Взаимодействие объектов в объектно-ориентированной среде выполнения

Б. Л. Романов, Д. Я. Слободецкий

Институт системного анализа Российской академии наук,
Россия, 117312 Москва, пр. 60-летия Октября, 9

Рассматриваются вопросы взаимодействия объектов, связанных друг с другом разнообразными отношениями целостности. Предлагаются алгоритмы, позволяющие поддерживать отношения целостности между объектами.

Введение

Высокие требования к безопасности и надежности информационных систем привели к тому, что подавляющее большинство современных решений имеют клиент-серверную архитектуру. Это позволяет ограничить доступ к информации, сосредоточив все функции непосредственного манипулирования данными в одном месте. В качестве серверного ПО, как правило, используются промышленные системы управления базами данных (СУБД), такие как MS SQL Server, Oracle и др. [5, 8], и, как следствие, для описания модели данных, бизнес-правил и бизнес-процедур, обеспечивающих целостность и непротиворечивость данных систем, используются средства, предлагаемые СУБД. Большинство современных СУБД являются реляционными [4], поэтому такими средствами являются таблицы (отношения), связи, триггеры и хранимые процедуры. Одним из недостатков реляционной модели является тот факт, что получаемая в результате проектирования системы совокупность таблиц очень часто является крайне сложной для понимания и анализа. Это приводит к тому, что при росте сложности системы отслеживать все необходимые места (триггеры и процедуры) для внесения изменений, становится крайне сложно, процесс развития системы становится неуправляемым или вовсе останавливается.

Альтернативой реляционной модели данных является объектно-ориентированный подход. При нем программа представляет собой описание объектов, их свойств (или атрибутов), совокупностей (классов), отношений между ними, способов их взаимодействия и операций над объектами (методов). Несомненным преимуществом данного подхода является концептуальная близость к предметной области произвольной структуры и назначения. Механизм наследования атрибутов и методов позволяет строить производные понятия на основе базовых и таким образом создавать модель сколь угодно сложной предметной области с заданными свойствами, сохраняя возможность анализа и внесения необходимых изменений. Объекты, классы и методы могут быть полиморфными, что делает реализованное программное обеспечение более гибким и универсальным [7].

Несмотря на преимущества объектного подхода, объектные СУБД не получили широкого распространения [2, 6]. Вместо этого получили распространение гибридные решения — объектно-реляционные СУБД, которые реализуют некоторые объектно-ориентированные принципы работы с данными, при этом хранение и представление данных выполняется с использованием реляционной модели.

1. Обзор архитектурных решений

Рассмотрим типичную задачу разработки информационной системы, обеспечивающей одновременный доступ к информации множества пользователей. Для работы с данными, хранящимися в информационной системе, пользователи используют клиентские автоматизированные рабочие места (АРМ) (специализированные приложения или веб-браузер); информация об объектах системы хранится в базе данных. Есть несколько типичных архитектурных решения, применяемых для такого рода систем.

1.1. Непосредственная работа с СУБД

В качестве клиентского АРМ используется специализированное приложение, которое обращается непосредственно к СУБД. Такая схема применяется обычно для небольших прикладных систем, поскольку, как было отмечено ранее, использование такого подхода для сложных структур данных затрудняет поддержку и развитие системы.

1.2. Использование промежуточного слоя

В качестве клиентского АРМ используется специализированное приложение или браузер, которые обращаются не непосредственно к СУБД, а к некоторому промежуточному слою, который, в свою очередь, обращается непосредственно к СУБД. Интерфейс взаимодействия, реализуемый сервером приложений и используемый клиентским АРМ, целесообразно выполнять

в терминах предметной области информационной системы и в соответствии с принципами объектно-ориентированного подхода. Использование промежуточного слоя в качестве дополнительного звена между клиентским АРМ и СУБД дает возможность клиентскому АРМ не зависеть явно от используемой СУБД, увеличивает возможности развития, интеграции и масштабирования всей системы. Можно сказать, что промежуточный слой реализует объектно-ориентированный интерфейс над средой хранения (СУБД).

