УДК 681.3.068

О.А. Змеев, А.Н. Моисеев

СРАВНИТЕЛЬНЫЙ АНАЛИЗ НЕКОТОРЫХ МЕТОДОВ О – R-ПРЕОБРАЗОВАНИЯ

Рассмотрены подходы к представлению объектов программной системы в реляционной базе данных (РБД). Производится сравнительный анализ с позиций скорости разработки, гибкости программной системы, времени выполнения операций над хранимыми объектами для различных моделей их представления в РБД.

1. ПОСТАНОВКА ЗАДАЧИ

Объектно-ориентированное проектирование и программирование в настоящее время является самым распространенным подходом к разработке программного обеспечения. В составе общего производственного процесса разработчики решают множество задач, связанных с различными особенностями данного подхода. Обычно различные части программной системы разрабатываются в рамках подсистем – достаточно независимых компонентов. Одной из таких подсистем является подсистема управления данными [1]. Таким образом, задача проектирования и реализации подсистемы хранения данных в объектно-ориентированных системах является актуальной.

Существует множество подходов к решению данной задачи: от применения объектно-ориентированных баз данных до создания собственных средств хранения, поддерживающих объектные типы данных. Вообще, под хранением объектов в рамках настоящей работы будем подразумевать сохранение и восстановление их состояния, то есть значений атрибутов, из источников долговременного хранения информации. В литературе по объектно-ориентированному анализу и проектированию (ООАП) эти операции называются материализацией и дематериализацией объектов.

В настоящей работе будем рассматривать задачу хранения объектов в реляционной базе данных. В этом случае выполняется так называемое О – R-преобразование, позволяющее перейти от объектной модели предметной области к реляционному способу хранения данных. Актуальность этого направления исследований уже неоднократно обсуждалась [2, 3]. В данной работе рассматривается несколько шаблонных решений этого вопроса, делается их сравнительный анализ.

Заранее оговорим, что реализация решений будет рассмотрена «в чистом виде», то есть отдельно от различных смежных вопросов, таких, как кэширование, блокировка, безопасность, а также без использования шаблонов, связанных с отложенной загрузкой.

2. ТЕСТОВЫЙ ПРИМЕР, ТЕХНОЛОГИЧЕСКАЯ БАЗА

В качестве примера реализации анализируемых подходов рассмотрим справочную информационную систему «Электронный каталог» книжного магазина. В основу положена база данных реального интернет-магазина. Полное описание структуры представлено в виде диаграммы классов предметной области (рис. 1).

База данных содержит информацию о 101 087 экземплярах товара, распределенных по 588 разделам, в том числе – 84 287 книг, 6 793 периодических изданий, 10 007 наименований кассет. База данных также содержит информацию о 5 243 издательствах и 42 857 авторах книг.

Все методы тестируются на одной технологической платформе:

- в качестве хранилища выбрана база данных формата MS Access 2000;
- все операции над базой данных выполняются посредством запросов SQL;
- выполнение запросов производится из приложения, созданного средствами Borland Delphi версии 7.0;
- измерение времени выполнения операций производилось на компьютере с процессором Pentium 3, 933 МГц, RAM 256 M, под управлением операционной системы MS Windows 2000 Server, причем каждая операция была выполнена по 100 раз. Время публикуется с точностью до сотых долей секунды.

Основные измерения будут касаться следующих операций:

- открытия всей базы данных (запуск программы);
- выборки всех данных по одному объекту «Книга» (максимальное количество атрибутов среди всех товаров);
- выборки всех данных по одному объекту «Кассета» (минимальное количество атрибутов среди всех товаров);
- выборки объекта-контейнера (справочника всех товаров со всеми их атрибутами);
- вычисление общей стоимости товаров, добавленных в корзину (взято 7 наименований с различным количеством);
- добавления нового объекта «Книга»;
- изменения данных объекта «Книга».

Сразу обратим внимание, что операции выборки одного объекта могут быть выполнены двумя способами:

1) без распознавания типа, то есть выборка всех атрибутов, какие только могут быть у наследников класса «Товар» в рамках одного запроса ко всем таблицам;

2) с распознаванием типа, при этом в программе необходимо предусмотреть такое распознавание и выполнение запросов к соответствующим таблицам.

Оценка методов будет производиться по следующим критериям:

- скорость выполнения операций;
- реализуемость – быстрота реализации метода в разрабатываемой программной системе;
- гибкость – объем работ при модификации модели предметной области;
- объем – физический объем файлов базы данных на диске
- возможные проблемы и сложность их разрешения.

Сразу оговорим, что независимо от подхода существует необходимость в поддержке уникальных идентификаторов («id») каждого объекта, по крайней мере – в ветке с корнем «Товар».

Результаты измерения времени выполнения операций и объема базы данных приведены в табл. 8 и на рис. 8 – 17 разд. 7.

