Методы реализации объектных статических моделей в приложениях баз данных

И.А. Микляев, П.П. Олейник, С.М. Салибекян

Аннотация. В статье рассмотрены три различных подхода к представлению статических моделей при реализации приложений баз данных. С целью сравнения решений, на основе выделенных критериев оптимальности создана унифицированная модель тестирования инструментов разработки объектно-ориентированных приложений в виде диаграммы классов языка UML. В качестве одной из реализаций рассматриваются классические методы объектно-реляционного отображения. Данные методы предлагают реализацию статических моделей в среде реляционной СУБД. Альтернативным решением, широко используемым одним из авторов, является объектно-атрибутивный подход. Приведены основные составляющие данного подхода и пример реализации унифицированной модели тестирования. В работе также предложено решение, основанное на матричной универсальной объектно-реляционной базе данных, основные составляющие и вопросы реализации которой представлены в соответствующих разделах.

Ключевые слова: базы данных, объектные модели, UML, методы (паттерны) объектно-реляционного отображения, объектно-атрибутный подход, матричная универсальная объектно-реляционная база данных.

Введение

В настоящий момент существует множество инструментов, предоставляющих объектный подход для разработки приложений. Это связано с тем, что объектно-ориентированная парадигма является доминирующей при разработке всех новых приложений не зависимо от сферы применения. Объектно-ориентированный подход все чаще применяется и при реализации приложений баз данных. Несмотря на наличие у каждого инструмента собственных достоинств и недостатков, основная цель подхода предоставить разработчику все преимущества объектно-ориентированной парадигмы при реализации приложений БД.

В данной статье представлены три различных подхода, использованных авторами при разработке приложений баз данных для прикладных предметных областей. Статья является результатом многолетних исследований и основана на многочисленных докладах, опубликованных в материалах Международной научно-практической конференции "Объектные системы" (objectsystems.ru).

Описанные подходы продемонстрированы на примере реализации унифицированной модели тестирования инструментов разработки объектно-ориентированных приложений. Это позволяет читателю самостоятельно сделать выводы о различиях в реализованных решениях.

1. Унифицированная модель тестирования

Проектирование статической модели при реализации приложений баз данных – наиболее сложный и ответственный этап, т.к. только

наличие всех выделенных сущностей и связей предметной области позволит корректно реализовать рабочую информационную систему. Для проектирования объектно-ориентированных приложений используют унифицированный язык моделирования UML, а для представления статической составляющей чаще всего применяют диаграммы класса этого языка.

Этот раздел подробно описывает унифицированную модель тестирования инструментов разработки объектно-ориентированных приложений, которая была ранее представлена в работе [1] и используется в данной статье при описании различных подходов реализации объектных статических моделей.

При проектировании унифицированной модели тестирования использован такой же подход, как при описании шаблонов (паттернов) проектирования в работе [2]. Этот подход предполагает представление многократно используемых задач широко распространённых проблем, возникающих при разработке программного обеспечения без привязки к конкретной предметной области. Основная цель данного раздела − описание модели и структурных элементов (классов и ассоциаций), а не обеспечения корректности модели и точность её соответствия конкретной предметной области.

В итоге, под унифицированной моделью тестирования инструментов разработки объектно-ориентированных приложений будем понимать диаграмму классов, состоящую из классов и атрибутов и содержащую наиболее часто встречающиеся на практике отношения между классами.

Идея, представленная здесь, не нова, имеются работы сходной тематики. Так в работе [3] уже была предпринята попытка построения унифицированной модели тестирования. Однако там отсутствовали множественные (n-арные) ассоциации и ассоциации с атрибутами, которые являются неотъемлемым элементом любой сложной информационной системы.

В работе [4] представлена тестовая модель для обучения проектированию объектно-ориентированных баз данных. Но эта модель относительно проста, что оправдано её назначением. В данной статье использованы достоинства имеющихся ранее работ и исправлены недостатки, присущие им.

Перед проектированием унифицированной модели тестирования были выдвинуты критерии оптимальности (КО), представляющие требования о наличии определенных структурных элементов на диаграмме классов и которым должна соответствовать готовая реализация. Были выдвинуты следующие требования для унифицированной модели тестирования инструментов разработки объектно-ориентированных приложений:

1. Необходимо наличие глубоких иерархий наследования. В реальных приложениях очень часто появляются глубокие иерархии, представляемые отношением наследования и объединяющие транзитивно не менее трех классов.

2. Присутствие нескольких иерархий наследования. Это позволит продемонстрировать различные опции и режимы, имеющиеся в инструменте.

3. Наличие абстрактных классов в иерархии. Абстрактные классы не могут иметь экземпляров в системе, и описаны в качестве контейнеров для атрибутов и методов, используемых в производных реальных (инстанцируемых) классах.

4. Присутствие множественных (n-арных) ассоциаций. В приложениях, автоматизирующих реальные прикладные области, часто возникают ассоциации, в которых участвуют три и более классов. Подобные отношения называют множественными или n-арными ассоциациями.

5. Наличие ассоциаций с атрибутами. Многие предметные области содержат атрибуты, которые не принадлежат определенным сущностям, и их значения появляются только при организации связей между экземплярами классов. Поэтому проектируемая унифицированная модель должна иметь ассоциации с атрибутами.

6. Присутствие композиции между классами. Композиция - это ассоциация между классами, представляющими Часть и Целое. Особенность в том, что класс, представляющий Часть, может входить только в один экземпляр класса, представляющий Целое. При этом класс, представляющий Целое, управляет жизненным циклом класса, представляющего Часть. Т.е. при удалении Целого, Части также удаляются. Эта особенность поведения очень важна для многих прикладных предметных областей.

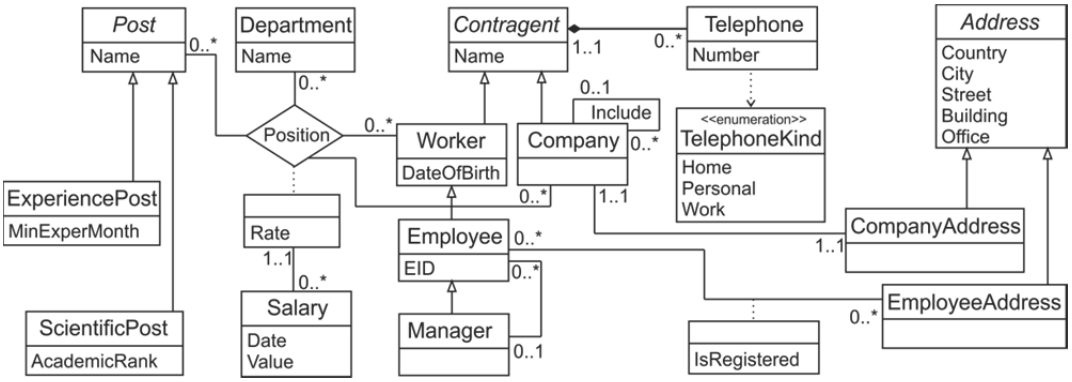

Рис. 1. Унифицированная модель тестирования инструментов разработки объектно-ориентированных приложений

7. Наличие рекурсивных ассоциаций. Рекурсивными называют ассоциации, концы которых связывают один и тот же класс. Подобные отношения позволяют реализовать иерархию подчинения.

8. Наличие ассоциаций между классами, входящими в одну иерархию наследования. С точки зрения реализации, необходимо предусмотреть ассоциации, края которых связывают классы, входящие в одну иерархию наследования, т.е. противопоставляют базовый и дочерний класс друг другу.

9. Присутствие класса-ассоциации. Класс-ассоциация - это ассоциация, которая в то же время является и классом. Особенность использования в том, что класс-ассоциация представляет собой уникальную ассоциацию, т.е. комбинация экземпляров классов в этой ассоциации является уникальной.

10. Наличие ассоциаций между классом-ассоциацией и другим классом. С теоретической точки зрения, класс-ассоциация - это класс, поэтому он может участвовать в других ассоциациях. С точки зрения реализации, класс-ассоциация представляет собой класс, содержащий атрибуты (поля или свойства языка программирования), которые ссылаются на другие классы. В свою очередь, для организации ассоциации с классом-ассоциацией необходимо в зависимом классе создать атрибут, типом которого выступает класс-ассоциация.

11. Присутствие в модели перечислений. С теоретической точки зрения перечисление - это набор предопределенных констант, при этом пользователю нельзя расширить этот набор путем добавления новых значений.

Для проверки удовлетворения модели вышеперечисленным критериям была реализована тестовая иерархия, представленная на Рис. 1.

Рассмотрим назначение основных классов представленной диаграммы. Как говорилось ранее, данная диаграмма классов является вымышленной и не предназначена для описания конкретной предметной области, поэтому содержит некоторые нелогичные (вымышленные) классы и ассоциации.

Для представления сотрудников и организаций выделен базовый абстрактный класс Contragent. Унаследованный класс Company представляет организацию, а класс Worker является базовым для сотрудника организации. Унаследованный класс Employee представляет служащего и содержит атрибут EID, представляющий табельный номер. Класс Manager представляет сотрудников, которые являются руководителями для других работников.

Абстрактный класс Post представляет собой должность, которую может занимать сотрудник. Унаследованный класс ExperiencePost представляет должность, которая требует от претендента минимального количество опыта, выражаемого количеством месяцев работы (атрибут MinExperMonth). Второй реализованный класс ScientificRank описывает должность соискателя, для которой требуется наличие ученой степени; название степени записано в атрибуте AcademicRank.

Для представления отделов организации и вхождения в n-арную ассоциацию введен класс Department. Класс Salary представляет выплаченную заработную плату, начисленную работ-

нику, занимающему должность, представленную сложной ассоциацией Position.

Класс Telephone позволяет представить номер телефона контрагента. Тип телефона (Домашний, Личный, Рабочий) представлен перечислением TelephoneKind. Для представления адреса используется базовый абстрактный класс Address. Два производных класса CompanyAddress и EmployeeAddress используются для представления адреса организации и адреса служащего соответственно.

