DOI: 10.12731/2227-930X-2023-13-4-159-174
УДК 004.658

Научная статья | Системный анализ, управление и обработка информации

СХЕМЫ РАЗВЕРТЫВАНИЙ БАЗ ДАННЫХ ДЛЯ ОБЕСПЕЧЕНИЯ ВЫСОКОЙ ДОСТУПНОСТИ

Р.М. Хамитов

Повсеместная цифровизация и развитие технологий больших данных, требуют обеспечения высокого уровня доступности (24/7) информационных ресурсов. Обеспечение высокой доступности базы данных требует значительного объема работы по проектированию, тестированию и обслуживанию нескольких компонентов и конфигураций. Высокая доступность может быть достигнута или максимизирована для следующих вариантов развертывания: один экземпляр базы данных (с репликами чтения или без них) и несколько основных (с координатором или без него). Высокая доступность обеспечивает бесперебойную работу системы за счет устранения отдельных точек отказа, в то время как аварийное восстановление фокусируется на восстановлении нормальной работы после сбоя или простоя. Существует несколько вариантов развертывания, поскольку не существует универсального решения. Выбор наилучшего варианта зависит от вашего конкретного варианта использования и требований. В данной статье представлены основные концепции обеспечения высокой доступности и наиболее эффективные схемы для обеспечения высокодоступных развертываний баз данных.

***Цель** – определение эффективных схем развертываний серверов баз данных для обеспечения высокой доступности.*

***Метод или методология проведения работы.** В работе используются результаты зарубежных и отечественных научных*

исследований. Автором применяются теоретические методы исследования, связанные с поиском и анализом информации для выявления связей и получения уникальных выводов.

Результаты. *Выделены наиболее эффективные схемы развертывания для обеспечения высокой доступности и отказоустойчивости баз данных.*

Область применения результатов. *Полученные результаты целесообразно применять в области DevOps разработки с целью оптимизации процесса разработки и выпуска приложений путем устранения известного узкого места: изменений кода базы данных.*

Ключевые слова: *DevOps; управление данными; платформа данных; база данных; СУБД; высокая доступность; MySQL; NoSQL; PostgreSQL; паттерны*

Для цитирования. *Хамитов Р.М. Схемы развертываний баз данных для обеспечения высокой доступности // International Journal of Advanced Studies. 2023. Т. 13, № 4. С. 159-174. DOI: 10.12731/2227-930X-2023-13-4-159-174*

Original article | System Analysis, Management and Information Processing

DATABASE DEPLOYMENT PATTERN FOR HIGH AVAILABILITY

R.M. Khamitov

Ubiquitous digitalization and the development of big data technologies require a high level of availability (24/7) of information resources. Ensuring high availability of the database requires a significant amount of work on the design, testing and maintenance of several components and configurations. High availability can be achieved or maximized for the following deployment options: one database instance (with or without read replicas) and several main ones (with or without a coordinator). High availability ensures the smooth operation of the sys-

tem by eliminating individual points of failure, while disaster recovery focuses on restoring normal operation after a failure or downtime. There are several deployment options because there is no universal solution. Choosing the best option depends on your specific use case and requirements. This article presents the main concepts of ensuring high availability and the most effective patterns for providing highly accessible database deployments.

Purpose. *Definition of effective deployment patterns for database servers to ensure high availability.*

Methodology. *The results of foreign and domestic scientific research are used in the work. The author uses theoretical research methods related to the search and analysis of information to identify connections and obtain unique conclusions.*

Results. *The most effective deployment schemes for ensuring high availability and fault tolerance of databases are highlighted.*

Practical implications *it is advisable to apply the results obtained in the field of DevOps development in order to optimize the process of developing and releasing applications by eliminating a known bottleneck: changes to the database code.*

Keywords: *DevOps; Data Management; Data Platform; Database; DBMS; High Availability; MySQL; MSSQL; PostgreSQL; patterns*

For citation. *Khamitov R.M. Database Deployment Pattern for High Availability. International Journal of Advanced Studies, 2023, vol. 13, no. 4, pp. 159-174. DOI: 10.12731/2227-930X-2023-13-4-159-174*

Введение

Повсеместная цифровизация и развитие технологий больших данных [2], требуют обеспечения высокого уровня доступности (24/7) информационных ресурсов не только в корпоративном секторе [1], но и в образовании и других отраслях[4, 10].

