УДК 004.65

ТЕХНОЛОГИЯ АГЕНТНО-РЕЛЯЦИОННОГО ОТОБРАЖЕНИЯ

Лукьянчиков О.И.[1,2]

[1]*Московский технологический институт, Москва, Россия, e-mail: luk-it6@yandex.ru;*
[2]*Московский государственный университет информационных технологий, радиотехники и электроники, Москва, Россия*

Тенденции развития информационных систем ориентированы на обработку огромных объемов данных и обслуживание большого количества клиентов. Поэтому эффективным при построении информационных систем является распределение вычислений и данных в системах. Так же в системах широко применяются реляционные баз данных, которые также могут иметь распределенный характер. Поэтому при разработке систем используются технологии удаленного доступа и технологии взаимодействия в базах данных, наиболее предпочтительной является технология объектно-реляционного отображения. Унификация принципов технологий и их объединение позволяет создать новую технологию агентно-реляционного отображения, которая порождает новые парадигмы программирования. Для разработанной технологии предложена методология проектирования и разработки систем. Проведены исследования, выявившие зависимость времени выполнения операций от количества клиентов и баз данных.

Ключевые слова: база данных (БД), распределенная база данных (РБД), технологии удаленного взаимодействия, агентно-ориентированное программирование (АОП), объектно-реляционное отображение (ОРО), агентно-реляционное отображение (АРО).

TECHNOLOGY AGENT-RELATIONAL MAPPING

Lukyanchikov O.I.[1,2]

[1]*Moscow Technological Institute, Moscow, Russia e-mail: luk-it6@yandex.ru;*
[2]*Moscow state University of information technologies, radio engineering and electronics, Moscow*

Trends in the development of information systems focused on the processing of big data and maintenance of a large number of clients. Therefore, effective information in building systems is to distribute computing and data systems. The same systems are often used relational databases, which can also be distributed nature. Therefore, when developing systems using remote access technology and data access technology, especially preferred is the technology of object-relational mapping. Unification of the principles of technology and their union allows to create a new technology agent-relational mapping, which gives rise to new programming paradigms. For the proposed methodology developed technology design and development of systems. The investigations, which revealed the dependence of the operations on the number of clients and databases.

Keywords: database (DB), a distributed database (DDB), remoting technology, agent-oriented programming (AOP), object-relational mapping (ORM), agent-relational mapping (ARM).

Существующие тенденции развития информационных систем ориентированы на обработку огромных объемов данных и обслуживание большого количества клиентов одновременно на разных уголках земного шара. Поэтому эффективным при построении информационных систем является распределение вычислений и данных в системах. Развитие распределенных систем дошло и до применения их в бизнесе, вследствие чего появился ряд архитектурных решений построения систем, ориентированных на предоставление и продажу информационных услуг, таких как: сервис-ориентированная архитектура и архитектура облачных вычислений.

Для упрощения разработки и интеграции компонентов распределенных систем используется промежуточный уровень (англ. *middleware*), который обеспечивает требуемый

уровень прозрачности, масштабируемости и открытости взаимодействия. К таким технологиям можно отнести [3, 10, 13]:

– системы и технологии удаленных объектов, которые представляют собой реализации стандартов CORBA (*Common Object Request Broker Architecture* – общая архитектура брокера объектных запросов), COM (*Component Object Model* — объектная модель компонентов), DCOM (*Distributed COM* – распределенный COM), COM+;

– технологии RPC (*Remote Procedure Call* – вызов удаленных процедур);

– технологии MOM (*Message-Oriented Middleware* – ориентированные на обработку сообщений);

– технологии взаимодействия с БД (англ. *database access middleware*).

При использовании различных технологий промежуточного уровня для разработки распределенных систем, проектирование и разработка приложений осуществляется на основе парадигмы объектно-ориентированного программирования, поэтому технологии удаленного доступа промежуточного уровня распределенных систем используются как обычная библиотека с доступным набором интерфейсов. Изменение парадигмы программирования при использовании технологий удаленного доступа позволит унифицировать существующие технологии и изменить принципы использования данных технологий.

Так же в системах широко применяются реляционные БД, являющиеся наиболее эффективным хранилищем данных, потому что реляционная структура хранения данных позволяет быстрее выполнять операции с данными, особенно поиск. Реляционные БД также могут иметь распределенный характер, при которой информация, хранящаяся на разных узлах, связана таким образом, чтобы составлять единую совокупность данных. Кристофер Дейт в 1987 году сформулировал один основной принцип и двенадцать правил, которым, по его мнению, должны следовать распределенные базы данных [8]. Основной принцип заключается в "прозрачности распределения", т. е. с точки зрения разработчика приложения информация в распределенной БД не должна отличаться от локальной БД. Реляционная модель БД сильно отличается от объектной модели приложений, что затрудняет проектирование и разработку информационных систем. Поэтому при разработке объектно-ориентированных приложений используются различные технологии взаимодействия с реляционными БД, особенно предпочтительной является технология объектно-реляционного отображения.

