3. *Mahadik V.A., Wu X., Reeves D.S.* Detection of Denial-of-QoS Attacks Based On x^2 Statistic And EWMA Control Charts. http://arqos.csc.ncsu.edu/papers/2002-02-usenixsec-diffservattack.pdf.
4. *Абрамовиц М., Стиган И.* Справочник по специальным функциям. М., 1979.

Ростовский государственный университет ***3 февраля 2006 г.***

УДК 519.710.73

РЕПЛИКАЦИОННЫЕ ПРИЛОЖЕНИЯ РАСПРЕДЕЛЕННЫХ БАЗ ДАННЫХ В ИНФОРМАЦИОННЫХ СИСТЕМАХ С НИЗКОСКОРОСТНЫМИ КАНАЛАМИ СВЯЗИ

© 2006 г. О.М. Омаров, А.А. Абдулгамидов

In article is given benchmark analysis of the most wide-spread models to organizations of the access to database and different model репликации and synchronizing database. The Offered model репликации on base two parallel transactionses with introduction mechanism репликационного exhibits that allows to organize efficient work in portioned information system in inferior condition channel relationship.

Одним из путей развития технологий распределенных вычислений и параллельной обработки является внедрение механизмов *репликации*, основная идея которой заключается в том, чтобы создать и поддерживать несколько копий разделяемого хранилища данных. Репликация рассматривается как средство распределения данных разделяемого хранилища. Ее основная задача – поддержание всех реплик в согласованном состоянии, что в условиях проблемных каналов связи становится довольно сложной и проблемной задачей.

Анализ состояния. Репликационные приложения (РП) в данной статье рассматриваются как наиболее эффективное средство управления трансакциями при построении распределенной системы в условиях слабой инфраструктуры – при отсутствии качественных, высокоскоростных каналов связи.

Основные задачи, решаемые при выборе стратегии репликации в распределенной системе: обеспечение отказоустойчивости (при отказе любой или даже нескольких реплик система должна продолжать работать); повышение производительности, позволяющее эффективнее использовать сетевые и вычислительные ресурсы (достигается за счет статистического распределения запросов чтения между многими репликами); локализация трафика (уменьшение задержки при обращении к базам данных (БД) и снижение загрузки дорогостоящих «дальнобойных» или перегруженных магистральных каналов).

Исследование проблем репликации сводится к решению этих трех основных вопросов. От эффективности этого решения зависит эффективность модели распределенной системы.

Обновление любого логического объекта должно распространяться на все хранимые копии этого объекта. Трудности возникают из-за того, что некоторый узел, содержащий данный объект, может быть недоступен именно в момент обновления. В таком случае очевидная стратегия немедленного распространения обновлений на все копии может оказаться неприемлемой, поскольку предполагается, что обновление (а значит и исполнение трансакции) будет провалено, если одна из копий будет недоступна в текущий момент. Это – основная проблема репликации данных.

Виды репликационных архитектур. Рассмотрим наиболее распространенные модели организации доступа к базам данных.

Единая БД с удаленным доступом на рабочих местах. Эта модель организации доступа к базе является наиболее простой в плане установки и администрирования. Работа осуществляется прямым доступом в удаленную БД через механизмы управления СУБД или РП и какой-либо сетевой протокол, в большинстве случаев TCP/IP. В данном случае необходимость в репликации отсутствует вообще, и проблем с несоответствием данных на различных рабочих местах быть не может. Однако такая модель имеет довольно существенный недостаток – высокая требовательность к качеству каналов связи. Смена экранных форм, информации на экране, реакция от управляющих элементов происходят крайне медленно, делая работу приложения практически неприемлемой.

Единая БД с обновляемыми копиями на других серверах. В этом виде архитектуры существуют основная БД и такие, чьи репликационные таблицы, используемые в приложении, но не относящиеся к административным схемам, регулярно обновляются как копии основной БД (снимки) или как копии друг друга под управлением процессов в основной БД. При этом обновление может осуществляться как автоматически через указанный интервал времени, так и вручную, по мере необходимости, путем репликации изменений или полного пересоздания. Достоинство этой архитектуры – в ее высокой производительности и малой зависимости от наличия и производительности каналов связи. Репликацию можно настроить на ночное время или на время, когда отсутствует интенсивный почтовый обмен или работа в интрасети. Неисправности в сети или на серверах реплицируемых БД не мешают продолжать работу, а после восстановления связи все накопившиеся изменения, ссылки на которые копятся в журналах снимков или трансакционных таблицах, поступают в БД-копию.

