УДК 004.75:004.657

Т.А. Медведев, И.В. Стручков
Санкт-Петербург, Россия

АРХИТЕКТУРА ПРОГРАММНОГО СЛОЯ ДОСТУПА К РЕЛЯЦИОННЫМ БАЗАМ ДАННЫХ ДЛЯ МАСШТАБИРУЕМЫХ ГИБРИДНО-ОБЛАЧНЫХ СИСТЕМ

T.A. Medvedev, I.V. Struchkov
St.-Petersburg, Russia

MIDDLEWARE ARCHITECTURE FOR RELATIONAL DATABASE ACCESS IN A SCALABLE HYBRID CLOUD SYSTEM

Рассмотрены возможные подходы к обеспечению масштабируемости реляционных СУБД в гибридно-облачной инфраструктуре. В качестве требований выступают: эффективное распределение данных при создании множественных копий виртуальных машин, миграция данных между частной и публичной инфраструктурами, обеспечение возможности автономного функционирования информационных систем при разрыве сетевого соединения между инфраструктурами и обеспечение контроля передаваемых данных через глобальную сеть. Предложен подход на основе расширения широко распространенной реляционной модели данных, а также архитектура программного слоя доступа к данным.

РЕЛЯЦИОННАЯ МОДЕЛЬ ДАННЫХ. РАСПРЕДЕЛЕНИЕ И МИГРАЦИЯ ДАННЫХ. ГИБРИДНО-ОБЛАЧНАЯ СИСТЕМА. МАСШТАБИРУЕМОСТЬ. КОНТРОЛЬ ПЕРЕДАЧИ ДАННЫХ.

The problem of relational DBMS scalability in a hybrid cloud infrastructure is investigated with respect to the following requirements: efficient data distribution when multiple virtual machine instances are run in the cloud, data migration between public and private cloud infrastructures, offline availability of hybrid cloud information services on network failure, data transfer control policies in the global network.

RELATIONAL DBMS. DATA DISTRIBUTION AND MIGRATION. HYBRID CLOUD SYSTEM. SCALABILITY. DATA TRANSFER CONTROL POLICY.

Технология облачных вычислений основывается на использовании виртуальных арендуемых вычислительных средств, предоставляемых по требованию. Благодаря тому, что в облачных технологиях ресурсы выделяются в кратчайшие сроки и в условно неограниченном объеме, возможен существенный шаг вперед в части обеспечения важнейших показателей качества современных информационных систем: масштабируемости и надежности. Это достигается путем своевременного автоматического подключения к системе новых вычислительных ресурсов в виде типовых виртуальных машин в случаях нехватки вычислительной мощности или выхода из строя каких-либо компонентов системы.

Появляющиеся в последнее время *гибридно-облачные технологии* позволяют размещать часть ресурсов в публичной облачной инфраструктуре, оставляя часть ресурсов в частной инфраструктуре организации. Такой подход позволяет повысить надежность, безопасность и масштабируемость информационных сервисов, сохраняя возможность использовать имеющиеся собственные ресурсы и прибегать к аренде

дополнительных ресурсов только при необходимости.

Имеющиеся технологии дают возможность масштабировать программные сервисы в облачной инфраструктуре путем создания множества копий идентичных программных компонентов. Один из способов реализации такого подхода описан в [1]. Данный способ позволяет осуществлять перенос компонентов программного обеспечения (ПО) между облачной инфраструктурой и собственными ресурсами предприятия, централизованное администрирование программных компонентов. Тем не менее недостаточно внимания уделено вопросу перераспределения данных при масштабировании сервисов. В отличие от программных компонентов, данные не могут быть механически размножены при масштабировании – остаются проблемы обеспечения целостности множества копий данных. В случае применения гибридно-облачных технологий дополнительную сложность создает тот факт, что данные могут храниться либо в локальной сети организации, либо в облачной инфраструктуре, либо в двух этих местах одновременно, разделяемые относительно медленным и слабо защищенным интернет-каналом.

