования (OMG) в ответ на ее запрос о едином стандартном языке моделирования. Вскоре был очерчен круг семантических задач для формализации новой спецификации UML с целью его интеграции с другими решениями для стандартизации.

В актуальный стандарт UML версии 2.3 помимо базовых диаграмм входят важнейшие механизмы расширения, такие, как стереотипы и ограничения. В рамках механизма ограничений создан целый язык, язык объектных ограничений (OCL) [6], а стереотипы расширяют базовый словарь UML, позволяя определять новые элементы, наследованные от существующих, и, таким образом, создавать метамодель для нового языка – подмножества UML, который помог бы разрабатывать более специфичные модели. Механизмы

расширения позволяют приспосабливать UML под нужды конкретных проектов, а также дают самому языку возможность адаптироваться в новую программную технологию. Вместе эти возможности выводят объектно-ориентированное моделирование на новый уровень, тем самым, еще больше сокращая дистанцию между моделью и готовым программным кодом прикладной системы.

Еще одним подходом, внесшим существенный вклад в решение задачи получения полного кода по модели, стал предложенный Ю-П. Толваненом и С. Келли в 2000 г. подход разработки приложений, основанный на доменно-специфичном моделировании. К 2008 г. этот подход сформировался в законченном виде и подробно описан в работе [7]. В этом подходе предлагается понизить уровень абстракции до уровня конкретной прикладной области, что существенно снижает затраты на реализацию генератора кода целевого приложения за счет оперирования менее общими по сравнению с разработкой, управляемой моделью, понятиями.

Существующие подходы к разработке ПО на основе моделей

Таким образом, к настоящему времени существует несколько проработанных вариантов использования моделей как основных артефактов разработки, т. е. подразумевается, что, разработав модель или набор моделей, мы можем получить код готовой прикладной системы автоматически. Схематически это представлено на рис. 1.

Как видно из рисунка, никаких «ручных» действий над кодом системы проводить нет необходимости, более того, ручные действия сводят на нет все преимущества подобных подходов.

Для разработки ПО главным становится правильное определение модели. Остальное – задача программного обработчика или набора обработчиков. Различие в подходах заключается в том, как именно необходимо разрабатывать этот набор моделей, и какими характеристиками модели должны обладать.

Рис. 1. Модели как основной артефакт разработки

Среди множества характеристик моделей можно выделить две основных: язык моделирования и уровень абстракции.

На сегодняшний день группой объектного моделирования OMG в качестве наиболее перспективного подхода выбран подход, основанный на архитектуре, управляемой моделью: MDA (Model Driven Architecture). В данном подходе предлагается строить несколько уровней моделей для того, чтобы повысить итоговый уровень абстракции. Вторым по распространенности является подход Келли и Толванена [7] DSM (Domain Specific Modeling), предлагающий для повышения уровня абстракции сужать область применения языка моделирования, делая его специфичным для конкретной прикладной области. Рассмотрим оба эти подхода более подробно.

Подход MDA. В подходе MDA предлагаемая схема разработки принимает вид, показанный на рис. 2.

На рисунке представлены отношения между моделями в MDA. Первичной моделью является модель PIM (Platform independent model), не зависящая от платформы. По модели PIM с помощью некоторых механизмов и характеристик платформы автоматически создаются специфичные модели PSM (Platform specific model), которые имеют жесткую привязку к конкретной платформе. В качестве языков для описания моделей предлагается использовать UML и его профили, также указывается возможность использования языка OCL для повышения детализации моделей и обеспечения возможности осуществить автоматизированный переход, по крайней мере, в теории.

Применительно к веб-технологиям модель PIM оперирует такими понятиями, как экранное представление, статические компоненты экранного представления, динамические компоненты экранного представления, пользовательский ввод и т. д. [8]. Основываясь на информации из PIM, необходимо описать правила построения веб-приложений под конкретную платформу. Если это платформа ASP.NET, то понятия, которыми будет оперировать PSM, это ASP.NET страница, ASP.NET элемент управления и т. д. Для поддержки функциональности на минимальном уровне потребуется описать большое количество вариантов преобразований, что сводит преимущества MDA на нет. Проанализировав подход MDA, можно