Использование промежуточного слоя, реализующего объектно-ориентированный интерфейс, дает возможность использовать для описания бизнес-процедур, бизнес-правил и логических ограничений целостности средства объектно-ориентированных языков программирования. Очень часто этот подход выражается в создании клиентского интерфейса прикладного программирования, при этом вся бизнес-логика реализуется непосредственно в клиентском прикладном приложении. Однако такое решение часто не соответствует предъявляемым к современным системам требованиям по безопасности.

1.3. Использование серверного приложения

Этот подход является развитием предыдущего. Клиентский АРМ обращается к специализированному серверному приложению, которое обращается к СУБД непосредственно или с использованием промежуточного слоя. Трехуровневая архитектура дает возможность сосредоточить бизнес-логику на сервере, повышает безопасность в работе с данными, увеличивает масштабируемость. Использование объектно-ориентированного интерфейса промежуточного слоя дает возможность описывать ограничения целостности в терминах объектов, что упрощает понимание и увеличивает возможности развития системы.

2. Взаимодействие объектов

При использовании технологии клиент-сервер с промежуточным слоем на сервере ответственность за сохранение отношений целостности между различными объектами, определяемых бизнес-правилами предметной области, ложится на этот слой.

При использовании объектно-ориентированной модели для реализации взаимодействия между объектами (классами) используется событийная модель, схема работы которой представлена на рис. 1. Основная идея взаимодействия с использованием событийной модели состоит в следующем. Объект-обработчик (В) события, состояние или данные которого зависят от данных другого объекта (А), осуществляет подписку на событие изменения данных или состояния объекта — источника события. При наступлении события (изменении данных или состояния) объект-источник (А)

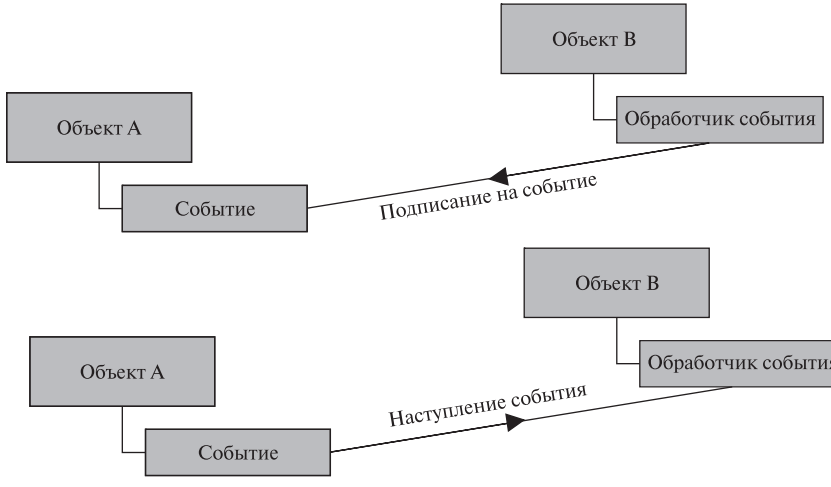

Рис. 1. Схема взаимодействия событий

генерирует событие, которое приводит к вызову и выполнению соответствующего обработчика (В). Использование событийной модели для реализации взаимодействия объектов с целью сохранения отношений целостности позволяет сделать зависимости между объектами неявными, упрощает процедуры добавления новых и удаления существующих зависимостей.

В простейшей реализации событийной модели, доступной в языках про-граммирования, событие представляется в виде объекта, хранящего адреса функций (методов объектов-обработчиков), которые необходимо вызвать при наступлении события. Этот механизм хорош, когда все объекты конструируются однократно и живут все время выполнения приложения или до их необходимости. Но при построении систем, предназначенных для одновременной работы с большим количеством взаимосвязанных объектов (например, системы электронного документооборота), объекты хранятся в СУБД и загружаются в память только при выполнении непосредственных операций с ними. Представление обработчика события как адреса функции в этом случае неприменимо, необходимы дополнительные средства, аналогичные триггерам в СУБД.