1. Агентно-ориентированное программирование

Развитие принципов программирования пришло от монолитных программ к модульным, при которых программа разбивается на относительно независимые составные части – программные модули. Из-за использования модульного подхода к проектированию

программных средств возникают трудности и проблемы на этапах интеграции и тестирования. Модульное программирование так и не смогло полностью реализовать все свои концепции в полной мере. Например, оформление каждого модуля в виде самостоятельного программного продукта (исполняемого файла) оказалось неосуществимо. Модули подгружались в процессе выполнения программы, размещаясь в том же адресном пространстве, что и основная программа. Независимость при этом довольно условная. Раздельное выполнение этих модулей на разных машинах нереально [1].

Использование технологий удаленного доступа промежуточного уровня распределенных систем, позволяет решить ряд трудностей и проблем модульного программирования, так как позволяют модулям приобрести следующие свойства: независимость модулей, обеспечивающая гибкость систем; сетевое взаимодействие, благодаря чему системы имеют широкое географическое распределение; распределение вычислений, позволяющее достигать высокой эффективности систем.

Обобщает все эти свойства агентно-ориентированный подход (АОП) программирования, предложенный Шохемом в работах [11, 12]. В этом случае проектирования основополагающим понятием является агент, который находится в среде, позволяющей взаимодействовать с другими агентами и определяющей его поведение. В данной парадигме программирования под агентом понимается не обязательно «агент» из терминологии многоагентных системы. В данной парадигме минимальной единицей декомпозиции может так же выступать и объект, и актор (универсальный примитив параллельного численного расчёта, который может принимать и посылать сообщения, устанавливая, как следует реагировать на последующие сообщения, а так же создавать новые акторы) [9], и отдельный процесс или сервис.

2. Объектно-реляционное отображения

Развитие структуры взаимодействия информационных систем и приложений с БД, привело к появлению таких технологий как *Open Database Connectivity* (ODBC), *Data Access Object* (DAO), *Borland Database Engine* (BDE), предоставляющих общий программный интерфейс для работы с различными СУБД. В дальнейшем потребности развития программно-аппаратных средств, работающих с данными, потребовали обеспечения доступа не к SQL-хранилищам данным, а к электронной почте и службе каталогов. Для обеспечения этих функций были разработаны технологии *Object Linking and Embedding, Database* (OLEDB) и *ActiveX Data Objects* (ADO). С появлением мощных систем класса Frameworks, таких как *.Net* и *Qt*, технологии взаимодействия с БД стали встраиваться в данные системы, обеспечивая полную интеграцию с ними, а так же интеграцию со слабоструктурированными данными в формате XML. Последний стал единым форматом для хранения данных в файлах.

Однако, независимо от выбранной технологии, разработчикам приложений, в основном, приходится оперировать SQL-запросами для выполнения операций с данными, и при развитии технологий сложность конструкций запросов увеличивается [2]. Для связи реляционных данных с объектами приложения предпочтительной является технология объектно-реляционного отображения ORM (Object-Relational Mapping – объектно-реляционное отображение). Однако полное отражение реляционных данных в объекты невозможно, поскольку объектная парадигма основана на создании приложений из объектов, имеющих данные и поведение, в то время как реляционная парадигма основана на хранении данных [7].

Существующие реализации данной технологии, такие как *QxORM, EntityFramework, Dapper, Hibernate* и прочие, в основном выполняют генерацию SQL-запросов при выполнении методов объектов. Помимо этого, в данную технологию встраивают функционал по кэшированию данных, работу с несколькими СУБД и поддержание ссылочной целостности данных на стороне клиента, что делает данную технологию удобной при использовании в распределенных БД. При построении гибридной облачной инфраструктуры [4, 5] описывается и рекомендуется использование технологии ORM, однако, данной технологии недостаточно, чтобы обеспечить все необходимые возможности, в частности — отсутствие взаимодействия между объектами. Поэтому эффективней будет применить совместно с технологией ORM технологии удаленного доступа. В этом случае каждый отображенный объект БД сможет обеспечивать взаимодействие с подобными ему, тем самым, также выполняя функции агента. Поэтому можно сказать, что производится реляционное отображение в агентно-ориентированную модель.