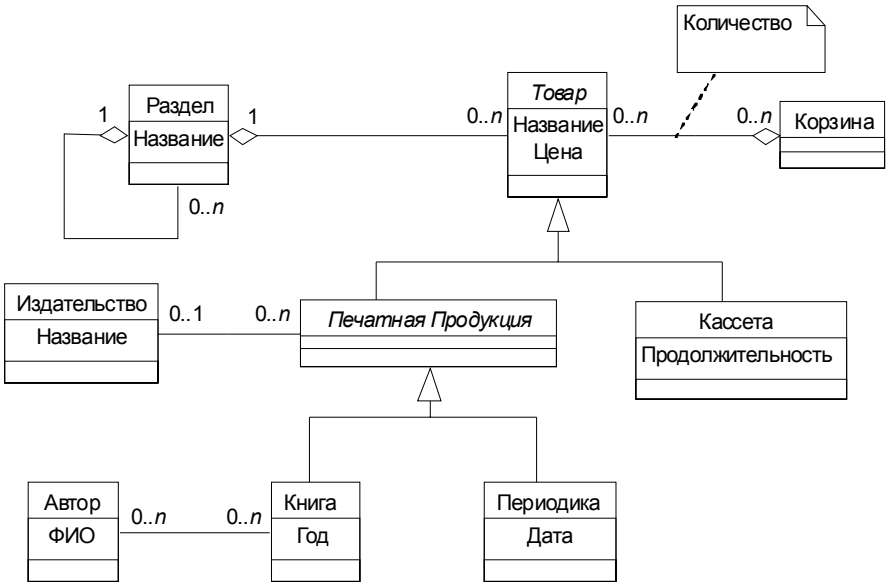

Рис.1. Диаграмма классов предметной области

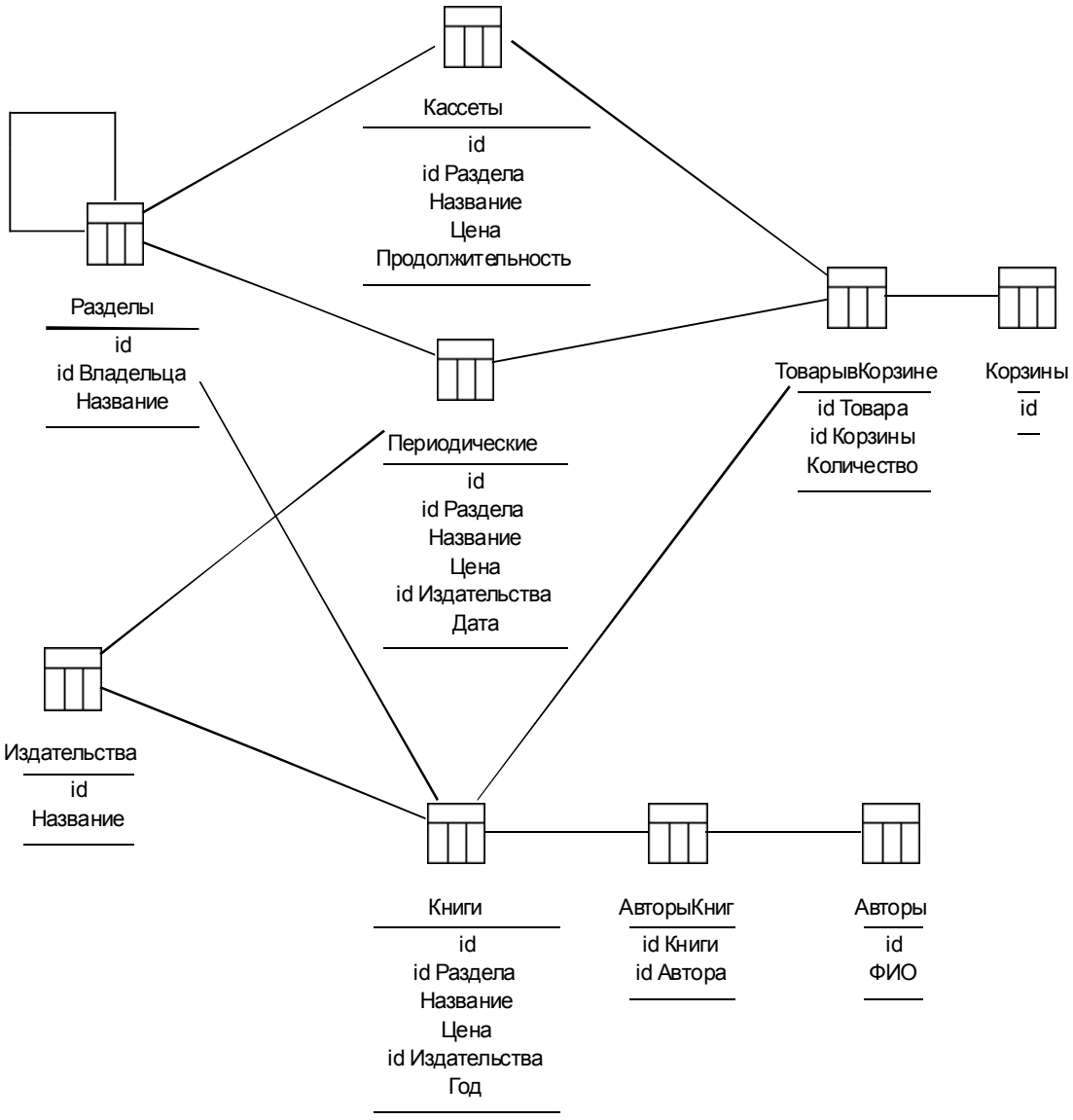

Рис. 2. Схема базы данных модели ROT

3. ОБЪЕКТЫ КАК ТАБЛИЦЫ (МОДЕЛЬ ROT)

Первым из методов рассмотрим подход, известный под названием Representing Objects as Tables (объекты как таблицы) [5] – ROT. Данный подход является наиболее «естественным» для реляционных баз данных. Суть его заключается в том, что каждому классу предметной области ставится в соответствие одна таблица реляционной базы данных с соответствующими атрибутами. Связи реализуются по соответствующим правилам ER-проектирования реляционных баз данных. При этом таблицы для абстрактных классов не создаются, а атрибуты классов-предков присутствуют и в таблицах для классов-потомков.

Схема базы данных для примера «Электронный каталог» представлена на рис. 2.

Реализуемость: подход удобен для быстрой разработки программ, так как все операции над базой данных легко реализуются стандартными конструкциями SQL. **Гибкость:** очень низкая, практически любые изменения в структуре базы данных неизбежно приводят к необходимости исправления исходного кода. **Особенности:** нет необходимости в поддержке уникальных идентификаторов в пространстве всей базы данных, в принципе, можно обойтись даже без уникальных идентификаторов ветки «Товар». **Объем базы данных и время выполнения** почти всех **операций** – минимальные среди всех рассмотренных подходов.