Существует несколько возможных вариантов перераспределения данных при масштабировании, которые будут рассматриваться на примере баз данных (БД) как самого распространенного способа хранения данных для программных сервисов.

1. Репликация БД – создание множества идентичных копий БД. В этом случае встает вопрос синхронизации и поддержания непротиворечивого состояния всех копий данных.

2. Распределенное хранение (шардинг – sharding) – каждый элемент распределенной системы хранит свою собственную уникальную часть данных, которая определяется значением заданного параметра распределения. В этом случае появляется вопрос поиска и извлечения информации из распределенного хранилища.

3. Распределенное хранение с дублированием – каждый элемент распределенной системы хранит часть данных, которая не обязательно является уникальной. Данный вариант наиболее универсальный, однако в этом случае требуется решать обе задачи, указанные выше.

Цель данной статьи – обоснование подхода и описание архитектуры программного сервиса для повышения масштабируемости гибридно-облачных систем за счет эффективного распределения данных, миграции данных между частной и публичной инфраструктурами, обеспечения возможности автономного функционирования информационных систем при разрыве сетевого соединения между инфраструктурами, обеспечения контроля передаваемых данных через глобальную сеть на основе политик, задаваемых пользователем. При этом существенным требованием к разрабатываемому подходу является возможность использования имеющегося ПО без необходимости внесения существенных изменений.

Анализ существующих решений

Сервис Relational Cloud [2] является реализацией реляционной СУБД в облачной инфраструктуре. Эффективная работа в многопользовательском режиме, обеспечивающая хорошую масштабируемость, достигается за счет использования централизованного сбора статистики и принятия решений о переносе и/или репликации логических БД между серверами СУБД. Использование разбиения запросов между несколькими узлами позволяет эффективно выполнять запросы, для которых не хватает ресурсов одного узла. Безопасность реализуется за счет шифрования данных. Сервис не предполагает возможность использования в гибридно-облачной инфраструктуре и не предполагает контроля передаваемых данных через глобальную сеть.

Система хранения Cloudy [3] представляет собой настраиваемую распределенную среду, в которой можно задать интерфейс обращения к данным и механизм их хранения. Реализация основана на использовании предложенной модели представления данных, позволяющей единообразно описывать как запросы, так и результаты. Модульная архитектура позволяет настроить систему хранения под нужды пользователя. Каждый модуль реализует три базовые операции: чтение, добавление и удаление. Масштабируемость данных ограничена, поскольку не рассматривается вариант сложных запросов, которые не могут быть выполнены на одном узле. Также не рассматривается выполнение запросов, включающих данные из разных физических хранилищ. В системе не описаны механизмы защиты данных

и не предполагается использование в гибридно-облачной инфраструктуре.

Система HadoopDB [4], разработанная в Йельском университете для выполнения запросов в хранилище данных (Data Warehouse), реализует механизм разделения запросов для уменьшения используемых ресурсов. Предлагается несколько техник оптимизации, с помощью которых можно эффективнее использовать имеющиеся ресурсы на узлах хранения для выполнения больших запросов. Однако идеология Data Warehousing предполагает эффективное распределенное чтение данных из различных источников, но не предполагает запись в эти источники.

Базы данных класса NoSQL [5], такие, как Hadoop, MongoDB, CouchDB, проектировались в расчете на использование в облачной инфраструктуре и имеют существенно более высокую масштабируемость по сравнению с реляционными БД. В то же время они ориентированы на работу в условиях интенсивных операций чтения, поэтому проигрывают в производительности, когда более интенсивными являются операции записи, и не обеспечивают ACID свойств, характерных для реляционных СУБД. Обычно в них реализован механизм миграции между облачными инфраструктурами и они могут обеспечивать автономную работу при потере соединения. Тем не менее в них нет механизмов контроля передачи данных через глобальную сеть. Наиболее существенный недостаток такого класса БД – необходимость изменения имеющегося ПО, ориентированного на реляционную модель данных.

Постановка задачи

По проведенному анализу можно сделать вывод, что в данный момент ни одно из существующих решений не позволяет полностью достичь поставленных целей. В связи с этим сформулируем более подробно задачи, которые требуется решить.