Теперь проверим соответствие представленной модели выделенным ранее критериям оптимальности. Необходимость наличия глубокой иерархии классов, представленной как минимум тремя транзитивно унаследованными классами, описана $КО_1$ и реализована классами Contragent, Worker, Employee, Manager. Кроме этой имеются еще две иерархии: 1) Post, ExperiencePost (ScientificPost); 2) Address, CompanyAddress (EmployeeAddress). Т.е. в модели присутствуют несколько иерархий наследования, следовательно, выполнено условие $КО_2$. Наличие абстрактных классов в иерархии обусловлено $КО_3$ и выполнено классами Post, Contragent и Address.

Требования $КО_4$ также выполнены, т.к. имеется n-арная ассоциация Position, объединяющая классы Post, Department, Worker, Company. Т.к. описываемая ассоциация содержит атрибут Rate, реализованный классом-ассоциацией и бинарная ассоциация, соединяющая классы Employee и EmployeeAddress также содержит атрибут (IsRegistered), то можно утверждать, что требование $КО_5$ выполнено.

У каждого контрагента, описываемого производными от Contragent классами, имеется список номеров телефонов, представленных экземплярами класса Telephone, и оба класса связаны композицией, т.е. требование $КО_6$ выполнено. Унифицированная модель позволяет сохранять информацию о группе компаний, т.е. организовать древовидную структуру с помощью рекурсивной ассоциации, соединяющей класс Company с собой. Наличие рекурсивной ассоциации продиктовано $КО_7$.

В $КО_8$ записано требование наличия ассоциаций между классами, входящими в одну иерархию наследования. На Рис. 1 между классами Employee и Manager имеется подобная ассоциация, удовлетворяющая $КО_8$. Как отмечалось ранее, в модели имеется наличие класса-ассоциации Position, что соответствует $КО_9$. Описанный класс-ассоциация соединен дополнительной ассоциацией с классом Salary. Это – следствие выполнения $КО_{10}$. Присутствие в модели перечислений обусловлено выполнением $КО_{11}$. Из представленного описания видно, что унифицированная модель полностью соответствует всем выделенным ранее критериям оптимальности. Поэтому можно переходить к реализации унифицированной модели.

2. Статическая модель на основе классического объектно-реляционного отображения

2.1. Методы классического объектно-реляционного отображения

Практическая реализация представленной модели тестирования в данном разделе продемонстрирована на примере применения классических методов (паттернов) объектно-реляционного отображения (ОРО). Таким образом, объектная модель приложения реализуется в среде реляционной СУБД в виде набора связанных реляционных отношений (таблиц). Такой подход наиболее оправдан, потому что РСУБД - самый популярный тип систем управления БД. Для реализации представленной модели использовалась среда разработки программных комплексов на основе организации метамодели объектной системы, описанной в работах [5-6]. Эта среда разработки получила название SharpArchitect RAD Studio и в качестве хранилища информации использует реляционную СУБД. Т.к. информационная система проектируется в понятиях объектно-ориентированной парадигмы, а реализуется в среде реляционной СУБД, то возникает так называемое «объектно-реляционное несоответствие», для преодоления последствий которого используются методы (паттерны, шаблоны) объектно-реляционного отображения. Наиболее часто используются паттерны для представления иерархии классов.

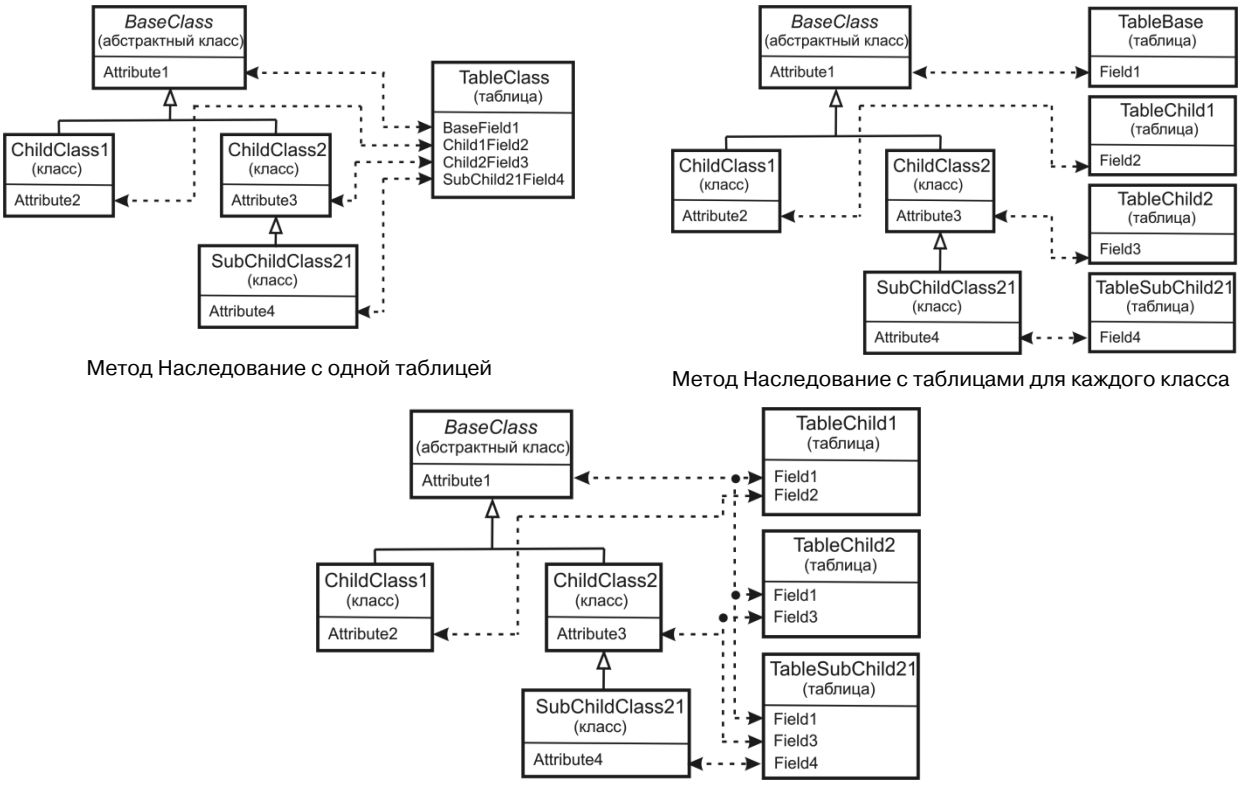

Рис. 2. Классические методы (паттерны, шаблоны) объектно-реляционного отображения, используемые для представления наследования классов в виде реляционной структуры (реляционных отношений)

В SharpArchitect RAD Studio реализуются три классических метода реализации объектно-ориентированного отношения наследования классов в виде реляционной структуры (реляционных отношений), представленные на Рис. 2 [3, 5].

Рассмотрим основные представленные методы более подробно. Метод Наследование с одной таблицей (Single Table Inheritance) физически представляет иерархию наследования классов в виде одной таблицы реляционной базы данных, столбцы которой соответствуют атрибутам всех классов, входящих в иерархию и позволяющих отобразить структуру наследования и минимизировать количество соединений, которые необходимо выполнить для извлечения информации. В данном методе каждому экземпляру класса соответствует одна строка таблицы. При создании объекта значения заносятся только в те столбцы таблицы, которые соответствуют атрибутам класса, а все остальные остаются пустыми (имеют null-значения).

Метод имеет достоинства:

- В структуре БД присутствует лишь одна таблица, представляющая все классы иерархии.
- Для извлечения экземпляров классов иерархии нет необходимости выполнять соединение таблиц.
- Перемещение полей из базового класса в производный (так же из производного в базовый) не требует внесения изменений в структуру таблицы.

Методу присущи недостатки:

- При изучении структуры таблиц БД могут возникнуть проблемы, ведь не все имеющиеся столбцы таблицы предназначены для описания каждого класса предметной области. Это усложняет процесс доработки системы в будущем.
- При наличии глубокой иерархии наследования с большим количеством атрибутов многие столбцы могут иметь пустые значения (null-маркеры). Это приводит к неэффективному использованию свободного пространства в БД. Однако современные СУБД способны

сжимать строки, содержащие большое количество отсутствующих значений.

• Итоговая таблица может оказаться слишком большой и содержать огромное число столбцов. Основной способ оптимизировать запрос (сократить время выполнения) – создать покрывающий индекс. Однако множество индексов и большое количество запросов к одной таблице способны привести к частым блокировкам, что будет оказывать негативное влияние на производительность программного приложения.

Альтернативным является метод Наследование с таблицами для каждого класса (Class Table Inheritance), представляющий иерархию классов по одной таблице для любого класса (как абстрактного, так и реализованного). Атрибуты класса отображаются непосредственно на столбцы соответствующей таблицы. При использовании данного метода ключевой является задача соединения соответствующих строк нескольких таблиц БД, представляющих единый объект предметной области.

Метод имеет следующие преимущества:

• В каждой таблице присутствуют лишь поля, соответствующие атрибутам определённого класса. Поэтому таблицы легки для понимания и занимают мало места на жёстком диске.

• Взаимосвязь между объектной моделью и схемой базы данных проста и понятна.

Однако имеются недостатки:

• При создании экземпляра определённого класса необходимо выполнить загрузку данных из несколько таблиц, что требует либо их естественного соединения, либо множества обращений к базе данных с последующим соединением результатов в оперативной памяти.

• Перемещение полей в производный класс или суперкласс требует изменения структуры нескольких таблиц.

• Таблицы суперклассов могут стать слабым местом в вопросах производительности, поскольку доступ к таким таблицам будет осуществляться слишком часто, что приведёт к множеству блокировок.

• Высокая степень нормализации может стать препятствием к выполнению незапланированных заранее запросов.

Метод Наследование с таблицами для каждого конкретного класса (Concrete Table Inheritance) представляет иерархию наследования классов, используя по одной таблице для каждого реализованного (неабстрактного) класса этой иерархии. С практической точки зрения данный метод предполагает, что каждый экземпляр класса (объект), который находится в оперативной памяти, будет отображён на отдельную строку таблицы. При этом каждая таблица в нашем случае содержит столбцы, соответствующие атрибутам как конкретного класса, так всех его предков.