По данным ManageForce [11], стоимость простоя базы данных оценивается в среднем в 474 000 долларов в час. Длительные простои базы данных являются результатом плохого про-

ектирования, касающегося отсутствия методов обеспечения высокой доступности. С резким ростом объема данных, генерируемых через Интернет (который, как ожидается, достигнет сотен зеттабайт к концу 2025 года [5], и растущей зависимостью от различных технологий для предоставления этих данных их предполагаемым пользователям стоимость простоя базы данных будет продолжать расти.

В данной статье сначала будут представлены основные концепции обеспечения высокой доступности. Затем мы перечислим наиболее эффективные схемы для обеспечения высокодоступных развертываний баз данных.

Материалы и методы исследования

В работе используются результаты зарубежных и отечественных научных исследований. Автором применяются теоретические методы исследования, связанные с поиском и анализом информации для выявления связей и получения уникальных выводов.

Результаты и обсуждение

Прежде чем углубляться в детали того, как добиться высокой доступности баз данных, давайте убедимся, что у нас есть общее понимание нескольких концепций.

Доступность – это показатель времени безотказной работы данной службы в течение определенного периода времени. Это можно понимать, как противоположность времени простоя. Например, ежемесячная доступность 99,99% подразумевает максимальное время простоя около 4 минут в месяц.

Долговечность – это мера способности данной системы сохранять данные при определенных сбоях (например, при отказе оборудования). Например, годовая надежность 99,9999999% означает, что вы можете терять один объект в год на каждый 1 миллиард объектов, которые вы храните!

Обратите внимание, что доступность данных и долговечность данных сильно различаются. Возможно, вам не удастся получить доступ к данным во время сбоя базы данных, но вы все равно ожидаете, что сохраненные данные будут доступны после возобновления работы базы данных.

Вычисления и хранение

В остальной части этой статьи будет использовано понятие *экземпляр (instance)* для обозначения *вычислительной* части развертывания базы данных на данном хосте. Экземпляр – это интерфейс, к которому подключается клиент базы данных (рисунок 1).

Кроме того, будет использоваться термин *«база данных»* (рисунок 2) для обозначения части *хранилища*, по сути, “файлов” данных, управляемых связанными экземплярами.

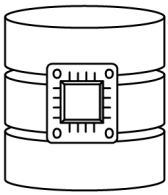

Рис. 1. Экземпляр базы данных

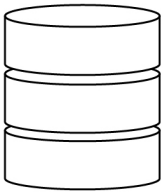

Рис. 2. База данных

Обратите внимание, что экземпляр и база данных могут находиться на разных хостах.

Схемы для достижения высокой доступности базы данных:

Высокая доступность обычно достигается с помощью конструкций *избыточности и изоляции. Избыточность* реализуется путем дублирования некоторых компонентов базы данных. *Изо-*

ляция достигается путем размещения избыточных компонентов на независимых хостах. Термин ***«кластер»*** относится ко всей совокупности компонентов развертывания базы данных, включая избыточные. Вместе эти компоненты обеспечивают доступность решения.

Давайте рассмотрим в следующем разделе некоторые Модели кластеризации.

*Избыточность / Кластеризующие конструкции***.** Как мы видели ранее, есть 2 основные части развертывания базы данных, которые мы можем сделать избыточными для достижения высокой доступности:

- экземпляр базы данных, который относится к вычислительной части.

- база данных, которая относится к части хранения.

*Кластеризация на уровне экземпляра***.** При таком типе кластеризации мы защищаем часть экземпляра базы данных, развертывая несколько экземпляров на разных хостах. База данных обычно находится в удаленном хранилище, видимом всем заинтересованным хостам.

У нас может быть 2 типа кластеризации на уровне экземпляра:

- *Активный* тип, при котором экземпляры активно обрабатывают запросы клиентов базы данных параллельно (рисунок 3). Это случай реального кластера приложений Oracle database [6]. При возникновении проблемы с одним экземпляром запросы клиентов базы данных будут перенаправлены на остальные работоспособные экземпляры.

