Олег Мостовлянский

Вечный двигатель третьего рода

Неканонические размышления о бизнес-системах, или О чём стоит сначала подумать. Модели данных и бизнес-логика

Олег Анатольевич Мостовлянский

Вечный двигатель третьего рода. Неканонические размышления о бизнес-системах, или О чём стоит сначала подумать. Модели данных и бизнес-логика

http://www.litres.ru/pages/biblio_book/?art=31508736
ISBN 9785449066596

Аннотация

Автор на основе своего опыта работы предлагает перечень тем и ситуаций, над которыми желательно предварительно хорошенько подумать.

Содержание

От автора 5
НАЧАЛО 6
Тема первая: Модели данных в бизнес-системах 7
Размышление первое: БИБЛИОТЕКА 8
Конец ознакомительного фрагмента. 14

Вечный двигатель третьего рода Неканонические размышления о бизнес-системах, или О чём стоит сначала подумать. Модели данных и бизнес-логика

Олег Анатольевич Мостовлянский

© Олег Анатольевич Мостовлянский, 2018

ISBN 978-5-4490-6659-6
Создано в интеллектуальной издательской системе Ridero

От автора

«О чём задумался, детина? -»

Русская народная песня – «Вот мчится тройка почтовая»

«Таким образом, передо мной встала особая проблема – объяснить, чем же я, собственно, занимаюсь.»

Карлос Кастанеда – «Дар Орла»

Проблема передо мной встала тогда, когда я (не вполне серьёзно – ибо достаточно реально оценивал тогда и оцениваю сейчас свои способности к литературному творчеству как очень близкие к нулевым) пообещал своим дочкам изложить мои соображения по поводу того, чем же это я занимался во время своей деятельности на ниве IT.

Но слово – не воробей, и в конце концов совесть заставила заняться этим неким подобием инвентаризации своих соображений по поводу того, что же, как и исходя из чего я считал необходимым делать (хотя частенько реализовать это и не удавалось в силу, скажем так, различных иногда объективных, иногда субъективных, но всегда непреодолимых обстоятельств).Так что записанные мною размышления – это не пересказ обязательных к исполнению пунктов из учебников, а просто перечень тем и ситуаций, над которыми желательно предварительно хорошенько подумать, а не сразу кидаться ляпать по какой-либо из зазубреных схем нечто картонное по принципу – зато скорее будет, чем отчитываться.

НАЧАЛО

Что представляют собой IT-бизнес-системы. В общем виде это некий информационный скелет реального бизнес-процесса. Причём это не некое искусственное добавление типа карикатуры «Диспетчер, куда подавать груз?«на тему «автоматизации» из старого номера журнала «Крокодил». Нет смысла навешивать на грузчика рацию и телевизор, если он по-прежнему катит тачку. IT-бизнес-система должна быть частью автоматизированного бизнес-процесса, определяя его архитектуру, обеспечивая его функционирование, в том числе и управление. То есть, по аналогии с биологическими объектами, это совокупность скелета, кровеносной и нервной систем.

Однако лучше не тратить время на болтологию – оставим это тем, кто на этом зарабатывает себе на жизнь, организуя и проводя различные «школы», «семинары» и т.п., а заняться делом и подумать о «вариациях на тему»: как строятся бизнес-системы? Как это делать – хотя бы в первом приближении – *оптимально*?

Тема первая: Модели данных в бизнес-системах

По порядку. Первым делом – ибо любая бизнес-система оперирует данными – поразмышляем о структурах данных. И начнём, естественно, с конкретного примера, с варианта, кажущегося наиболее простым: простой-примитивной – на первый взгляд – библиотечной картотеки…

Размышление первое: БИБЛИОТЕКА

Нет, под библиотекой здесь будем иметь в виду не учреждение, расположенное в собственном либо арендованном здании, обременённое всевозможными расходами на персонал, коммунальные услуги и т. п., обслуживающее клиентов в читальных залах и посредством абонемента…

Займёмся только фондом библиотеки – книгохранилищем (как это называлось в доэлектронную эру) вкупе с каталогами (причём логистическая составляющая – сбор заявок, доставка литературы в залы, выдача/приёмка и т. п. – останется вне поля нашего внимания).

И представим всё это в виде структур данных.

…а представлять будем не в виде набора таблиц и относящихся к ним списков полей – а в виде диаграмм-графов, показывающих объекты и связи между ними. Атрибуты (поля), конечно же, упоминания не избегнут – но лишь те, которые призваны однозначно идентифицировать объект; и то – не в конкретно-обязательной форме, а только *концептуально*. Ибо это – универсальное представление, поддающееся реализации в любых терминах – хоть реляционных баз данных, хоть графовых… да вообще как угодно.

И будем безбожно путать термины «сущность» и «объ-

ект», «атрибут» и «поле», ибо в подобных рассуждениях это не принципиально: что таблица, что структура – всё едино.

Основные сущности

Начальный этап дизайна структур данных – определение сущностей, т. е., объектов, из которых состоит рассматриваемая система. В данном случае – библиотечный фонд.