3. Подсистема управления взаимодействием объектов

При разработке подсистемы, реализующей событийную модель взаимодействия объектов в рамках промежуточного слоя на сервере, возникает необходимость в поддержке двух типов событий:

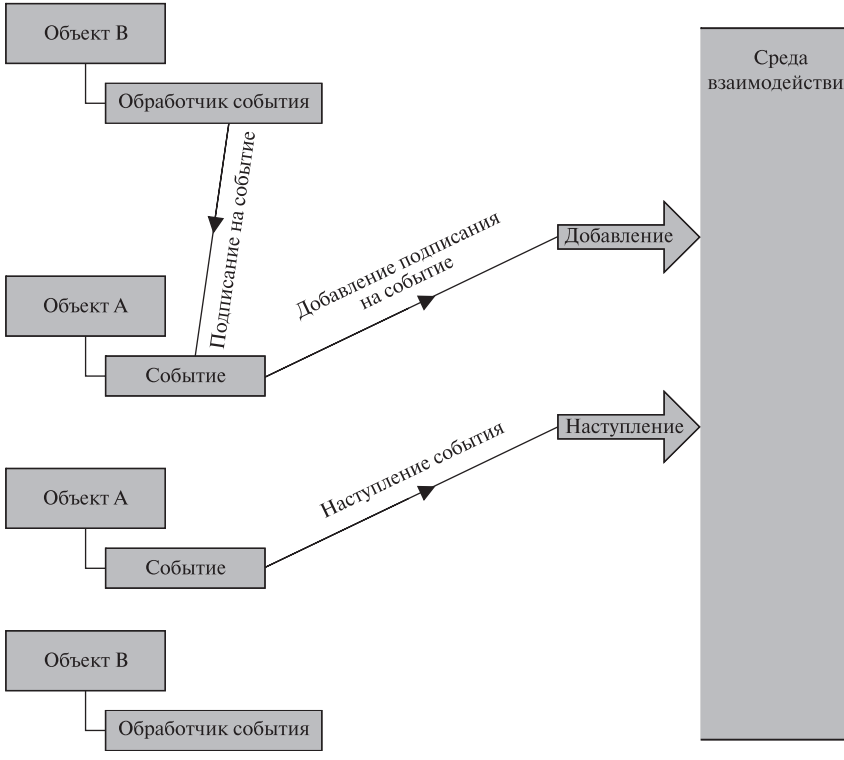

Рис. 2. Схема взаимодействия через промежуточный слой

- объектные, возникающие при выполнении действий с каким-либо объектом системы;
- временные, возникающие при наступлении определенного момента времени.

Обработчики событий также можно разделить на два типа:

- синхронные — обработка события непосредственно в момент наступления события, в той же транзакции, в которой возникло событие;
- асинхронные — обработчик события помещается в очередь и выполняется во время простоя системы (в отдельной транзакции).

Использование асинхронных обработчиков позволяет уменьшить время выполнения транзакции и, как следствие, уменьшить время реакции системы на действия пользователей. Возможно и увеличение производительности системы за счет добавления дополнительных процессов (в том числе на отдельных компьютерах) для вызова асинхронных обработчиков.

Событийная модель взаимодействия объектов промежуточного слоя реализуется с помощью *подсистемы взаимодействия объектов*, представленной на рис. 2.

Основными элементами подсистемы взаимодействия объектов являются хранилище подписок на событие и очередь асинхронных обработчиков наступивших событий.

При добавлении подписки на событие информация о нем добавляется в хранилище подписок. Каждая запись содержит информацию об объекте-источнике, событии объекта-источника, типе подписки (синхронная или асинхронная), объекте-обработчике и методе объекта обработчика, который надо вызвать для обработки события.

При наступлении события выполняется поиск в хранилище всех подписок на наступившее событие объекта-источника и, в зависимости от типа подписки, происходит либо непосредственное обращение к объекту-обработчику, либо добавление записи в очередь асинхронных обработчиков. Вызов асинхронных обработчиков выполняется во время простоя системы, что значительно повышает ее производительность.

