К ВОПРОСУ О ЦЕЛЕСООБРАЗНОСТИ ИСПОЛЬЗОВАНИЯ ПОСТРЕЛЯЦИОННЫХ СУБД ПРИ РАБОТЕ С ДАННЫМИ СЛОЖНОЙ АРХИТЕКТУРЫ

Ю.А. Тимофеев

Московский инженерно-физический институт
115409 Москва, Каширское ш. 31

Аннотация. Поставлена проблема использования классических реляционных баз данных при работе с динамичными блоками данных, представляющих себя объектами реализованными в виде классов объектно-ориентированных языков программирования. Предложена концепция организации построения постреляционной базы данных.

Ключевые слова: реляционные базы данных, классы.

Актуальность проблемы: ускоряется темп жизни и все большее значение приобретает вопрос о скорости разработки технического решения взаимодействия программного продукта и БД. Время, которое готовы предоставить на разработку данной связки ежегодно уменьшается. На сегодняшний день по опыту реализованных нами проектов (от кастинговых до блочных систем работы с данными) оно составляет от 14 до 60 дней, в то время как год назад эти сроки были вдвое больше. Тенденция к сокращению сроков разработки налицо. В связи с этим вопрос эффективного использования «классических» СУБД (систем управления базами данных) переживает актуализацию.
Кроме того, нам предоставляется все меньше возможностей специализировать БД под конкретный вид данных. Все больше требуются универсальные решения, позволяющие сохранять объектные классы «как есть» [1].

Комплекс сложных научно-технических проблем, связанных с созданием производительных и надежных систем баз данных, не только сохраняет, но и усиливает свою актуальность с каждым днем с ростом объемов и сложности информации в целом. В настоящее время системы управления базами данных используются практически во всех сферах человеческой деятельности, связанных с хранением и обработкой информации.

Но не верно было бы считать классические СУБД панацеей для организации хранения данных. Решая часть задач по автоматизации производственных и иных процессов, они порождают проблемы, свойственные самой технологии. При организации хранения сложных данных периодически встает вопрос принудительной сериализации данных, которая далеко не всегда бывает удобна. Зачастую разработчики вынуждены искусственно ограничивать размеры данных исходя из формата полей базы данных.

Существует типичный набор проблем, связанных с СУБД: недостаточная производительность приложений, сложности при построении сложных систем данных [2], трудности моделирования данных в БД, скорость и простота разработки приложений, в том числе и для Web.

Частично вышеперечисленные проблемы могут решаться с помощью сериализации данных и нормализации БД в разрезе использования классических реляционных СУБД. Однако если размер структуры данных резко разнится от случая к случаю, данный подход является неэффективным.

При попытках создать так называемые «чисто объектные» или постреляционные системы удается избавиться от многих недостатков реляционных СУБД [3], особенно при проектировании систем и моделировании данных, при работе с «сущностями», как с едиными информационными объектами. Но при этом теряются и такие достоинства реляционного подхода, как удобство и быстрота получения отчетов, хранение информации в структурах, пригодных для её многоаспектного анализа.

Известный факт: объектная база данных очень хорошо подходит для областей, где требуется работа с данными сложной структуры переменного объема.

В отличие от объектно-реляционных СУБД (Oracle, Microsoft SQL Server, PostgreSQL, Sav Zigzag и другие), где ядро остается реляционным, а «объектность» реализуется в виде специальной надстройки, ядро ООСУБД (объектно-ориентированной системы управления базами данных) может быть оптимизировано для типовых операций с объектами и производить операции кэширования объектов, ведение версий объектов, разделения прав доступа к конкретным объектам.

Поэтому ООСУБД имеет более высокое быстродействие на операциях, требующих доступа и получения данных, упакованных в объекты, по сравнению с реляционными СУБД, для которых необходимость выборки связных данных ведет к выполнению дополнительных внутренних операций.

Наиболее важным новым качеством ООБД (объектно-ориентированной базы данных), которое позволяет реализовать объектно-ориентированный подход, является поведенческий аспект объектов. В прикладных информационных системах, основывавшихся на БД с традиционной организацией (вплоть до тех, которые базировались на семантических моделях данных – имена собственные), существовал принципиальный разрыв между структурной и поведенческой частями. Структурная часть системы поддерживалась всем аппаратом БД, ее можно было моделировать, верифицировать и т.д., а поведенческая часть создавалась изолированно. В частности, отсутствовали формальный аппарат, системная поддержка совместного моделирования и гарантии согласованности этих структурной (статической) и поведенческой (динамической) частей. В среде ООБД проектирование, разработка и сопровождение прикладной системы становится процессом, в котором интегрируются структурный и поведенческий аспекты. Конечно, для этого нужны специальные конструкции в существующих языках, позволяющие определять объекты и создавать на их основе прикладную систему. В настоящее время с переходом на разработку программных продуктов по принципу объектно-ориентированного программирования (ООП) данная задача не представляет большой сложности [4].

