АЛГОРИТМ ДИНАМИЧЕСКОГО МАСШТАБИРОВАНИЯ РЕЛЯЦИОННЫХ БАЗ ДАННЫХ В ОБЛАЧНЫХ СРЕДАХ

УДК 004.65

Александр Викторович Бойченко,
к.т.н., начальник УНИР,
Московский государственный университет экономики, статистики и информатики (МЭСИ)
Тел.: (495) 442-60-11
Эл. почта: ABoichenko@mesi.ru

Дмитрий Константинович Рогожин,
аспирант, инженер-программист научно-образовательного центра УНИР,
Московский государственный университет экономики, статистики и информатики (МЭСИ)
Тел.: (495) 442-82-33
Эл. почта: DRogojin@mesi.ru

Дмитрий Геннадьевич Корнеев,
к.э.н., начальник отдела фандрайзинга УНИР,
Московский государственный университет экономики, статистики и информатики (МЭСИ)
Тел.: (495) 442-60-11
Эл. почта: DKorneev@mesi.ru

В статье анализируются основные методы масштабирования баз данных (репликация, шардинг) и их поддержки на уровне популярных реляционных СУБД и NoSQL решений с различными моделями данных: документо-ориентированной, ключ-значение, поколоночной, графовой; приводится алгоритм динамического масштабирования реляционной базы данных (БД), учитывающий специфику различных типов даталогической модели БД.

Ключевые слова: *динамическое масштабирование, облачные вычисления, реляционные БД, NoSQL, шардинг, репликация.*

Alexander V. Boichenko,
PhD, Head of the Research Department,
Moscow State University of Economics, Statistics and Informatics (MESI)
Tel.: (495) 442-60-11
E-mail: ABoichenko@mesi.ru

Dminry K. Rogojin,
Post-graduate student, Moscow State University of Economics, Statistics and Informatics (MESI),
Tel.: (495) 442-82-33
E-mail: DRogojin@mesi.ru

Dmitry G. Korneev,
PhD, Head of Fundraising Division,
Moscow State University of Economics, Statistics and Informatics (MESI),
Tel.: (495) 442-82-33
E-mail: DKorneev@mesi.ru

ALGORITHM FOR DYNAMIC SCALING RELATIONAL DATABASE IN CLOUDS

This article analyzes the main methods of scaling databases (replication, sharding) and their support at the popular relational databases and NoSQL solutions with different data models: document-oriented, key-value, column-oriented and graph. The article presents an algorithm for the dynamic scaling of a relational database (DB), that takes into account the specifics of the different types of logic database model. This article was prepared with the support of RFBR (grant № 13-07-00749).

Keywords: *dynamic scaling, cloud computing, relational DB, NoSQL, sharding, replication*

1. Введение

В условиях постоянно изменяющейся нагрузки, связанной с возрастанием количества обращений к сервису и возрастанием объемов обрабатываемых данных способность сервиса к быстромудинамическому масштабированию в облачной инфраструктуре является главным преимуществом перед классическим подходом доставки сервисов. Эта способность позволяет повысить отказоустойчивость, уровень доступности и производительность сервиса. Под динамическим масштабированием подразумевается свойство системы регулировать ресурсы, не требуя оперативного вмешательства пользователя. Динамическое масштабирование позволяет снизить время реакции на инциденты, увеличить уровень доступности сервиса и, тем самым, повысить качество предоставляемых сервисов. Возможности динамического масштабирования серверов на основе набора триггеров и метрик предоставляют своим клиентам такие компании, как Amazon (на основе решений *CloudWatch* и *ElasticCloud*), *Rackspace* (на основе решений *Rackspace Cloud Monitoring* и *Otter*). Создаваемая система *TIRAMOLA* [1] – совместный труд греческих университетов *National Technical University of Athens и Ionian University* – позволяет динамически масштабировать некоторые *NoSQL* СУБД с поколоночной моделью данных. В настоящее время для масштабирования обычно используются алгоритмы, основанные только на показателях функционирования технических устройств. Таким образом, разработка алгоритмов, учитывающих особенности структур БД, используемых сервисом, для возможности динамически выполнять масштабирование БД в облачной инфраструктуре является актуальной темой исследований.