Тем не менее в случае длительной потери связи и интенсивных трансакциях накопившиеся изменения могут после восстановления связи переполнить возможности физической обработки данных БД-приемником в указанный интервал времени и подвесить систему на длительное и, возможно, неограниченное время. Кроме того, при возникновении сбоя или ошибки выполнения трансакции, что, несмотря на их безупречную работу в БД-источнике, может быть связано с БД-приемником – например, изменение структуры таблиц или ее ограничений, удаление родительской запи-

си и пр., ошибки могут нарастать в каскадном режиме, что приводит к значительному затруднению их анализа и устранения. По указанной выше причине таблицы, регулярно участвующие в интенсивных трансакциях (десятки и сотни тысяч добавлений, изменений и удалений записей), необходимо, по мере возможностей, исключать из репликации и периодически обновлять в БД путем экспорта-импорта.

Недостатком данного вида архитектуры является необходимость и сложность ее регулярного администрирования при ежедневных операциях с БД – ежедневного просмотра и устранения возможных ошибок, а также сложность установки, в том числе привязывания клиентской части. Кроме того, какие-либо изменения свойств объектов требуют перевода групп реплицируемых таблиц в состояние, при котором запись в БД из клиентской части невозможна.

Распределенная БД является более сложным видом вышеописанной архитектуры с обновляемыми БД. В этом виде вся БД разделена на сектора, каждый из которых расположен на удаленном сервере, таблицы которого реплицируются на другие серверы. Иначе: общая БД состоит из таблиц удаленных БД.

В этом случае общая часть БД реплицируется на серверы филиалов, а таблицы, принадлежащие филиалам, или только их часть более-менее регулярно реплицируются на общий сервер. Вариантом является репликация этих таблиц, в том числе различающейся структуры, в отдельные схемы и затем их объединение в общее представление.

Такая архитектура удобна возможностью существования разнородных БД и низкой загрузкой сети. Неудобство – в установке и развитии – детальное предоставление прав на таблицы и другие объекты различным пользователям. Однако неудобства эти становятся не столь очевидными и значительными сравнительно с проблемами организации высокоскоростных каналов связи в труднодоступные, удаленные подразделения.

Методы, стратегии и схемы репликации. Каждая репликационная архитектура, в свою очередь, классифицируется различной организацией реализованными в ней моделями репликации.

Механизм распространения изменений по принципу ведущий-ведомый используется менеджером учетных записей безопасности Security Access Manager (SAM) домена SMB NT4, который имеет ведущую реплику первичного контроллера домена – PDC (Primary Domain Controller) и возможно несколько ведомых реплик, представляющих резервные контроллеры домена BDC (Backup Domain Controller) [1]. PDC запоминает все вносимые в SAM изменения в отдельном хранилище и оповещает о них все BDC. Так как все серверы домена перечислены в SAM, PDC знает адреса этих контроллеров и распространяет информацию целенаправленно. Как только все BDC подтвердят получение записи, PDC удаляет ее. Главный недостаток этой, на первый взгляд, весьма оптимальной схемы состоит в

том, что все ведомые реплики доступны только для чтения, и все остальные недостатки являются производными от этого.

Постоянного соединения между серверами требуют две наиболее распространенные модели репликации – *репликация вытягиванием (pull replication)* и *репликация проталкиванием (push replication)*. Альтернативой этим двум методам является *репликация по расписанию*, при которой реплики согласуются не чаще, чем с определенным интервалом. Но такая модель возможна только в случае приемлемости временной рассинхронизации данных.

Репликация вытягиванием чаще всего находит применение в технологии кэширования. Некоторые авторы даже объединяют все типы репликации вытягиванием в одну общую группу кэшей (cache) [1]. Кэш, представляя собой частичную реплику некоторого одного или нескольких центральных хранилищ, посредством системы собственных индексов некоей ассоциативной памяти сопоставляет адрес записи в центральном хранилище, а при использовании нескольких таких хранилищ – сочетание из адреса хранилища и адреса записи и локальную реплику этой записи. Поскольку реплика частичная и содержит только те записи, которые реально востребованы клиентом, такая репликация порождает меньше сетевого трафика, чем стратегии, предполагающие согласование полных реплик. При объеме кэша всего в несколько процентов от объема хранилища и неравномерном распределении обращений к записям хранилищ – как правило, на основании распределения Парето [2], удается достичь коэффициента попадания (hit rate), измеряемого отношением количества попаданий к общему количеству обращений – 90, а то и 99 %.