Масштабируемость гибридно-облачных СУБД. Отсутствие эффективных механизмов распределения данных является существенным ограничением масштабируемости информационной системы в целом. Чтобы получить преимущество использования облачной и тем более гибридно-облачной инфраструктуры, необходимо обеспечить механизм доступа к данным, который допускал бы гибкое перераспределение данных при масштабировании системы с возможностью создания нескольких пересекающихся копий данных, обеспечивая синхронизацию данных, эффективный поиск, возможность разбиения сложных запросов при обращении к ним.

Базы данных класса NoSQL удовлетворяют поставленному требованию, тем не менее при интенсивной работе с большим количеством операций записи они проигрывают в эффективности по сравнению с традиционными реляционными БД и не обеспечивают ACID свойств, традиционных для реляционных СУБД [6]. Система Cloudy не позволяет разбивать запросы, что делает невозможным выполнение запросов, относящихся к данным на разных физических узлах. Сервис Relational Cloud наиболее полно решает задачу, но не предоставляет возможности использования в гибридно-облачной инфраструктуре.

Синхронизация данных в публичной и частной инфраструктурах. Для эффективной работы системы в гибридно-облачной инфраструктуре необходимо иметь копии часто используемых данных в обеих инфраструктурах. При этом требуется обеспечить возможность автоматической синхронизации копий данных и обеспечения их непротиворечивого состояния.

Способность автономного функционирования без подключения к сети Интернет. Требуется организовать доступ к данным таким образом, чтобы при временном разрыве соединения между публичной и частной инфраструктурами работоспособность информационных систем сохранялась, возможно, с некоторыми ограничениями. При восстановлении соединения работоспособность должна полностью восстанавливаться.

Контроль передачи данных через глобальную сеть. Многие информационные системы, в которых имеются конфиденциальные данные, не используют облачные и гибридно-облачные инфраструктуры, поскольку имеющиеся технологии не позволяют обеспечить требуемый уровень безопасности. В частности, при использовании гибридно-облачной инфраструктуры требуется обеспечить контроль передачи конфиденциальных данных в сеть Интернет на основе задаваемых политик, чтобы требуемый уровень безопас-

ности был обеспечен (например, при работе с персональными данными). Имеющиеся системы в основном обеспечивают безопасность данных с помощью их шифрования, без возможности ограничения их передачи.

Требуется создать модель политик безопасности передачи данных, которая позволит задавать правила допустимости передачи определенных данных между инфраструктурами.

Совместимость с имеющимся программным обеспечением. Система должна обеспечивать работу имеющегося ПО без внесения существенных изменений. Данное требование удовлетворяется использованием промежуточного слоя доступа к данным, который будет обеспечивать привычный интерфейс для имеющегося ПО и в то же время удовлетворять требуемым свойствам.

Расширенная модель представления запросов к СУБД

В качестве основы для разрабатываемой математической модели данных используется традиционная реляционная модель. Данная модель является наиболее распространенной. Кроме того, набирающая в последнее время популярность объектная модель данных в большинстве случаев также сводится в реляционной модели через механизмы объектно-реляционного отображения (Object-Relational Mapping – ORM). Таким образом, выбор реляционной модели данных в наибольшей степени отвечает поставленному требованию сохранения совместимости с имеющимся ПО.

Условные обозначения. Приведенные ниже математические объекты и их обозначения соответствуют общепринятой реляционной модели данных:

4) отношение (таблица) $R(A_1, A_2, ..., A_n)$;

5) атрибут отношения (столбец данных) $A_i \in A$;

6) кортеж (строка данных) $(a_1, a_2, ..., a_n)$, где $a_1 \in A_1, a_2 \in A_2, \ldots, a_n \in A_n$;

7) условие выборки – предикат атрибута отношения $cw(A_i) \in A_i$;

8) условие соединения атрибутов отношений $cj \in C$; $cj = <A_i, A_j, \text{condition}>$, причем A_i и A_j – атрибуты разных отношений