4. МОДИФИКАЦИЯ ROT С УЧЕТОМ НАСЛЕДОВАНИЯ

Модель ROT напрямую отображает классы предметной области в таблицы реляционной базы данных. Однако, как это было показано выше, это приводит к тому, что некоторые общие данные классов, находящихся в отношении наследования, оказываются расположены в разных таблицах и однотипную обработку данных приходится реализовывать для каждой таблицы в отдельности (в нашем примере – это класс «Товар» и его наследники).

Существует подход, позволяющий решить эту проблему. Суть данного подхода заключается в том, что для суперклассов, имеющих атрибуты или ссылки на другие объекты, создаются соответствующие таблицы, а в таблицы наследников помещаются только идентификатор и атрибуты, доопределенные в классах-наследниках. При этом между таблицей родителя и таблицей наследника устанавливается связь 1:1.

Схема базы данных, основанная на модели ROT с учетом наследования, приведена на рис. 3.

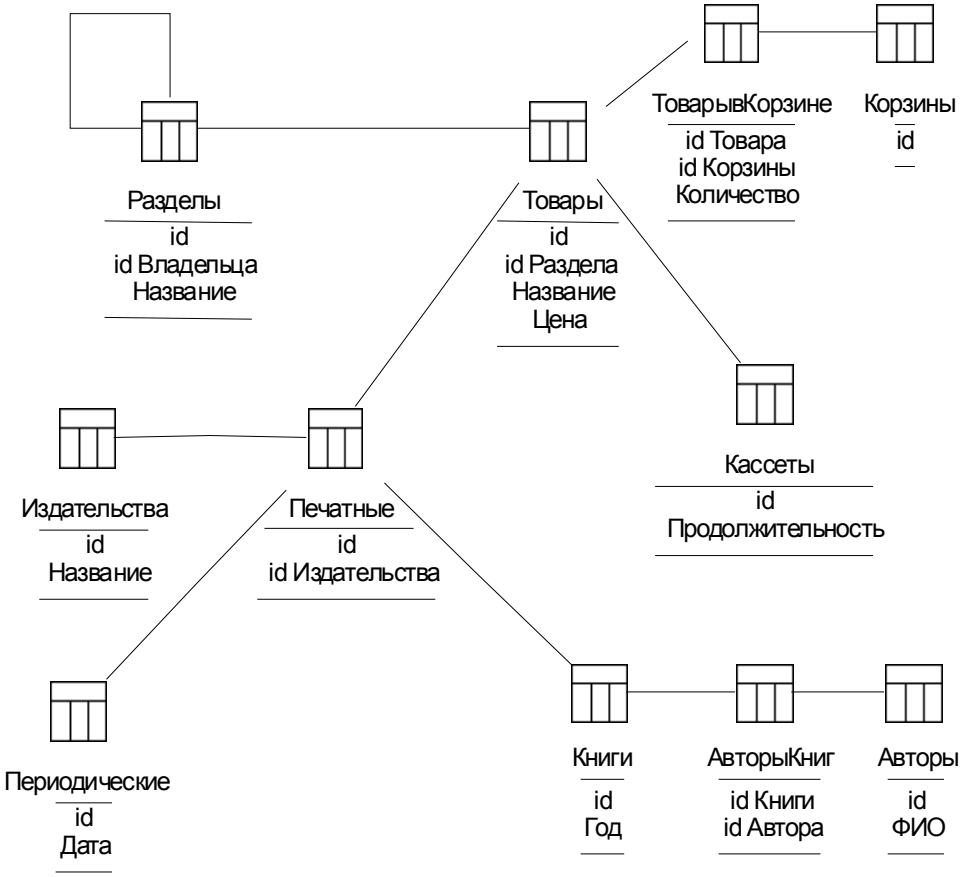

Рис. 3. Схема базы данных для модели ROT с учетом наследования

Реализуемость: данный подход в реализации немного сложнее предыдущего, большинство операций также реализуются заранее подготовленными запросами SQL. **Гибкость:** выше, чем у модели ROT, однако большинство изменений в структуре базы данных также требуют исправления исходного кода программы. **Особенности:** благодаря выделению таблицы для суперкласса «Товар» и связей 1:1 между таблицами для наследуемых классов этой ветки поддержка уникальных идентификаторов выполняется легко. **Объем базы данных и время выполнения операций** – чуть больше, чем у модели ROT.

5. МОДЕЛЬ А. ТЕНЦЕРА «БАЗА ДАННЫХ – ХРАНИЛИЩЕ ОБЪЕКТОВ»

Основы данного подхода изложены в [6]. Главной идеей является неизменность схемы базы данных: по сути дела, предложенная в работе схема является универсальной и готовой к использованию в любых объектно-ориентированных приложениях. К тому же заранее определена универсальная реализация методов материализации и дематериализации объектов.

К сожалению, в упомянутой работе не описаны некоторые нюансы, свойственные объектно-ориентированным системам. В связи с этим ниже оговорим некоторые важные с точки зрения практической реализации моменты этой модели.

Во-первых, в работе отсутствует развернутое описание реализации в модели механизмов наследования. В частности, не ясно нужно ли для наследников в таблицах StrDesc, PropDesc и т.д. дублировать свойства, которые были уже приписаны родителю. То же касается и связей в таблице AllowedLinks. Будем считать, что описание свойств классов и связей между ними не дублируется у наследников, а восстанавливается благодаря зависимости Id–ParentId в таблице ObjType.

Во-вторых, не ясно, создаются ли все записи для свойств нового объекта в таблицах свойств (Strings, Properties и т.д.), даже если они пусты. Будем считать, что у «живого» экземпляра объекта определены все свойства, и поэтому в упомянутые таблицы вносятся записи, соответствующие всем свойствам объекта.