- *Активный / пассивный* тип, при котором в любой момент времени только один экземпляр активно обрабатывает запросы клиентов базы данных (рисунок 4). При возникновении проблемы со старым активным экземпляром запросы клиентов будут перенесены во вновь выбранный активный экземпляр. Экземпляр отказоустойчивого кластера Microsoft SQL Server (FCI) [12] предлагает такой тип кластеризации.

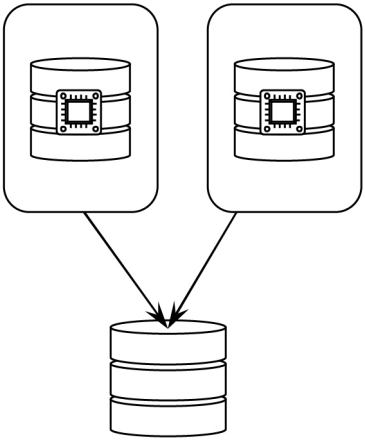

Рис. 3. Активная кластеризация на уровне экземпляра.

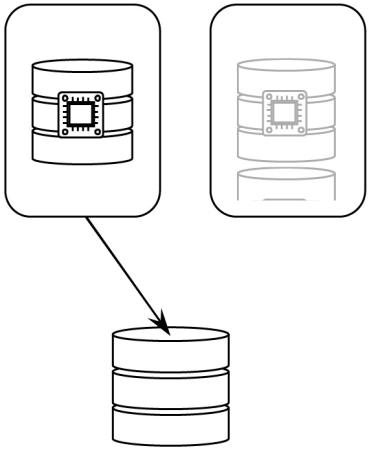

Рис. 4. Кластеризация на уровне активного / пассивного экземпляра

Кластеризация на уровне базы данных

При таком типе кластеризации мы защищаем как экземпляр базы данных, так и саму базу данных. Защита только базы данных может привести к ситуации, когда ваши данные защищены, но нет возможности быстро получить к ним доступ. Обратите вни-

мание, что мы можем создавать автономные копии базы данных для резервного копирования. Тем не менее, основной целью в таком случае является защита данных. Высокая доступность базы данных достигается за счет репликации. Мы можем выделить 2 типа репликации в зависимости от уровня, выполняющего ее:

- *Репликация на основе хранилища,* при которой протокол уровня хранилища / файловой системы передает изменения, происходящие в одной базе данных, в другие базы данных (рисунок 5). Это тип кластеризации, предлагаемый комбинацией экземпляра отказоустойчивого кластера Microsoft SQL Server (FCI) и Storage Space Direct. При этом типе репликации дополнительные экземпляры обычно находятся в режиме ожидания, а протокол репликации обычно синхронный.
- *Репликация на основе решения для баз данных*, при которой протокол уровня базы данных передает изменения, происходящие в одной базе данных, в другие базы данных. Такой тип кластеризации обеспечивается такими решениями, как ReplicaSet от MongoDB, AlwaysOn Availability Group от Microsoft SQL Server и Data Guard от Oracle Database. У нас может быть 2 подтипа репликации на основе решения для базы данных:

– *Логическая репликация*: когда мы реплицируем запросы клиента базы данных от одного экземпляра к другому. Репликация Oracle Golden Gate и MySQL на основе инструкций поддерживают этот тип репликации.

– *Физическая репликация:* когда мы реплицируем влияние клиентских запросов базы данных на данные из одной базы данных в другую (проходя через экземпляр). Oracle Data Guard и репликация на основе строк MySQL поддерживают этот вид репликации.

Мы используем термин «*реплики*» для обозначения дополнительных баз данных, возникающих в результате механизма репликации. *Основная* база данных / экземпляр – это та, которая получает трафик записи клиента.