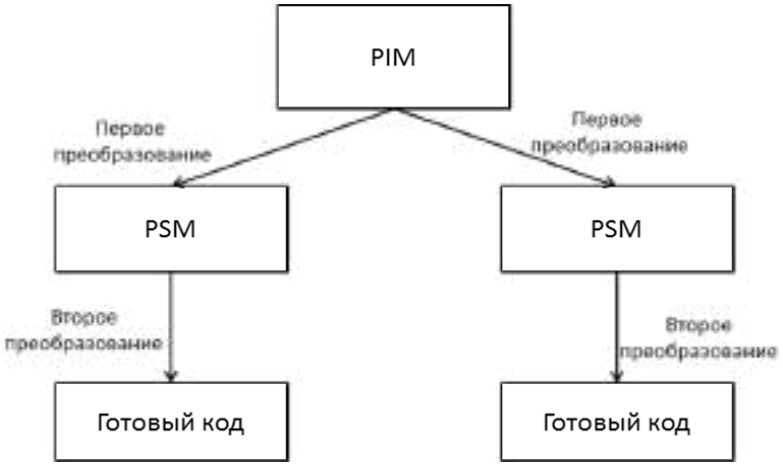

Рис. 2. Процесс разработки MDA

сделать вывод о том, что использование моделей слишком высокого уровня абстракции в качестве первичного артефакта разработки позволяет, при соблюдении условий четкой формализации языка моделирования, получить теоретически обоснованную технологию разработки ПО, однако, применительно к веб-приложениям такой подход не является целесообразным ввиду крайне высокой трудоемкости реализации механизмов перехода между моделями и готовым кодом системы.

Подход DSM. Подход DSM опирается на похожие предпосылки, однако в нем для повышения уровня абстракции предлагается использовать специальные языки для каждой конкретной прикладной области. Сужение области применения позволяет плотнее увязать язык моделирования с реальными проблемами и снижает сложность написания генератора кода, решающего задачу получения полного кода ПО по модели.

На рис. 3 представлена схема структуры DSM проекта и те ее части, которые видимы разработчикам. Данная схема показывает, что с точки зрения разработчика ПО процесс упрощается, по сравнению с MDA. Из двух уровней моделей и двух автоматизированных переходов, платформенно-независимого и платформенно-зависимого, остается лишь один, платформенно-зависимый. Подобное сокращение уровней значительно снижает затраты на разработку модулей генерации конечного кода. Структура генератора, в свою очередь, обычно скрыта от аналитиков, и ее можно сравнить с компилятором. Продолжая аналогию, преобразование моделей в исполняемый код однозначное и исключает модификацию

Рис. 3. Архитектура DSM проекта

этого кода. Эта полнота является ключевым фактором и других успешных подвижек в области языков программирования. Платформа и целевое окружение могут быть доступны как в виде кода, так и в форме интерфейсов, поддерживающих интеграцию с генератором. При необходимости, сгенерированный код может также объединяться с кодом, написанным вручную. В качестве языка моделирования конкретной прикладной области также может выступать профиль UML. Особенность подхода, заключающаяся в значительном сужении прикладной области, существенно ограничивает возможности языка моделирования. Для разработки веб-приложения в смежной области потребуется заново разрабатывать соответствующий UML профиль и генератор кода.

Формулирование требований к методике разработки веб-приложений

Таким образом, и у подхода MDA, и у подхода DSM есть серьезные недостатки, снижающие эффективность их использования при разработке веб-приложений. В данной статье предлагается методика разработки веб-приложений, учитывающая опыт разработанных техник моделирования. В качестве требований к методике выбраны:

1) высокий уровень абстракции, позволяющий избежать написания кода приложения вручную;

2) гибкость и функциональность, не ограниченная конкретной прикладной областью;

3) эффективность разработки веб-приложений.

Для удовлетворения этим требованиям методика основана на следующих положениях.

Процесс разработки должен иметь многоуровневую модель. На среднем уровне располагается платформа разработки, на верхнем уровне находятся модели высокого уровня абстракции.

В качестве платформы необходимо использовать технологию разработки определенного типа приложений в целом, т. к. это позволит избежать узкой направленности готового решения.

Количество уровней разработки необходимо выбирать исходя из соображений минимизации трудозатрат на разработку программной поддержки процесса и реализацию переходов между уровнями.

Переходы между уровнями должны выполняться в автоматическом режиме. Введение ручного анализа кода и его модификации негативно сказывается на эффективности конечного решения. Подобный подход допустимо использовать для разработки не только веб-приложений, но и приложений вообще. Далее эти положения будут рассмотрены более подробно.