Традиционная модель. Традиционные реляционные СУБД исходят из предположения, что все данные хранятся централизованно. По этой причине *запрос данных* $z_{t1} \in Z_t$, где Z_t – это множество всех возможных запросов данных, представляет собой кортеж $(A_1, A_2, ..., A_k, cw_1, cw_2, ..., cw_l, cj_1, cj_2, ..., cj_l)$, который содержит запрашиваемые столбцы, их условия выборки и соединения, поскольку остальные параметры запроса не важны с точки зрения получения данных. Кроме этого СУБД имеет информацию об учетной записи пользователя, сделавшего данный запрос, что реализует базовый механизм безопасности доступа к данным.

В случае использования шардинга определением места хранения запрашиваемых данных занимается прикладное ПО. Таким образом, запрос данных z_{t2} в этом случае представляет собой кортеж, идентичный указанному выше, с дополнительным ограничением: все столбцы данных Ai должны относиться к одному источнику данных. Сложные запросы, требующие объединения результатов из разных источников, не могут быть выполнены стандартными средствами.

На уровне контроля доступа к данным при обработке запросов проверяются только права пользователя в отношении данных, с которыми он работает, что не позволяет контролировать передачу этих данных через глобальные сети.

Таким образом, для эффективной работы с данными, хранящимися распределенно, требуется расширить язык представления запросов.

Предлагаемое расширение традиционной модели. Предлагаемая модель обеспечивает возможность маршрутизации запросов в гибридно-облачной среде с учетом распределенного хранения данных и возможность контроля передачи запрашиваемых данных через глобальную сеть.

С этой целью в кортеж запроса данных добавлена информация о сетевом узле, с которого поступил запрос данных (*номер узла* $n \in N$): $z = (A_1, A_2, ..., A_k, cw_1, cw_2, ..., cw_l, cj_1, cj_2, ..., cj_l, n) \in Z$, где Z – это множество всех возможных запросов данных, расширенных номером узла, что позволяет отслеживать передачу запрашиваемых и вспомогательных данных в рамках запроса.

Кроме этого в модель добавлены такие объекты, как *распределение хранящихся данных* $s = (A_i, cw_i, N_1) \in S$, где $A_i \in A$, $N_1 \subset N$, и *политика*

допустимого перемещения данных $p = (A_i, cw_i, n) \in P$, где $A_i \in A$, $n \in N$. С помощью этих объектов можно проверить доступность для передачи запрашивающему узлу данных запроса, а также провести разбиение и распределенное выполнение запроса.

Сформированный указанным выше способом расширенный запрос направляется в промежуточный слой доступа к данным, где происходит поиск данных, проверка доступности и маршрутизация на основе заданного распределения хранящихся данных. Если данные распределены так, что один узел не может выполнить весь запрос целиком, промежуточный слой разбивает запрос таким образом, чтобы он был выполнен наиболее эффективно.

Пример. Исходный запрос от узла n: $z = (A_1, A_2, ..., A_k, cw_1, cw_2, ..., cw_l, cj_1, cj_2, ..., cj_l, n)$.

Распределение данных: $(A_1, ..., A_i, cw_1, ..., cw_j, n_1)$; $(A_{i+1}, ..., A_k, cw_j, ..., cw_l, n_2)$.

Итоговые запросы:

$z1 = (A_1, ..., A_i, cw_1, ..., cw_j, cj_1, ..., cj_t, n)$ для отправки на узел n_1;

$z2 = (A_{i+1}, ..., A_k, cw_{j+1}, ..., cw_l, cj_{t+1}, ..., cj_l, n)$ для отправки на узел n_2.

Кроме того, применяются политики перемещения данных к запрашивающему узлу из мест хранения. Процедура применения политик отличается для данных разных типов:

1. *Данные результата запроса* обязательно будут переданы источнику запроса, поэтому проверка необходима.

2. *Данные соединения* могут быть переданы на узел сборки или источник запроса, поэтому проверка также необходима.

3. *Данные выборки* не передаются, поскольку участвуют в выборке на узле хранения, поэтому проверка не требуется.