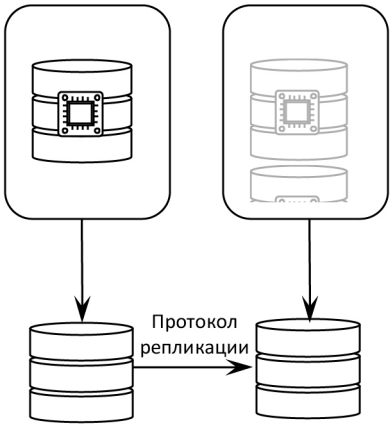

Рис. 5. Репликация на основе хранилища для кластеризации базы данных

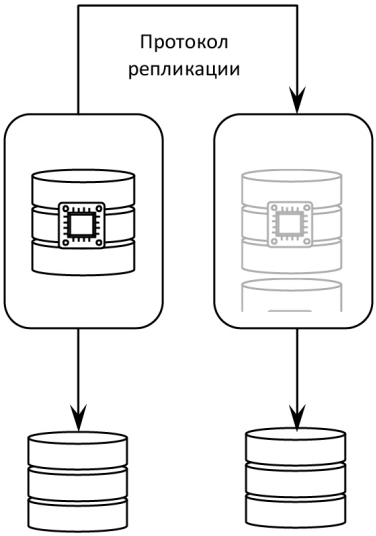

Рис. 6. Репликация на основе решения для кластеризации баз данных

Развертывания баз данных без общего доступа состоят из независимых серверов, каждый из которых имеет свою собственную выделенную память, вычислительные мощности и хранили-

ще. Они, как правило, обеспечивают более высокую доступность, поскольку позволяют нам использовать все конструкции изоляции, которые мы увидим в следующем разделе.

Конструкции изоляции. Изоляция заключается в уменьшении радиуса воздействия данного сбоя / аварийного события на компоненты вашего кластера. Чем дальше расположены ваши резервные компоненты, тем меньше вероятность того, что все они выйдут из строя одновременно (рисунок 7).

Рис. 7. Изменение изоляции данных

Изоляция сервера. Это самая элементарная форма изоляции. Размещение резервных компонентов на разных серверах предотвращает воздействие сбоя в сетевой карте, подключенном устройстве хранения данных или процессоре на все ваши резервные компоненты.

Изоляция в стойке. Стойка – это стандартизированный корпус для установки серверов и различного другого электронного оборудования. Серверы, размещенные в одной стойке, могут совместно использовать ряд элементов, таких как сетевые коммутаторы и кабели питания. Размещение ваших резервных компонентов на серверах, размещенных в разных стойках, предотвратит влияние сбоя в одном из совместно используемых компонентов на все ваше развертывание.

Изоляция центра обработки данных (ЦОД). Как правило, все серверы, размещенные в данном центре обработки данных, совместно используют инфраструктуру питания и охлаждения. Использование нескольких центров обработки данных для размещения развертывания базы данных сделает ее устойчивой к более

широкому спектру событий, таких как сбои питания и операции обслуживания в масштабах всего центра обработки данных.

Изоляция зоны доступности. Поставщики общедоступных облачных вычислений популяризировали концепцию "зоны доступности". Она состоит из одного или нескольких центров обработки данных, которые географически расположены близко друг к другу. Использование нескольких зон доступности для размещения развертываний базы данных может защитить ваши службы от некоторых "стихийных" бедствий, таких как пожар и наводнения.

Региональная изоляция. Мы можем пойти еще дальше в плане изоляции и использовать несколько регионов для развертывания нашей базы данных. Такого рода настройки могут защитить вашу базу данных от крупных стихийных бедствий, таких как штормы, извержения вулканов и даже политическая нестабильность (подумайте о переносе рабочей нагрузки из зоны боевых действий в другой регион). Теперь, когда у нас есть хороший обзор того, как сделать данное развертывание базы данных устойчивым к определенным событиям сбоя, нам нужно убедиться, что мы можем *автоматически* использовать его отказоустойчивость.

Автоматическое управление отказоустойчивостью

Отказоустойчивость – это процесс, посредством которого мы передаем право собственности на службу базы данных с неисправного сервера на исправный. Отказоустойчивость может быть инициирована вручную пользователем или автоматически компонентом развертывания базы данных. Использование вмешательства человека может привести к увеличению времени простоя по сравнению с автоматическими отказами, поэтому этого следует избегать.