В-третьих, в модели Тенцера отсутствует (явно не выражена) возможность создания атрибутов связи. В нашем примере – это атрибут «Количество», значение которого определяется только при сочетании одного конкретного экземпляра класса «Товар» с одним конкретным экземпляром класса «Корзина». Реализуем решение этого вопроса в виде ассоциативного класса «ТоварВКорзине» с атрибутом «Количество» (рис. 4).

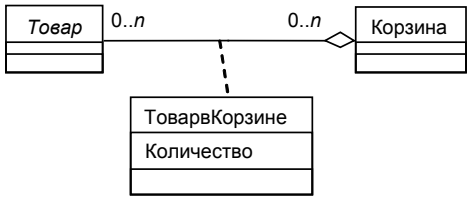

Рис. 4. Представление атрибута связи в виде ассоциативного класса

В-четвертых, связи (таблицы AllowedLinks и Links) в этой модели реализованы только бинарные. В «классическом» ER-проектировании баз данных существует возможность представления трех- и более сторонних связей. Трехстороннюю связь на диаграмме рис. 4 преобразуем в две бинарные путем введения дополнительного класса, описывающего связь. В данном случае классом связи может выступить сам ассоциативный класс, поэтому диаграмму на рис. 4 можно представить в следующем виде (рис. 5).

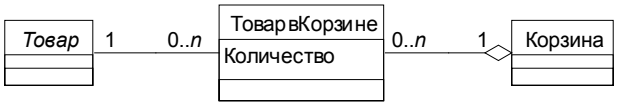

Рис. 5. Представление трехсторонней связи в виде совокупности бинарных отношений

Перейдем теперь к представлению нашего примера в базе данных, составленной согласно модели Тенцера. Атрибуты классов нашей предметной модели относятся к следующим простым типам данных: строка («Название», «ФИО»), число («Цена», «Год», «Количество»), дата/время («Дата», «Продолжительность»). Согласно модели Тенцера, для их описания и хранения потребуется три пары таблиц: StrDesc, Strings – для строк, PropDesc, Properties – для чисел, DTDesc, DateTimes – для дат/времени. Таким образом, получаем следующую схему базы данных (рис. 6).

Кроме того, в таблицы необходимо внести следующую информацию (табл. 1 – 5). Для краткости опустим таблицу AllowedLinks, так как она не влияет на выполняемые нами измерения.

Таблица 1

Таблица ObjType

Id	Code	ItemName	ParentId
1	Раздел		0
2	Товар		0
3	Корзина		0
4	ТоварвКорзине	Экземпляр объекта-связи	0
5	Издательство		0
6	Автор		0
7	ПечатнаяПродукция		2
8	Кассета		2
9	Книга		7
10	Периодика	Периодическое издание	7

Таблица 2

Таблица StrDesc

Id	TypeId	Code	ItemName
11	1	Название	Название раздела
12	2	Название	Название товара
13	5	Название	Название издательства
14	6	ФИО	Фамилия И.О. автора

Таблица 3

Таблица PropDesc

Id	TypeId	Code	ItemName
15	2	Цена	Цена единицы товара
16	4	Количество	Количество в корзине
17	9	Год	Год издания книги

Таблица 4

Таблица DTDesc

Id	TypeId	Code	ItemName
18	8	Продолжительность	Продолжительность кассеты
19	10	Дата	Дата периодического издания

Таблица 5

Таблица LinkType

Id	Code	ItemName
20	Входит	Связь в дереве разделов
21	Относится	Связь товара с разделом
22	Соответствует	Связь товара с экземпляром Товар в Корзине
23	Относится	Связь экземпляра Товар в Корзине с Корзиной
24	Издательство	Связь печатной продукции с издательством
25	Автор	Связь книги с автором

Реализуемость: для реализации в программе данной модели необходимо, в первую очередь, реализовать подсистему управления данными, которая «на лету» собирает объекты. Подход применим при долговременной разработке либо при наличии уже реализованной подсистемы управления данными. **Гибкость:** высокая, практически все изменения в предметной области требуют внесения исправлений на уровне данных (конфигурирования) и не влияют на исходный код. **Особенности** – для универсальности программы имеется необходимость поддержки уникального идентификатора в пространстве всей базы данных. Разделение таблиц для атрибутов по типам приводит к необходимости выборки или поиска по всем таким таблицам. Благодаря полям Code и ItemName в таблицах такой базы данных очень удобно строить списки элементов (справочники). **Объем базы данных и время выполнения операций** – достаточно высокие, в большинстве тестов – максимальные.

