ОРИГИНАЛЬНАЯ СТАТЬЯ



DOI: 10.26794/3030-7097-2026-2-1-63-72
УДК 004.043+004.6004+7336.7+1004.056(045)

# Анализ подходов и решений для разработки системы хранения клиентских данных банковской экосистемы

**Н.А. Бурыкин**[1], **Н.В. Гринева**[1], **П.Е. Голосов**[2]
[1] Финансовый университет при Правительстве Российской Федерации, Москва, Российская Федерация;
[2] Российская академия народного хозяйства и государственной службы при Президенте Российской Федерации, Москва, Российская Федерация

## АННОТАЦИЯ

Статья посвящена анализу подходов и решений для разработки единого, масштабируемого и высоконагруженного хранилища клиентских данных в рамках цифровой экосистемы банка. В условиях цифровой трансформации и развития цифровых экосистем крупные организации, в том числе банки, сталкиваются с тем, что хранение и обработка клиентских данных становится критически важным вопросом для обеспечения бизнес-процессов компании, персонализации сервисов для клиентов, соответствия регуляторным требованиям, поддержания конкурентного преимущества. **Цель исследования** — определить оптимальный набор архитектурных принципов и технологических решений для построения масштабируемой, отказоустойчивой и высокопроизводительной системы хранения клиентских данных. **Методы исследования:** системный анализ и синтез — для изучения специфики предметной области банковской цифровой экосистемы; сравнительный анализ — для оценки технологий хранения данных по следующим критериям: производительность, масштабируемость, уровень согласованности данных, отказоустойчивость; анализ научной литературы и информационных источников — для изучения предметной области, современных решений и технологий в системах хранения данных. **Результаты исследования:** выбраны оптимальные паттерны и технологии для реализации платформы хранения клиентских данных; сформулированы рекомендации по проектированию архитектуры единого клиентского хранилища, способного эффективно функционировать в динамичном ИТ-ландшафте современной банковской экосистемы. **Выводы.** Результаты исследования подтверждают, что использование гибридных хранилищ данных с применением различных технологий и решений наиболее оптимально для построения высоконагруженных и масштабируемых систем хранения и обработки данных (что также согласуется с выводами других научных работ). **Перспективы исследования** включают дальнейшее углубленное изучение интеллектуальных систем управления данными; технологий хранения и обработки данных, используемых в машинном обучении и ИИ; подходов Data Mesh и Data Fabric; возможностей применения перечисленных технологий в банковской сфере.

***Ключевые слова:*** банковская экосистема; клиентские данные; высоконагруженные системы; NoSQL; реляционные базы данных; In-Memory хранилища; архитектура информационных систем

***Для цитирования:*** **Бурыкин Н.А., Гринева Н.В., Голосов П.Е. Анализ подходов и решений для разработки системы хранения клиентских данных банковской экосистемы. *Цифровые решения и технологии искусственного интеллекта*. 2026;2(1):63-72. DOI: 10.26794/3030-7097-2026-2-1-63-72**

ORIGINAL PAPER

# Analysis of Approaches and Solutions for Developing a Customer Data Storage System for the Banking Ecosystem

**N.A. Burykin**[1], **N.V. Grineva**[1], **P.E. Golosov**[2]
[1] Financial University under the Government of the Russian Federation, Moscow, Russia;
[2] Russian Presidential Academy of National Economy and Public Administration, Moscow, Russia

## ABSTRACT

The article is devoted to the analysis of approaches and solutions for the development of a single, scalable and highly loaded customer data warehouse within the bank's digital ecosystem. In the context of digital transformation and the development of digital ecosystems, large organizations, including banks, are faced with the fact that the storage and processing of customer data is becoming a critical issue for ensuring company processes, personalization of services for customers, compliance with regulatory requirements and maintaining a competitive advantage. The purpose of the research is to determine the optimal set of architectural principles and technological solutions applicable to building a scalable, fault-tolerant and high-performance customer data storage system. The following methods were used to conduct the research: system analysis and synthesis to study the specifics of the subject area of the banking digital ecosystem, comparative analysis to evaluate data storage technologies based on criteria such as performance, scalability, data consistency, fault tolerance, and analysis of scientific literature and information sources to study the subject area, modern solutions and technologies in systems data storage.

Based on the conducted research, optimal patterns and technologies for implementing a customer data storage platform have been selected and recommendations have been formulated for designing a single customer storage architecture capable of operating effectively in the dynamic IT landscape of the modern banking ecosystem. Based on the results of the study, it was concluded that other scientific papers and articles also confirm that the use of hybrid data warehouses using different technologies and solutions is most optimal for building highly loaded and scalable data storage and processing systems. The prospects of the research include further in-depth study of intelligent data management systems, data storage and processing used in machine learning and AI, Data Mesh and Data Fabric approaches and their application in the banking sector.
***Keywords:*** banking ecosystem; client data; high-load systems; NoSQL; relational databases; In-Memory storage; information systems architecture