2. Существующие методы масштабирования БД

Различают вертикальное и горизонтальное масштабирование БД. При вертикальном масштабировании новая виртуальная машина с сервером БД будет не добавляться в кластер, а заменять существующий сервер, обеспечивая вертикальное масштабирование. Однако, у данного подхода есть свои существенные недостатки. Во-первых, во время замены одного сервера другим сервис становится недоступен. Задержка в этом случае складывается из времени выполнения следующих операций: остановка программного сервера СУБД и размонтирование раздела с данными (эти действия необходимы для обеспечения сохранности данных), запуска новой виртуальной машины, монтирования раздела с данными к новой виртуальной машине. В итоге мы имеем задержку в несколько минут, которая может оказаться не такой важной для многих типов систем, но для других станет критичной. Во-вторых, и это главный недостаток, бесконечно масштабироваться вертикально невозможно, и в какой-то момент вертикальное масштабирование перестанет быть оптимальным решением. В частности, увеличение производительности сервиса за счет добавление новых процессоров для параллельной обработки запросов к базе данных ограничивается законом Амдаля-Уэра:

$$S_p = \frac{1}{\alpha + \frac{1-\alpha}{p}},$$

где p – количество процессоров (ядер), а $(1 - \alpha)$ – доля задач, которые могут быть распараллелены. Для СУБД с поддержкой транзакций эта доля невелика, поэтому с определенного момента добавление процессор-

[1] Статья подготовлена при поддержке РФФИ (грант № 13-07-00749).

ных мощностей перестанет давать ожидаемый эффект. [2] Поэтому вертикальное масштабирование уровня БД не является универсальным средством масштабирования баз данных.

Далее более подробно рассмотрим «горизонтальное масштабирование, предполагающее добавление новых аппаратных ресурсов. Рост интереса к горизонтальному масштабированию баз данных привел к появлению нового течения в хранении и обработке данных – *NoSQL* решениям, большинство из которых изначально проектировались с поддержкой устойчивости к разделению данных по сети. Классические реляционные СУБД: *Oracle*, *MySQL*, *MS SQL Server* также стремятся к поддержке устойчивости к разделению по сети в специальных "кластерных" версиях своих продуктов, жертвуя полноценными *ACID* транзакциями в пользу eventual consistency – "согласованность в конечном итоге" [3].

В настоящее время среди существующих методов горизонтального масштабирования, имеющих поддержку в реляционных БД и *NoSQL* решениях, принято выделять репликацию, секционирование (партицирование) и шардинг. [4, 5]

Репликация — это процесс, под которым понимается копирование данных из одного источника на множество других и наоборот. Существует около 10 видов репликации серверов баз данных: мульти-мастер (*multi-master*, *master-master*), мастер-подчинённый (*master-slave*), мульти-источник (*multy-source*), мастер-подчинённый-мастер (*master-slave-master*), мульти-сервер (*multi-server parallel query execution*), сектор-подчинённый (*mirror data partitioning*) и др. [6] Несмотря на то, что репликация теоретически является бесконечно наращиваемым решением, она способна решить только проблемы чтения данных из БД. Увеличение числа серверов становится нецелесообразным, когда появляются проблемы с записью данных в БД. [5]

Секционирование (партицирование) – средство масштабирования многих современных реляционных баз данных, которое позволяет разбивать таблицы, индексы и индекс-таблицы на части, таким образом, обеспечивая контроль и доступ к данным объектам базы данных на более низком уровне. Каждая из этих частей объекта базы данных называется секцией (или подсекцией для составных секционированных объектов. Таблицы секционируются с использованием «ключа секционирования», набора столбцов, определяющих, в какой секции будет располагаться заданная запись. Более подробно механизмы реализации секционирования описаны в [2, 7]. Существенным минусом механизма секционирования применительно к горизонтальному масштабированию уровня БД является то, что секционирование не выходит за рамки одного сервера. То есть, полученные в результате секции объектов физически располагаются на том же сервере.

Шардинг – разделение данных на уровне ресурсов – процесс, по смыслу схожий с секционированием, однако полученные в результате разделения объекты разносятся по разным серверам БД. Принято выделять *вертикальный шардинг*, при котором таблица (коллекция) выносится на другой сервер целиком, и *горизонтальный шардинг*, при котором на разные сервера выносятся части одной таблицы (коллекции). Подобно процессу секционирования, при шардинге выбирается «ключ шардирования», определяющий, на каком сервере будут располагаться части данных. Логика поиска сервера, на котором располагаются необходимые данные, заимствована из алгоритма разбиения пространства ключей DHT (англ. *Distributed Hash Table* — «распределённая хеш-таблица»). [8] В основе этого способа разбиения лежит функция $\delta(k_1, k_2)$, определяющая абстрактное понятие расстояния между ключами k_1 и k_2. Каждому узлу присваивается единичный ключ, называемый его идентификатором (*ID*). Узел с *ID* i_n владеет всеми ключами *km*, для которых i_n – самый ближайший *ID*, вычисленный с помощью $\delta(k_m, i_n)$. [9] Наиболее удобным является механизм так называемого авто-шардинга (*auto-sharding*), при котором реализация логики процесса шардинга осуществляется средствами самой СУБД. В таком случае нет необходимости реализовывать ее самостоятельно в рамках программного кода приложения или применять сторонние решения в качестве дополнительного промежуточного слоя. Помимо оптимизации операций чтения – читаются данные из более узкого набора, а не из всей таблицы (коллекции) целиком, шардинг решает также проблемы записи в БД.