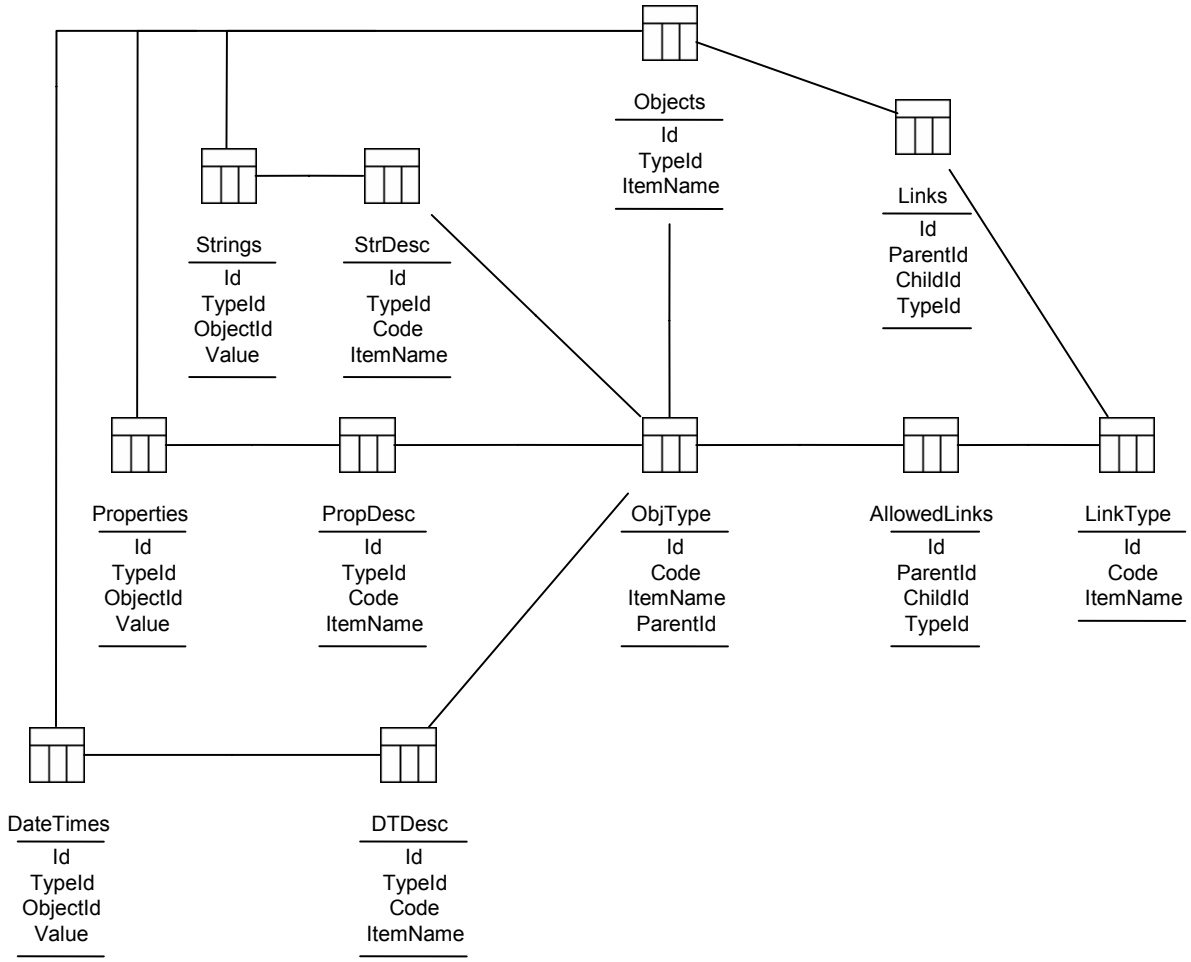

Рис. 6. Схема базы данных согласно модели Тенцера

6. МОДИФИКАЦИЯ МОДЕЛИ ТЕНЦЕРА

Рассмотрим возможную модификацию модели Тенцера. Внесенные изменения, с одной стороны, вызваны обобщением подхода: описание атрибутов всех объектов производится в одной таблице Attributes со структурой, похожей на XXXDesc в модели Тенцера. Хранение значений всех атрибутов объектов производится в пределах одной таблицы, подобной таблице Strings, в которой столбец Value имеет тип Variant. Конечно, не все существующие форматы реляционных баз данных поддерживают такой тип столбцов, поэтому на практике можно использовать тип String. При этом возникают естественные проблемы, связанные с увеличением объема таких данных и их агрегатной обработкой. Однако объем в данном случае является ценой за универсальность подхода, а агрегатная обработка в объектно-ориентированных системах непосредственно в базе данных обычно не производится (для этого используют сервер приложений [6]).

С другой стороны, введение таблицы Containers продиктовано соображениями, вытекающими из практического опыта работы. Дело в том, что наиболее часто встречающимся видом ассоциации (кроме наследования) между классами является агрегация. Например, подсистема нормативно-справочной информации любой информационной системы обязательно будет содержать такой вид связи [7].

Наконец, другие виды связи в этой модели реализуются таким же образом, как и в исходном коде объектно-ориентированных программ – через атрибуты, являющиеся ссылками на другие объекты. В данном случае эти ссылки представляют собой внешние ключи, ссылающиеся на идентификатор соответствующего объекта. Значения этих атрибутов также располагаются в таблице ObjectData.

На рис. 7 представлена схема базы данных для предлагаемой модели.

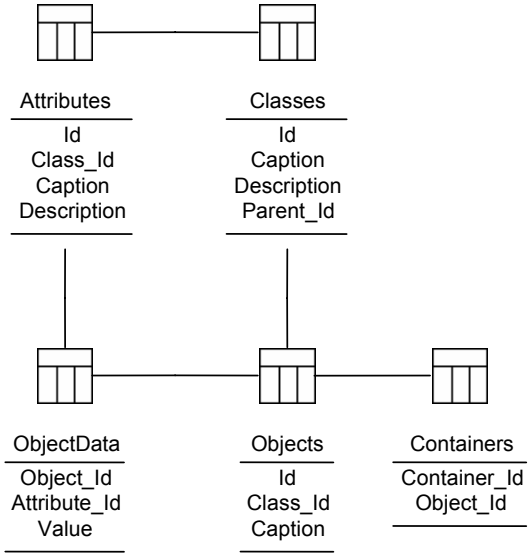

Рис. 7. Схема базы данных модифицированной модели Тенцера

Таблица Classes содержит описание классов системы. Атрибут Id – уникальный идентификатор, Parent_Id – ссылка на Id предка в этой же таблице, Caption – короткое (возможно, англоязычное – для исходного кода программы) название класса, Description – подробное описание класса.

Таблица Attributes содержит описание атрибутов классов. Id – уникальный идентификатор атрибута, Class_Id – ссылка на Id класса (в таблице Classes), к которому относится атрибут, Caption и Description – аналогично предыдущему. Для классов-наследников описание атрибутов не повторяется. Таким образом, для материализации описания класса необходимо восстановить всю структуру наследования «вверх» и выбрать описания всех атрибутов классов этого поддерева.

Таблица Objects содержит список экземпляров объектов. Id – уникальный идентификатор объекта, Class_Id – ссылка на Id класса (в таблице Classes), к которому относится экземпляр, Caption – краткое обозначение объекта (для быстрого вывода в списках), формируется программной системой, обычно соответствует значению одного (или комбинации нескольких) из свойств.