***For citation:*** **Burykin N.A., Grineva N.V., Golosov P.E. Analysis of approaches and solutions for developing a customer data storage system for the banking ecosystem.** ***Digital solutions and artificial intelligence technologies.*** **2026;2(1):63-72. DOI:10.26794/3030-7097-2026-2-1-63-72**

## ВВЕДЕНИЕ

Современные банковские цифровые экосистемы находятся в состоянии цифровой трансформации, в рамках которой данные о клиентах превращаются из пассивного актива в ключевой стратегический ресурс, что позволяет создавать персонализированные сервисы, управлять рисками и обеспечить кредитной организации конкурентное преимущество. Процесс связан с обработкой растущих объемов различной информации. Помимо структурированных транзакционных данных, банки вынуждены анализировать массивы неструктурированных и слабоструктурированных данных, включая поведенческую аналитику, данные о взаимодействии и данные со сторонних платформ экосистемы [1]. Задача усложняется ужесточением регуляторных требований и растущих ожиданий клиентов в отношении скорости, доступности и персонализации сервисов в режиме 24/7.

В таких условиях классические централизованные и монолитные архитектуры систем хранения данных демонстрируют свою несостоятельность, становясь «узким местом» в развитии экосистемы. Их ограниченная масштабируемость, высокая стоимость владения и негибкость при интеграции новых источников данных формируют принципиально новые требования к инфраструктуре, где главными задачами становятся: обеспечение бесперебойной работы в условиях пиковых нагрузок, обработка данных в реальном времени, гарантированная согласованность и безопасность при распределенной обработке. Это соответствует принципам построения высоконагруженных отказоустойчивых систем, подробно исследуемых в фундаментальных работах по проектированию программного обеспечения.

**Актуальность данного исследования** обусловлена острой необходимостью в систематизации и поиске оптимальных архитектурных решений для построения распределенных высоконагруженных систем хранения данных, способных удовлетворить требования цифровой банковской экосистемы. Как отмечают исследователи, современные системы должны сочетать в себе гарантии согласованности ACID и гибкость принципов BASE для обработки больших данных и обеспечения высокой доступности [2]. Необходим комплексный подход, объединяющий достижения в области микросервисной архитектуры, контейнеризации, сегментирования (sharding) данных, а также синергетического использования различных классов систем хранения.

**Научная новизна** работы заключается в комплексном и системном анализе архитектурных паттернов и технологических решений для построения распределенных высоконагруженных систем хранения клиентских данных. Исследование предлагает целостную модель, рассматривающую проблему через призму интеграции разнородных систем хранения (SQL, NoSQL, in-memory), при одновременном соблюдении строгих требований к производительности, надежности и соответствию регуляторным нормам.

**Цель исследования** — разработка теоретико-методологического подхода и выбор решений для построения надежной высоконагруженной системы хранения данных клиентов с высокой производительностью, удовлетворяющей все требования к системам хранения клиентских данных в крупных организациях.

**Задачи исследования:** проанализировать предметные области и выявить специфические требования к современным системам хранения данных; выполнить сравнительный анализ современных подходов к организации распределенных систем хранения данных и выявить их применимость в контексте банковской цифровой экосистемы; оценить и систематизировать современные технологии хранения данных и подходы к разработке систем хранения клиентских данных.

## ИССЛЕДОВАНИЕ СУЩЕСТВУЮЩИХ ПОДХОДОВ К ХРАНЕНИЮ И ОБРАБОТКЕ ДАННЫХ ДЛЯ РАЗРАБОТКИ ВЫСОКОНАГРУЖЕННЫХ СИСТЕМ ХРАНЕНИЯ ДАННЫХ

Разработка высоконагруженной распределенной системы хранения для цифровой экосистемы банка

представляет собой комплексную задачу, требующую соответствия фундаментальным ограничениям, сформулированным в теореме CAP (Брюера). Согласно данной теореме, в распределенной системе невозможно одновременно обеспечить более двух из трех свойств: согласованности данных (Consistency), доступности (Availability) и устойчивости к разделению сети (Partition Tolerance) [3].

Этот принцип, как отмечают Таненбаум и ван Стейен, является краеугольным камнем при проектировании и делит все системы на два основных класса: CP-ориентированные (с акцентом на согласованность и устойчивость к разделению) и AP-ориентированные (с акцентом на доступность и устойчивость к разделению) [4].

Исторически основой для построения надежных систем хранения в финансовой сфере являлись реляционные СУБД (Oracle Database, PostgreSQL), основанные на принципах ACID (Atomicity, Consistency, Isolation, Durability — атомарность, согласованность, изолированность, долговечность). Они обеспечивают строгую транзакционную семантику, что критически важно для банковских систем, где целостность данных является неприкосновенной [5].

В ответ на вызовы масштабируемости и высокой доступности в эпоху Big Data и веб-сервисов сформировалась альтернативная парадигма BASE[1]. Данный набор принципов, подробно описанный Садаладжем и Фаулером, лежит в основе NoSQL-систем, которые жертвуют строгой согласованностью в пользу доступности и горизонтального масштабирования [5].