При репликации проталкиванием [2] сервер реплики, в которой были модифицированы данные, оповещает об этом все остальные серверы. Для обеспечения такой идеологии требуется, чтобы каждый сервер реплицируемой системы знал о наличии всех остальных, что зачастую требует прямого соединения между серверами, в реплики которых могут вноситься изменения, т.е. создания полносвязной сети соединений хотя бы транспортного уровня.

На практике в чистом виде редко можно встретить какую-то конкретную модель репликации – в большинстве случаев распределенные системы построены на смешанных идеологиях. Одной из моделей гибридной репликации является алгоритм MESI [1] обеспечения когерентности кэшей (cache coherency), используемый процессорами Intel и др. MESI продолжает идеологию кэширования: в кэш скачиваются и вообще передаются по шине только те записи, которые непосредственно востребованы системой. Обрабатываются в первую очередь те операции, которые в данный момент являются причиной каких-то задержек. Эта идеология оправдана вышеупомянутым принципом распределения Парето, т.е. соображением, что значительная часть центрального хранилища большую часть времени невостребована.

Рассмотренные модели обеспечивают надежную репликацию отдельных записей БД, гарантируя, что если запись изменена в одной из реплик, то она рано или поздно изменится во всех остальных. Однако такая репликация приемлема лишь постольку, поскольку каждая запись автономна, т.е. изменения в ней не затрагивают другие записи. На практике разработчики приложений на основе реляционных СУБД зачастую сталкиваются с обратной ситуацией, когда записи в базе данных в том или ином смысле взаимозависимы, взаимосвязаны и не могут быть разделены без нарушения своей сущности.

Репликационное приложение как эффективный инструмент управления трансакциями. Попытки построения модели с централизованным хранением данных и доступом удаленных пользователей к центральной БД в условиях географически распределенной системы приводят к необходимости установления соединений между центральным сервером, хранящим данные, и компьютерами-клиентами, поэтому в этом случае ключевое значение имеет качество каналов связи. Однако большинство компьютеров-клиентов могут быть отделены от центрального сервера медленными и недостаточно надежными линиями связи. В этом случае работа в режиме удаленного клиента становится почти невозможной. Доставка данных к месту назначения не системными средствами, а путем экспорта/импорта файлов приводит к необходимости участия человека в процессе обмена, что влечет за собой низкую скорость поступления данных и необходимость реализации внешних механизмов контроля целостности и непротиворечивости. Выход из создавшегося положения – возложение функции тиражирования данных на существующие специализированные программные средства. Внедрение технологии тиражирования данных обеспечивает гарантированную доставку, своевременность и целостность передаваемых данных. Поэтому центральным вопросом проектирования является выбор технологии распределенной обработки и доставки данных.

Проблемы реализации распределенной информационной системы в Дагестанском отделении Пенсионного фонда РФ появились сразу же по мере формирования структуры. Главная причина – сложный географический рельеф местности, влияющий на отсутствие каких-либо существенных каналов, пригодных для передачи информации между подразделениями предприятия. Вторая наиболее существенная проблема, которая пока не потеряла актуальности, – гетерогенность и отсутствие достаточной интероперабельности унаследованных информационных систем и локальных программных модулей. Предприятие имеет трехуровневую иерархию [3]: район – регион – центр. Одной из основных задач является сбор, учет, хранение и распределение информации по индивидуальным сведениям начисленных и уплаченных страховых взносов пенсионного обеспечения. Данная задача в плане взаимодействия уровня район – регион не требует реализации режима «реального времени» в информационной системе, однако степень актуальности должна быть достаточно высокой.

В основу механизма репликации, предложенного и реализованного специалистами организации в качестве основы распределения данных в распределенной системе, был положен уже реализованный и обкатанный в ряде систем механизм «объектной репликации», дополненный системой предварительного анализа выборки реплицируемых объектов с созданием реплицируемого приложения управления глобальными и локальными трансакциями.