Выбор того или иного средства для обеспечения масштабирования уровня БД зависит в первую очередь от комплексного анализа состояния БД и технических средств и выявления возможной точки отказа. Зачастую, при очень высокой нагрузке, необходимо применить вышеописанные методы масштабирования в комплексе. Выбор какого-то конкретного метода зависит не только от поставленной задачи, но и от поддержки метода на уровне самой СУБД. В Таблице 1 приведены сведения о поддержке и реализации репликации и шардинга в современных СУБД с различными моделями данных: документо-ориентированной, ключ-значение, поколоночной, графовой и реляционной. Поскольку механизмы секционирования не имеют прямого отношения к организации масштабируемого кластера в облачной инфраструктуре, в данной таблице и далее они не рассматриваются.

3. Алгоритм динамического шардинга реляционных БД

Как видно из таблицы, большинство *NoSQL* решений (кроме графовых БД) обладают встроенными механизмами автоматического шардинга в базовой поставке, в отличие от реляционных БД, где шардинг (точнее, инструментарий для осуществления шардирования, который использует администратор БД) доступен только в специальных версиях. Такое ограничение в реляционных БД волне логично – проектировщику БД предоставляется выбор: что важнее – полноценная поддержка транзакций в нераспределенной среде или поддержка устойчивости к разделению по сети с согласованностью в конечном итоге (ограничения теоремы *CAP:* ***Теорема***

Таблица 1

Реализации репликации и шардинга в современных СУБД

База данных	Режимы репликация	Механизм шардинга
Документо-ориентированные		
MongoDB	Мастер-подчиненный	*Auto-sharding* (на основе «осколков» (*Shards*) и «кусков» (*Chunks*)) [12]
CouchDB	Мульти-мастер	Встроенной поддержки нет. Осуществляется сторонними решениями [13]: *CouchDb-Lounge*, *BigCouch*, *Gizzard*
Ключ-значение		
Riak	Мульти-мастер	*Auto-sharding* (на основе *Ring Partitions*)
Redis	Мастер-подчиненный	Встроенной поддержки нет. Осуществляется сторонними решениями [14]: *Redis Cluster*, *Twemproxy*, *Predis*
Поколоночные (Столбцовые)		
HBase	Мастер-подчиненный	*Auto-sharding* (на основе Регионов (*Regions*))
Cassandra	Мульти-мастер	*Auto-sharding* (на основе *Ring Partitions*) [8]
Графовые		
Neo4j	Мастер-подчиненный	Полноценной поддержки нет. Частично реализован в технологии *Cache Sharding* [11]
OrientDB	Мульти-мастер	Нет
Реляционные		
Oracle Database	Мульти-мастер, Мастер-подчиненный	Только для *Oracle Real Application Clusters* [15]
MySQL	Мульти-мастер, Мастер-подчиненный, «Круговая» репликация	*Sharding* (только в версии *MySQL Cluster*) [16]
MS SQL Server	Мастер-подчиненный, *Multi-source*	*Data-Dependent Routing* (для *MS SQL Server*), *Federation* (для *SQL Azure*) [17]
PostgreSQL	Мульти-мастер, Мастер-подчиненный	Встроенной поддержки нет. Осуществляется сторонними решениями [18]: *PL/Proxy*, *HadoopDB*, *PgBouncer*.