Однако, как подчеркивает Мартин Фаулер, при масштабировании такие системы сталкиваются с «бутылочным горлышком» в виде единой точки отказа и сложностями горизонтального масштабирования (sharding), что требует сложных процедур решардинга и может негативно сказываться на производительности при высоких нагрузках [5].

Для различных моделей данных и сценариев использования разработаны специализированные типы хранилищ:

- *«ключ-значение»* (Redis или Apache Ignite) — применяются в высоконагруженных проектах для кэширования и работы с данными в оперативной памяти, требующими минимальной задержки;
- ширококолоночные (Apache Cassandra). Отсутствие единой точки отказа и предсказуемая производительность при линейном масштабировании делают их привлекательными для задач, требующих записи и чтения больших объемов данных с временными рядами или событийными журналами [6];
- документоориентированные (MongoDB или Yandex Database). YDB от Яндекса — отличный пример современной гибридной системы, поддерживающей как документную модель, так и реляционные запросы с распределенными транзакциями[2].

Интересный тренд — конвергенция подходов, выраженная в появлении класса NewSQL-систем. Эти системы, такие как Google Spanner [7] и его открытые аналоги (CockroachDB и YDB), сочетают гарантии ACID реляционных баз данных с горизонтальной масштабируемостью NoSQ L. В их основе лежат алгоритмы распределенного консенсуса, такие как Paxos или Raft, для управления репликацией данных, что обеспечивает строгую согласованность в распределенной среде [8].

Еще один важный архитектурный паттерн — CQRS (Command Query Responsibility Segregation — разделение ответственности команд и запросов) в сочетании с Event Sourcing (источники событий)[3]. CQRS разделяет модели для операций обновления (команд) и чтения (запросов), позволяя независимо масштабировать и оптимизировать каждую из них.

Event Sourcing предполагает сохранение не конечного состояния объекта, а последовательности всех событий, которые к этому состоянию привели. Это не только обеспечивает полный аудит изменений, что крайне важно для банковской деятельности, но и позволяет строить различные материализованные представления (проекции) данных, оптимальные для конкретных сценариев чтения.

На *рис. 1* схематически представлено разделение моделей данных с применением подхода CQRS, где можно увидеть изоляцию моделей данных на уровне хранилищ данных: выполнение запросов на запись — в одно хранилище с конкретной моделью данных, выполнение запросов на чтение — в другое хранилище с другой моделью данных[4].

Активное развитие искусственного интеллекта (ИИ) и его внедрение в автоматизацию бизнес-процессов и производство в рамках цифровой банковской экосистемы сформировали новый подход к хранению и обработке данных — RAG (Retrieval-Augmented Generation), применяемый в интеллек-

[1] BASE. Basically Available, Soft state, Eventual consistency — фундаментальная доступность, неустойчивое состояние, потенциальная согласованность.

[2] Документация YDB. URL: https://ydb.tech/docs/ru/?version=v25.2

[3] CQRS Documents. URL: https://cqrs.wordpress.com/wp-content/uploads/2010/11/cqrs_documents.pdf

[4] Там же.

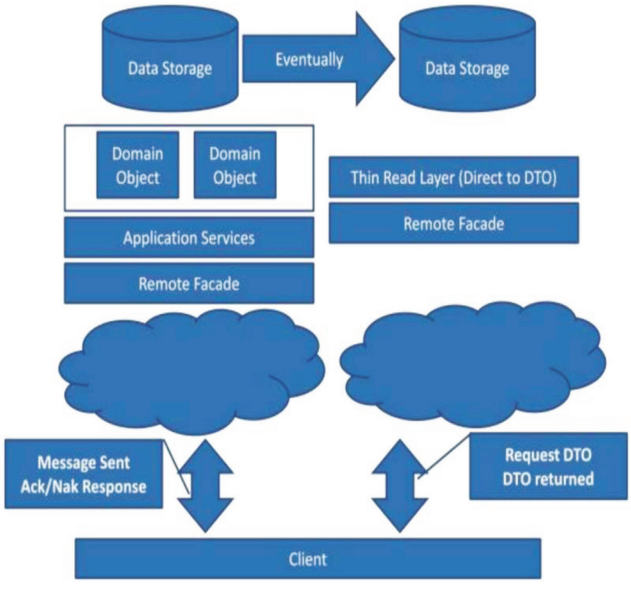


*Рис. 1 / Fig. 1.* **Разделение моделей данных с применением подхода CRQS / Separation of Data Models Using the CRQS Approach**

*Источник / Source:* документы CRQS / CQRS Documents.

туальных сервисах, где для извлечения информации используются запросы в виде натурального языка, сгенерированные большими языковыми моделями (LLM) или непосредственно человеком.