Определение оптимального количества уровней моделирования

Как уже было отмечено, эффективнее всего организовывать многоуровневый процесс разработки. Обобщая опыт подходов MDA и DSM, на нижнем уровне необходимо располагать исполняемый код готового приложения, в то время как на верхнем уровне должны находиться модели высокого уровня абстракции. Для того чтобы снять ограничения узкой направленности подхода, в качестве промежуточной платформы необходимо выбирать непосредственно технологию разработки приложений, т. е. инструментальную систему, включающую в себя языковую поддержку и поддержку некоторой платформы. В данной статье рассматриваются веб-приложения, поэтому платформой разработки выступает технология ASP.NET. Промежуточный уровень является необходимым условием применения данного подхода на практике, т. к. в его отсутствие задача реализации перехода от моделей к готовому коду, минуя поддержку платформы, крайне трудоемка, и на сегодняшний день не существует инструментальных систем – примеров ее успешного решения. При необходимости, промежуточный уровень можно разделять на подуровни, с целью дальнейшего упрощения переходов от моделей к готовому приложению. Однако для веб-приложений и технологии ASP.NET это разделение не является целесообразным, т. к. трудоемкость реализации переходов в случае нескольких промежуточных уровней превышает трудоемкость реализации переходов для одного промежуточного уровня. Таким образом, эффективнее всего использовать трехуровневую схему разработки, которая схематично представлена на рис. 4.

В качестве моделей высокого уровня абстракции выступают модели веб-приложения, созданные с помощью языка моделирования высокого уровня, специфичного для платформы, но независимого от прикладной области. С помощью автоматического преобразования по такой модели создаются модели, специфичные для платформенного генератора и оперирующие понятиями, реализованными в коде на языке высокого уровня. Следующее автоматическое преобразование позволяет получить готовый исполняемый код веб-приложения ASP.NET.

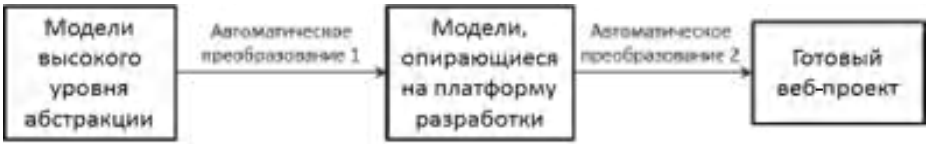

Рис. 4. Схема комбинированного подхода разработки веб-приложений

Описание платформы разработки

Платформа разработки, наряду с автоматизированными переходами, является ключевой частью предлагаемой методики. Она должна сочетать в себе интеграцию с технологией разработки, в нашем случае, ASP.NET, позволять описывать приложения с помощью программных моделей на языке программирования высокого уровня, такого, как C#, а также допускать возможность создания этих программных моделей с помощью некоторых внешних инструментов. Всем этим требованиям удовлетворяет инструментальная система генерации сайтов ASP.NET, разработанная в рамках сотрудничества ООО «Деловые Консультации» и ООО «Брэйн Системс». Разработанная платформа является многоуровневой, и содержит в себе несколько модулей и наборов объектов, реализующих соответствующие функциональные возможности. Еще одним преимуществом данной платформы является поддержка корпоративного ядра «СКАУТ», разработанного ООО «Деловые Консультации» и позволяющего описывать бизнес-логику приложений в привязке к базе данных. Наборы объектов платформы можно разделить на следующие группы.

Набор объектов, обеспечивающий базовые пользовательские функции ввода–вывода, опирающиеся на стандартные компоненты технологии ASP.NET.

Набор объектов уровня бизнес-логики; эти объекты непосредственно привязаны к ядру «СКАУТ» и реализуют его функции по созданию объектов и операциям с ними.

Сложные объекты, сочетающие в себе функции уровня бизнес-логики и функции отображения информации, удовлетворяющие требованиям заказчика. В этот набор объектов входят специальные управляющие элементы, стандартные компоненты отображения информации ASP.NET, а также другие объекты для выполнения функций, не укладывающихся в стандартные возможности компонентов ASP.NET.

Все эти объекты располагаются на соответствующих уровнях программной модели. Программная модель основана на языке высокого уровня и состоит из классов C#.

Помимо перечисленных выше объектов в платформу входят:

модуль генерации кода;

набор шаблонов для генерации кода;

модули, отвечающие за хранение информации и получение ее из выбранного хранилища (поддерживается несколько вариантов хранения созданных объектов).

Данную платформу предлагается использовать как прикладную область – домен в терминах подхода DSM. Это позволяет одновременно повысить уровень абстракции с веб-приложения под конкретную прикладную область до веб-приложения ASP.NET, сохранив невысокие расходы на разработку модулей генерации кода.