Отказоустойчивость на стороне сервера. Чтобы автоматически инициировать переход от основного экземпляра / базы данных к реплике, компонент должен отслеживать состояние каждого из экземпляров и решать, в какой из исправных следует перенести службу. Вот список средств для восстановления работоспособности некоторых популярных баз данных:

- Orchestrator для MySQL/MariaDB [8];
- Patroni (включая необходимое распределенное хранилище конфигурации) для PostgreSQL [2, 9];
- Отказоустойчивый кластер Windows Server (WSFC), включающий свидетелей голосования участников для SQL Server [12];
- Oracle Data Guard (включая компонент observers) для Oracle database [6] .

Если вы планируете использовать такие инструменты, то вам необходимо выполнить серию тестов (как на первичных, так и на репликах), чтобы убедиться, что поведение кластера соответствует вашим ожиданиям:

- корректно остановить основной процесс создания экземпляра базы данных;
- внезапно завершить основной процесс создания экземпляра базы данных
- перезагрузить хостинг-сервер;
- отключить процессы кластеризации / отказа;
- изолировать, с точки зрения сети, соответствующий экземпляр базы данных от остальных экземпляров базы данных.

В зависимости от поведения выбранного решения вам может потребоваться реализовать собственную настройку в соответствии с вашими требованиями.

Отказоустойчивость на стороне приложения. Важным аспектом высокой доступности базы данных, который часто упускается из виду, является способность приложения быстро (повторно) подключаться к исправным экземплярам базы данных после сбоя сервера.

Для быстрого перехода ваших приложений к работоспособным экземплярам может потребоваться:

- установить разумные настройки сохранения работоспособности TCP, которые позволят вашим приложениям вовремя обнаруживать обрыв соединения;

- установить тайм-ауты для всех ваших обращений к базе данных, чтобы избежать зависания соединений;
- реализовать логику повторных попыток в своих приложениях для устранения определенных типов ошибок базы данных / сети;
- использовать прокси-серверы базы данных, чтобы скрыть изменения основного хоста от клиента базы данных.

Как видно, обеспечение высокой доступности базы данных требует значительного объема работы по проектированию, тестированию и обслуживанию нескольких компонентов и конфигураций. Высокая доступность может быть достигнута или максимизирована для следующих вариантов развертывания: один экземпляр базы данных (с репликами чтения или без них) и несколько основных (с координатором или без него). Высокая доступность обеспечивает бесперебойную работу системы за счет устранения отдельных точек отказа, в то время как аварийное восстановление фокусируется на восстановлении нормальной работы после сбоя или простоя. Существует несколько вариантов развертывания, поскольку не существует универсального решения. Выбор наилучшего варианта зависит от вашего конкретного варианта использования и требований.

Список литературы

1. Бочаров А. А. Архитектура серверов корпоративных баз данных // Диалог культур: Материалы XVI Международной научно-практической конференции на английском языке. В 3-х частях, Санкт-Петербург, 17–19 мая 2023 года / Под общей редакцией В.В. Кирилловой. Том Часть I. Санкт-Петербург: Санкт-Петербургский государственный университет промышленных технологий и дизайна, 2023. С. 158-162.
2. Будзко В. И. Развитие систем высокой доступности с применением технологии «большие данные» // Системы высокой доступности. 2013. Т. 9, № 4. С. 003-011.