CAP *(известная также как* ***теорема Брюера****) – эвристическое утверждение о том, что в любой реализации распределённых вычислений возможно обеспечить не более двух из трёх следующих свойств: согласованность данных (англ. consistency) – во всех вычислительных узлах в один момент времени данные не противоречат друг другу; доступность (англ. availability) – любой запрос к распределённой системе завершается корректным откликом; устойчивость к разделению (англ. partition tolerance) – расщепление распределённой системы на несколько изолированных секций не приводит к некорректности отклика от каждой из секций*. [10]). Для графовых БД отсутствие шардинга объясняется теми же ограничениями, поскольку большинство СУБД, основанных на графовых моделях, имеют полноценную поддержку *ACID*-транзакций (Atomicity-атомарность, Consistency-согласованность данных, Isolation-изолированность, Durability-надежность) подобно традиционным реляционным базам данных. [11] В свою очередь, реляционные СУБД имеют более широкий диапазон выбора методов репликации, апробированных в процессе многолетнего использования реляционных БД.

Предлагается следующий алгоритм реализации динамического шардинга реляционных БД, учитывающий специфику наиболее часто встречающихся типов даталогической модели БД и максимально обеспечивающий поддержку *ACID*-транзакций. По сравнению с существующими методами шардирования, предполагающими обязательное участие администратора БД, который осуществляет шардинг «в ручном» режиме, предлагаемый ниже алгоритм позволяет автоматизировать процесс шардирования. При выполнении алгоритма таблицы БД будут разделены таким образом, чтобы операторы соединения (JOIN) выполнялись в одном узле аппаратной архитектуры (шарде). Такие транзакции будут выполняться без задержек, связанных с необходимостью коммуникаций с другими узлами (за исключением задержек, возникающих при синхронизации реплик таблиц БД).

Шаг 1. Определение типа даталогической модели реляционной БД.

Шаг 2. а) Если тип модели «звезда». Авторы [19] в своих исследованиях предполагают, значительное количество «промышленных» БД (и преобладающее большинство БД, являющихся хранилищами данных для аналитических систем) имеют даталогическую схемы, в которой таблицы соединяются только с одной таблицей-потомком – центральной (корневой) таблицей. Для этого случая в [19] предлагается следующий метод для горизонтального шардинга: центральная таблица разделяется по диапазонам значений первичного ключа; таблицы-потомки разделяются по строкам, требуемым для соединения с выделенными разделами корневой таблицы. Выделенные разделы таблиц помещаются в различные шарды.

б) Если тип модели – «дерево». В этом случае различные таблицы соединяются друг с другом связями мощностью 1:N последовательно. То есть некоторая таблица соединяется по связями 1:N с несколькими таблицами, которые в свою очередь соединяются связями 1:N с другими таблицами и т.д. В этом случае на нижних

уровне будут находится таблицы, в которых находится максимальное количество записей и которые в первую очередь являются кандидатами на шардирование. В данном случае предлагается разделение таких таблиц по дипазонам внешних ключей с использованием таблицы индексов, построенной по внешнему ключу, в которой значения внешнего ключа отсортированы и присутствуют указатели на физический адрес записи индексируемой таблицы. Выделенные разделы таблиц помещаются в различные шарды.

Шаг 3. Определение таблиц, которые минимально подвержены корректировки (существенно преобладают операции чтения по сравнению с операторами корректировки, удаления, добавления записей).

Шаг 4. Репликация таблиц, выделенных на шаге 2, во всех шардах.

4. Направления для продолжения исследований.

В статье приводится возможный алгоритм для динамического разделения реляционных таблиц по узлам распределенной системы с целью минимизировать задержки при выполнении запросов, связанные с необходимостью коммуникаций с различными узлами аппаратной архитектуры (за исключением задержек, возникающих при синхронизации реплик таблиц БД). Исследования предполагается продолжить с целью разработки алгоритмов для графовых БД, для которых, как видно из Таблицы 1, в настоящий момент такие алгоритмы отсутствуют.

Литература

1. Tsoumakos D., Konstantinou I., Boumpouka C. On the Elasticity of NoSQL Databases over Cloud Management Platforms. / Proceeding CIKM ‘11 Proceedings of the 20th ACM international conference on Information and knowledge management, Pages 2385–2388, ISBN: 978-1-4503-0717-8, / URL: http://www.cslab.ntua.gr/~ikons/elastic_nosql.pdf (дата обращения: 07.11.2014).

2. Tsoumakos D., Konstantinou I., Boumpouka C. On the Elasticity of NoSQL Databases over Cloud Management Platforms. / Proceeding CIKM ‘11 Proceedings of the 20th ACM international conference on Information and knowledge management, Pages 2385–2388, ISBN: 978-1-4503-0717-8, / URL: http://www.cslab.ntua.gr/~ikons/elastic_nosql.pdf (дата обращения: 07.11.2014).