3. Агентно-реляционное отображение

Под агентно-реляционным отображением (АРО) будем понимать отображение реляционной модели в объектную модель, экземпляры которой имеют возможность удаленного взаимодействия между собой. При АРО, как и при ORM, каждой сущности реляционной модели ставится в соответствие объект. Однако при АРО полученные объекты имеют возможность не только генерировать SQL запросы, но и обеспечивать взаимодействие через сеть, что позволяет выполнять синхронизацию данных между всеми объектами.

Технологией, реализующей АРО, можно назвать гибрид технологий удаленного доступа и технологий ORM. Для совместного использования этих двух технологий должны быть добавлены дополнительные возможности, позволяющие реализовать АРО.

В базовый набор операций входят выборка и 3 метода манипулирования данными: добавление; удаление; изменение. Данные операции обеспечивают работу с реляционными БД и присутствуют практически во всех реализациях технологии ORM, соответственно, при

объединении технологий ORM и удаленного доступа данные операции должны выполнять помимо реляционных запросов к БД, команды к удаленным объектам, а именно:

- выборка (*select*) — получение данных удаленного объекта;
- изменение (*update*) – установка данных удаленного объекта;
- добавление (*insert*) – добавление в удаленный контейнер из объектов нового объекта;
- удаление (*delete*) – удаление из удаленного контейнера объекта.

Методы позволят обеспечить синхронизацию данных между всеми удаленными объектами, для обеспечения чего предпочтительным является метод синхронизации через транзакции, потому что данным способом обеспечивается синхронизация данных в реляционных СУБД и поддерживается согласованность данных. Для реализации механизма транзакций необходимы две операции при удаленном взаимодействии: зафиксировать транзакцию (commit) и откатить транзакцию (rollback). Последовательность алгоритма синхронизации между двумя объектами на разных клиентах показана на рис. 1.

Когда один из объектов выполняет некую операцию манипулирования данными (изменение, добавление, удаление), он выполняет соответствующий SQL-запрос, а так же посылает команду и данные для выполнения этой операции другим удаленным объектам. Инициатор операции ожидает подтверждение выполнения свой операции и, получив его от всех удаленных объектов, посылает команду зафиксировать (commit) всем объектам и СУБД. Если во время ожидания инициатором операции, т. е. во время выполнения транзакции, какой-нибудь из удаленных объектов попробует выполнить конфликтующую операцию, то сначала будет произведена проверка, какая из операций была раньше вызвана. Поэтому каждая транзакция должна иметь свой уникальный номер и хранить значение момента времени начала выполнения. Инициатор, чья операция началась позже, получит отказ и будет вынужден откатить операцию (rollback).

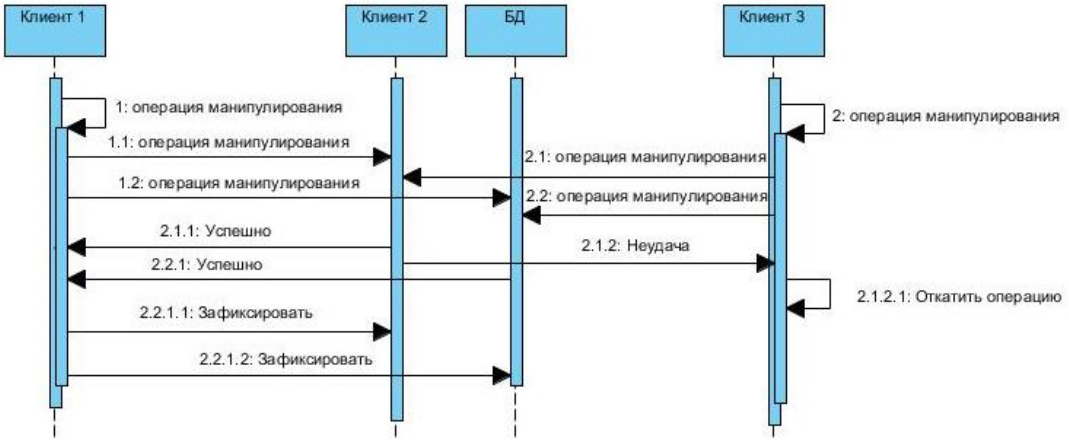


Рис. 1. Последовательность операций синхронизации данных

Для реализации механизма транзакции необходимо знать информацию обо всех удаленных объектах. Таким образом, каждое приложение, использующее данный механизм, должно иметь «карту» всех объектов системы, содержащее следующую информацию о расположении объекта: название объекта (в качестве чего может быть использовано название таблицы); приложение (название и PID номер приложения, уникально идентифицирующий приложение на узле); вычислительный узел (название вычислительного узла и IP-адрес вычислительного узла в сети).