Пример. Предположим, что узел n сформировал запрос: $z = (A_1, A_2, ..., A_k, cw_1, cw_2, ..., cw_l, cj_1, cj_2, ..., cj_l, n)$.

В этом случае:

данные результата запроса – $(A_1, A_2, ..., A_k, cw_1, cw_2, ..., cw_k)$;

данные соединения – $(cj_1, cj_m, ..., cj_l)$;

данные выборки – $(cw_1, cw_2, ..., cw_l) \setminus (A_1, A_2, ..., A_k)$.

В результате проверки формируется список столбцов $A_i : \forall\, i\ A_i \subset (z \cap s = (A_i, cw_i, n_j) \cap p = (A_i, cw_i, n))$.

Если полученный список A_i содержит все столбцы, участвующие в запросе, значит запрос может быть выполнен. Иначе запрос выполнить невозможно.

$z \cap (A_i) = z \Rightarrow$ запрос удовлетворяет политикам.

$z \cap (A_i) \neq z \Rightarrow$ запрос нарушает одну или более политик.

Архитектура и принципы функционирования масштабируемого слоя доступа к данным гибридно-облачной платформы

В статье [7] предложена архитектура для распределенного хранения данных в одноранговой сети. Ее отличительной особенностью является групповая организация узлов с различными характеристиками для достижения требуемых свойств хранения. Также одноранговые сети предоставляют эффективные механизмы поиска по

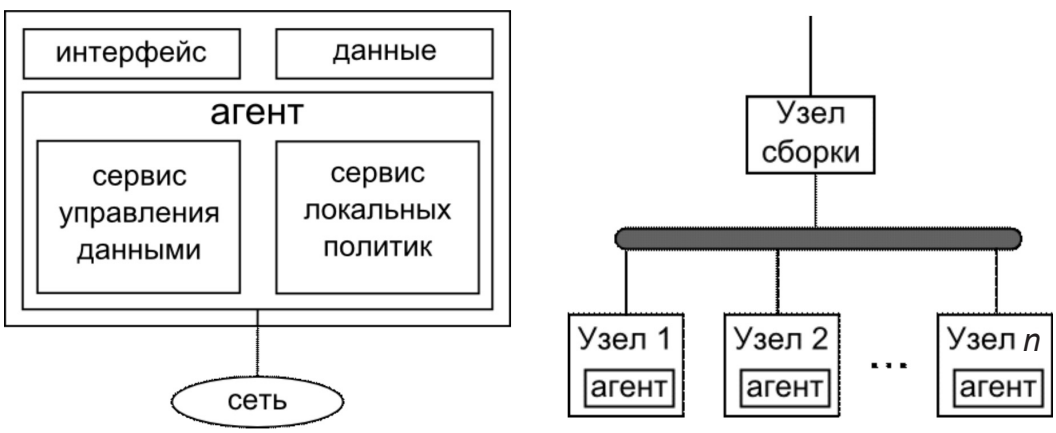

Рис. 1. Архитектура узла системы и пример группы узлов с узлом сборки

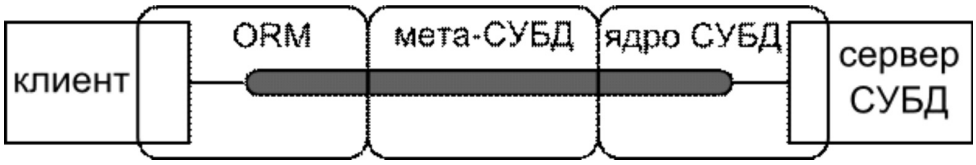

Рис. 2. Различные уровни встраивания промежуточного слоя доступа к данным

распределенным данным.

В нашем случае имеются две группы узлов, обладающие разными свойствами: ресурсы публичной и частной инфраструктур. Таким образом, мы можем использовать предложенную ранее архитектуру, чтобы учитывать гетерогенность этих групп узлов.

На рис. 1 представлена архитектура ПО узла системы и пример группы узлов с узлом сборки, о котором будет сказано позже.