Распределенная система построена с использованием комбинации VFP9.0/DB2. Данный выбор обусловлен именно теми штампами, которые неизбежно приводятся при упоминании об этих технологиях: независимостью от платформ и полной совместимостью с представлением данных уже реализованных систем (VFP9.0) и настоящей «объектностью» и нейтральностью к языкам программирования (DB2). Имели место и чисто прагматические соображения: ориентация на грандов компьютерной отрасли – IBM и Microsoft. И DB2, и VFP9.0 – открытые системы, описанные достаточно подробно и внятно, имеющие предсказуемую линию развития [4].

Репликационное приложение (РП) – это программный модуль, причем функционально различающийся для регионального и районного уровней, с внедренными в нем механизмами предварительного анализа выборки требуемой информации. Оно обеспечивает не только управление трансакциями, но и определяет оптимальное значение актуальности документов для экономии трафика при последующей репликации. Таким образом, внедренный механизм обеспечивает достаточный уровень актуальности реплик в условиях проблемных, низкоскоростных каналов связи.

Установка (загрузка) РП в филиале производится на одну из машин в рамках локальной сети, имеющей доступ к локальной БД, с одной стороны, и имеющей периодический либо постоянный доступ к РП, находящемуся на головном предприятии, с другой. РП может работать в двух режимах: ожидания репликационных соединений (сервер) и режиме, при котором производятся периодические попытки соединения с другими РП, работающими в режиме сервера (клиент). Для каждой из БД можно запускать одновременно два экземпляра РП в разных режимах. Если на связь с сервером выходят одновременно несколько удаленных узлов, то сервер осуществляет взаимодействие с каждым из них в рамках отдельного потока выполнения, запускаемого в виртуальной машине. В случае аварии имеющихся каналов связи, либо если удаленное подразделение не имеет таковых вовсе, предусмотрена так же off-line-репликация. В минимальном варианте все программное обеспечение, включая коммуникационные программы, РП и БД, может быть размещено на одном компьютере.

На рис. 1 показаны взаимодействия РП между собой и с БД при передаче объектов между БД.

В процессе сеанса репликации из одной БД в другую производится формирование предварительного протокола выборки информации из реплицируемой базы для обеспечения необходимого и достаточного уровня

актуальности реплики; выявляются и исправляются потенциальные ошибки, возможные при разрывах сеансов связи; передаются объекты, которые были созданы, но еще не переданы в конкретную БД; передаются изменения содержимого ранее переданных объектов; производится удаление объектов в БД дальнего узла, которые были удалены в локальной БД и ранее переданы в дальний узел. Удаление объектов в БД дальнего узла производится в порядке регистрации их удаления в локальной.