3. Burckhardt S., Leijen D. Eventually Consistent Transactions/ URL: http://research.microsoft.com/pubs/158085/ecr-esop2012.pdf / (дата обращения: 07.11.2014).

4. Рогожин Д. Средства масштабирования в облачной инфраструктуре / III научно-практическая конференция молодых ученых «Инновационное развитие российской экономики». 10 декабря 2012г. / Сборник научных трудов. – М.: МЭСИ, 2012. – Предм. указ.: с. 140 – 148.

5. Den Golotyuk. Шардинг, партиционирование, репликация – зачем и когда? / URL: http://highload.com.ua/index.php/2009/05/06/шардинг-партиционирование-репликац/ (дата обращения: 07.11.2014)

6. Косьмина Я.О. Репликация. Master-slave, кластер и прочее. / URL: http://freehabr.ru/blog/database/1119.html (дата обращения: 07.11.2014).

7. SQL Server 2012. Create Partitioned Tables and Indexes/ URL: http://technet.microsoft.com/en-us/library/ms188730.aspx (дата обращения: 07.11.2014).

8. Lomakin A. Apache Cassandra, part 1 – principles, data model. Презентация Exigen Services 28 июня 2011. / URL: http://www.slideshare.net/lomakin.andrey/apache-cassandra-part-1-principles-data-model (дата обращения: 07.11.2014).

9. Распределённая хеш-таблица. Материал из Википедии – свободной энциклопедии / URL: http://ru.wikipedia.org/wiki/Распределённая_хеш-таблица (дата обращения: 07.11.2014).

10. Seth Gilbert, Nancy Lynch. Brewer’s Conjecture and the Feasibility of Consistent, Available, Partition-Tolerant Web Services. – Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139. URL: http://people.csail.mit.edu/sethg/pubs/BrewersConjecture-SigAct.pdf (дата обращения: 07.11.2014).

11. How To Use Cache Sharding: Scale Out Neo4j / FromDev: A Technology Blog About Programming, Web Development, Tips, Tutorials and Books Recommendation for Developers / 14.10.2013 / URL: http://www.fromdev.com/2013/10/neo4j-cache-sharding-scale-out.html (дата обращения: 07.11.2014).

12. Sharding and MongoDB. MongoDB Documentation Project. November 01, 2013/ URL: http://docs.mongodb.org/v2.4/MongoDB-sharding-guide.pdf (дата обращения: 07.11.2014)

13. CouchDB. The Definitive Guide. Clustering. / URL: http://guide.couchdb.org/draft/clustering.html (дата обращения: 07.11.2013).

14. Partitioning: how to split data among multiple Redis instances./ URL: http://redis.io/topics/partitioning (дата обращения: 07.11.2013).

15. Oracle Real Application Clusters (RAC) . An Oracle White Paper, June 2013 / URL: http://www.oracle.com/technetwork/products/clustering/rac-wp-12c-1896129.pdf?ssSourceSiteId=ocomen (дата обращения: 07.11.2014).

16. Guide to Scaling Web Databases with MySQL Cluster. Accelerating Innovation on the Web and in the Cloud, A MySQL® White Paper, June 2013 / URL: http://www.mysql.com/why-mysql/white-papers/guide-to-scaling-web-databases-with-mysql-cluster/ (дата обращения: 07.11.2014).

17. MSDN. Scaling Out SQL Server. / Microsoft Corporation, April 2012. / URL: http://msdn.microsoft.com/en-us/library/aa479364.aspx (дата обращения: 07.11.2013).

18. Васильев А.Ю. Работа с Postgresql: настройка, масштабирование, – Справочное пособие, 2010. / URL: http://postgresql.ru.net/pgtune/postgresql.html#SECTION00600000000000000000 (дата обращения: 07.11.2014) / Глава 5. Шардинг.

19. Michael Stonebraker, Samuel Madden, Daniel J. Abadi, Stavros Harizopoulos, Nabil Hachem, Pat Helland.The End of an Architectural Era (It’s Time for a Complete Rewrite). Proceedings of the 33rd International Conference on Very Large Data Bases, 2007, pp. 1150–1160.

References

1. Tsoumakos D., Konstantinou I., Boumpouka C. On the Elasticity of NoSQL Databases over Cloud Management Platforms. / Proceeding CIKM '11 Proceedings of the 20th ACM international conference on Information and knowledge management, Pages 2385–2388, ISBN: 978-1-4503-0717-8, / URL: http://www.cslab.ntua.gr/~ikons/elastic_nosql.pdf (date of request: 07.11.2014).