Для формирования «карты» объектов каждое приложение должно выполнять регистрацию своих объектов в других приложениях. Операция регистрации должна выполняться широковещательным пакетом при запуске каждого приложения, использующего технологию APO и обеспечить снятие с учета при завершении такого приложения.

В результате возможности удаленного доступа технологии APO приобретают характер работы с реляционными БД, повторяя поведение основных операций работы с данными и механизмы синхронизации данных с помощью транзакций.

При проектировании систем, которые используют APO, необходимо применять методы объектно-ориентированного проектирования и методы проектирования реляционных БД, так как технология APO объединяет их. Но непосредственное использование APO происходит на стороне приложения, которое имеет объектно-ориентированную модель. Технология APO позволяет отобразить реляционную модель БД в объектное приложение. Таким образом, проектирование системы должно идти от проектирования БД, которое состоит из трех этапов: концептуальное (инфологическое) проектирование; логическое (даталогическое) проектирование; физическое проектирование.

На этапе концептуального проектирования строится модель предметной области, в которой определяются основные сущности системы и связи между ними. В ходе данного этапа строится диаграмма «сущность-связь», предпочтительнее в нотации Чена. Множества сущностей изображаются в виде прямоугольников, множества отношений изображаются в виде ромбов. Если сущность участвует в отношении, они связаны линией. Атрибуты изображаются в виде овалов и связываются линией с одним отношением или с одной сущностью. Пример диаграммы «сущность-связь» в нотации Чена приведена рис. 2.

На следующем этапе логического проектирования создается схема базы данных на основе реляционной модели данных — набор схем отношений, обычно с указанием первичных ключей, а также «связей» между отношениями, представляющих собой внешние ключи. Преобразование концептуальной модели в логическую модель, как правило, осуществляется по формальным правилам (пример — см. 2). Предпочтительным способом

построения объектной модели является построение диаграммы классов в нотациях языка UML. Так как на предыдущем этапе у нас была построена логическая модель данных, то для построения объектной модели можно так же сформировать формальные правила. Пример построения объектной модели приведен на рис. 2.