Таблица ObjectData содержит значения свойств объектов. Object_Id – ссылка на Id объекта (в таблице Objects), Attribute_Id – ссылка на Id атрибута (в таблице Attributes), Value – значение этого атрибута для данного объекта (тип Variant или, возможно, String).

Таблица Containers содержит ключи ассоциации агрегирования. Здесь Container_Id – ссылка на Id объекта-владельца (в таблице Objects), Object_Id – ссылка на Id обекта (там же), содержащегося в контейнере. Сразу оговорим, что агрегирование в данном случае означает «содержит ссылку» и не подразумевает полное владение вложенными объектами.

Таким образом, таблицы Clsses и Attributes содержат описание (метаданные), а остальные таблицы – сами данные.

В представленной статье не будем касаться вопросов реализации описания типов атрибутов, контроля за совместимостью типов для ссылок и описания ролей классов в ассоциациях, так как это не является существенным для производимых измерений.

Для нашей модели предметной области (диаграммы на рис. 1 и 5) таблицы, содержащие метаданные, следует заполнить следующим образом (табл. 6 – 7).

Таблица 6

Таблица Classes

Id	Caption	Description	Parent_Id
1	Раздел		0
2	Товар		0
3	Корзина		0
4	ТоварвКорзине	Экземпляр объекта-связи	0
5	Издательство		0
6	Автор		0
7	ПечатнаяПродукция	Печатная продукция	2
8	Кассета		2
9	Книга		7
10	Периодика	Периодическое издание	7

Таблица 7

Таблица Attributes

Id	Class_Id	Code	Description
11	1	Название	Название раздела
12	2	Название	Название товара
13	5	Название	Название издательства
14	6	ФИО	Фамилия И.О. автора
15	2	Цена	Цена единицы товара
16	4	Количество	Количество в корзине
17	9	Год	Год издания книги
18	8	Продолжительность	Продолжительность кассеты
19	10	Дата	Дата периодического издания
20	4	Товар	Ссылка на Id товара
21	7	Издательство	Ссылка на Id издательства

Реализуемость: аналогично предыдущей модели требует наличия подсистемы управления данными, облегчает ее реализацию благодаря представлению данных всех типов в одной таблице и требует лишь приведения типов. **Гибкость:** очень высокая, изменения в предметной области требуют внесения исправлений на уровне данных и не влияют на исходный код программы. **Особенности** – аналогичны модели Тенцера, для универсальности программы имеется необходимость поддержки уникального идентификатора в пространстве всей базы данных. Однако здесь эту проблему решить проще благодаря меньшему количеству таблиц. Объединение всех атрибутов в одну таблицу облегчает реализацию поиска и выборки их значений. **Объем базы данных и время выполнения операций** – очень высокие.

7. СРАВНИТЕЛЬНЫЙ АНАЛИЗ

Результаты измерения времени выполнения операций и объема базы данных приведены в табл. 8 и на рис. 8–17.

Произведем краткий анализ полученных результатов.

Открытие всех таблиц базы данных. Суммарное время открытия всех таблиц базы данных в модели Тенцера и ее модификации значительно превышает этот показатель для ROT и ее модификации (рис. 8). Это связано с «гигантскими» таблицами Objects, Links, Strings, Properties и т.д.

Выборка всех данных по одной Книге. На проведение выборки данных по одной Книге (рис. 9, 10) не влияет определение типа на этапе выполнения, так как в любом случае количество атрибутов остается одинаковым – максимальным. Хорошие результаты показывает модифицированная модель Тенцера, так как выборка самих значений производится по одной таблице ObjectData.

Выборка всех данных по одной Кассете. Общая картина практически не изменяется при универсальном запросе (без определения типа). В случае динамического определения типа товара для модели ROT с наследованием и модели Тенцера происходит резкое уменьшение времени выполнения операции, так как в этом случае запрос строится по меньшему количеству таблиц.

Выборка всех данных по всем товарам. Результатом такой операции является не просто список наименований, а полная информация о каждом товаре. Поэтому модель Тенцера проигрывает модели ROT, однако значительно опережает ROT с наследованием благодаря меньшему количеству таблиц в таком запросе.

Вычисление суммы товаров в корзине. Время выполнения данной операции в модели Тенцера и ее модификации значительно больше, чем в модели ROT. Связано это с необходимостью учета ссылок через таблицу Links (или Containers) и вытекающим отсюда «каскадированием» запроса. К тому же, в модифицированной модели время вычисления суммы еще больше из-за необходимости преобразования данных строковых типов в числовые.

Добавление и изменение данных одной Книги. Увеличенное время выполнения данных операций в модели Тенцера и ее модификации по сравнению с ROT, опять же, объясняется более длинными таблицами, в которые вносятся данные. Причиной большого разброса времени выполнения этих операций, видимо, является ненаблюдаемое кэширование данных.

Таблица 8

Время выполнения операций и объемы баз данных

Параметр	Модель ROT	Модель ROT с наследованием	Модель Тенцера	Модифицированная модель Тенцера
Открытие всех таблиц базы данных, с	5,72–5,74	4,71–4,72	14,01–15,11	14,04–15,16
Выборка всех данных по одной Книге без определения типа, с	6,57–6,63	15,10–16,02	31,29–31,36	7,83–8,93
Выборка всех данных по одной Книге с определением типа, с	6,29–6,33	14,18–14,24	30,05–30,82	8,56–10,17
Выборка всех данных по одной Кассете без определения типа, с	6,43–6,49	15,12–15,14	15,67–16,46	8,71–10,05
Выборка всех данных по одной Кассете с определением типа, с	0,53	1,06–1,08	4,11–5,29	4,80–5,92
Выборка всех данных по всем товарам, с	60,12–91,90	206,47–236,32	97,81–135,25	94,83–113,11
Вычисление суммы товаров в корзине, с	8,61–8,84	8,45–9,35	12,08–13,06	13,50–14,86
Добавление новой Книги, с	0,03–0,05	0,04–0,13	0,14–0,55	0,17–0,57
Изменение данных по одной Книге, с	1,31–1,35	2,50–2,61	3,12–8,81	3,62–9,25
Объем базы данных, МБ	39,5	43,7	49,7	56,6