Для реализации такой подсистемы необходимо, чтобы каждый объект, являющийся источником или обработчиком события, имел уникальный идентификатор, используя который можно при необходимости загрузить необходимый объект из базы данных в память приложения. Такой идентификатор позволяет в описании подписки на события сохранять ссылки на объекты, являющиеся источниками и обработчиками событий.

4. Реализация подсистемы управления взаимодействием для системы электронного документооборота Евфрат

Приведем пример реализации подсистемы управления взаимодействием для системы электронного документооборота (СЭД) ЕВФРАТ. Объекты бизнес-логики данной системы очень тесно связаны между собой отношениями логической целостности, для корректной работы системы необходимо сохранение этих отношений.

Подсистема реализована на платформе .NET [1] с использованием технологии .NET Reflection для динамического вызова метода класса-обработчика. В качестве среды хранения используется хранилище данных Xnika [3].

4.1. Загрузка объектов в оперативную память

Данные всех объектов хранятся в хранилище данных XNika. Для непосредственной работы с объектом его данные должны быть предварительно загружены в оперативную память. В связи с этим возникает необходимость в идентификации объектов.

Объекты СЭД ЕВФРАТ образуют иерархию, поэтому идентификатор каждого объекта имеет сложный вид и состоит из идентификаторов всех родительских объектов. Например, полный идентификатор учетной записи в адресной книге имеет вид: «addressbook/123». Идентификатору «addressbook» соответствует объект адресной книги, который, в свою очередь, выполняет загрузку учетной записи с идентификатором «123». Иерархическая идентификация позволяет сделать процедуру загрузки объектов в память простой, расширяемой и гибкой.

Для конструирования в оперативной памяти объекта с известным идентификатором рекурсивно выполняются следующие действия:

- выделение идентификатора объекта верхнего уровня;
- конструирование объекта верхнего уровня;
- передача полученному объекту оставшийся «хвост» идентификатора для конструирования подобъектов.

Данный алгоритм обеспечивает прозрачное получение любого объекта по его идентификатору и не имеет никаких преград для расширения новыми объектами.

4.2. Подписка на событие

Для обеспечения взаимодействия между объектами используется объект — описание события, содержащий в себе идентификатор события и идентификатор объекта — источника события.

Подсистема взаимодействия управляет коллекцией подписок на события. Коллекция подписок соответствует коллекции объектов хранилища Xnika. При подписке на событие информация о ней добавляется в коллекцию подписок. При снятии подписки с события, происходит удаления соответствующей записи из хранилища.

Каждая подписка включает в себя следующую информацию:

- идентификатор события (только для события объектов);
- идентификатор объекта-источника (только для события объектов);
- время наступления события (только для события времени);
- идентификатор объекта-обработчика;
- информация о методе объекта-обработчика;
- тип обработки (синхронная/асинхронная).

4.3. Возникновение события

При наступлении события объект-источник информирует об этом среду взаимодействия, вызывая специализированный метод, в который в качестве параметров передаются идентификаторы события и объекта-ис-