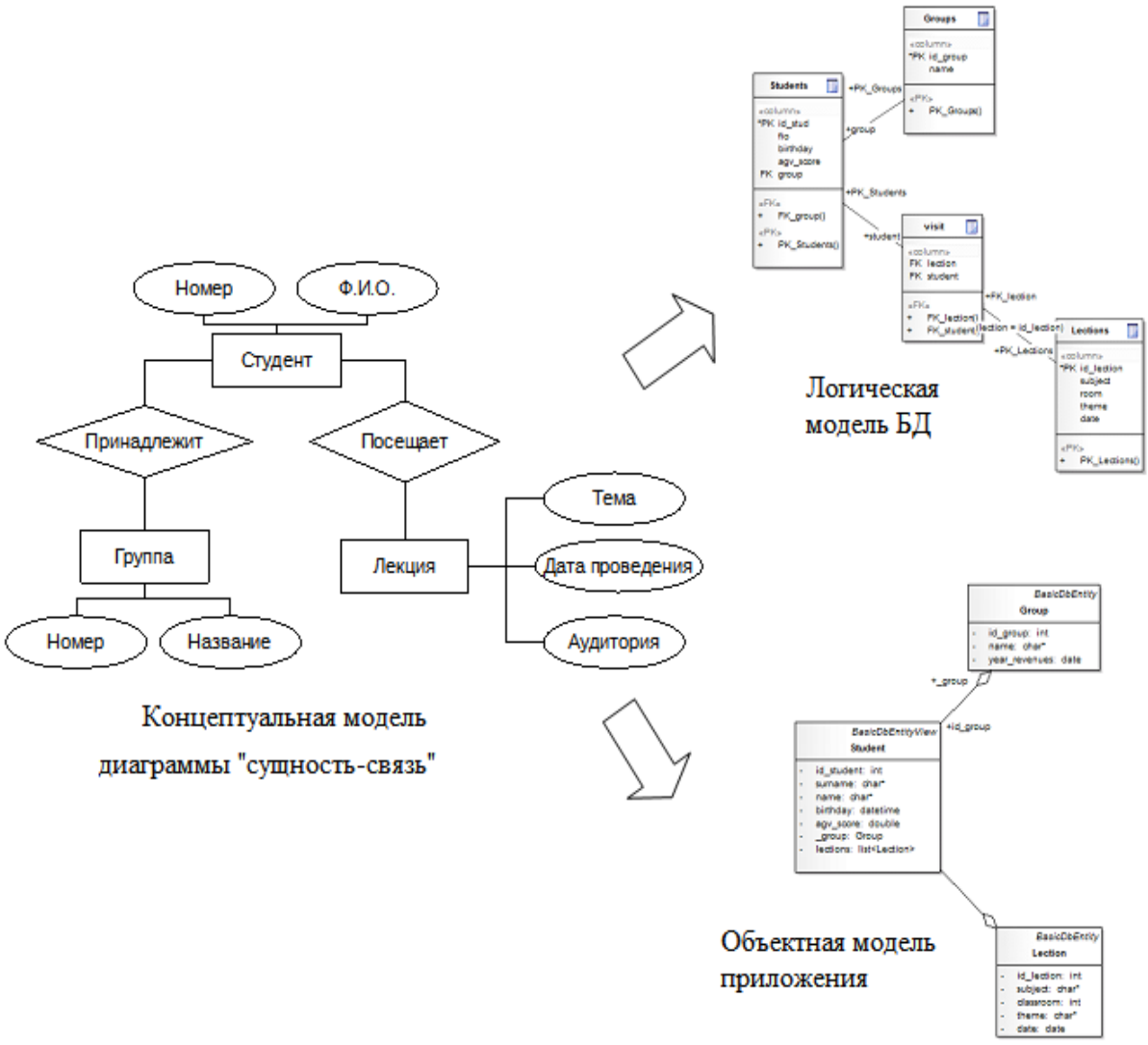


Рис. 2. Примеры диаграмм при проектировании

В ходе построения логической модели БД и объектной модели приложения из концептуальной модели не все сущности могут использоваться впоследствии при наследовании объектов. Технология APO может использоваться и как отдельная технология удаленного доступа, обеспечивая только взаимодействие объектов между приложениями без хранения данных в реляционной БД. Так же могут и присутствовать сущности, которые не используются приложениями, а используются только внутри реляционной БД.

После построения логической модели БД, следует этап физической реализации БД на

конкретной СУБД. Параллельно с физической реализацией может идти и разработка самого приложения по объектной модели, однако предварительно возможно и дополнительное проектирование, в ходе которого с помощью диаграмм UML в полном объеме описывается система, в том числе и описание взаимодействия объектов. Общая схема этапов проектирования системы при использовании АРО, изображена на рис. 3.

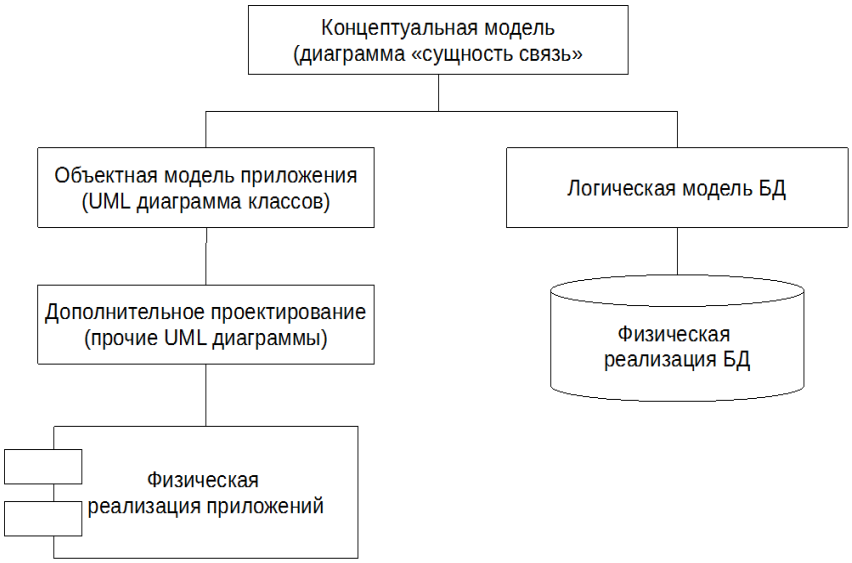


Рис. 3. Этапы проектирования систем с использованием АРО

Описанные теоретические основы АРО и методология проектирования систем с применением приложенной технологии порождают новую парадигму программирования. Гербер Шилдт, известный американский программист, принимавший стандарты С и С++, сказал об появлении ООП: «На каждом этапе развития программирования появлялись методы и инструментальные средства для “обуздания” растущей сложности программ. И на каждом таком этапе новый подход вбирал в себя все самое лучшее из предыдущих, знаменуя собой прогресс в программировании» [6]. Так же и АРО, в соответствии с текущими тенденциями в необходимости обработки больших объемов данных и обслуживания большого количества клиентов, получив лучшее от современных технологий, порождает новый подход в программирование.