Специфика применения объектно-ориентированного подхода для организации и управления БД требует уточненного толкования классических концепций и значительного их расширения. Это определяется потребностями долговременного хранения объ-

ектов во внешней памяти, ассоциативного доступа к объектам, обеспечения согласованного состояния ООБД в условиях мультидоступа и тому подобных возможностей, свойственных базам данных. Выделяются три аспекта, отсутствующие в традиционной парадигме, но обязательно требующиеся в ООБД.

Первый аспект касается потребности в средствах спецификации блоков данных при определении класса (ограничений целостности, правил обработки и т.п.)

Второй аспект – потребность в механизме определения разного рода семантических связей между объектами, вообще говоря, разных классов. Фактически это означает требование полного распространения на ООБД средств семантического моделирования данных. Потребность в использовании абстракции ассоциирования отмечается и в связи с возможностью использования ООБД в сфере автоматизированного проектирования и инженерии.

Наконец, третий аспект связан с пересмотром понятия класса. В контексте ООБД оказывается более удобным рассматривать класс как множество объектов данного типа, т.е. одновременно поддерживать понятия и типа и класса объектов[5]. Как мы можем заметить, в сообществе исследователей ООБД и разработчиков систем отсутствует полное согласие, но в большинстве практических работ используется некоторое расширение объектно-ориентированного подхода.

Как правило, база данных объединяет несколько хранилищ данных, каждое из которых ассоциируется с одним или несколькими файлами. Хранятся как метаданные (класс, атрибут, определения операций), так и пользовательские данные. В предложенной нами схеме выделяются три типа хранилищ:

- **системное хранилище** используется для хранения системы классов, создается на этапе формирования базы данных и содержит информацию о классах, о наличии и месторасположении пользовательских хранилищ;
- **пользовательское хранилище** предназначено для хранения пользовательских объектов;
- **служебное хранилище** содержит временную информацию, например сведения о заблокированных объектах, об активных транзакциях, различного вида списки запросов пользователей и т.д.

Перейдем к вопросу организации взаимодействия программной среды и базы данных. Варианты языков запросов можно разделить на две группы. Первая объединяет языки, унаследованные от SQL и представляющие собой разновидность OQL (Object Query Language) [6], языка, стандартизованного ODMG (Object Data Management Group). Объектно-реляционные СУБД используют различные варианты ограниченных объектных расширений SQL.

Вторая, сравнительно новая группа языков запросов базируется на XML. Собирательное название языков этой группы — XML QL (или XQL). Они могут применяться в качестве языков запросов в объектных и объектно-реляционных базах данных. Использование в чисто реляционных базах нецелесообразно, поскольку языком предусматривается объектная структура запроса.

Объектная модель предлагаемой архитектуры СУБД в большой степени соответствует объектной модели стандарта ODMG. В соответствии с ODMG каждый объект имеет определенный, единственный тип.

Следует заметить, что в целом стандарт ODMG оставляет противоречивые ощущения. С одной стороны, в документе содержится много интересных идей и технических спецификаций. С другой стороны, некоторые важные части стандарта кажутся недостаточно проработанными и плохо отредактированными. Местами встречаются противоречия и прямые ошибки. Но в настоящий момент это единственный общепризнанный стандарт ООСУБД.

Поведение объекта определяется операциями (методами), а состояние объекта – значениями его свойств (метаданные). Свойства и операции составляют характеристики типа. Тип определяется одним интерфейсом, которому может соответствовать одна или большее число реализаций. Объектная модель представлена на рис. 1.

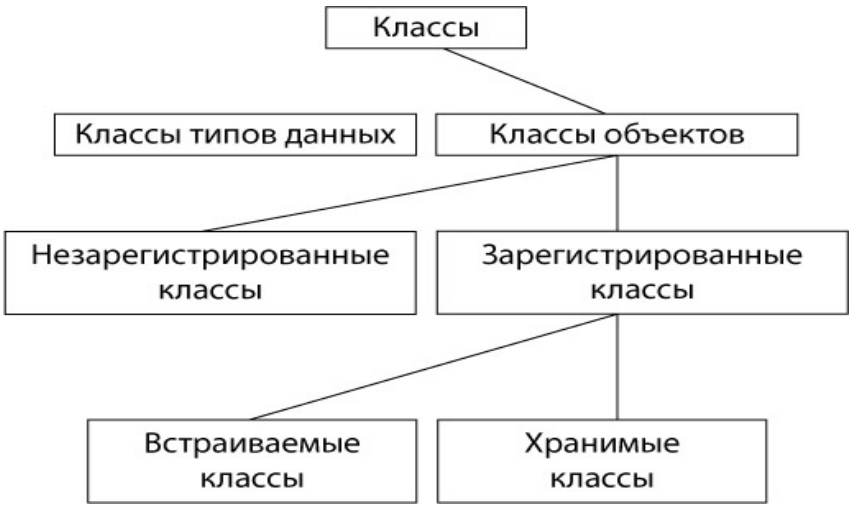

Рис 1. Объектная модель

При разработке модели данных часть объектов, исходя из их структуры, имеет смысл размещать в структурах в классических реляционных таблицах. Комбинируя подходы и сохраняя данные в разных структурах можно добиться наибольшего удобства и эффективности при работе с базами данных подобного типа [7].