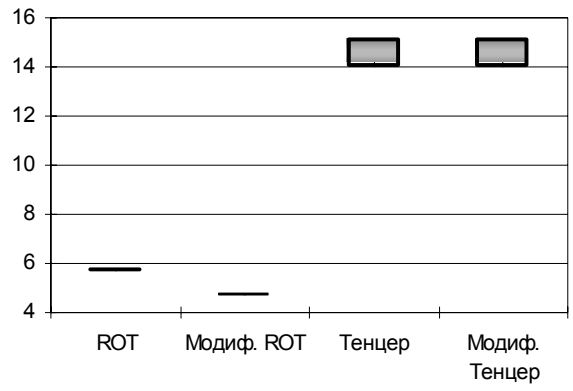

Рис. 8. Открытие всех таблиц базы данных, с

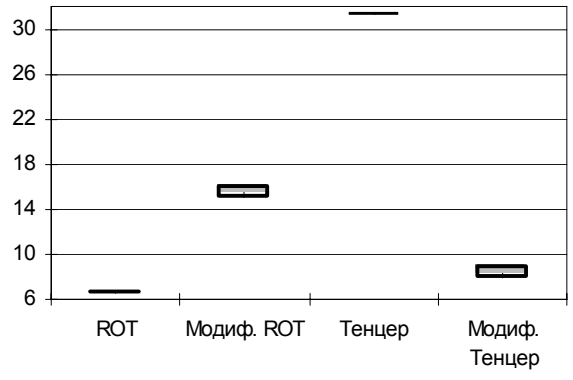

Рис. 9. Выборка всех данных по одной Книге без определения типа, с

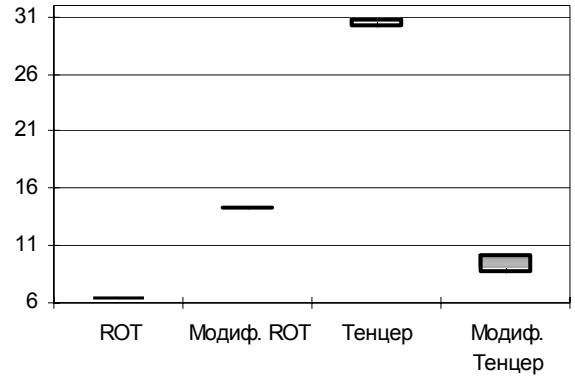

Рис. 10. Выборка всех данных по одной Книге с определением типа, с

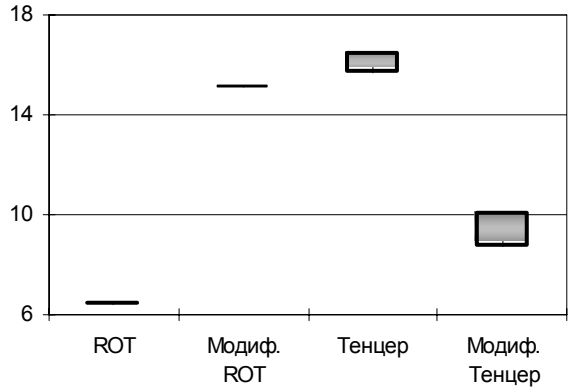

Рис. 11. Выборка всех данных по одной Кассете без определения типа, с

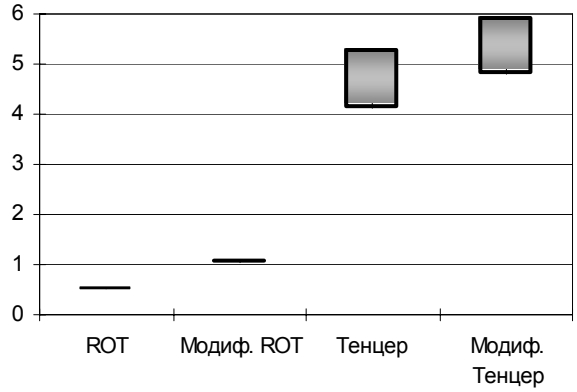

Рис. 12. Выборка всех данных по одной Кассете с определением типа, с

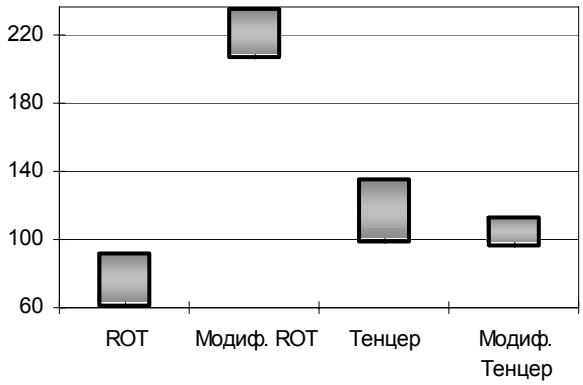

Рис. 13. Выборка всех данных по всем товарам, с

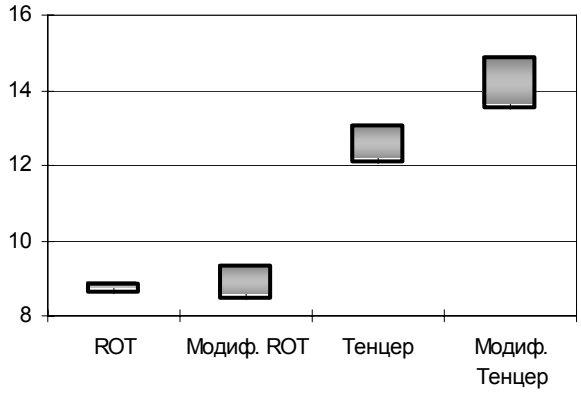

Рис. 14. Вычисление суммы товаров в корзине, с

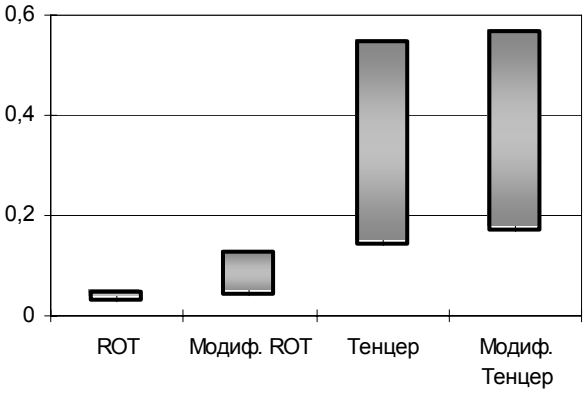

Рис. 15. Добавление новой Книги, с

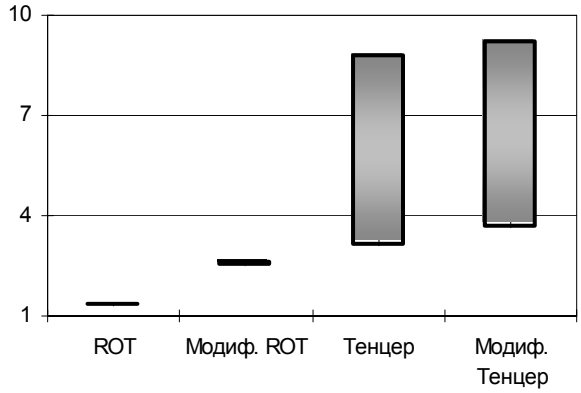

Рис. 16. Изменение данных по одной Книге, с

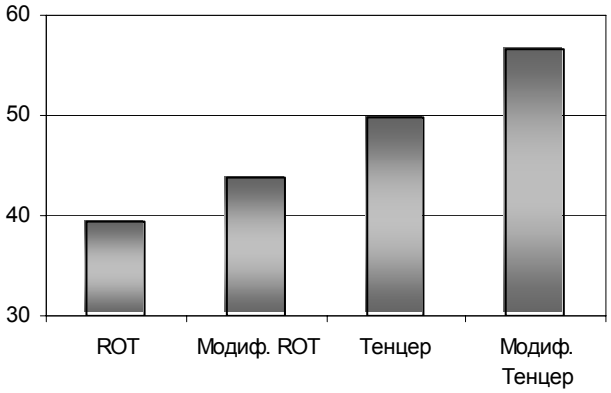

Рис. 17. Объем базы данных, МБ

ВЫВОДЫ

Полученные результаты показывают, что для создания небольших информационных систем, не требующих дальнейшего сопровождения, вполне подходит модель представления классов как таблиц (ROT): она позволяет быстро реализовать все необходимые операции над данными, при этом сами операции выполняются очень быстро, а размер базы данных получается минимальный. Для придания большей гибкости такой системе можно использовать модификацию модели ROT с наследованием (средние системы). При разработке сложных информационных систем, требующих длительного сопровождения и предполагающих дальнейшее развитие, рекомендуется использовать модифицированную модель Тенцера. При этом основной объем работ будет посвящен разработке подсистемы управления данными, оптимизации скорости выполнения операций (при проведении тестов задачи решались «в лоб», без рассмотрения проблемы оптимизации). И хотя время выполнения многих операций в такой модели все равно останется достаточно большим, стоимость владения такой сложной информационной системой будет