RAG разработан для повышения точности, актуальности и проверяемости выходных данных LLM за счет их обогащения релевантной информацией, извлеченной из внешних корпусов знаний (knowledge bases) [9]. Перед записью в RAG данные

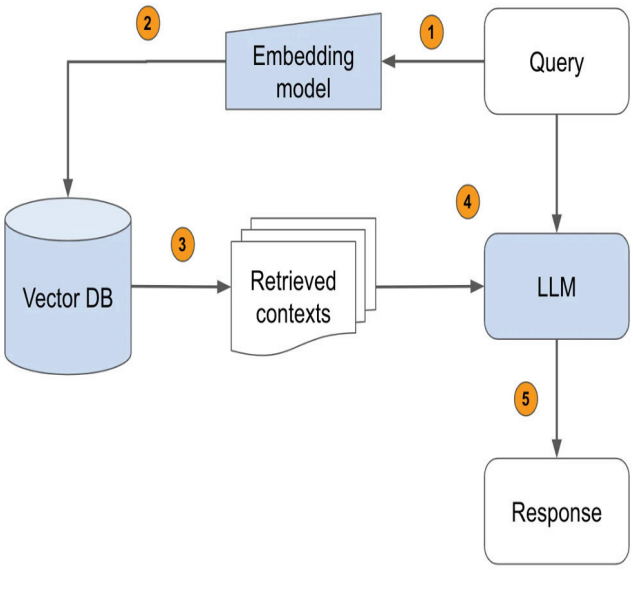


*Рис. 2 / Fig. 2.* **Схематическое описание работы RAG / Schematic description of RAG operation**

*Источник / Source:* RAG (Retrieval-Augmented Generation): основы и продвинутые техники*.

*Примечание / Note:* *URL: https://habr.com/ru/articles/871226/

индексируются путем преобразования в векторное представление данных (*embedding*) с помощью специализированных моделей машинного обучения.

Для извлечения данных из RAG запросы, полученные от пользователя (человек или LLM), также преобразуются в *embedding* для дальнейшего вычисления векторного расстояния между проиндексированными данными и поступившим от пользователя запросом — с целью извлечения информации, наиболее близкой по векторному расстоянию к поступившему запросу и смысловому значению.

Распространенными решениями векторных баз данных, применяемых в RAG, являются ElasticSearch, Qdrant, плагин PG Vector для СУБД PostgreSQL и Milvus.

На *рис. 2* представлено схематическое описание работы RAG.

Помимо технологий и решений хранения данных необходимо также рассмотреть архитектурные паттерны для разработки распределенных систем хранения. Глобально можно выделить главные архитектурные подходы разработки программного обеспечения (ПО) — монолитная и микросервисная архитектуры.

Монолитная архитектура характерна тем, что вся логика приложения, включая пользовательский интерфейс, бизнес-логику и доступ к данным, находится внутри одной единицы развертывания приложения. При монолитной архитектуре разработка, тестирование и развертывание монолитного приложения может быть относительно проще, так как все части находятся в одном месте из-за чего изменение или добавление новых функций может быть сложным, а масштабирование приложения может достигнуть своих пределов при росте объема данных или пользовательской нагрузки [10].

На *рис. 3* представлен пример классической монолитной архитектуры, где компоненты и слои приложения программного обеспечения объединены в единицу развертывания, а взаимодействие между компонентами осуществляется с помощью локальных вызовов.

Рассмотрим альтернативу монолитной архитектуре — микросервисную архитектуру — еще один подход к разработке программного обеспечения, при котором приложение строится как набор небольших, независимых и легко масштабируемых сервисов, взаимодействующих между собой через API (Application Program Interface). Каждый микросервис представляет собой отдельную и независимую функциональность или бизнес-задачу [10].

На *рис. 4* изображен пример микросервисной архитектуры. На схеме видно отсутствие общей единицы развертывания для ПО, как в случае мо-

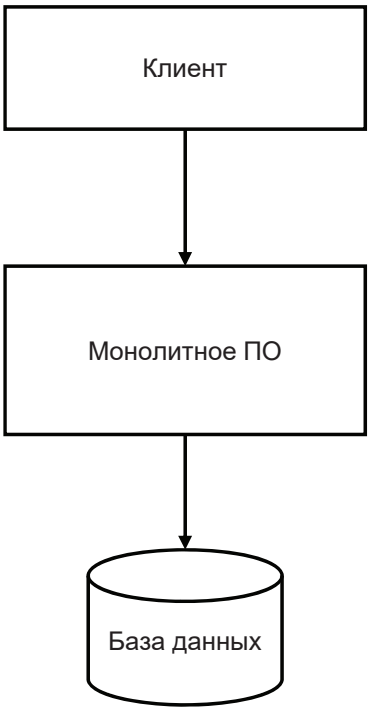


*Рис. 3 / Fig. 3.* **Монолитная архитектура ПО / Monolithic Software Architecture**

*Источник / Source:* составлено авторами / Complied by the authors.

нолитной архитектуры, и разделение ПО на микросервисы, выполняющие свои независимые функции и передающие вызовы другим микросервисам через API.

Анализ существующих подходов демонстрирует отсутствие универсального решения. Выбор архитектуры системы хранения для цифровой экосистемы банка должен определяться спецификой бизнес-требований к каждому типу данных, что предполагает переход от монолитной базы данных к полиглотному хранению (Polyglot Persistence)[5] — стратегии, где в рамках одной экосистемы используются различные системы хранения, наиболее подходящие для решения конкретных задач. Например, транзакционные данные могут храниться в CP-системе (NewSQL, такая как YDB), кэш-сессии — в кластере ключ-значение (Redis или Tarantool), а журналы операций — в ширококолонном хранилище (Apache Cassandra). Ключевая задача здесь — обеспечение надежной оркестрации (на основе таких платформ, как Kubernetes), мониторинга и согласованности данных между компонентами [11].