4. Реализация технологии агентно-реляционного отображения и оценка её эффективности

На основе описанных теоретических основ АРО, предложена реализация технологии АРО, включающая в себя библиотеку, названную «ArLib», которая непосредственно реализует АРО, и служба (демон) «ArNotifyService», обеспечивающий обмен данными между приложениями, использующими библиотеку «ArLib». Данная служба постоянно должна работать на вычислительных узлах. Она выполняет получение сообщение с сериализованным объектом, с информацией «от кого и кому» и с названием операции по протоколу UDP. Затем

определяет, кому был адресовано сообщение и через локальный сокет пересылает его приложению. Принцип передачи сообщений через службу «ArNotifyService» показан на рис. 4.

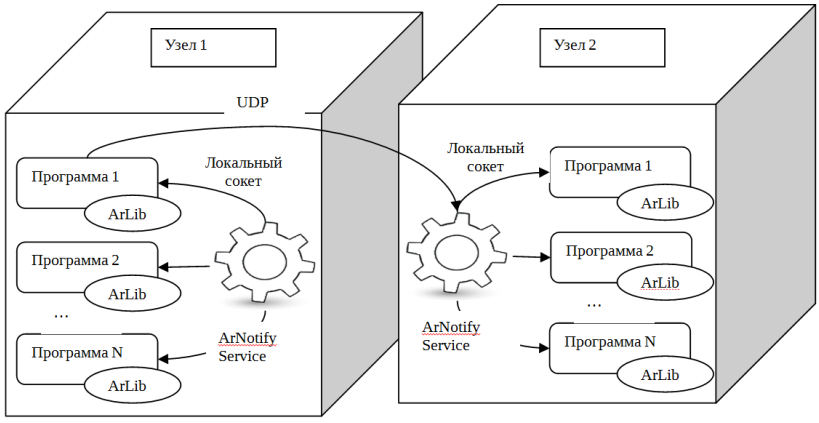


Рис. 4. Передача сообщений через службу «ArNotifyService»

Основным классом является «ArBasicEntity» библиотеки «ArLib», наследуясь от которого, объекты получают возможности выполнять запросы в БД и взаимодействовать с другими объектами. Таким образом, данный класс позволяет: скрыть всю работу с SQL, так как данный класс выполняет самостоятельно генерацию всех необходимых запросов; предоставляет функции для удаленного взаимодействия с другими подобными ему объектами; предоставляет возможности синхронизации всех объектов в программном комплексе.

Чтобы объекту предоставлялись такие возможности, необходимо указать и передать ряд параметров в его класс. Для создания объекта, который соответствует сущности БД, необходимо создать наследника от класса «ArBasicEntity» и в конструкторе указать 2 параметра:

- название сущности, которое будет применяться при формировании строк на формах и прочих сообщений пользователю;
- название таблицы в БД (необходимо указать с разделом БД), которое используется при генерации запросов.

Далее в конструкторе объекта нужно указать соответствие атрибутов объектов к полям сущности БД, используя функции «void setField» и «void setIdentifier». Пример использования методов «void setField» и «void setIdentifier»:

```
setIdentifierField(&_id,"id_stud",QObject::tr("Номер студента"),QVariant::Int);
setField(&_fio,"fio",QObject::tr("Фамилия Имя Отчество"),QVariant::String);
setField(&_avgScore,"average_score",QObject::tr("Средний балл"),QVariant::Double);
```

Так же полученный объект необходимо зарегистрировать в системе, для этого в функции «main» необходимо вызвать метод «void RegisterMetaType<T>», например

следующим образом: *ArEntityParameters::RegisterMetaType<Student>();*

После этого объект полностью готов к использованию и можно выполнять все стандартные операции: выборку из БД (функцией «bool selectData»); добавление новой записи в БД (функцией «bool insertData»); изменение записи в БД (функцией «bool updateData»); удаление записи из БД (функцией «bool deleteData»).

Данные методы могут принимать в качестве параметров параметры подключения к БД, тем самым эти методы будут выполнять соответствующее действие на указанных БД. Помимо этого, данные методы имеют возможность принимать в качестве параметров расположение удаленных узлов в специальном формате, которые возможно получить из «карты» объектов. При этом данные методы будут выполнять соответствующие действия на указанных удаленных объектах.