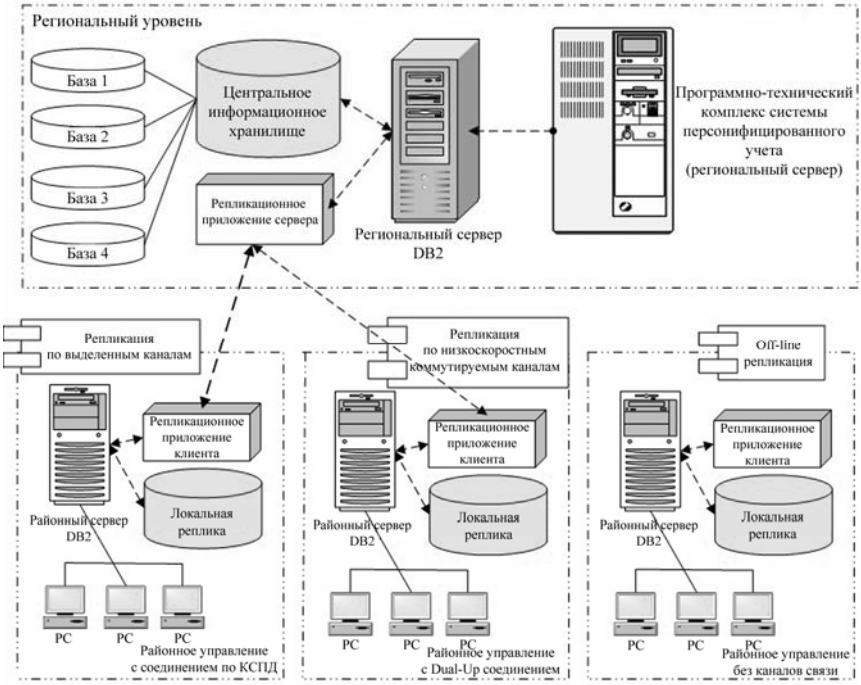

Рис. 1. Взаимодействия РП

Передача каждого объекта или действий над объектами проводится двумя параллельными трансакциями: одной в удаленной БД, другой – в местной. Передающая сторона управляет обеими трансакциями, перед передачей очередного объекта (действия) они открываются, при этом инициатором открытия трансакций является передающая сторона. После завершения факта передачи объекта и окончания всех регистрационных действий передающая сторона производит попытку фиксации трансакции на удаленном узле и откат трансакции в случае неудачи – неполучения от удалённой СУБД подтверждения. После получения положительного ответа об окончании фиксации трансакции на удаленном узле производится фиксация местной, при возникновении какой-либо ошибки в процессе репликации – откат обеих трансакций и завершение сеанса репликации.

Используемое программное обеспечение в ПФР построено на различных информационных платформах. Связано это, главным образом, с тем, что предприятием ряд информационных пакетов был унаследован от других ведомств. Сыграло роль отсутствие четкой единой политики в плане информационных технологий, что в общем присуще крупным предприятиям бюджетной сферы. Таким образом, глобальная информационная система ПФР представляет собой гетерогенный мультибазовый комплекс аппаратно-программных средств, интеграцию автономных систем, в которых реализовывается путем построения механизмов взаимодействия глобальных и локальных трансакций.

Разработка эффективных подходов к управлению глобальными трансакциями требует выбора критерия корректности выполнения трансакций; определения механизмов разрешения конфликтов последних в системах управления БД; разработки протокола управления ими в распределенной информационной среде на основе автономных систем БД.

При помощи математического аппарата теории множеств определяются в формализованном виде глобальные трансакции и подтрансакции, локальные трансакции. Последние неизвестны на глобальном уровне распределенной информационной системы, что значительно осложняет управление глобальными трансакциями. При этом подтрансакция является обычной локальной с точки зрения системы БД, в которой она выполняется [5].

На рис. 2 можно отследить порядок обработки трансакций РП. Всю информацию БД локальной реплики района можно разделить на три группы: 1 – объекты, требующие немедленного обновления для реализации текущего запроса; 2 – объекты с достаточной степенью актуальности, не требующие обновления для данного запроса; 3 – актуальные объекты.

РП-1 перехватывает первичный запрос и проводит анализ сущности и наличия актуальных данных в локальной реплике. При этом глобальные данные делятся на составляющие их объекты. Оценка степени актуальности проводится дифференцировано для каждого объекта. В процессе выполнения этих операций формируется так называемая таблица очередности реплицируемых объектов, в которой описывается, какие именно объекты локальной реплики должны быть синхронизированы немедленно для исполнения данного запроса, а какие могут подождать, т.е. степень их актуальности достаточна на данный момент времени, при текущих условиях и обстоятельствах.

Сформированная таким образом таблица очередности реплицируемых объектов передается РП–2, которое отвечает за формирование запроса к центральному хранилищу и корректную выборку объектов с рациональным расходованием трафика. Таким образом, происходит обновление информации в локальном хранилище, после чего управление снова передается РП–1, которое взаимодействует непосредственно с приложением и выдает результаты запроса.