[5] Polyglot Persistence. URL: https://martinfowler.com/bliki/PolyglotPersistence.html

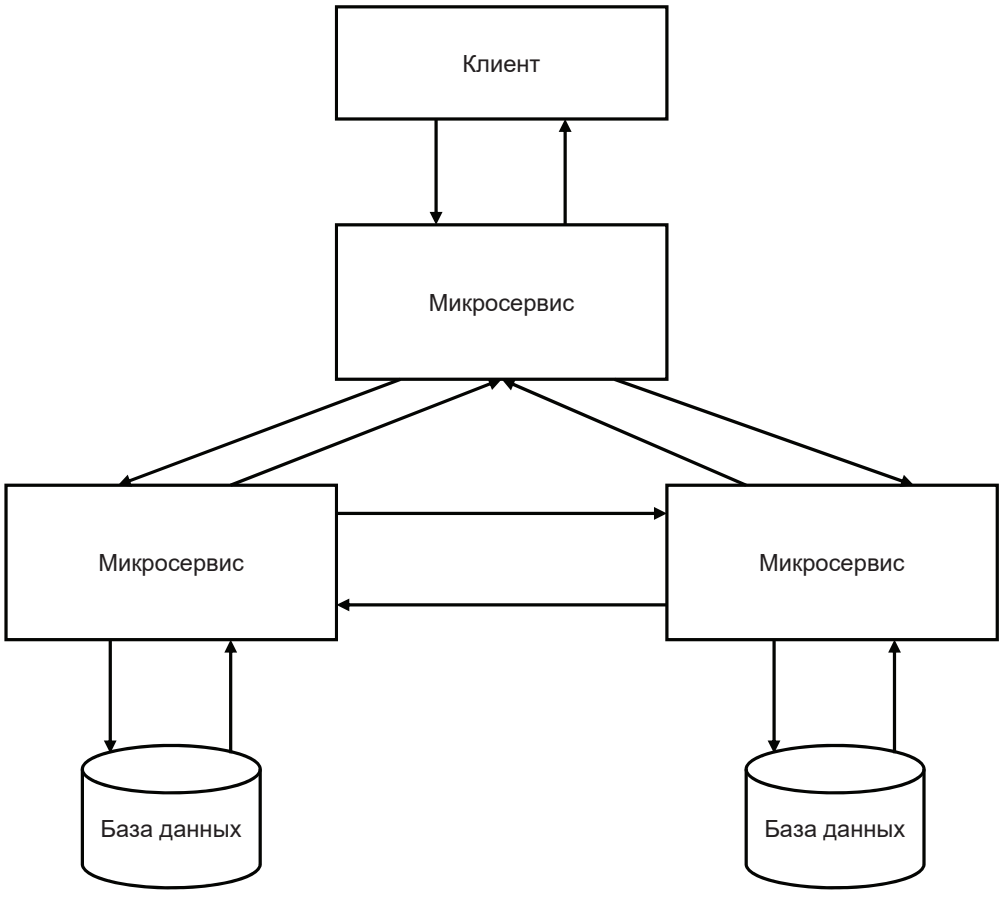


*Рис. 4 / Fig. 4.* **Микросервисная архитектура ПО / Software Microservice Architecture**

*Источник / Source:* составлено авторами / Complied by the authors.

## СРАВНИТЕЛЬНЫЙ АНАЛИЗ РЕШЕНИЙ И ПОДХОДОВ К РАЗРАБОТКЕ СИСТЕМ ХРАНЕНИЯ И ОБРАБОТКИ ДАННЫХ

В ходе исследования проведен сравнительный анализ подходов и технологий хранения данных, таких как традиционные реляционные СУДБ, NoSQL- и NewSQL-системы. Выбраны ключевые критерии сравнения при выборе решения для разработки распределенной и высоконагруженной систем хранения данных:

- теоретическая основа и модель данных;
- архитектурный паттерн масштабирования;

*Таблица 1 / Table 1*

**Сравнительный анализ подходов и решений к хранению данных / Comparative Analysis of Approaches and Solutions to Data Storage**