Второй важный класс библиотеки «ArLib» — «ArNotifer», который обеспечивает взаимодействие между объектами, созданных на основе класса *«DbBasicEntity»*. Объект класса «ArNotifer» является одиночкой (англ. *Singleton*), то есть объект этого класса может быть только один. Объект «ArNotifer» осуществляет непосредственную передачу и прием сообщений, содержащих информацию с сериализованным объектом, с информацией «от кого и кому» и с названием операции. Объект «ArNotifer» получает сообщения от службы «ArNotifyService»; фильтрует не относящиеся к данному приложению сообщения; десериализует объект и по уникальному идентификатору объекта дает команду на выполнения операции, полученной в сообщении. Объекты пересылают сообщения с помощью статических функций «notify» класса «ArNotifer».

Объект «ArNotifer» содержит «карту» всех объектов системы. Регистрация объекта производится при запуске приложения в функции «main» вызовом метода «void ArNotifer::connectToLocalServer». При этом всем приложениям по широковещательному пакету отправляется сообщение с запросом на регистрацию. По этому запросу приложения отправляют сообщения затребовавшему регистрацию информацию обо всех объектах своего приложения. Объект «ArNotifer» получает эти сообщения и регистрирует их у себя.

Помимо самой разработанной технологии необходимы исследования по эффективности использования технологии АРО при построении различных классов информационных систем. Рассмотрение систем без реляционной БД не позволит в полном объеме оценить технологию АРО, потому что возможностей удаленного доступа объектов достаточно для обеспечения необходимого функционала системы.

Технологии АРО, которые наиболее эффективно применяются для проектирования распределенных систем с реляционными БД, являются предпочтительным вариантом применения. В данной системе присутствует множество клиентских приложений, которые

имеют возможность обмениваться удаленно данными между собой, и распределенная БД, как показано на рисунке 5 (где N – количество клиентов в системе, а M – количество БД в системе). В данных системах предпочтительны оперативные данные, которые часто хранятся на клиентах, а долгосрочные данные, которые редко востребованы, хранятся в реляционных БД. При данной архитектуре клиент, выполняющий синхронизацию, может быть любым. При выполнении операций клиентское приложение выполняет операции в БД и рассылает команду на выполнение другим клиентам. Клиентские приложения выполняют операцию и при успешном выполнении отсылают обратно инициатору сообщение об успешном выполнении. Инициатор при сборе от всех клиентов успешных сообщений фиксирует у себя изменение, фиксирует изменения в БД и отсылает всем остальным клиентам сообщение с командой фиксации изменений.

Другой интересной системой для рассмотрения является сервис-ориентированная архитектура (рис. 5). При данном построении системы имеется централизованный сервис, который будет отвечать за синхронизацию данных и выступать в качестве инициатора операций. Поэтому исследования эффективности данной архитектуры направлены на сравнение временной эффективности системы данной архитектуры и распределенной архитектуры с реляционной БД.

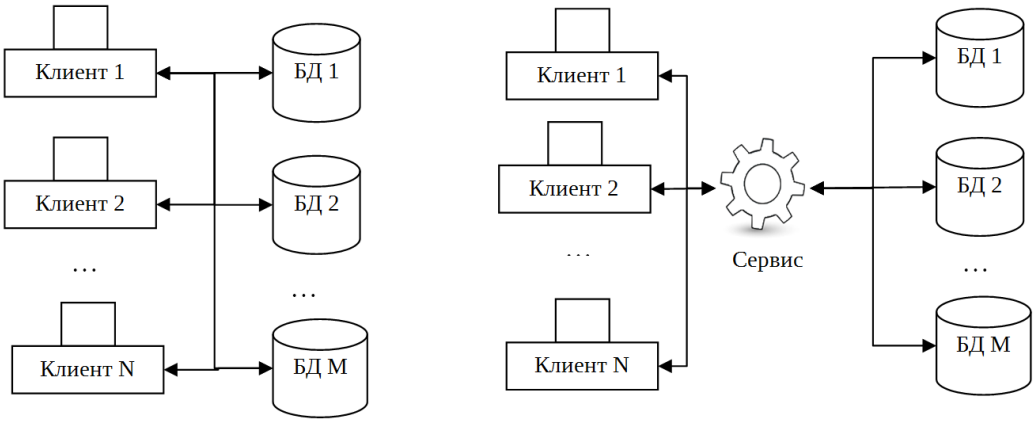
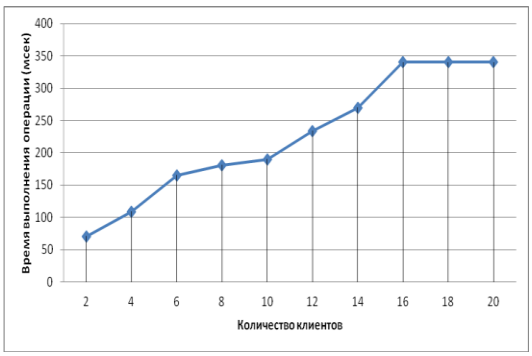