3. Смирнов А. А. Репликация и высокая доступность в Postgresql // Научный Лидер. 2023. № 30(128). С. 30-31.
4. Хамитов Р. М. Цифровизация образования и ее аспекты // Современные проблемы науки и образования. 2021. № 3. С. 8. https://doi.org/10.17513/spno.30771
5. Data Attack Surface Report Steve Morgan, Editor-in-Chief Northport, N.Y. June 8, 2020. URL: https://cybersecurityventures.com/wp-content/uploads/2020/12/ArcserveDataReport2020.pdf (дата обращения: 14.10.2023).
6. Data Guard. URL: https://www.oracle.com/technical-resources/articles/smiley-fsfo.html (дата обращения: 14.10.2023).
7. Mohamed Wadie Nsiri Patterns to achieve database High Availability. URL: https://ubuntu.com/blog/database-high-availability (дата обращения: 14.10.2023).
8. Orchestrator. URL: https://github.com/openark/orchestrator (дата обращения: 14.10.2023).
9. Patroni. URL: https://patroni.readthedocs.io/en/latest/ (дата обращения: 14.10.2023).
10. Shevchenko O.M., Torkunova Yu.V., Upshinskaya A.E., Shorina T.V. Learning Data Visualization in Assessing Linguistic Competence in the International Baccalaureate // European Proceedings of Social and Behavioral Sciences. Conference proceedings. London, 2020. P. 1155-1164.
11. System Failure - The Cost of Database Downtime. URL: https://www.manageforce.com/blog/dba-suffering-from-system-failure-infographic (дата обращения: 14.10.2023).
12. WSFC. URL: https://docs.microsoft.com/en-us/sql/sql-server/failover-clusters/windows/windows-server-failover-clustering-wsfc-with-sql-server?view=sql-server-ver16 (дата обращения: 14.10.2023).

References

1. Bocharov A. A. *Dialog kul'tur: Materialy XVI Mezhdunarodnoy nauchno-prakticheskoy konferentsii na angliyskom yazyke. V 3-kh chastyakh, Sankt-Peterburg, 17–19 maya 2023 goda* [Dialogue of Cultures: Proceedings of the XVI International Scientific and Prac-

tical Conference in English. In 3 parts, St. Petersburg, May 17-19, 2023] / ed. V.V. Kirillova. Part 1. St. Petersburg: St. Petersburg State University of Industrial Technologies and Design, 2023, pp. 158-162.
2. Budzko V. I. *Sistemy vysokoy dostupnosti,* 2013, vol. 9, no. 4, pp. 003-011.
3. Smirnov A. A. *Nauchnyy Lider,* 2023, no. 30(128), pp. 30-31.
4. Khamitov R. M. *Sovremennye problemy nauki i obrazovaniya,* 2021, no. 3, p. 8. https://doi.org/10.17513/spno.30771
5. Data Attack Surface Report Steve Morgan, Editor-in-Chief Northport, N.Y. June 8, 2020. https://cybersecurityventures.com/wp-content/uploads/2020/12/ArcserveDataReport2020.pdf
6. Data Guard. https://www.oracle.com/technical-resources/articles/smiley-fsfo.html
7. Mohamed Wadie Nsiri Patterns to achieve database High Availability. https://ubuntu.com/blog/database-high-availability
8. Orchestrator. https://github.com/openark/orchestrator
9. Patroni. https://patroni.readthedocs.io/en/latest/
10. Shevchenko O.M., Torkunova Yu.V., Upshinskaya A.E., Shorina T.V. Learning Data Visualization in Assessing Linguistic Competence in the Inter-national Baccalaureate. *European Proceedings of Social and Behavioral Sciences. Conference proceedings*. London, 2020, pp. 1155-1164.
11. System Failure - The Cost of Database Downtime. https://www.manageforce.com/blog/dba-suffering-from-system-failure-infographic
12. WSFC. https://docs.microsoft.com/en-us/sql/sql-server/failover-clusters/windows/windows-server-failover-clustering-wsfc-with-sql-server?view=sql-server-ver16

ДАННЫЕ ОБ АВТОРЕ

Хамитов Ренат Минзашарифович, доцент кафедры «Информационные технологии и интеллектуальные системы», кандидат технических наук, доцент
ФГБОУ ВО «Казанский государственный энергетический университет»

ул. Красносельская, 51, г. Казань, 420066, Российская Федерация
hamitov@gmail.com

DATA ABOUT THE AUTHOR

Renat M. Khamitov, Associate Professor of the Department of Information Technologies and Intelligent Systems, Candidate of Technical Sciences
Kazan State Power Engineering University
51, Krasnoselskaya Str., Kazan, 420066, Russian Federation
hamitov@gmail.com
SPIN-code: 7401-9166
ORCID: https://orcid.org/0000-0002-9949-4404
Scopus Author ID: 57222149321

Поступила 13.11.2023
После рецензирования 28.11.2023
Принята 02.12.2023

Received 13.11.2023
Revised 28.11.2023
Accepted 02.12.2023