На каждом узле имеется программный агент, с помощью которого реализуется промежуточный слой доступа к данным. Агент хранит информацию о том, какие данные хранятся на узле и политики безопасности, связанные с этими данными.

С целью повышения масштабируемости в сети может иметься *узел сборки*, который используется для связи с другими сетями, по аналогии с подходом, предложенным в [7]. Под сетью будем подразумевать либо набор узлов без узла сборки, либо узел сборки и связанные с ним узлы. Каждый узел сети может быть как самостоятельным узлом, хранящим данные, так и узлом сборки другой сети, через который происходит доступ к данным этой сети. Узел сборки хранит информацию о данных, имеющихся в сети, и политики безопасности для передачи этих данных в другие сети.

При необходимости получения данных с других узлов агент ищет узлы сети, которые могут предоставить эти данные. Если в сети данные не были найдены, запрос направляется на узел сборки, при его наличии. Узел сборки повторяет процедуру поиска в сети более высокого уровня.

В корне иерархии сетей находится сеть без узла сборки, и если в ней данные не были найдены, запрос выполнить невозможно.

Поддержка имеющегося программного обеспечения

С целью сохранения обратной совместимости с имеющимся ПО, предлагается реализовать механизм доступа к данным в гибридно-облачной системе в виде промежуточного слоя ПО. При работе с СУБД имеется несколько возможных вариантов встраивания промежуточного слоя ПО (рис. 2).

Уровень ORM (Object-relational mapping – Объектно-реляционное отображение) предполагает встраивание на стороне клиента в процессе преобразования объектного запроса в реляционный. Такой вариант потребует существенного изменения ПО, написанного без использования ORM.

Вариант встраивания на уровне ядра СУБД требует учета особенностей внутреннего представления запросов и данных, а также принципов работы каждой отдельной СУБД.

Мета-СУБД представляет собой абстрактную СУБД, являющуюся посредником между клиентским ПО и основной СУБД. Она получает запросы от клиентского ПО в исходном виде, запрашивает данные у основной СУБД и возвращает клиенту. В большинстве случаев использование такого подхода не потребует внесения изменений в имеющееся ПО, т. к. сохраняется стандартный программный интерфейс реляционной СУБД. Таким образом, требование совместимости с имеющимся ПО без внесения существенных изменений наилучшим образом обеспечивается вариантом с мета-СУБД.

Результаты исследования

Повышение масштабируемости гибридно-облачных СУБД достигнуто за счет использования модели представления запросов, дополняющей реляционную модель характеристиками распределения данных по множеству узлов. Это позволяет хранить копии данных на различных узлах, выполнять эффективный поиск методами одноранговых сетей. Наличие условий соединения позволяет разделять сложные запросы и выполнять их части на разных узлах.

Синхронизация данных в публичной и частной инфраструктурах обеспечивается за счет

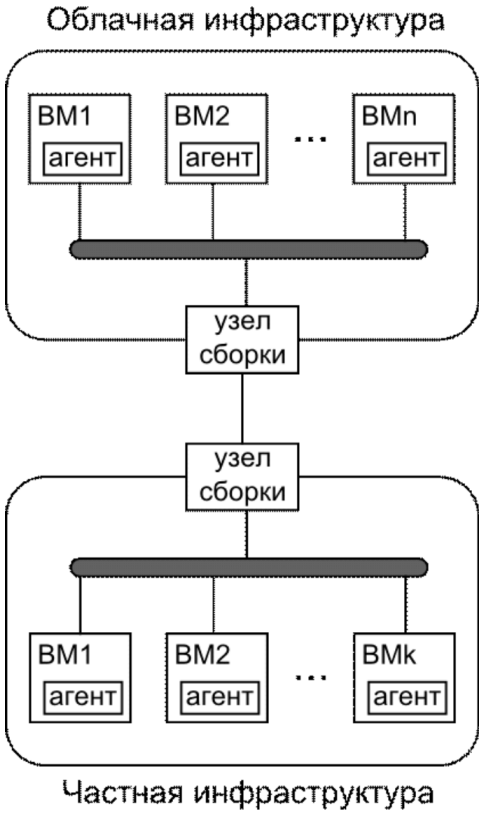

Рис. 3. Пример архитектуры системы

применения слоя мета-СУБД и распределенной архитектуры ПО на основе программных агентов. На рис. 3 представлен пример архитектуры, описанной выше.