Преимуществами являются то, что:

• Каждая таблица не содержит лишних полей, вследствие чего ее удобно использовать в других приложениях, в которых не использован инструмент объектно-реляционного отображения.

• При создании объектов определённого класса в памяти приложения и извлечения данных из реляционной базы данных выборка выполнятся из одной таблицы, т.е. не требуется выполнять реляционные соединения.

• Доступ к таблице осуществляется только в случае доступа к конкретному классу, что позволяет снизить количество блокировок, накладываемых на таблицу, и распределить нагрузку на систему.

При этом имеются недостатки:

• Первичные ключи могут быть неудобны в обработке.

• Отсутствует возможность моделировать отношения (связи) между абстрактными классами.

• Если атрибуты классов перемещаются между суперклассами и производными классами, требуется изменять структуру нескольких таблиц. Эти изменения будут не так часты, как в случае наследования с таблицами для каждого класса, однако их нельзя игнорировать (в отличие от метода Наследование с одной таблицей, в котором эти изменения вообще отсутствуют).

• Если в суперклассе изменить определение хотя бы одного атрибута (например, поменять тип данных), это потребует изменить структуру каждой таблицы, представляющей производный класс, поскольку поля суперк-

ласса дублируются во всех таблицах его производных классов.

• При реализации метода поиска данных в абстрактном классе потребуется просматривать все таблицы, представляющие экземпляры производных классов. Это требует большого количества обращений к базе данных.

Выбор соответствующего метода ОРО зависит от исходной логической модели, т.е. от иерархии классов предметной области. При этом одновременно могут использоваться два и более метода ОРО, что связано с необходимостью оптимизировать структуру реляционной БД и сократить количество используемых таблиц, что позволит увеличить скорость выполнения запросов на выборку данных.

Описав используемые в SharpArchitect RAD Studio паттерны объектно-реляционного отображения, которые доступны разработчику, можно приступить к реализации унифицированной модели тестирования.

2.2. Унифицированная модель тестирования на основе объектно-реляционного отображения

С целью упрощения процесса реализации разобьем три имеющиеся на Рис. 1 иерархии классов в соответствии с имеющимися методами ОРО. Результат представлен на Рис. 3.

Метод Наследование с одной таблицей будет использован для иерархии классов Post, ExperiencePost (ScientificPost). В результате предполагается, что в РБД будет создана одна единственная таблица (отношение БД), в которой сохраняются экземпляры всех перечисленных неабстрактных классов. Для иерархии классов Contragent, Worker (Company), Employee, Manager используется метод Наследование с таблицами для каждого класса. Т.е. для всех классов независимо от того, абстрактный он или реализованный, будет создана отдельная таблица РБД. Класс Address является абстрактным и не имеет ассоциаций с другими классами модели, поэтому для него не будет создаваться отдельная таблица в РБД. А для дочерних классов будут созданы две таблицы (по одной для каждого наследника). Т.е. для иерархии Address, CompanyAddress (EmployeeAddress) применен метод Наследование с таблицами для каждого конкретного класса. Для остальных классов, не входящих в описанные иерархии, будет создано по отдельному отношению.

Одной из основных особенностей SharpArchitect RAD Studio является поддержка множественного наследования, реализуемого с помощью интерфейсов (interface) языка C#, что подробно описано в [5]. Применяемый язык C# не поддерживает такой синтаксической конструкции как ассоциация. Для представления бинарных ассоциаций независимо от множественности используются свойства (property), содержащие одно значение или коллекцию значений.

Множественные n-арные ассоциации представляются отдельным классом, атрибуты этих

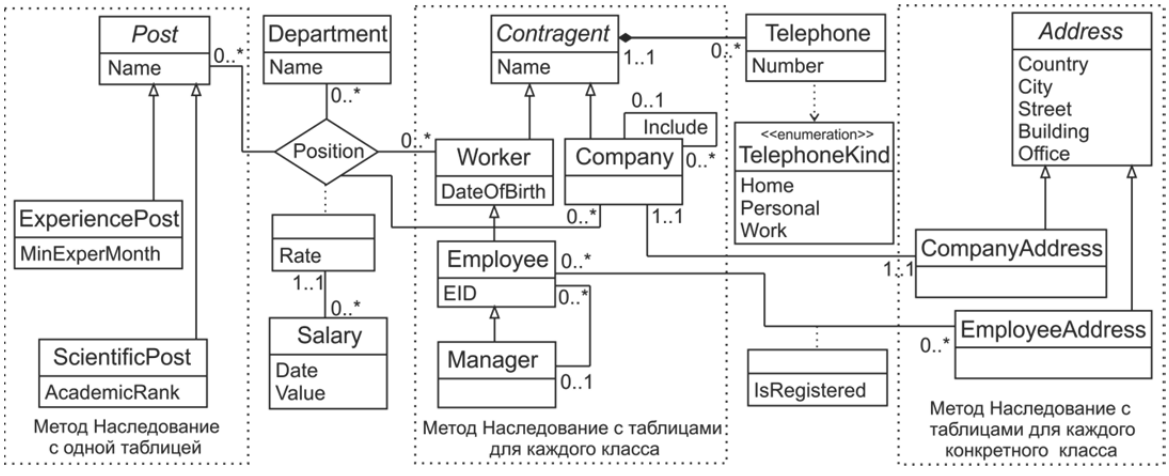

Рис. 3. Применение классических методов ОРО для реализации унифицированной модели тестирования инструментов разработки объектно-ориентированных приложений

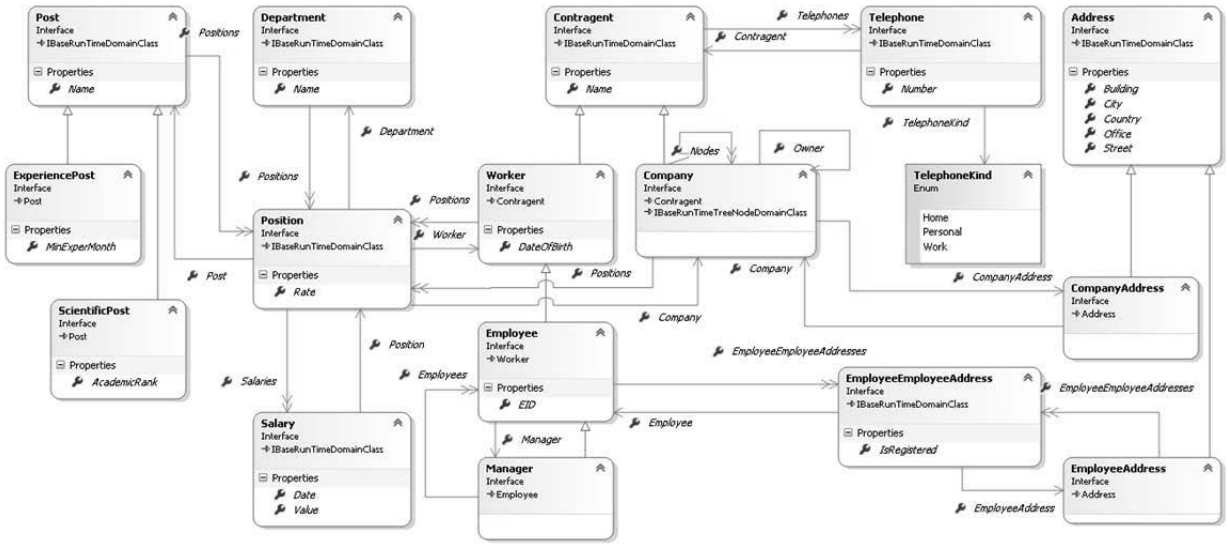

Рис. 4. Унифицированная модель тестирования инструментов разработки объектно-ориентированных приложений, реализованная в SharpArchitect RAD Studio на языке C#

ассоциаций (как и атрибуты бинарных ассоциаций) превращаются в свойства классов. Для упрощения поиска информации и извлечения все ассоциации являются двунаправленными, т.е. на обоих краях в соответствующих классах присутствуют свойства, тип которых соответствует классу противоположного конца ассоциации. Все описанные выше рассуждения графически представлены на Рис. 4.

При реализации использовались интерфейсы языка C#, поэтому невозможно курсивом выделить абстрактные классы. Двунаправленные ассоциации показаны соответствующими стрелками, соединяющими классы. При реализации ассоциаций использовался следующий подход. Со стороны «один» создавалось свойство (property), типом которого является список (C# тип IList<>), содержащий элементы, типом которых является класс, расположенный со стороны «ко-многим». При этом со стороны «ко-многим» в классе объявляется свойство, типом которого является класс, расположенный со стороны «один». Ассоциация типа «многие-ко-многим» (без атрибутов) может быть представлена двумя списками, объявленными в противоположных классах. В среде разработки SharpArchitect RAD Studio имеется ряд базовых классов, в которых реализован наиболее общий функционал. Например, класс IBaseRunTimeDomainClass является корневым для всех классов предметной области. Для реализации древовидной структуры достаточно унаследоваться от IBaseRunTimeTreeNodeDomainClass. В момент кодогенерации автоматически будут генерироваться дополнительные атрибуты Nodes и Owner, позволяющие сохранять ссылки на вложенные и родительский узлы, соответственно. Именно таким способом реализуются рекурсивные ассоциации. Для представления перечислений и множеств используется синтаксическая конструкция enum.

Теперь, применяя классические методы ОРО, было получено реляционное представление унифицированной модели. На Рис. 5 изображён результат.

Рисунок требует пояснения. Для всех должностей, представленных тремя классами Post, ExperiencePost и ScientificPost, создана одна единственная таблица Post, в которой присутствуют все атрибуты классов. Дополнительно в таблице присутствует столбец OID, представляющий объектный идентификатор (первичный ключ в реляционной модели). Столбец ObjectType содержит идентификатор класса, объекты которого сохранены в виде строк таблицы. Это значение используется приложением для создания класса объектно-ориентированного языка программирования и для загрузки значений атрибутов.