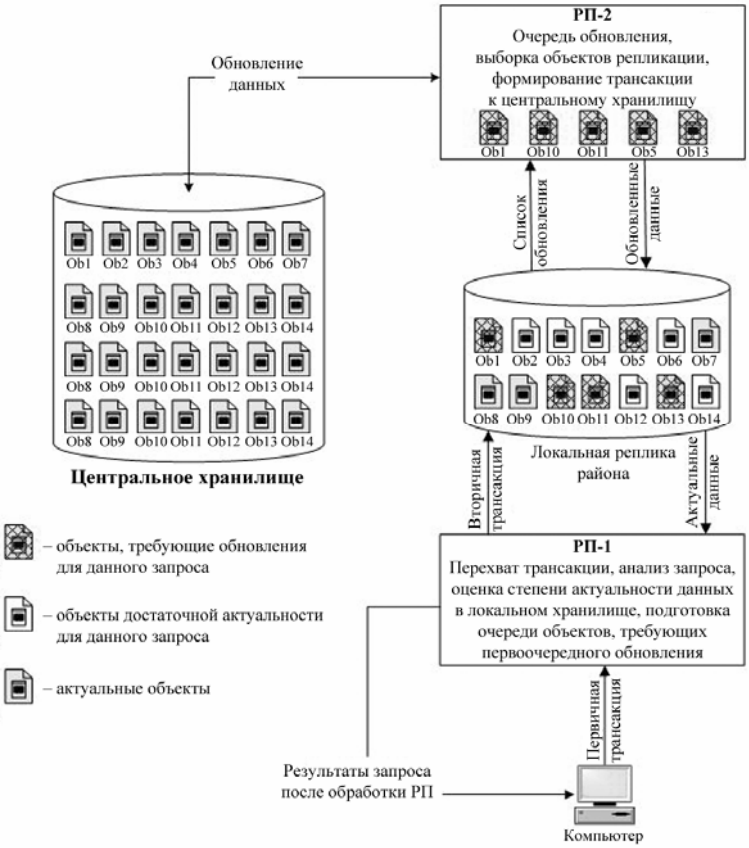

Рис. 2. Схема обработки трансакций РП

Помимо РП, отвечающего за корректное выполнения всех глобальных и локальных трансакций, в автоматизированную систему отделения внедрены механизмы мониторинга трансакций. В распределенной трансакции каждый участвующий ресурс имеет локального диспетчера трансакции (TM – transaction manager) для отслеживания входящих и исходящих трансакций на данном компьютере. Монитор обработки трансакций назначает одному диспетчеру трансакции дополнительную задачу координации действий между локальными диспетчерами трансакций. Диспетчер трансакций, координирующий их действия, называется корневым или координирующим. Он координирует и управляет всеми функциями обработки трансакций, но не оснащен средствами для непосредственного управления данными. Диспетчеры ресурсов обрабатывают операции, связанные с данными.

При этом на всех этапах построения механизма трансакций соблюдаются свойства атомарности, согласованности, автономности и надежности (ASID), гарантирующих предсказуемый характер поведения трансакций,

усиливая их роль в качестве утверждения «полностью или никак», для обеспечения целостности данных нагрузки управления при большом количестве переменных.

Заключение. Создание распределенных информационных систем является весьма актуальной задачей. Связано это, прежде всего, с возрастающими требованиями к оперативности и достоверности информации.

Достаточная производительность и высокая надежность в условиях проблемных каналов связи – одни из важнейших целей, на достижение которых направлена рассмотренная в статье технология репликации распределенной СУБД. Как правило, она обеспечивается за счет сочетания нескольких взаимодополняющих решений – выбор оптимальных режимов репликации, оценка баланса загруженности сети и внедрение взвешенных графиков, разработка гибкого реплицируемого приложения с предварительным анализом условий достаточности. Наличие системы, «осведомленной» о специфических потребности баз данных в определенный временной интервал, упрощает реализацию функций баз данных нижнего уровня и способствует снижению трафика. Надежность системы также достигается за счет исключения одиночных точек отказа. Отказ одного узла или сбой на линии связи не приводит к выходу из строя всей системы. Даже если часть данных становится недоступной, при правильной организации системы пользователи могут иметь доступ к остальной части информации. Под «правильной организацией» понимается поддержка распределенных трансакций и протоколов обеспечения надежности, т.е. протоколов фиксации и восстановления.

Предложенная модель репликации на основе двух параллельных трансакций с внедрением механизмов репликационного приложения, основанного на поддержании необходимого и одновременно строго достаточного уровня согласованности реплик, представляет возможность построения распределенных информационных систем в условиях низкого качества связи [5], что, несмотря на значительные достижения в области телекоммуникаций, до сих пор еще является весьма актуальной проблемой во многих регионах России.

Литература

1. *Танненбаум Э., Ван Стеен М.* Распределенные системы, принципы и парадигмы. СПб., 2003.
2. *Иртегов Д.В.* Введение в сетевые технологии. СПб., 2004.
3. *Омаров О.М., Ахмедов Р.К., Гаджиев М.А. и др.* // Информационные и телекоммуникационные системы: сетевые технологии: Материалы III республиканской науч.-практ. конф. Махачкала, 2004. С. 151–165.
4. *Кренке Д.* Теория и практика построения баз данных. СПб., 2003.
5. *Бурковский В.Л., Дорофеев А.Н., Копсяев А.П.* // Вестн. Вычислительные информационно-телекоммуникационные системы. Воронеж, 2002. Вып. 8. № 2. С. 53–56.

Дагестанский государственный технический университет,
Государственное учреждение отделения Пенсионного фонда России
по Республике Дагестан, г. Махачкала

8 февраля 2006 г.