Интеграция с технологией разработки ASP.NET заключается в том, что по программной модели веб-приложения в терминах платформы строится готовое веб-приложение ASP.NET, со страницами aspx, библиотеками, содержащими реализацию бизнес-функций и связи с базой данных, а также с функциями ядра «СКАУТ», которые позволяют более гибко реализовывать бизнес-логику веб-приложения.

Платформа опирается на технологии Microsoft.NET, поэтому ее программная модель допускает создание объектов с помощью скриптов Windows PowerShell, что и было использовано при реализации перехода от моделей высокого уровня абстракции к программной модели в терминах платформы.

Модели верхнего уровня и реализация автоматических преобразований

Данный подход предлагается применять в среде разработки MS Visual Studio 2010 Ultimate, поэтому естественным выглядит решение использовать функциональные возможности моделирования, предоставляемые пакетом дополнения Visualization and Modeling Feature Pack. Это позволяет получить поддержку языка моделирования высокого уровня, такого, как UML, вместе с мощным механизмом его расширения в

виде профилей, а также доступ на программном уровне к разработанным моделям, что помогает анализировать эти модели и создает основу для реализации переходов в платформенные модели. Для разработки моделей верхнего уровня используются возможности метамоделирования, предоставляемые Microsoft Visualization & Modeling SDK, в частности, включающие в себя механизмы стереотипов UML.

Для автоматического преобразования 1 предлагается использовать шаблоны t4 для генерации текста. Таким образом, сначала в терминах разработанного профиля UML создаются модели высокого уровня абстракции, не привязанные к конкретной прикладной области, но содержащие связь с платформой разработки. Создание моделей осуществляется в рамках проекта моделирования в Microsoft Visual Studio 2010. Созданные в этом проекте диаграммы затем анализируются в рамках другого проекта Visual Studio, включающего в себя поддержку шаблонов генерации текста t4 templates и ссылку на библиотеку Microsoft. VisualStudio.ArchitectureTools.Extensibility.dll. Реализация преобразования 1 осуществляется с помощью названных выше шаблонов путем анализа разработанных в проекте моделирования диаграмм и написания на их основе скриптов PowerShell, содержащих необходимые команды для воспроизведения модели веб-приложения в программной модели платформы генерации сайтов. Функциональность получения по моделям платформы готового кода проекта реализована в механизмах платформы. Таким образом, предлагаемая методика позволяет разрабатывать веб-проекты для любой прикладной области, используя в качестве первичного артефакта разработки модели высокого уровня абстракции. С помощью реализованных автоматических преобразований возможно получение полного кода веб-приложения по таким моделям.

Предложенную в данной статье методику можно формализовать как последовательность следующих этапов.

1. Определение функциональных требований к веб-приложению.

2. Определение необходимых уровней расположения бизнес-логики приложения. Некоторые бизнес-функции удобнее реализовывать серверными компонентами ASP.NET, для других эффективнее использовать механизмы, предоставляемые базой данных.

3. В случае, если существующих компонентов платформы не хватает для реализации требуемых бизнес-функций, разработка новых компонентов, реализующих требуемые функции.

4. Доработка профиля UML, включение в него появившихся на третьем этапе компонентов.

5. Построение наглядных моделей веб-приложения в профиле UML, согласование с заказчиком.

6. Автоматизированные переходы от построенных моделей в готовый к публикации на сервере код веб-приложения.

Предполагая определенные затраты на внедрение методики, данный подход позволяет значительно сократить затраты на разработку веб-приложений в дальнейшем. Чем больше приложений реализовано, тем более богатыми возможностями обладает платформа, и тем меньше средств понадобится на разработку последующих приложений.

Следует отметить, что предложенный подход был опробован и показал хорошие практические результаты в следующих программных продуктах: «Скаут–Заявки для Сбербанка»; «Скаут–Навигатор»; «Телемикс».

Для выполнения поставленных в ТЗ задач по каждому из этих проектов разработаны спецификации необходимых пользовательских элементов управления, произведена разработка и добавление этих элементов в библиотеки платформы, описаны необходимые классы для языка моделирования и механизмы переходов от моделей, содержащих новые элементы, в готовый серверный код.

СПИСОК ЛИТЕРАТУРЫ

1. **Chen, P.** The Entity-Relationship Approach to Logical Data Base Design [Текст] / P. Chen. –Wellesley, MA: Q.E.D. Information Sciences, Inc. –1977.

2. **Nijssen, G.M.** Current Issues in Conceptual Schema Concepts [Текст] / G.M. Nijssen (ed.) // Proc. 1977 IFIP Working Conf. on Modelling in Data Base Management Systems, Nice, France. –Amsterdam: North-Holland Publishing, 1977. –P. 31–66.