При реализации метода Наследование с таблицами для каждого класса были созданы

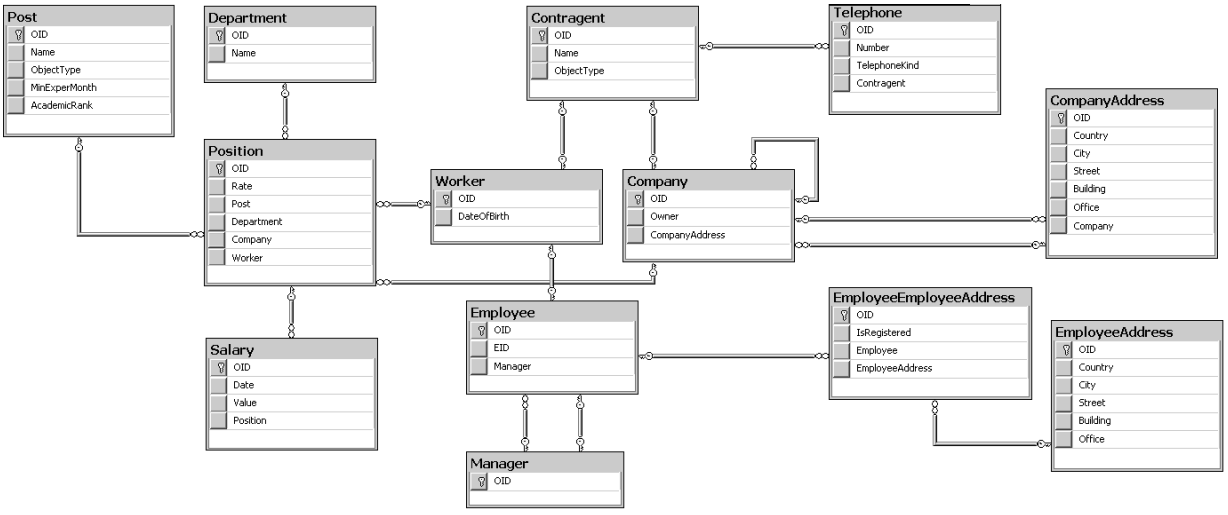

Рис. 5. Реляционное представление реализации унифицированной модели тестирования в SharpArchitect RAD Studio

таблица Contragent для абстрактного класса и таблицы Worker, Company, Employee, Manager для реализованных классов. Таким образом, экземпляры классов физически хранятся в нескольких таблицах базы данных. А экземпляр класса Manager хранится во всех таблицах.

При реализации метода Наследование с таблицами для каждого конкретного класса, применимого для классов Address, CompanyAddress и EmployeeAddress, были созданы две таблицы: CompanyAddress и EmployeeAddress, т.к. класс CompanyAddress является абстрактным. Все атрибуты абстрактного класса физически сохраняются в таблицах конкретных классов.

Для n-арной ассоциации Position создана отдельная таблица также, как и для бинарной ассоциации, соединяющей классы Employee и EmployeeAddress, для которой создана таблица EmployeeEmployeeAddress, содержащая внешние ключи.

Отметим, что для перечисления TelephoneKind отдельная таблица не создавалась. Использовался подход представления значений перечисления в виде битовой маски и сохранения полученного значения в виде целого числа там, где используются соответствующие атрибуты. Так в таблице Telephone имеется столбец TelephoneKind, SQL-тип которого Integer.

Проанализировав перечисленное выше, можно утверждать, что представленная на Рис. 5 реализация, созданная в среде разработки SharpArchitect RAD Studio, полностью соответствует унифицированной модели тестирования инструментов разработки объектно-ориентированных приложений, представленной на Рис. 1.

3. Статическая модель на основе объектно-атрибутного подхода

3.1. Основные элементы подхода

На протяжении последних 30 лет доминирующими являются реляционные базы данных (БД) [7]. Несмотря на это реляционная модель перестала удовлетворять потребностям времени: ограниченность в средствах описания онтологии не позволяет применять ее в интеллектуальных системах, трудность масштабирования затрудняет реализацию подобных СУБД на параллельных и распределенных вычислительных системах. В ответ на создавшийся кризис СУБД [8] было предложено множество путей выхода из него – объектно-ориентированный (ОО) подход к БД, древовидная БД, сетевая модель, движение NoSQL (Amazon, Google). Однако ни один из них, кроме ОО-модели, которой также присуща масса недостатков [9], не сумел составить достаточно серьезной конкуренции реляционной модели. В разделе представлен новый способ построения БД, основанный на объектно-атрибутном (ОА) подходе к организации вычислительного процесса и структур данных [10, 11]. ОА-БД относится к классу сетевых

Рис. 6. Основные типы связей в ОО-модели БД

(графовых) – БД представляет собой граф (или ОА-граф), вершинами которой являются описания объектов и смысловых связей (отношений), а дуги отображают ассоциации между ними. Для иллюстрации такого способа организации БД покажем, каким образом можно реализовывать основные типы связей между сущностями ОО- и реляционной модели БД и приведем пример реализации небольшой БД на основе иерархии с Рис. 1.

Основой для проводимого анализа выберем UML-диаграмму классов [12], с помощью которой можно задать основные сущности ОО-модели: класс, атрибут класса, операции класса; а также следующие типы связей: зависимость, обобщение, множественное наследование, ассоциация, агрегация, композиция. Не следует забывать и о понятиях кратность отношения и роль (Рис. 6). Теперь опишем методы реализации каждого типа отношений в ОА-БД. Мы будем применять ОА-язык – специализированный язык для описания ОА-графов [10, 11].

Начнем с описания класса в ОА-БД. Итак, класс (шаблон, по которому создается объект) представляет собой набор полей и методов. В ОА-системе объект представляется в виде информационной капсулы (ИК); каждому полю объекта соответствует информационная пара (ИП), входящая в эту ИК. Имена атрибутов (индексу атрибута сопоставляется уникальная мнемоника) будут являться атрибутами ИП, в нагрузке ИП будет размещаться значение поля класса. Методы в объекте – это ссылка на подпрограмму. В ОА-системе программа представляет собой последовательность ИП (милликоманд). Эта последовательность может передаваться на функциональные устройства (ФУ), которые под управлением потока милликоманд производят определенные вычисления, преобразование данных и ввод/вывод. Опишем на ОА-языке класс на Рис. 6, а : Человек{**Пол**=М **ДатаРожд**=07.02.1970 **ФамилияИмя**= ИвановИван **ВыдатьВозраст**= ВыдатьВозрастПрог **СохранитьТекущийДоход**= СохранитьТекущДоходПрог **ВыдатьОбщийДоход**= ВыдатьОбщийДоходПрог}, где в нагрузках последних трех ИП находятся указатели на информационные капсулы (ИК) программ.

Связь-обобщение (подклассы) реализуется с помощью добавления ИП в ИК описания объекта. Атрибут добавленной ИП идентифицирует подкласс, в нагрузке помещается – указатель на ОА-граф, описывающий подкласс. Например, для Рис. 6, б) связь задается так: ЧеловекИзУниверситета{**Студент**= СтудентУк {…} **Преподаватель**=ПреподавательУк {…}}, где вместо многоточия помещается описание объекта, относящегося к подклассам Студент и

Преподаватель; Студент и Преподаватель – мнемоники атрибутов. Мнемоники СтудентУк и ПреподавательУк – это указатели на капсулу с описанием студента и преподавателя соответственно. Наследование в ОА-БД осуществляется с помощью копирования ОА-графа описания объекта-родителя и добавления в корневую ИК одной или нескольких ИП, описывающих новые поля и методы. Например, на Рис. 6, б представлено множественное наследование: класс Студент-Преподаватель наследует свойства классов Студент и Преподаватель. На ОА-языке такое наследование будет описано так: СтудентПреподаватель{СтудентУк, ПреподавательУк}. Если в описании капсулы в ОА-языке присутствует мнемоники указателя на ИК (но не в нагрузке ИП), то данная ИК (и весь ОА-граф, в который она входит) копируется в формируемую ИК. Таким образом, в ИК под именем Студент-Преподаватель будут скопированы все ИП из капсул Студент и Преподаватель.

Связи-ассоциации задаются с помощью ИП, добавленную в ИК описания объекта: в нагрузке ИП находится указатель на ИК с описанием объекта, с которым происходит ассоциация, в атрибуте – мнемоника роли. В ОА-подходе связь один к одному задается с помощью двух ИП, ссылающихся на ИК описаний объектов. Например, между объектом класса Человек и Университет (Рис. 6, в) существует связь, в которой Человек выступает в роли Работник, а Университет в роли Работодатель. На ОА-языке эта связь описывается следующим образом: Человек{ … **Студент**= Университет **Работник**=Университет … } Университет{ … **Наниматель**= Человек **Обучающий**=Человек … } Человек={**Студент**= Университет **Работник**=Университет}. Человек может работать в нескольких университетах, а университет нанимать несколько человек на работу (связь n-арной кратности), тогда ОА-граф будет выглядеть следующим образом: Человек1{… **Работник**=Университет1 **Работник**=Университет2 …} Человек2{… **Работник**=Университет1 **Работник**=Университет2 …} Университет1{… **Нанимает**= Человек1 **Нанимает**= Человек2 …} Университет2{…**Нанимает**= Человек1 **Нанимает**= Человек 2…}. Как видно, название роли в данном случае выступает как имя (атрибут) связи.

Композицию (неотъемлемая часть объекта) можно реализовать, например, так: составить список атрибутов, в нагрузке которых имеется указатель объекта, неотъемлемого от своего родителя. Тогда если в удаляемой ИК встречаются ИП с такими атрибутами, то и капсулы, указатель на которые хранится в нагрузке, также удаляются.

Сложные отношения и отношения-классы реализуются с помощью ИК. Например, необходимо описать отношение «действие» (такая необходимость возникает в задаче составления БД, исходя из текста на естественном языке), в котором участвуют достаточно много объектов: субъект (тот, кто производит действие), объект (то, на что направлено действие), посредник (например, инструмент, которым действие производят) и т.д. Такое отношение может выглядеть так: Действие{**Субъект**={...} **Объект**={...} **Посредник**={...} **Адресат**={...} **ОбразДействия**={...}...}.

Отметим, что ОА-подход построения БД реализует все типы связей, существующие в ОО- и реляционном подходах, поэтому может быть применим для реализации приложений любых прикладных областей.