| Критерий / Criteria | Реляционные СУБД / Relational databases | NoSQL-системы / NoSQL systems | NewSQL-системы / NewSQL systems |
|---|---|---|---|
| Теоретическая основа и модель данных | Реляционная алгебра, теория множеств. Жесткая, предопределенная схема. ACID — неотъемлемое свойство | Отказ от реляционной модели в пользу специализированных структур: документной, «ключ-значение», колоночной, графовой. Гибкая или отсутствующая схема. Базируются на принципе BASE | Сохранение реляционной модели и SQL-интерфейса как фундамента. Переосмысление архитектурного уровня для распределенного выполнения с сохранением ACID |
| Архитектурный паттерн масштабирования | Преимущественно используется вертикальное масштабирование. Кластерные конфигурации ориентированы на отказоустойчивость, а не на линейный прирост производительности при записи | Горизонтальное масштабирование как основополагающий принцип. Шардирование и партиционирование данных и репликация реализованы на архитектурном уровне | Горизонтальное масштабирование как базовая возможность. Автоматическое шардирование и ребалансировка данных в рамках единого логического кластера |
| Позиционирование относительно теоремы CAP | Приоритет CA (Consistency, Availability). В классических одноузловых и кластерных конфигурациях с синхронной репликацией устойчивость к разделению сети не достигается | Явный выбор AP (Availability, Partition Tolerance) или CP (Consistency, Partition Tolerance) в зависимости от предназначения системы | Стратегическая цель — CA-системы с минимальным компромиссом по P благодаря инновационным распределенным протоколам и управляемой задержки репликации |
| Модель транзакций и консистентность | Полноценная поддержка ACID-транзакций, включающих множественные операции над разнородными данными | Транзакционность часто ограничена операциями в рамках одного документа, ключа или партиции. Глобальные распределенные транзакции либо отсутствуют | Ключевое архитектурное достижение: поддержка распределенных ACID-транзакций на уровне всего кластера |

*Источник / Source:* составлено авторами / Complied by the authors.

• позиционирование относительно теоремы CAP;

• модель транзакций и консистентность.

Результаты сравнительного анализа представлены в *табл. 1*.

Выбор варианта хранения данных — не просто оптимизационная задача, а стратегическое решение, определяющее онтологические свойства будущей системы. При создании хранилища для банковской цифровой экосистемы приоритет отдается не единой парадигме, а стратифицированной архитектуре, где каждый подход используется там, где его свойства и ограничения дают наилучший результат [12]. Для высоконагруженных распределенных систем в исследовании сопоставлены монолитная и микросервисная архитектуры. Итоги сравнения представлены в *табл. 2*.

Результаты сравнительного анализа показывают: и микросервисная, и монолитная архитектуры ПО обладают как преимуществами, так и недостатками — в зависимости от критериев оценки. При этом выбор архитектуры определяется функцио-

*Таблица 2 / Table 2*

**Сравнительный анализ монолитной и микросервисной архитектур ПО / Comparative Analysis of Monolithic and Microservice Software Architectures**

| Критерий / Criterion | Архитектура / Architecture | |
|---|---|---|
| | монолитная / Monolithic | микросервисная / Microservice |
| Структурная организация | Единая, тесносвязанная кодовая база. Все функциональные модули компилируются и развертываются как единое целое | Набор слабосвязанных, независимо развертываемых сервисов. Каждый сервис инкапсулирует специфическую бизнес-возможность и обладает автономным циклом данных |
| Взаимосвязанность компонентов | Жесткая взаимосвязанность между модулями через общие структуры данных, вызовы функций и единую базу данных, что приводит к распространению изменений | Взаимодействие через четко определенные, стабильные API (REST/ gRPC, асинхронные взаимодействия). Распространение изменений ограничено контрактом API |
| Масштабируемость | Вертикальное масштабирование: развертывание дополнительных идентичных инстансов всего монолита. Горизонтальное масштабирование неэффективно, так как копируется вся функциональность, даже если нагрузка пришлась на один модуль | Гранулярное горизонтальное масштабирование: возможность независимо масштабировать только те сервисы, которые испытывают высокую нагрузку. Позволяет оптимизировать использование ресурсов и стоимостные показатели |
| Гибкость технологического стека | Единый технологический стек для всего приложения. Технологическая эволюция затруднена, требует глобального обновления | Принцип полиглотного стека: каждый сервис реализуется на наиболее подходящем языке и используется специализированное хранилище данных. Ускоряет инновации и адаптацию новых технологий |
| Отказоустойчивость | Единая точка отказа: сбой в любом модуле приводит к падению всего приложения | Распределение и изоляция сбоев повышает общую доступность системы |

*Источник / Source:* составлено авторами / Complied by the authors.

нальными и нефункциональными требованиями к системе хранения данных. В частности, для цифровой банковской экосистемы при разработке системы хранения клиентских данных эффективнее использовать микросервисную архитектуру: она обеспечивает необходимое масштабирование и отказоустойчивость.

## ВЫВОДЫ

Проведенное исследование позволяет сформулировать комплексный подход к проектированию и разработку распределенной высоконагруженной системы хранения клиентских данных в условиях современной цифровой банковской экосистемы. Анализ эволюции требований к подобным системам — экспоненциальный рост объемов и разнообразия данных, необходимость обработки информации в режиме реального времени, ужесточение регуляторных норм и повышенные ожидания клиентов к доступности и персонализации сервисов — выявил несостоятельность традиционных подходов к построению систем хранения данных.

Сравнительный анализ технологических парадигм хранения данных — реляционных СУБД, NoSQL- и NewSQL-систем — подтвердил отсутствие универсального решения. Каждый подход — уникальный компромисс в рамках теоремы CAP (Брюера), что детерминирует его применимость для конкретных типов данных и сценариев использования внутри экосистемы. Транзакционные данные, требующие строгой ACID-семантики, могут эффективно обслуживаться современными NewSQL-системами (например, CockroachDB, YDB), в то время как для задач кэширования и работы с временными рядами оптимальны хранилища «ключ-значение» (Redis, Apache Ignite) и широколонные СУБД (Apache Cassandra).