Рис. 5. Распределенная система с реляционными БД и сервис - ориентированная архитектура

Результаты оценки эффективности выполнения данных операций с разным количеством клиентов показаны на рис. 6.



Распределенная система с реляционными БД

Сервис – ориентированная система

Рис. 6. Эффективность выполнения операций в зависимости от количества клиентов

По полученным результатам можно сказать, что увеличение количества клиентов и БД влечет за собой увеличение времени выполнения операции, то есть эффективность системы уменьшается. Однако так же можно отметить, что за счет асинхронности выполнения операций увеличение времени выполнения при большом количестве клиентов сильно не будет увеличиваться, а вся нагрузка будет ложиться только на клиента инициатора, что видно из графика, где значения от 16 до 20 клиентов практически не изменяются. Для сервис-ориентированной архитектуры в среднем операции выполнялись на 20 мсек. дольше, чем в прошлом случае. Это вызвано тем, что имеются задержки на передачу дополнительной команды сервису, получив которую он выполняет операции подобные прошлым экспериментам. Однако при данной архитектуре, т. к. выявлен узел выполняющий синхронизацию, то при увеличении производительности на большом количестве клиентов будет выигрыш по эффективности.

Заключение

На основе анализа технологий удаленного доступа и технологии ORM предложена новая технология АРО. Для данной технологии сформулированы теоретические основы и предложена методология проектирования и разработки систем с её использованием.

Помимо этого, предложена архитектура технологии и проведен ряд исследований, показавших, что увеличение количества клиентов и БД влечет за собой увеличение времени выполнения операции, то есть эффективность системы уменьшается.

Список литературы

1. Бреслер И.Б., Семенов С.А., Корниенко В.В., Борисов В.В. Перспективный подход к организации программных комплексов // Радиопромышленность, 2009. Вып. 1. С. 72–88.

2. Иванников В.П., Гайсарян С.С, Антипин К.В., Рубанов В.В. Объектно-ориентированное окружение, обеспечивающее доступ к реляционным СУБД // Труды Института системного программирования РАН. 2001. Т. 2. С. 89–114.

3. Касаткин. А. Средства middleware и их классификация // PCWeek, 1999. № 19 (193).
4. Лукьянчиков О.И., Никульчев Е.В., Паяин С.В., Плужник Е.В. Проектирование распределенных информационных систем обработки больших объемов данных в гибридной облачной инфраструктуре // Прикаспийский журнал: управление и высокие технологии. 2014. № 4(28). С. 76–87.
5. Никульчев Е.В., Плужник Е.В., Лукьянчиков О.И. Проектирование распределенных информационных систем обработки больших объемов данных в гибридной облачной инфраструктуре // Вестник РГРТУ. 2014. № 50-1. С. 135–138.
6. Шилдт Г. C# 4.0 полное руководство, изд. — М.: "Вильямс", 2011
7. Ambler S. W. Mapping Objects to Relational Databases: O/R Mapping In Detail. 2010.
8. Date C.J. 1987. What is distributed database? // InfoDB, 2:7
9. Hewitt C., Bishop P., Steiger R. A. Universal Modular Actor Formalism for Artificial Intelligence.// Proceedings of IJCAI'73. P. 235–245.
10. Ruh, W.A. and Maginnis, F.X. and Brown, W.J. Enterprise Application Integration: A Wiley Tech Brief. — Wiley, 2002. P. 52-59.
11. Shoham Y. Agent Oriented Programming // Computer Science Department, Stanford University 1990.
12. Shoham Y. *Agent-Oriented Programming* // Artificial Intelligence. 1993. З. 51–92.
13. Stojmenovic I., Thulasiram R.K., Yang L.T. Parallel and Distributed Processing and Applications // Proceedings 5th International Symposium, ISPA 2007. P. 563-656.

Рецензенты:

Никульчев Е.В., д.т.н., профессор, проректор по научной работе Московского технологического института, г. Москва;

Горяшко А.П., д.т.н., профессор, профессор кафедры программных систем Московского технологического института, г. Москва.