3.2. Унифицированная модель тестирования на основе ОА-подхода

Для иллюстрации предложенного принципа реализуем структуру БД унифицированной модели тестирования, описанную с помощью UML-нотации (диаграммы классов), представленную в разделе 1. На Рис. 7 на ОА-языке (// - символы комментария) представлено описание БД, модель которой изображена на Рис. 1. Компилятор ОА-языка входит в состав ОА-среды программирования и моделирования, которая позволяет моделировать вычислительный процесс в ОА-системе.

В листинге (Рис. 7) приводится ОА-программа инициализации ОА-БД. Рассмотрим его подробно. В первой строке ОА-программы осуществляется создание функционального устройства (ФУ), обрабатывающего ОА-граф. Далее происходит инициализация мнемоник атрибутов и констант (знак «#» обозначает инициализацию константы, изолированная мнемоника – инициализация атрибута). Милли-

команда (ИП, адресованная для ФУ) с мнемоникой «DB.Set» задает операцию «установить указатель на ОА-граф для ФУ «DB» - после знака «=», идет описание ОА-графа на ОА-языке. Знак «>» обозначает начало новой записи в списке

```
  NewFU={Mnemo="DB" FUType=FUGraph}
  // Инициализация атрибутов
  Department Post
  ExperiencePost MinExperMonth
  ScientificRank AcademicRank
  Position Rate Salary Date
  Value Worker DateOfBirth
  Company Contragent Employee
  EID Manager Include Telephone
  Number TelephoneKind
  Home#1 Personal#2 Work#3
  Name Adress Country
  City Street Building
  Office CompanyAdress
  EmployeeAdress IsRegistred

  DB.Set=
  >Position1{ // Position1 –метка ИК
    Post={Name="Full professor"
    ExperiencePost={MinExperMonth=12}}
    Department={
      Name="Computer science"}
    Department={
     Name="Applied mathematics"}
    Salary={Date="1.10.2014"
     Value=4000}
   Worker={
    DateOfBirth="13.04.1968"
    Telephone={Number=1234567
      TelephoneKind=Personal}
    Telephone={Number=7654321
      TelephoneKind=Home}
   }
    Employee=Employee1{// Метка
     EID=12876  Manager=null
     EmployeeAdress=
      {IsRegistred=true
       Adress={Country="USA"
     City="New York" Street="Street 20"
     Building=10 Office=1}}
   Company=
    >{Name="IBM"
      Telephone={Number=1111111
        TelephoneKind=Personal}
      Telephone={Number=2222222
        TelephoneKind=Work}
     CompanyAdress={Country="USA"
     City="New  York"  Street="Street  10"
Building=10 Office=1}
     }}}
  >Position2{…// Вторая запись БД
```

Рис. 7. Листинг ОА-БД, модель которой представлена на Рис. 1

(список организуется из списка сущностей «Position», этот список «сшивает» все записи в единую БД). После инициализации с помощью ФУ «DB» можно производить модификацию ОА-графа и поиск информации в нем. ОА-граф, созданный в среде ОА-программирования и моделирования представлен на Рис. 8 (для вывода ОА-графа в среде ОА-программирования и моделирования реализован специальный программный компонент). На рисунке в фигурных скобках помещаются индексы информационных капсул (ИК): т.к. ОА-граф представляет собой граф произвольной топологии, а на экране отображается только его остов, то по индексам можно отследить все связи между вершинами, не вошедшими в остов, – в этом случае после знака «=» (обозначение нагрузки ИП) указывается индекс той ИК, на которую указывает ссылка в нагрузке (например, «ID_14»).

ОА-подход обеспечивает такие ключевые качества БД как: описание любой онтологии, масштабируемость вычислений, целостность БД, эргономичность (запросы к БД вводятся на естественном языке) [13], высокая скорость поиска в БД (для ОА-БД разработана методика поиска с применением индексации ИК).

ОА-БД относится к классу сетевых (графовых), когда описание предметной области представляет собой множество вершин (семов), обозначающих объекты и дуги, задающие связи между объектами. Сетевая БД не имеет ограничений на топологию, и поэтому потенциально имеет возможность описывать любую предметную область. Этим она качественно отличается от реляционной и ОО (фреймовой) моделей: в первой из них практически невозможно описать предметную область с большим числом типов связей (для каждого типа необходимо создавать собственное отношение (таблицу), что очень громоздко), у второго существует ограничение на топологию (граф типа «дерево»).

Масштабируемость ОА-БД достигается благодаря применению управления вычислениями с помощью потока данных (dataflow) [14]. Dataflow позволяет «отвязать» адресацию данных от оперативной памяти компьютера: доступ к данным производится через функциональные устройства (ФУ), которые являются устройствами виртуальными, т.е. жестко не привя-

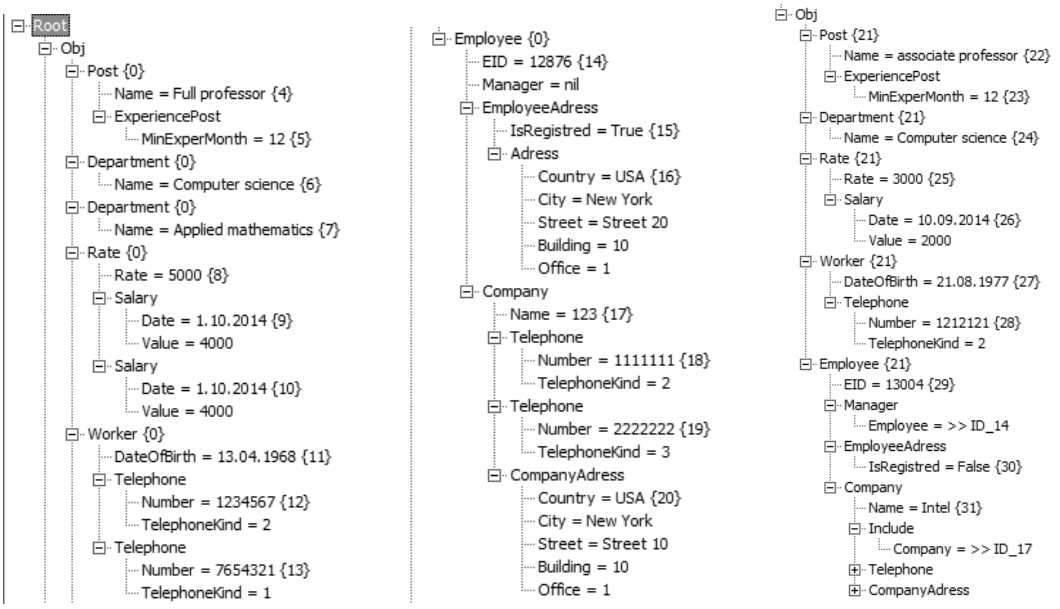

Рис. 8. Представление БД в виде ОА-графа

занными к аппаратуре вычислительных узлов. Совокупность ФУ формирует так называемый ОА-образ – виртуальную систему, производящую вычисления. Затем ОА-образ «накладывается» на конкретную вычислительную систему (ВС) путем распределения ФУ по вычислительным узлам ВС и настройки самих ФУ и маршрутизаторов, ответственных за пересылку информационных пар (ИП) между вычислительными узлами. При изменении конфигурации ВС необходимо только произвести перераспределение ФУ между вычислительными узлами и задать новые настройки маршрутизаторов [10].

Целостность ОА-БД обеспечивается благодаря формату и технологии обработки ОА-графа. Так, модификация ОА-БД заключается в добавлении или удалении некоторых ИП из ИК, составляющих ОА-граф. Допустим (Рис. 9), в ИК имеется ИП с атрибутом «1», затем мы добавили туда же ИП с атрибутом «2». И теперь уже можно производить поиск данных либо по атрибуту «1», либо по атрибуту «2» независимо: если мы ищем данные с атрибутом «1», то атрибут «2» мы игнорируем, и наоборот. Таким образом, с помощью анализа атрибутов ИП можно в одном ОА-графе выделять несколько подграфов… Т.е. мы можем менять структуру БД без опасения за нарушение её целостности. В результате отпадает необходи-

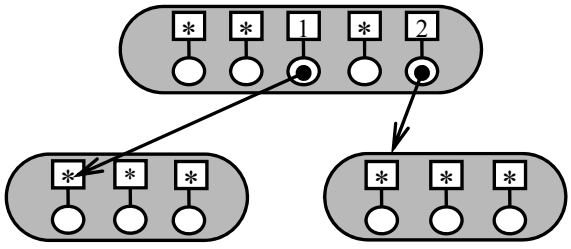

Рис. 9. Обеспечение целостности ОА-БД при ее модификации

мость в трудоемком рефакторинге, без чего практически нельзя обойтись при серьезной модификации ОО-БД.

Эргономичность обеспечивается благодаря тому, что запросы к ОА-БД выполняются на естественном языке: запрос преобразуется в ОА-граф запроса и далее производится определение, является ли запрос подграфом ОА-графа БД. О технологии преобразования текста не естественном языке в ОА-граф рассказывается в [13]. Ответы на запросы так же могут выводиться на естественном языке.

Подобно ОО-подходу, ОА-БД обеспечивает широкие возможности абстракции данных. Например, ОА-граф можно выстроить по топологии «дерево»: тогда ИК высшего уровня инкапсулирует информационную структуру под ней (в ИК помещаются основные характери-

стики нижележащего уровня). Если ОА-граф имеет произвольную топологию, то в нем можно выделить подграфы (например, по принципу описания какого-либо объекта, множества объектов или явления) и включить в ОА-граф ИК, описывающие основные характеристики таких подграфов – теперь при необходимости пользователь может работать с данной ИК без обращения к подробному описанию объекта.