Программные агенты выступают в качестве посредников в любой операции запроса или модификации данных за счет слоя мета-СУБД. Таким образом, агенты способны формировать снимки изменений и рассылать их другим агентам по аналогии с подходами, используемыми в одноранговых сетях [7].

Автономное функционирование без подключения к сети Интернет обеспечено за счет использования абстракции, когда другие сети воспринимаются как обычные узлы. В этом случае разрыв соединения между сегментами сети воспринимается аналогично отключению одного узла. Оба сегмента сети могут продолжать функционировать автономно с некоторыми ограничениями:

невозможность использования данных, которые стали недоступны;

кэшированные данные отключенного сегмента сети становятся доступны только для чтения.

После восстановления соединения производится синхронизация данных, и работоспособность системы полностью восстанавливается.

Обеспечен контроль передачи данных через глобальную сеть за счет применения политик допустимости передачи в слое доступа к данным. Применение политик – это процедура определения допустимости передачи данных узла на запрашивающий узел. Проверка считается успешной, если допустимо передать все данные, которые запросил узел.

В статье проанализированы возможные подходы к обеспечению масштабируемости реляционных СУБД в гибридно-облачной инфраструктуре и сформулированы цели и задачи развития существующих методов для достижения гибкого распределения данных, сочетающего разделение и дублирование данных, а также введение контроля данных, передаваемых между отдельными инфраструктурами, на основе политик. При этом важным требованием к создаваемому подходу является обеспечение возможности использования существующего ПО без существенных модификаций.

Предложен подход на основе расширения широко распространенной реляционной модели данных, а также архитектура программного слоя доступа к данным, которые позволяют достичь поставленных целей. В качестве дальнейшего развития подхода предполагается создание действующего прототипа на основе предложенной архитектуры и проведение экспериментального исследования свойств полученной системы.

СПИСОК ЛИТЕРАТУРЫ

1. **Стручков, И.В.** Способ установки, настройки, администрирования и резервного копирования программного обеспечения: Патент РФ № 2445686 [Текст] / И.В. Стручков. –Дата приоритета 21.01.2010.

2. **Curino, Carlo.** Relational Cloud: A Database-as-a-Service for the Cloud [Электронный ресурс] / Carlo Curino [et al.] // Biennial Conf. on Innovative Data Systems Research, CIDR 2011. –Jan. 9-12, 2011. –Asilomar, California.

3. **Donald, Kossmann.** Cloudy: A Modular Cloud Storage System [Электронный ресурс] / Donald Kossmann [et al.] // Proc. VLDB Endow. –Sept. 2010. –Vol. 3.

4. **Bajda-Pawlikowski, K.** Efficient processing of data warehousing queries in a split execution environment [Электронный ресурс] / K. Bajda-Pawlikowski [et al.] // Proc. of the 2011 International conf. on Management of data (SIGMOD'11) – ACM, 2011. –P. 1165–1176.

5. **Strauch, C.** NoSQL Databases [Электронный ресурс] / C. Strauch. –Режим доступа: https://oak.cs.ucla.edu/cs144/handouts/nosqldbs.pdf

6. **Arora, I.** Cloud Databases: A Paradigm Shift in Databases [Text] / I. Arora, A. Gupta // International J. of Computer Science Issues. –July 2012. –Vol. 9. –Iss. 4. –№ 3. –P. 77–83.

7. **Медведев, Т.А.** Распределенная система резервного копирования в гибридно-облачной среде [Текст] / Т.А. Медведев, И.В. Стручков // Вычислительные, измерительные и управляющие системы: сб. науч. тр. –СПб.: Изд-во Политехн. ун-та, 2010.