3. **Jacobson, I.** Object Oriented Software Engineering: A Use Case Driven Approach [Текст]/ I. Jacobson; Revised ed. –Addison-Wesley Professional, 1992.

4. **Rumbaugh, J.** Object-Oriented Modeling and De-

sign. [Текст]/ J. Rumbaugh. –N.J.:–Prentice Hall, Englewood Cliffs, 1991.

5. **Shlaer, S.** Object-Oriented Systems Analysis: Modeling The Real World in Data [Текст] / S. Shlaer, S.J. Mellor. –N.J.:Prentice Hall, Englewood Cliffs, 1988.

6. **Warmer, J.** Object Constraint Language, The Getting Your Models Ready for MDA [Текст] / J. Warmer, A. Kleppe; 2 ed. –Addison Wesley, 2003. –240 с.

7. **Kelly, S.** Domain Specific Modeling. Enabling Full Code Generation [Текст] / S. Kelly, J-P. Tolvanen. –Wiley, 2008. –427 с.

8. **Conallen, J.** Building Web Applications with UML [Текст] / J. Connalen; 2 ed. –Addison Wesley, 2002. – 496 с.

УДК 004.651.4

А.М. Бородин, С.В. Поршнев

АНАЛИТИЧЕСКИЕ СПОСОБЫ ОЦЕНКИ ЭФФЕКТИВНОСТИ ПРИМЕНЕНИЯ ПРОСТРАНСТВЕННЫХ ИНДЕКСОВ В OLAP-СИСТЕМАХ

В работах [1, 2] показано, что выполнение агрегирующих запросов по наборам аналитических данных в OLAP-системах при использовании пространственных индексов оказывается весьма эффективным, в первую очередь, с точки зрения скорости выполнения запросов. Помимо этого оказалось, что выбранный подход позволяет при изменении набора данных ограничиться только частичным перестроением индекса. В то же время оценки эффективности выполнения агрегирующих запросов по аналитическим данным, приведенные в [1, 2], были получены экспериментально, поэтому зависят от конкретных наборов аналитических данных. В этой ситуации для обоснования возможности экстраполяции данных оценок на аналитические данные OLAP-систем произвольной структуры разработка аналитических методов оценки эффективности пространственных индексов является актуальной.

В статье изложено решение задачи создания аналитических способов оценки эффективности метода индексирования данных, разработанного в [1, 2], в т. ч. R*-дерева, Ra*-дерева и KDB-дерева.

Способ оценки ожидаемой эффективности R-дерева

R-дерево предназначено для индексирования набора пространственных объектов. При этом основной характеристикой объекта является многомерный параллелепипед (параллелотоп) минимально возможного объема, описанный вокруг объекта, ребра которого параллельны осям координат (minimum bounding rectangle − MBR)*. MBR-параллелотоп описывается заданием координат его вершин.

Здесь и далее мы считаем, что исходный OLAP-куб отображен на единичный OLAP-куб с помощью преобразования

$$\tilde{x}_k^{\langle i\rangle} = \frac{x_k^{\langle i\rangle} - x_{\min}^{\langle i\rangle}}{x_{\max}^{\langle i\rangle} - x_{\min}^{\langle i\rangle}},$$

где i − номер измерения куба; k − номер отсчета в i-м измерении; $x_{\min}^{\langle i\rangle}$ − минимальное значение координат i-го вектора (измерения); $x_{\max}^{\langle i\rangle}$ − максимальное значение координат i-го вектора (измерения). Использование единичного OLAP-куба, как очевидно, снимает проблему отличий в единицах измерения величин, расположенных в соответствующих измерениях исходного OLAP-куба.

Для получения аналитической оценки эффективности R-дерева докажем несколько промежуточных утверждений.

Назовем отрезок $s=[s_0,s_1]$ равномерно распределенным, если $s_0,s_1\in[0,1], s_0\le s_1, s_1=s_0+|s|$ и s_0 − случайное число с равномерным законом распределения на интервале $[0,1-|s|]$.

Лемма 1. Вероятность p пересечения отрезков $s=[s_0,s_1]$ и $q=[q_0,q_1]$, таких, что $|s|,|q|$ − заданные числа, s и q – равномерно распределенные отрезки, и $|q|+|s|\le 1$. Тогда p вычисляется по формуле:

* Отметим, что аббревиатура MBR в ряде статей используется также и для обозначения набора MBR-параллелотопов.