В ОА-БД имеется возможность производить достаточно быстрый поиск информации, благодаря индексации ИК. Как уже говорилось ранее, поиск в ОА-БД заключается в отыскании подграфа (подграфов) в ОА-графе БД, совпадающих с графом-запросом [13]. Процесс поиска упрощается благодаря тому, что вершины ОА-графа, как бы, помечены с помощью ИП, расположенных в ИК. Поэтому можно предложить следующую методику поиска подграфа, который состоит из нескольких этапов. Первый – индексация всех вершин ОА-графа БД и запроса. Вторая – составление двух списков атрибутов ИП: первый – атрибутов, встречающихся в ОА-граф БД; второй – в ОА-граф запроса. Если какие-то атрибуты из запроса не присутствуют в БД, то поиск прекращается и пользователю выдается сообщение об отрицательном результате. Иначе в графе БД выделяются все ИК, где встречаются атрибуты из графа-запроса. Третий – группировка найденных в ОА-графе ИП по капсулам: если у найденных ИП имеется одинаковый индекс ИК (т.е. они находятся в одной капсуле), то они объединяются в группу. Четвертый – сопоставление групп: если группа не совпадает ни с одной группой из графа-запроса, то она удаляется. Если в результате все группы были удалены, или количество групп стало меньше, чем в графе-запросе; то поиск завершается с отрицательным результатом. Пятый – сопоставление оставшихся групп с ИК графа-запроса. Для ускорения поиска нужных атрибутов можно использовать хеширование, которое значительно повышает скорость расчетов. Такой алгоритм имеет временную сложность порядка N*M, где N – число ИП в БД, M – число ИП в графе-запросе.

Следует отметить, что теоретически ОА-СУБД имеет достаточно хорошие перспективы, т.к. обладает множеством положительных качеств СУБД. Однако следует отметить и ее недостатки: во-первых, чрезмерный расход оперативной памяти (в ОА-БД для каждой связи выделяются две ИП, т.к. дуги ОА-графа должны быть двунаправленными); во-вторых, благодаря возможности добавления новых структур данных без нарушения старых в ОА-БД могут возникнуть структуры данных, которые уже никому не нужны и о которых разработчики уже давно забыли, но которые занимают оперативную память и замедляют поиск нужной информации.

4. Статическая объектная модель на основе матричного подхода

4.1 Основные принципы подхода

В матричной универсальной объектно-реляционной базе данных (МУОРБД) базовая реляционная модель неизменна, что следует из объявленной универсальности в названии [15-17].

Исходя из матричной структуры, которая соответствует пятимерному пространству [15-16] вытекает на сегодняшний день принцип минимизации субъективного разделения информации [18]. А оптимизация обработки информации достигается в том, что необходимо максимально задействовать математический аппарат адресной структуры и реляционной алгебры [19]. Для собственной СУБД МУОРБД этот инструментарий ещё больше расширен, максимально нацеливаясь на матричную структуру [20].

Для реализации логической модели, представленной на Рис. 1, внутри МУОРБД нужно всего лишь три объекта, которые реально существуют: человек, компания и географический объект.

Руководство между индивидуумами реализовано через руководство отделом компании. При необходимости можно установить циклическую связь любого уровня и на сущность Индивид.

Домены – это справочник значений, имеющий наименование и сокращение. Для доменов может быть определена зависимость, как между собой, так и от сущностей, которые их используют [21].

Отдельно требуется описать принципы организации связей между сущностями в МУОРБД (аналог ассоциаций в UML). В первую очередь

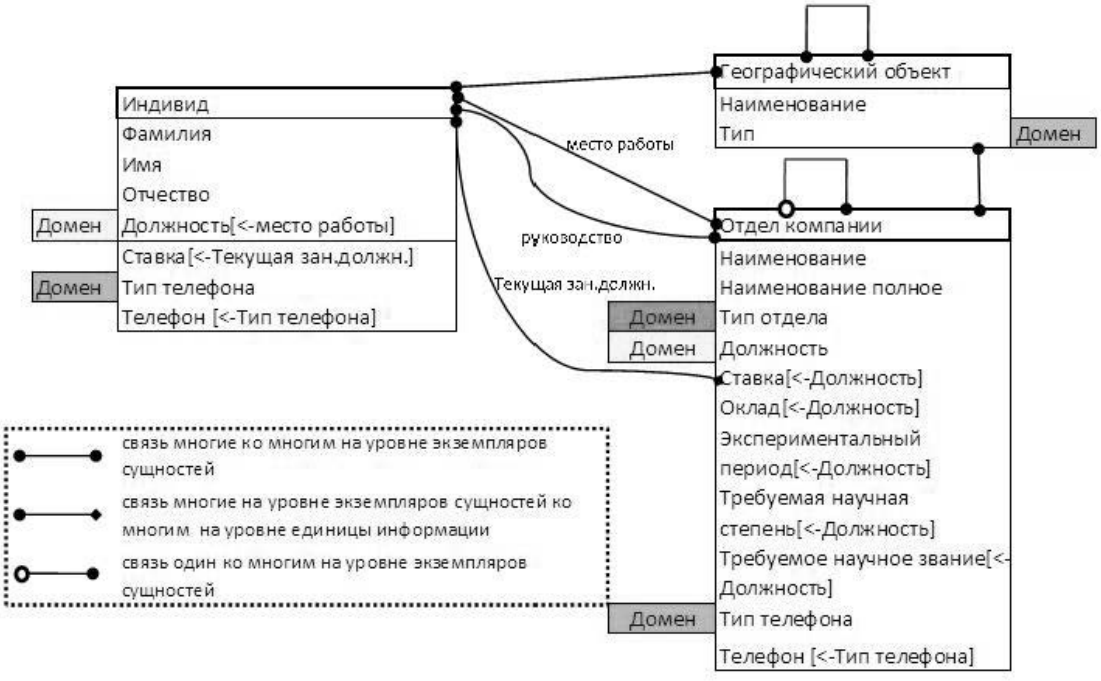

Рис. 10. Схема реализации модели тестовой предметной области в МУОРБД

необходимо отметить, что в отличие от реляционных БД в МУОРБД базовой связью является «многие-ко-многим» [16], а связи «один-ко-многим» и «один к одному» образуются установлением с одного из концов ограничения на уникальность. Также значительно расширен диапазон самих связей. Т.к. вместо первичного ключа используется адрес местоположения в пятимерном пространстве, то ссылка может иметь различный уровень: вся сущность (таблица), экземпляр сущности (строка, стандартная связь для реляционных БД), единица информации, элемент единицы информации. Для распределённых БД имеется возможность сделать ссылку на всю БД.

Например, на Рис. 10 Индивид может относиться к отделу – связь на уровне экземпляра сущности, может быть руководителем - связь в обратную сторону, также на уровне экземпляра сущности, и относиться к конкретной единице штатного расписания, выполняя в конкретный момент соответствующие обязанности по определённой должности – связь на уровне единицы информации.

Характерная отличительная черта МУОРБД – это синтез метаинформации данных [22]. Основной смысл состоит в одинаковом подходе как к данным пользователя, так и к метаинформации, включающей всю информацию о структуре базы данных. Таким образом, имеем возможность наложить весь аппарат реляционной алгебры и объектно-ориентированного подхода на саму метаинформацию. На Рис. 10, например, в сущности Отдел компании для полей Ставка, Оклад, Экспериментальный период, Требуемая научная степень и Требуемое научное звание установлено ограничение на использование, только после указания Должности и в подчинении к соответствующему элементу единицы информации. Для сущности Отдел Индивид для поля Ставка установлено ограничение на использование, только после указания Текущая зан.должн. и в подчинении к соответствующему элементу единицы информации, который определяется из списка должностей, конкретного Отдела компании.

4.2. Унифицированная модель тестирования на основе матричного подхода

Наследование - это ключевая особенность объектно-ориентированной парадигмы. Демонстрация реализации наследования на унифицированной тестовой модели представлена на Рис. 11. Создана сущность Worker, которая наследует все поля сущности Индивид. Для

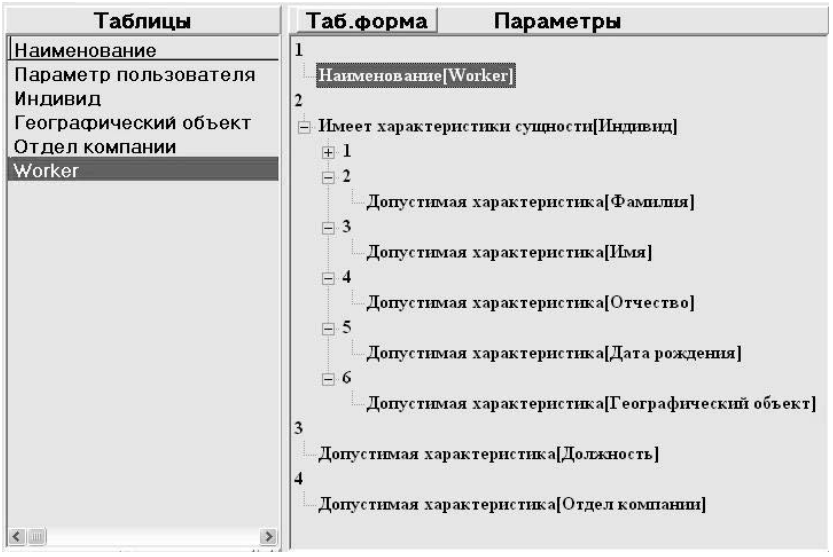

Рис. 11. Пример организации наследования в МУОРБД

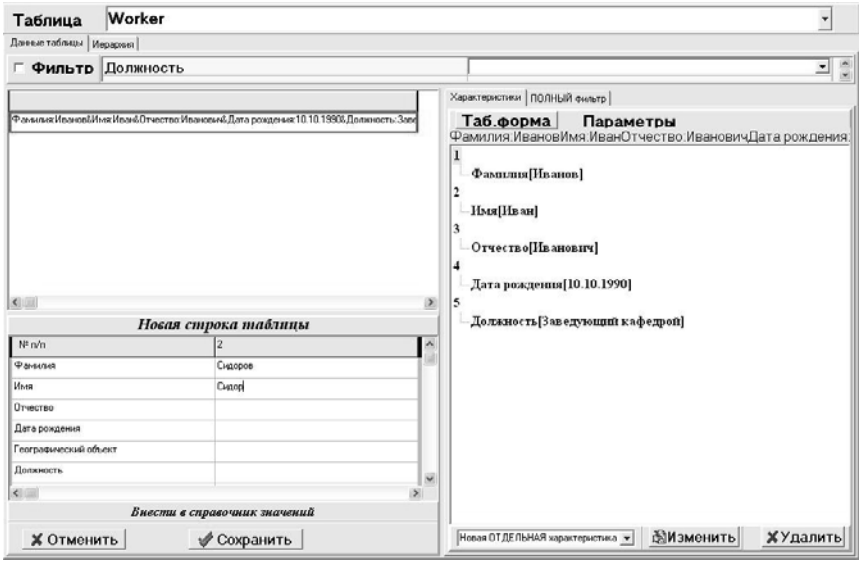

Рис. 12. Добавление новой записи и отображение уже имеющейся в сущности Worker, унаследовавшей поля от сущности Индивид

этого в параметрах сущности Worker добавлена единица информации «Имеет характеристики сущности» со значением-ссылкой на сущность Индивид.