2. Tsoumakos D., Konstantinou I., Boumpouka C. On the Elasticity of NoSQL Databases over Cloud Management Platforms. / Proceeding CIKM '11 Proceedings of the 20th ACM international conference on Information and knowledge management, Pages 2385–2388, ISBN: 978-1-4503-0717-8, / URL: http://www.cslab.ntua.gr/~ikons/elastic_nosql.pdf (date of request: 07.11.2014).

3. Burckhardt S., Leijen D. Eventually Consistent Transactions/ URL: http://research.microsoft.com/pubs/158085/ecr-esop2012.pdf / (date of request: 07.11.2014).

4. Rogogjin D. Solution of database scaling in cloud infrastructure / III nauchno-prakticheskaya konferenciya molodyh uchenyh "Innovacionnoe razvitie rossijskoj ekonomiki". 10 dekabrya 2012g. / Sbornik nauchnyh trudov. – M.: MESI, 2012. – Predm. ukaz.: s. 140 – 148.

5. Den Golotyuk, Sharding, partitsionirovaniye, replication – why and when?/ URL: http://highload.com.ua/index.php/2009/05/06/шардинг-партиционирование-репликац/ (date of request: 07.11.2014)

6. Kosmina Y. Replication. Master-slave, claster and so on./ URL: http://freehabr.ru/blog/database/1119.html (date of request: 07.11.2014).

7. SQL Server 2012. Create Partitioned Tables and Indexes/ URL: http://technet.microsoft.com/en-us/library/ms188730.aspx (date of request: 07.11.2014)

8. Lomakin A. Apache Cassandra, part 1 – principles, data model. Презентация Exigen Services 28 июня 2011. / URL: http://www.slideshare.net/lomakin.andrey/apache-cassandra-part-1-principles-data-model (date of request: 07.11.2014).

9. Distributed hash table. Wikipedia/ URL: http://ru.wikipedia.org/wiki/distributedhashtable (date of request: 07.11.2014).

10. Seth Gilbert, Nancy Lynch. Brewer's Conjecture and the Feasibility of Consistent, Available, Partition-Tolerant Web Services. – Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139. URL: http://people.csail.mit.edu/sethg/pubs/BrewersConjecture-SigAct.pdf (date of request: 07.11.2013).

11. How To Use Cache Sharding: Scale Out Neo4j / FromDev: A Technology Blog About Programming, Web Development, Tips, Tutorials and Books Recommendation for Developers / 14.10.2013 / URL: http://www.fromdev.com/2013/10/neo4j-cache-sharding-scale-out.html (date of request: 07.11.2014).

12. Sharding and MongoDB. MongoDB Documentation Project. November 01, 2013/ URL: http://docs.mongodb.org/v2.4/MongoDB-sharding-guide.pdf (date of request: 07.11.2014).

13. CouchDB. The Definitive Guide. Clustering. / URL: http://guide.couchdb.org/draft/clustering.html (date of request: 07.11.2013).

14. Partitioning: how to split data among multiple Redis instances./ URL: http://redis.io/topics/partitioning (date of request: 07.11.2013).

15. Oracle Real Application Clusters (RAC) . An Oracle White Paper, June 2013 / URL: http://www.oracle.com/technetwork/products/clustering/rac-wp-12c-1896129.pdf?ssSourceSiteId=ocomen (date of request: 07.11.2014).

16. Guide to Scaling Web Databases with MySQL Cluster. Accelerating Innovation on the Web and in the Cloud, A MySQL® White Paper, June 2013 / URL: http://www.mysql.com/why-mysql/white-papers/guide-to-scaling-web-databases-with-mysql-cluster/ (date of request: 07.11.2013).

17. MSDN. Scaling Out SQL Server. / Microsoft Corporation, April 2012. / URL: http://msdn.microsoft.com/en-us/library/aa479364.aspx (date of request: 07.11.2014).

18. Vasilev A.Yu. Work with Postgresql: control, scaling, – Spravochnoe posobie, 2010. / URL: http://postgresql.ru.net/pgtune/postgresql.html#SECTION00600000000000000000 (date of request: 07.11.2014) / Glava 5. Sharding.

19. Michael Stonebraker, Samuel Madden, Daniel J. Abadi, Stavros Harizopoulos, Nabil Hachem, Pat Helland.The End of an Architectural Era (It's Time for a Complete Rewrite). Proceedings of the 33rd International Conference on Very Large Data Bases, 2007, pp. 1150–1160.