Исходя из вышесказанного предлагается использовать два типа классов:

- классы типов данных;
- классы объектов.

Классы типов данных определяют допустимые значения констант (литералов) и позволяют их контролировать как в рамках реляционных таблиц, так и в виде объектных

моделей. Литерал не может существовать независимо от своего значения, в то время как объекты имеют уникальную идентификацию.

Классы типов данных подразделяется на два подкласса типов:

- атомарные;
- структурированные.

Атомарными литеральными типами являются традиционные скалярные типы данных (String, Integer, Float, Date и др.). Каждый литерал уникально идентифицируется индексом в массиве или порядковым номером в списке.

Структурированные классы позволят сохранять заранее предопределенные блоки данных, целостность которых будет отслеживаться на уровне СУБД. Подход к реализации данного класса похож на организацию хранения нормализованного блока данных в классической реляционной базе данных.

Классы объектов предлагается целиком передать разработчику для сохранения объектно-ориентированных представлений классов из языка ООП. Разработчик может хранить метаданные для класса как отдельно, так и назначать их в виде указателей на структуру данных внутри класса.

В настоящий момент нами была реализована подобная структура базы данных в рамках реализации проекта компании ТелеКастинг, которая хранит и обрабатывает абстрактные объекты с использованием механизма «хэш таблиц». Благодаря подобной организации данных в рамках проекта удалось избежать потребности реорганизовывать структуру БД при появлении новых типов данных со стороны пользователей и в значительной мере сократить сроки разработки, так и повысить скорость работы системы в целом.

В заключение хочется добавить, что в прошлые годы много внимания уделялось вопросу трудоемкости разработки программного обеспечения. Возросшая сложность программ и объемы используемых данных не позволяют начать разрабатывать новый продукт без построения модели базы данных, рассчитанной на строго определенную задачу. Теперь на первый план начинают выходить технологии, позволяющие создавать легко сопровождаемые программы за счет использования наиболее гибких и универсальных средств разработки программного обеспечения и средств обработки и хранения данных..

Но реляционные базы данных, скорее всего, по-прежнему останутся в качестве недорогих средств разработки приложений и, во многих случаях, естественных средств представления предметной области, подобно радио и кино, которых не вытеснило телевидение.

Благодаря четному разделению данных на скалярные таблицы и блоки объектов данный подход позволит хранить данные в наиболее удобном для разработчика виде с применением собственных метаданных и свойств для индексации и поиска, используя преимущества от эффекта синергии двух подходов в реализации СУБД, что безусловно положительно скажется на сроках разработки и качестве итогового продукта.

Изложенные в настоящей работе принципы объектного проектирования базы данных и объектная технология обработки информации позволяют разработать оригинальную структуру СУБД, сетевая версия которой, построенная по архитектуре «клиент-сервер», воплощающую в себе идеи блочного хранения абстрактных данных.

ЛИТЕРАТУРА

1. *Мутушев Д.М. Филиппов В.И.* Объектно-ориентированные базы данных. Программирование. - М., 1995 №6.
2. *Кирстен В., Иринге*р *М. и др*. СУБД Cache'. Объектно-ориентированная разработка приложений. Питер, 2001.
3. *Кренке Д.* Теория и практика построения баз данных .Питер, 2005.
4. *Буч Г.* Объектно-ориентированное проектирование (с примерами применения) М.Конкорд. 1992.
5. *Аткинсон М., Бансилон Ф. и др*. Манифест систем объектно-ориентированных баз данных // СУБД № 4 1995.
6. СУБД № 1 1996. Стандарт систем управления объектными базами данных ODMG-93: краткий обзор и оценка состояния Л.А.Калиниченко.
7. Data & Knowledge Engineering. Amsterdam 1996 Том 18 №1 Unifying data, behaviours, and messages in object-oriented databases. Sylvia L. Osborn, Li Yu.

ON THE QUESTION OF WHETHER TO USE POST-RELATIONAL DATABASE WHEN WORKING WITH COMPLEX DATA ARCHITECTURE

Yu. Timofeev

Moscow Engineering Physics Institute
115409 Moscow, Russia, Kashirskoye shosse, 31

Abstract. The following problem is raised: the use of classical relational databases when working with dynamic blocks of data representing the objects themselves implemented as classes in object-oriented programming languages. The concept of organization of building of post-relational database is offered.

Key words: relational databases, classes.

ТЕХНОЛОГИИ И МЕТОДИКИ ОБУЧЕНИЯ

УДК 681.3.07

МЕТОДИЧЕСКИЕ ОСОБЕННОСТИ ИСПОЛЬЗОВАНИЯ