В результате, при внесении и отображении информации сущности Worker доступны все поля базовой сущности Индивид Рис. 12.

При таком подходе механизм наследования может иметь неограниченную глубину. В базе данных для этого всё уже сделано, необходим лишь учёт его в приложении. Пример представлен на Рис. 13.

При внесении информации параметр ссылается уже на сущность, где атрибут был инициализирован. Что делает структурную безопасность информации от цепочки связей наследования. Наследование работает на уровне первичного описания сущности и добавлении новой информации. После внесения информации её безопасность передаётся в механизм целостности ссылок, а редактирование при изменении цепочки наследования устанавливается в дополнительные параметры каскадной обработки, зависимой от наследования информации пользователя.

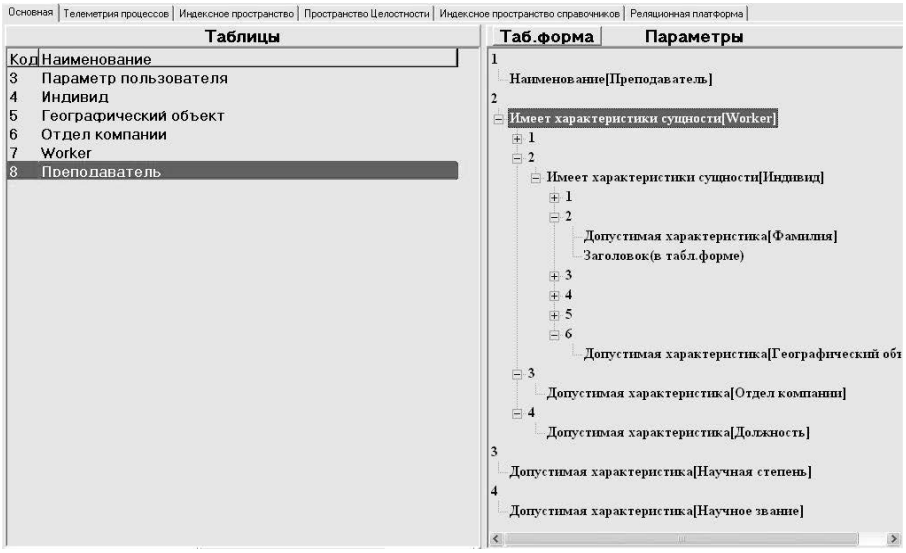

Рис. 13. Многоуровневое наследование в МУОРБД

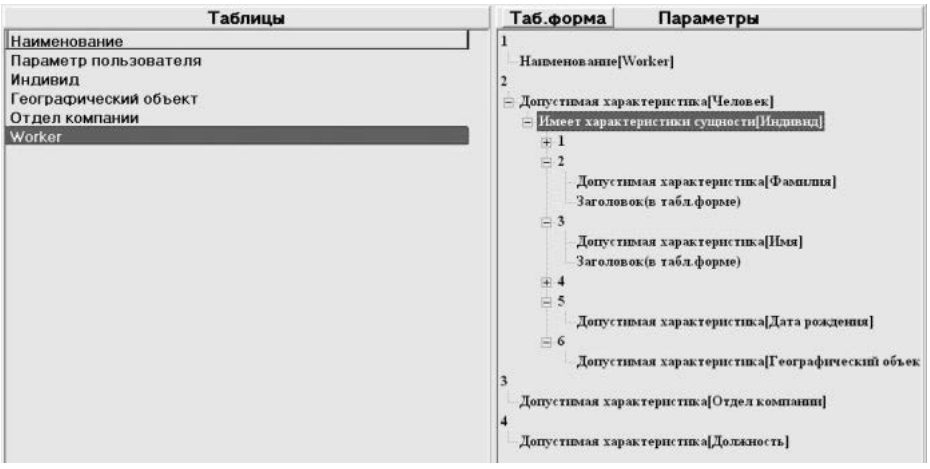

Рис. 14. Описание поля объектного типа в МУОРБД

Количество параллельных родительских сущностей также не ограничено и объявляется, и обрабатывается совершенно аналогично, как и для одного родителя.

Для добавления поля, обладающего набором характеристик другой сущности в МУОРБД, и реализации таким способом агрегации, необходимо создать новое поле (называем более обще - Допустимая характеристика) и приписать ему параметр Имеет характеристики сущности со значением-ссылкой на сущность другую сущность. На Рис. 14 у сущности Worker создана Допустимая характеристика Человек, и к этой записи добавлен параметр Имеет характеристики сущности со значением-ссылкой на сущность Индивид.

В результате оперирование информацией в сущности Worker в поле объектного типа производится для каждого значения в отдельности, но при указании базовой характеристики (Рис. 15).

Как видим, представление и работа с информацией в МУОРБД очень схожа с представлением и работой с объектами в объектно-ориентированных средах программирования таких, как Delphi, Visual Studio, Visual Basic и др. Именно в них, в 1990-хх годах отработан механизм работы с информацией в объектной форме. Здесь важным является самое широкое распространение и полная универсальность для задач любой предметной области. Не хватает лишь перехода от контекстного описания к

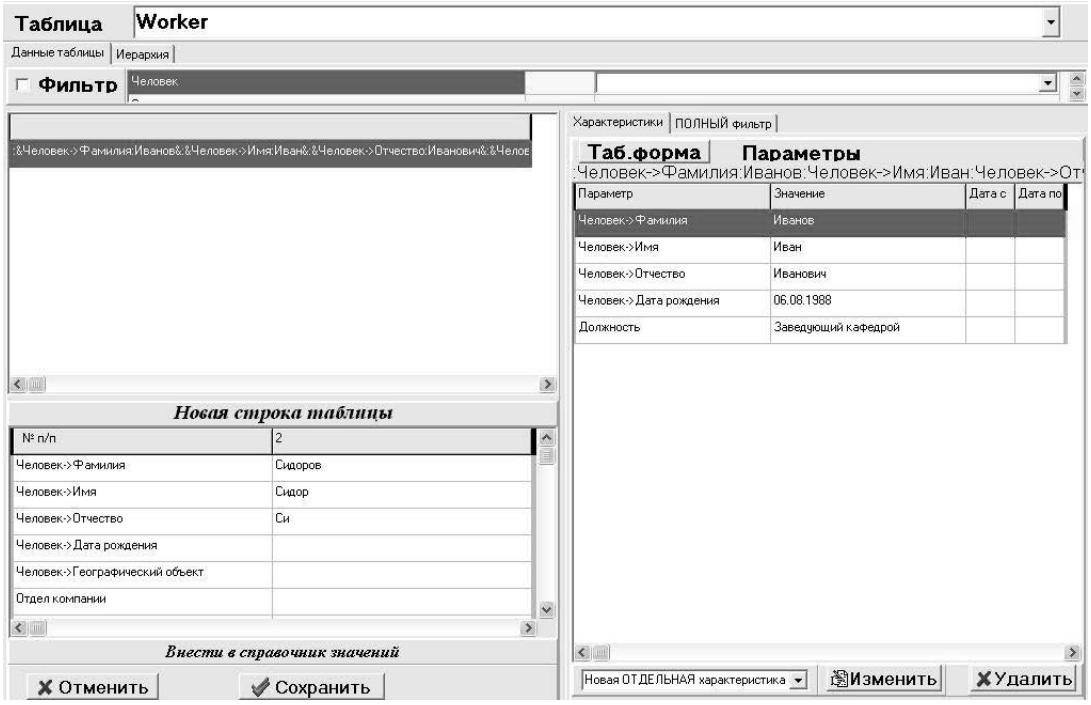

Рис. 15. Добавление и отображение информации экземпляров сущностей с полями объектного типа

строго структурированной форме как в базах данных. Это и было первостепенной задачей при разработке МУОРБД, где весь функционал объектно-ориентированного подхода реализован, а представление агрегированных параметров (Рис. 15) аналогично синтаксическому представлению объекта и его свойствам на языке С.

Используя синтез метаинформации и данных пользователя, появилась возможность наследования не только в абстрактной зоне описания сущностей, но и в области данных пользователя.

Исходя из определения баз данных и их главного объекта назначения – информации, можно описание сущностей самой базы данных пользователя отнести к такой же информации, как и информация его предметной области. Поэтому вполне логичным является и апробировать инструментарий объектно-ориентированного подхода, нацеленного на наложение логических связей единиц информации, аналогичных представлению информации в реальном мире, на данные пользователя.

В МУОРБД реализована система наследования характеристик экземпляров (строк) сущности (таблицы). В этом случае характеристики нового экземпляра (строки) точно совпадают с характеристиками экземпляра (строки), у которого он их наследует. На физическом уровне сохраняется лишь отметка о наследовании с параметром Имеет значения экземпляра сущности и значением-ссылкой на родительский экземпляр.

Впоследствии имеется возможность добавить, изменить или удалить характеристики этого экземпляра. В этом случае сохраняются все изменения и они являются собственностью только этого экземпляра (строк).

Для создания нового экземпляра (строки) таблицы на основе уже существующего необходимо выполнить действия, представленные на Рис. 16.