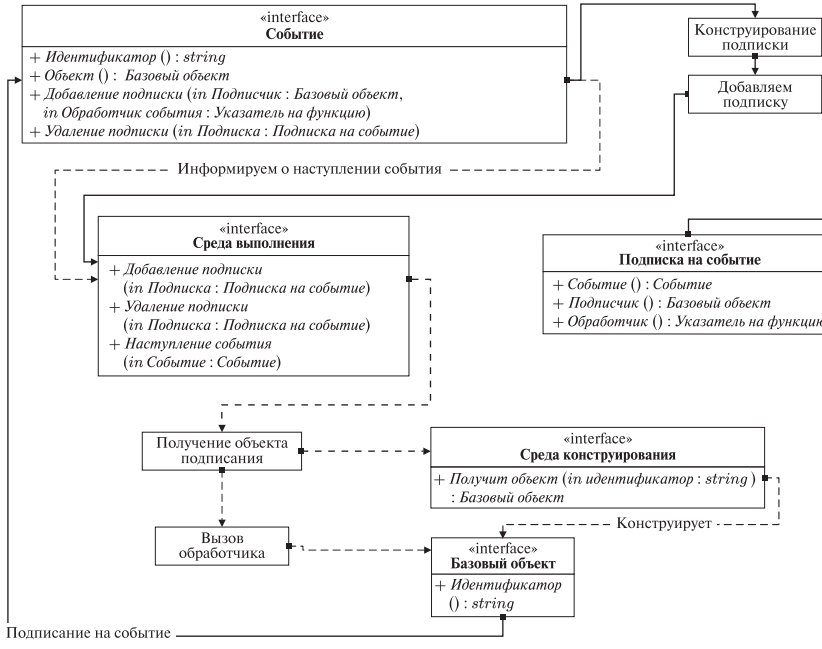

Рис. 3. Схема взаимодействия объектов

точника. Среда взаимодействия выполняет поиск в хранилище всех подписок на указанное событие указанного объекта и последовательно их обрабатывает.

Общая схема взаимодействия представлена на рис. 3, где сплошной линией отмечен этап подписания на событие, а штрихпунктирной — этап наступления события.

4.4. Вызов обработчика

Вызов обработчика события выполняется по-разному, в зависимости от типа обработки (синхронная или асинхронная). Для подписки с синхронным типом обработки немедленно конструируется объект-обработчик, у которого вызывается соответствующий событию метод. Для подписок с асинхронным типом обработки информация о событии помещается в специализированную очередь, которая обрабатывается во время простоя системы. Очередь асинхронных обработчиков хранится в хранилище данных, поэтому информация о них не теряется при завершении работы программы. При последующем запуске все асинхронные обработчики будут выполнены.

4.5. Целостность на уровне транзакций

Любое действие с объектом, приведшее к возникновению события, выполняется в рамках некоторой транзакции. Все подписки синхронного типа обрабатываются в рамках той же транзакции, что обеспечивает сохранение отношений целостности между объектами системы. Для асинхронных подписок сохранение отношений целостности достигается за счет того, что добавление в очередь асинхронных обработчиков осуществляется в исходной транзакции и очередь асинхронной обработки находится в том же хранилище, что и данные объектов. Это дает гарантию того, что информация о возникновении события не будет утеряна и обработчик будет вызван.

Заключение

При правильном проектировании связей между объектами использование описанной методологии построения взаимодействия позволяет содержать объекты в целостном состоянии. Использование асинхронных обработчиков событий позволяет уменьшить время отклика системы и расширить возможности масштабирования.

Литература

1. Microsoft. Документация по .NET Framework 3.5 // .NET Framework 3.5. Microsoft Corporation, 2009. http://msdn.microsoft.com/ru-ru/library/w0x726c2.aspx.
2. *Андреев А. М., Березкин Д. Ю.* Выбор СУБД для построения информационных систем корпоративного уровня на основе объектной парадигмы // CITForum. 2001. http://www.citforum.ru/database/articles/subd_cis.shtml.
3. *Богданов А. С., Емельянов Н. Е., Ерохин В. И., Скорняков В. А., Романов Б. Л.* НИКА-технология построения информационных систем // Организационное управление и искусственный интеллект. М.: URSS, 2003. Сборник трудов ИСА РАН.
4. *Зализняк Е.* «Три толстяка» СУБД оккупировали рынок пожизненно // CNews. 2005. 15 августа. http://www.cnews.ru/reviews/index.shtml?2005/08/15/184770
5. *Козленко Л.* Проектирование информационных систем. Ч. 5 // Interface. 2002. http://www.interface.ru/home.asp?artId = 3585.
6. *Крейг С.* Объектные СУБД пока особым спросом не пользуются // Computerworld. 1998.
7. *Мейер Б.* Объектно-ориентированное конструирование программных систем. М.: Русская редакция, 2005
8. *Свинарев С.* Рынок реляционных СУБД: стабильный рост // PCWeek. 2008.