значительно ниже, чем выполненной в ROT, так как определяется лишь ценой подсистемы управления данными и ядром интерфейса. При этом практически все вопросы сопровождения программного продукта будут решаться на уровне конфигурирования специалистом квалификации «опытный пользователь» или «администратор ИС». В случае использования ROT, как было описано выше, стоимость сопровождения определяется работой более дорогого специалиста (возможно, нескольких) квалификации «программист».

ЛИТЕРАТУРА

1. *Змеев О.А., Моисеев А.Н.* Шаблон диаграммы компонентов информационной системы корпоративного уровня // Вестник ТГУ. 2002. № 275. С. 130–133.
2. *Змеев О.А., Новиков Д.В., Моисеев А.Н.* К вопросу проектирования уровня хранения в виде ООРБД // Вестник ТГУ. Приложение №1 (II), сентябрь 2002. Доклады IV Всероссийской конференции с международным участием «Новые информационные технологии в исследовании сложных структур», Томск, 10-13 сентября 2002. С. 363–367.
3. *Войтиков К.Ю., Змеев О.А., Моисеев А.Н.* Основные функциональные требования к подсистеме «Брокер объектных запросов» в рамках унифицированного процесса разработки программного обеспечения // Обработка данных и управление в сложных системах. Томск: Изд-во Том. ун-та, 2003. Вып. 5. С. 3–13.
4. *Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж.* Приемы объектно-ориентированного проектирования. Паттерны проектирования. СПб.: Питер, 2001. 368 с.
5. *Ларман К.* Применение UML и шаблонов проектирования. 2-е изд. М.: Изд. дом «Вильямс», 2002. 624 с.
6. *Тенцер А.* База данных – хранилище объектов // КомпьютерПресс. 2001. № 8.
7. *Войтиков К.Ю., Змеев О.А., Моисеев А.Н.* Объектный подход к проблеме проектирования подсистемы нормативно-справочной информации // Обработка данных и управление в сложных системах. Томск: Изд-во Том. ун-та, 2002. Вып. 4. С. 13–20.

Статья представлена кафедрой прикладной информатики факультета информатики Томского государственного университета, поступила в научную редакцию 30 апреля 2003 г.