Объектно-ориентированные свойства структуры объектно-ориентированных баз данных имеют несколько другой смысл, в отличие от объектно-ориентированных сред программирования и иных форм реализации представления объектов реального мира в виртуальном пространстве. Так как во втором случае главной целью является полная идентичность отображения объекта, его свойств и поведения, а в

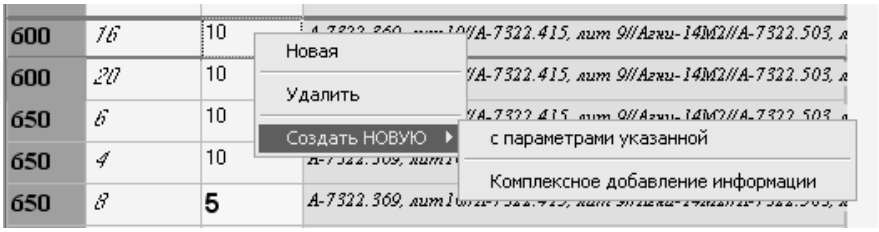

Рис. 16. Добавление нового экземпляра сущности, наследующего значения уже существующего

случае баз данных мы имеем дело не с самими объектами, а с информацией о них. Базы данных являются лишь информационной основой для реализации представления объектов в виртуальном мире. Исходя из этапов разработки программного обеспечения, построение логической, а затем физической моделей баз данных происходит сразу после изучения и анализа предметной области, и лишь на последующих этапах разрабатывается интерфейс реализации самих объектов в программном продукте, где на основе, заложенной в базе данных информации, моделируется представление и поведение объектов.

Таким образом, инструментарий объектно-ориентированного подхода в базах данных с одной стороны сужен, так, например, действие не воспроизводится, а состоит в описании алгоритма, по которому оно происходит, с другой стороны возможности расширены, т.к. стирается граница между абстрактной и реальной информацией, так как и та и другая являются объектом предназначения баз данных.

Заключение

В статье представлены три подхода к реализации статических моделей, проектируемых при реализации объектно-ориентированных приложений. В качестве примера использована унифицированная модель, представленная диаграммой классов языка UML. Описанные решения являются результатом многолетней работы авторов и протестированы на приложениях, реально используемых множествами пользователей. Дальнейшее развитие представленных подходов авторы видят в реализации принципов формирования графического интерфейса пользователя. Так же предполагается разработка формальных методов описания представленных решений.

Литература

1. Олейник П.П. Унифицированная модель тестирования инструментов разработки объектно-ориентированных приложений // Объектные системы – 2014 (Зимняя сессия): материалы IX Международной научно-практической конференции (Ростов-на-Дону, 10-12 декабря 2014 г.) / Под общ. ред. П.П. Олейника. – Ростов-на-Дону: ШИ (ф) ЮРГПУ (НПИ) им. М.И. Платова, 2014. - С. 25 - 35, http://objectsystems.ru/files/2014ws/Object_Systems_2014_Winter_session_Proceedings.pdf
2. Гамма Э. и др. Приёмы объектно-ориентированного проектирования. Паттёрны проектирования, СПб: Питер, 2001. – 368 с.: ил. (Серия «Библиотека программиста»).
3. Олейник П.П. Унифицированная модель для тестирования инструментов объектно-реляционного отображения // Объектные системы - 2011: материалы III Международной научно-практической конференции (Ростов-на-Дону, 10-12 мая 2011 г.) / Под общ. ред. П.П. Олейника. - Ростов-на-Дону, 2011. - С. 65-69., http://objectsystems.ru/files/Object_Systems_2011_Proceedings.pdf
4. Олейник П.П. Тестовая модель для обучения проектированию объектно-ориентированных баз данных // Объектные системы – 2014: материалы VIII Международной научно-практической конференции (Ростов-на-Дону, 10-12 мая 2014 г.) / Под общ. ред. П.П. Олейника. – Ростов-на-Дону: ШИ (ф) ЮРГПУ (НПИ) им. М.И. Платова, 2014. - С. 86-89., http://objectsystems.ru/files/2014/Object_Systems_2014_Proceedings.pdf
5. Олейник П.П. Элементы среды разработки программных комплексов на основе организации метамодели объектной системы // Бизнес-информатика. 2013. №4(26). – С. 69-76.
6. Олейник П.П., программа для ЭВМ "Унифицированная среда быстрой разработки корпоративных информационных систем SharpArchitect RAD Studio", свидетельство о государственной регистрации № 2013618212 от 04 сентября 2013 г.
7. Дейт, К. Дж. Введение в системы баз данных, 8-е издание.: Пер. с англ. — М.: Издательский дом "Вильяме", 2005.
8. Леонид Черняк. Смутное время СУБД // Открытые системы, №2, 2012 URL: http://www.osp.ru/os/2012/02/13014107/

9. Gabriel, R. Objects Have Failed: Notes for a Debate. (retrieved 17 May 2009). http://www.dreamsongs.com/Files/ObjectsHaveFailed.pdf
10. Салибекян С.М, Панфилов П.Б. ОА-архитектура построения и моделирования распределенных систем автоматизации // Автоматизация в промышленности. №11, 2011.
11. Сайт, посвященный работам в области ОА-архитектуры. URL: dataflow.miem.edu.ru
12. Новиков Ф.А., Иванов Д.Ю. Моделирование на UML. Теория, практика, видеокурс. – СПб.: Профессиональная литература, Наука и Техника, 2010. – 640 с.: ил. + цв. Вклейки (+ 2 DVD).
13. Салибекян С.М., Халькина С.Б., Тиновицкий К.Д. Объектно-атрибутный подход для семантического анализа естественного языка // Объектные системы - 2014: материал VI Международной научно-практической конференции (Ростов-на-Дону, 10-12 мая 2014 г.) / Под общ. ред. П.П. Олейника. - Ростов-на-Дону: ШИ ЮРГТУ (НПИ), 2014. - С. 80-86, http://objectsystems.ru/files/2014/Object_Systems_2014_Proceedings.pdf
14. Jurij Silk, Borut Robic and Theo Ungerer «Asynchrony in parallel computing: From dataflow to multithreading» Institut Jozef Stefan, Technical Report CDS-97-4, September 1997.
15. Микляев И.А., Матричная универсальная объектно-реляционная база данных // Объектные системы - 2010: Материалы I Международной научно-практической конференции. Россия, Ростов-на-Дону, 10-12 мая 2010 г / под общ. ред. П.П. Олейника. - Ростов-на-Дону, 2010. С. 34-39, http://objectsystems.ru/files/Sertificates/Object_Systems_2010_Proceedings.pdf
16. Микляев И.А., Концепция разработки матричной универсальной базы данных с поддержкой древовидной структуры единицы информации и её универсального приложения // Вестник Воронежского государственного университета. Серия: Системный анализ и информационные технологии. 2010. № 2., Воронеж, С. 101-108.
17. Микляев И.А. Матричная универсальная объектно-реляционная база данных на реляционной платформе: монография/ Сев.(Арктич.) федер. Ун-т им. М.В. Ломоносова. – Архангельск: ИД САФУ, 2014. – 226 с.
18. Микляев И.А., Черткова О.В. Синергетическое информационное пространство МУОРБД, // Объектные системы – 2011 (Зимняя сессия): материалы V Международной научно-практической конференции (Ростов-на-Дону, 10-12 декабря 2011 г.) / Под общ. ред. П.П. Олейника. – Ростов-на-Дону: ШИ ЮРГТУ (НПИ), 2011. С. 67-72, http://objectsystems.ru/files/2011WS/Object_Systems_2011_Winter_Session_Proceedings.pdf
19. Микляев И.А., Черткова О.В. Инструментарий оптимизации работы системы управления объектно-реляционной базы данных // Объектные системы – 2011 (Зимняя сессия): материалы V Международной научно-практической конференции (Ростов-на-Дону, 10-12 декабря 2011 г.) / Под общ. ред. П.П. Олейника. – Ростов-на-Дону: ШИ ЮРГТУ (НПИ), 2011. С. 72-76, http://objectsystems.ru/files/2011WS/Object_Systems_2011_Winter_Session_Proceedings.pdf
20. Микляев И.А., Свидетельство ОФЕРНиО №15405 (Объединённого фонда электронных ресурсов «Наука и образование») "Универсальный тип данных баз данных", 2010.
21. Микляев И.А., Свидетельство ОФЕРНиО № 14246 (Объединённого фонда электронных ресурсов «Наука и образование») Универсально приложение для матричной универсальной объектно-реляционной базы данных", 2010.
22. Жирнова М.А., Микляев И.А., Синтезирование мета-информации и данных Сборник докладов по материалам научно-практической конференции в рамках XLII Ломоносовских чтений / секция «Информационные системы и технологии в экономике и управлении» /Сборник докладов. – Федеральное государственное автономное образовательное учреждение высшего профессионального образования «Северный (Арктический) федеральный университет имени М.В. Ломоносова» филиал в г. Северодвинске Архангельской области институт судостроения и морской арктической техники, 2013. - с., стр. 38-43.

Микляев Иван Александрович. Доцент Института судостроения и морской арктической техники (Севмашвтуз) Северного (Арктического) федерального университета имени М. В. Ломоносова. Окончил Московский физико-технический институт в 1996 году. Кандидат физико-математических наук. Автор более 55 печатных работ и одной монографии. Область научных интересов: аэро- и гидродинамика и информационные структуры программного обеспечения.
E-mail: ivanmia1@rambler.ru

Олейник Павел Петрович. Системный архитектор программного обеспечения ОАО "Астон", доцент Шахтинского института (филиал) Южно-Российского государственного политехнического университета им. М.И. Платова. Окончил Шахтинский институт в 2004 году. Кандидат технических наук. Автор более 70 печатных работ и двух монографий. Область научных интересов: объектно-ориентированные системы, компонентное программирование, алгоритмы, системы построения пользовательского интерфейса, ERP – системы, объектно-реляционные преобразования, базы данных, шаблоны проектирования, распределённые системы. E-mail: xsl@list.ru

Салибекян Сергей Михайлович. Доцент Московского института электроники и математики НИУ «Высшая школа экономики». Окончил МИЭМ в 2000 году. Кандидат технических наук. Автор более 30 печатных работ. Область научных интересов: вычислительные системы с управлением потоком данных, параллельные вычисления, распределенные вычисления, компьютерная лингвистика, имитационное моделирование, поиск данных. E-mail**:** salibek@yandex.ru