Архитектурный анализ продемонстрировал, что микросервисная парадигма, в сравнении с монолитной, является более подходящим фундаментом для построения масштабируемой и отказоустойчивой системы. Преимущества данного подхода — слабая связанность компонентов, гранулярное горизонтальное масштабирование, технологическая гибкость и изоляция сбоев — прямо отвечают на вызовы динамичной банковской среды. Ключевым архитектурным принципом становится стратегия Polyglot Persistence, предполагающая синергетическое использование разнородных систем хранения, наиболее подходящих для решения конкретных бизнес-задач.

Таким образом, оптимальная архитектура единого хранилища клиентских данных для цифровой банковской экосистемы представляет собой стратифицированную, сервисно-ориентированную систему. Ее ядро должно сочетать гарантии целостности критически важных данных (через NewSQL или реляционные кластеры) с высокой доступностью и производительностью для аналитических и операционных нагрузок (через специализированные NoSQL-решения и in-memory-хранилища). Реализация таких архитектурных паттернов, как CQRS и Event Sourcing, позволит дополнительно повысить масштабируемость и поддержку разнообразных материализованных представлений данных.

## СПИСОК ИСТОЧНИКОВ


1. Клеппман М. Высоконагруженные приложения программирование, масштабирование, поддержка. Пер. с англ. И. Пальти, А. Тумаркин. СПб.: Питер; 2021. URL: https://rusneb.ru/catalog/000200_000018_RU_NLR_BIBL_A_012416514/
2. Мартин Р. Чистая архитектура. Искусство разработки программного обеспечения. СПб.: Питер; 2021. URL: https://search.rsl.ru/ru/record/01010779241
3. Wiling B. Scientific Study of CAP Theorem and Understanding its Different Implementation Methods. *Mathematical Statistician and Engineering Applications.* 2022;(1);133-137. DOI: 10.17762/msea.v71i1.55
4. Таненбаум Э., ван Стейен М. Распределенные системы. Принципы и парадигмы. СПб.: Питер; 2003. URL: https://rusneb.ru/catalog/000199_000009_002157753/?ysclid=mmw6lsm0tr819862687
5. Садаладж П., Фаулер М. NoSQL: новая методология разработки нереляционных баз данных. М. Вильямс; 2013. 192 с. URL: https://search.rsl.ru/ru/record/01006568860
6. Lakshman, A., Malik, P. Cassandra: A Decentralized Structured Storage System. *ACM SIGOPS Operating Systems Review*. 2021;(2):35-40. DOI: 10.1145/1773912.1773922
7. Corbett J. C. Spanner: Google's Globally-Distributed Database. *OSDI*. 2023;(12):251-264. URL: https://sayedalesawy.hashnode.dev/spanner-googles-globally-distributed-database
8. Игнатенко И.Д., Астахов В.В., Акинина Ю.С. Сравнительный анализ подходов к реализации распределенных транзакций. В сб.: Будущее науки — 2025. 2025;(1):209-216. URL: https://www.elibrary.ru/eoxwmm
9. Lewis P., Perez E., Piktus A. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. *Advances in Neural Information Processing Systems*. 2021;(33):9459-9474. DOI: 10.48550/arXiv.2005.11401

10. Ньюмен С. От монолита к микросервисам. СПб.: БХВ-Петербург; 2021. 272 с. URL: https://rusneb.ru/catalog/000200_000018_RU_NLR_BIBL_A_012548166/?ysclid=mmw6qujxqc209793652
11. Куликова О.М., Суворова С.Д. Облачные технологии: основа построения корпоративной архитектуры. *Инновационная экономика: перспективы развития и совершенствования*. 2021;(4):65-70. URL: https://www.elibrary.ru/xfjvta
12. Косарев В.Е., Городецкая О.Ю., Габарева Я.Л., Рычаго М.Е. Интеграция распределенных облачных вычислений и модульных информационных систем для повышения эффективности управления промышленным производством. *Кузнечно-штамповочное производство. Обработка материалов давлением*. 2025;(11):107-114. URL: https://www.elibrary.ru/qgcazs