В самом общем виде – в библиотеке хранится что-то где-то, содержащее какую-то информацию…

В самом конкретном – в комнатах стоят шкафы или стеллажи, на полках – книги, которые содержат произведения (одно или много; или же – часть) каких-либо авторов… Это – случай классической библиотеки. Более современный случай – в фонде имеется информация, хранимая в электронном (оцифрованном) виде, в виде файлов. Файлы могут находиться как на носителях типа дискет (а что? вдруг ещё остались где-нибудь), магнитных лент (это вообще раритеты), компакт-дисков (CD, DVD, BD и т. п.), а также на накопителях (т. н. жёстких дисках) серверов. Носители будут храниться как и книги, на полках (либо в ящиках) шкафов, сервера будут расположены в комнатах – так что принципиальных отличий здесь не видно.

Сделаем первый шаг.

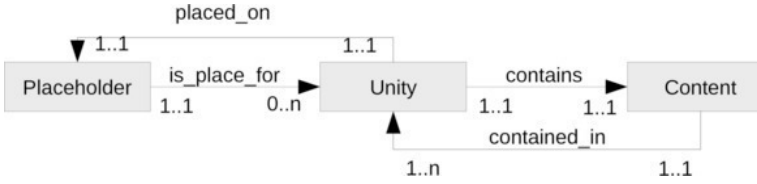
Назовём основные сущности:

– хранимая единица информации – *Unity*

– местоположение хранения этой единицы – *Placeholder*

– содержание хранимой единицы информации – *Content*

Связаны они следующим образом:



Что видно из этой диаграммы?

– В месте хранения *Placeholder* может находиться несколько хранимых единиц *Unity*, но может не быть и ни одной. Это показано связью (*relation*) *is_place_for*.

– Каждая хранимая единица *Unity* обязательно находится в одном (и не более! это вроде само собой понятно) месте хранения *Placeholder* (показано связью *placed_on*).

– Каждая хранимая единица *Unity* обязательно – поскольку речь идёт об *информации* – имеет вполне определённое содержание *Content*. Причём – только одно. Связь *contains*.

– Напротив, содержание *Content* вполне может относиться к нескольким (одной и более) хранимым единицам *Unity* (связь *contained_in*). Пример – несколько экземпляров одной книги.

Это – самое общее представление. Будем рассматривать эти сущности более детально, и в том же порядке, в котором они были декларированы: *Unity*, затем *Placeholder*, ну и на-

последок *Content*. Возможно (то есть, практически наверняка), что-то ещё добавится по пути, в процессе.

Естественно, для каждой сущности будет описан и минимальный набор обязательных – придающих ей уникальную значимость – атрибутов.

Да, вот что ещё здесь обязательно надо упомянуть: все имена сущностей (объектов) и их атрибутов (полей), а также связей между объектами – вымышленные, взятые исключительно для примера. Всякое совпадение с реальными именами чисто случайное…

И – о, Гринпис, где твой ледокол! – при рисовании схем и написании текста ни одно животное не пострадало.

Хранимая единица информации (Unity)

Итак, для начала – хранимая единица информации *Unity*. Очевидно, она должна описывать предметы, которые хранятся в фонде, учитываются как отдельные единицы и могут выдаваться абонентам. Что может храниться в библиотеке?

– Книги,
– журналы,
– газеты,
– рукописи,
– ксерокопии,
– микрофильмы,
– микрофиши,
– магнитные, оптические и прочие современные нам но-

сители,

– и т. д., и т. п. – в общем, всё, что только может содержать информацию.

Здесь надо немного остановиться. Вопрос касается файлов.

С одной стороны, файл – отдельная законченная единица информации. С другой стороны, в одном файле может содержаться несколько независимых единиц информации – например, оцифрованных книг, репродукций и т. п. И с третьей стороны – один хранимый экземпляр носителя информации (например, компакт-диск) может содержать несколько файлов. Кстати, этот компакт-диск может быть не самостоятельным – а приложением к какой-либо книге…

Остановка получается недолгая, ибо решение очевидно и просто: каждая единица информации (*Unity*) может быть в то же время и собранием (*Collection*) подобных единиц.

Вернёмся к теме – *Unity*.

Каждый экземпляр этой сущности должен представлять собой сочетание атрибутов, уникальным образом характеризующих хранимую единицу информации. На сами же отдельные атрибуты требование уникальности значений может распространяться не всегда. Если атрибут должен иметь уникальное значение, то оно обычно автоматически генерируется программным способом при создании экземпляра сущности (например, в случае целочисленных величин или весовых значений символов логично применить автоинкремент).

Если значение атрибута может быть не уникальным, то возможны случаи:

Конец ознакомительного фрагмента.

Текст предоставлен ООО «ЛитРес».

Прочитайте эту книгу целиком, купив полную легальную версию на ЛитРес.

Безопасно оплатить книгу можно банковской картой Visa, MasterCard, Maestro, со счета мобильного телефона, с платежного терминала, в салоне МТС или Связной, через PayPal, WebMoney, Яндекс.Деньги, QIWI Кошелек, бонусными картами или другим удобным Вам способом.