## REFERENCES

1. Kleppman M. Highly loaded applications programming, scaling, support. Transl. from Eng. by I. Palti, A. Tumarkin. Saint Petersburg: St. Petersburg; 2021. URL: https://rusneb.ru/catalog/000200_000018_RU_NLR_BIBL_A_012416514/ (In Russ.).
2. Martin R. Pure architecture. The art of software development. Saint Petersburg: St. Petersburg; 2021. URL: https://search.rsl.ru/ru/record/01010779241 (In Russ.).
3. Wiling B. Scientific Study of CAP Theorem and Understanding its Different Implementation Methods. *Mathematical Statistician and Engineering Applications*. 2022;(1);133-137. DOI: 10.17762/msea.v71i1.55
4. Tanenbaum E., van Steyen M. Distributed systems. Principles and paradigms. Saint Petersburg: Peter; 2003. URL: https://rusneb.ru/catalog/000199_000009_002157753 /?ysclid=mmw6lsm0tr819862687 (In Russ.).
5. Sadalaj P., Fowler M. NoSQL: a new methodology for developing non-relational databases. Moscow: Williams; 2013. 192 p. URL: https://search.rsl.ru/ru/record/01006568860 (In Russ.).
6. Lakshman A., Malik, P. Cassandra: A Decentralized Structured Storage System. *ACM SIGOPS Operating Systems Review*. 2021;(2):35-40. DOI: 10.1145/1773912.1773922
7. Corbett J.C. Spanner: Google's Globally-Distributed Database. OSDI. 2023;(12):251-264. URL: https://sayedalesawy.hashnode.dev/spanner-googles-globally-distributed-database (In Russ.).
8. Ignatenko I.D., Astakhov V.V., Akinina Yu.S. Comparative analysis of approaches to the implementation of distributed transactions. In the collection: The future of science — 2025. 2025;(1):209-216. URL: https://www.elibrary.ru/eoxwmm (In Russ.).
9. Lewis P., Perez E., Piktus A. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. *Advances in Neural Information Processing Systems*. 2021;(33):9459-9474. URL: https://doi.org/10.48550/arXiv.2005.11401
10. Newman S. From monolith to microservices. Saint Petersburg: BHV-Petersburg; 2021. 272 p. URL: https://rusneb.ru/catalog/000200_000018_RU_NLR_BIBL_A_012548166 /?ysclid=mmw6qujxqc209793652 (In Russ.).
11. Kulikova O.M., Suvorova S.D. Cloud technologies: the basis for building a corporate architecture. *Innovative economy: prospects for development and improvement*. 2021;(4):65-70. URL: https://www.elibrary.ru/xfjvta (In Russ.).
12. Kosarev V.E., Gorodetskaya O.Yu., Gobareva Ya.L., Rychago M.E. Integration of distributed cloud computing and modular information systems to improve the efficiency of industrial production management. *Forging and stamping production. Pressure treatment of materials*. 2025;(11):107-114. URL: https://www.elibrary.ru/qgcazs (In Russ.).

## ИНФОРМАЦИЯ ОБ АВТОРАХ / ABOUT THE AUTHORS

***Никита Андреевич Бурыкин*** — студент магистратуры факультета информационных технологий и анализа больших данных, Финансовый университет при Правительстве Российской Федерации, Москва, Российская Федерация

***Nikita A. Burykin —*** Master's degree student, Department of Information Technology and Big Data Analysis, Financial University under the Government of the Russian Federation, Moscow, Russian Federation

https://orcid.org/0009-0009-6631-230X

*Автор для корреспонденции / Corresponding author:*

244461@edu.fa.ru

***Наталья Владимировна Гринева*** — кандидат экономических наук, доцент, доцент кафедры информационных технологий факультета информационных технологий и анализа больших данных, Финансовый университет при Правительстве Российской Федерации, Москва, Российская Федерация
***Natalia V. Grineva*** — Cand. Sci. (Econ.), Assoc. Prof., Assoc. Prof. of the Department of Information Technology, Faculty of Information Technology and Big Data Analysis, Financial University under the Government of the Russian Federation, Moscow, Russian Federation
https://orcid.org/0000-0001-7647-5967
ngrineva@fa.ru

***Павел Евгеньевич Голосов*** — кандидат технических наук, директор Института общественных наук РАНХиГС, Москва, Российская Федерация
***Pavel E. Golosov —*** Cand. Sci. (Tech.), Director of the Institute of Social Sciences RANEPA, Moscow, Russian Federation
https://orcid.org/0000-0003-4313-0887
golosov-pe@ranepa.ru

***Заявленный вклад авторов:***
**Н.А. Бурыкин —** сравнительный анализ архитектурных подходов и технологических решений для разработки системы хранения клиентских данных, написание разделов «Введение», «Выводы».
**Н.В. Гринева —** разработка общей концепции статьи, исследование существующих подходов хранения и обработки данных.
**П.Е. Голосов** — исследование существующих технологических решений хранения данных, список литературы.

***Authors' contributions:***
**N.A. Burykin** — comparative analysis of architectural approaches and technological solutions for the development of a customer data storage system, writing the sections "Introduction", "Conclusions".
**N.V. Grineva** — development of the general concept of the article, research of existing approaches to data storage and processing.
**P.E. Golosov** — research of existing technological solutions for data storage, references.

*Конфликт интересов: авторы заявляют об отсутствии конфликта интересов.*
*Conflicts of Interest Statement: The authors have no conflicts of interest to declare.*

*Статья поступила 19.01.2026; после рецензирования 02.03.2026; принята к публикации 13.03.2026.*
*Авторы прочитали и одобрили окончательный вариант рукописи.*
*The article was submitted on 19.01.2026; revised on 02.03.2026 and accepted for publication on 13.03.2026.*
*The authors read and approved the final version of the manuscript.*