18. *Tolpaev V.A., Akhmedov K.S.* Dolgosrochnoe prognozirovanie raboty skvazhin gazokondensatnykh mestorozhdeniy metodami kubicheskoy splayn-approksimatsii [Long-term prediction of wells operation in gas condensate fields using cubic spline approximation methods], Neftepromyslovoe delo [Oilfield Engineering], 2024, No. 9, pp. 10-22. Available at: https://rucont.ru/efd/904508.
19. *Yusufov A. et al.* Application of Computer-Aided Design (CAD) Systems when Solving Engineering Survey Tasks, *Universum: tekhnicheskie nauki* [Universum: Techincal Sceinces], 2023, 3 (108). Available at: https://7universum.com/ru/tech/archive/item/15088.
20. *Ivanchenko A.N., Ivanchenko K.N.* Metod rascheta veroyatnosti svyaznosti dvukhpolyusnoy seti proizvol'noy struktury [Method for calculation of the probability of connectivity of a two-pol network of an arbitrary structure], *Izvestiya vuzov. Severo-Kavkazskiy region. Tekhnicheskie nauki* [University News. North-Caucasus Region. Technical Sciences], 2023, No. 2, pp. 25-33. Available at: https://doi.org/10.17213/1560-3644-2023-2-25-33.

Дорофеев Алексей Анатольевич – Донской государственный технический университет; e-mail: AlexeiDorofeyev@yandex.ru; г. Ростов-на-Дону, Россия; к.т.н.; доцент кафедры высшей математики; ORCID: https://orcid.org/0009-0004-2496-6691; AuthorID: https://elibrary.ru/author_items.asp?authorid=1305339.

Dorofeev Alexei Anatol'evich – Don State Technical University; e-mail: AlexeiDorofeyev@yandex.ru; Rostov-on-Don, Russia; cand. of eng. sc.; associate professor of the Department of Higher Mathematics; ORCID: https://orcid.org/0009-0004-2496-6691; AuthorID: https://elibrary.ru/author_items.asp?authorid=1305339.

УДК 004.652 DOI 10.18522/2311-3103-2025-6-105-121

А.А. Коблов, О.М. Ромакина, А.С. Клемешева, А.З. Арсеньева

ИССЛЕДОВАНИЕ ПРИМЕНИМОСТИ МУЛЬТИМОДЕЛЬНЫХ ХРАНИЛИЩ ДАННЫХ В ИГРОВОЙ ИНДУСТРИИ

Проводится исследование целесообразности и эффективности применения мультимодельных баз данных для хранения и обработки данных в игровой индустрии. Современные игровые проекты характеризуются высокой сложностью и разнородностью данных: от строго структурированной информации об игроках, предметах и квестах до слабоструктурированных и сильносвязанных данных, таких как системы рецептов, диалоговые деревья, отношения между кланами и внутриигровые энциклопедии. Существующие подходы, основанные на реляционных или одномодельных NoSQL-хранилищах, часто не обеспечивают необходимой гибкости, производительности и удобства разработки для таких комплексных сценариев. Целью исследования является проектирование и сравнительный анализ производительности мультимодельного решения в контексте типовых игровых механик. Авторами разработана структура мультимодельного хранилища на базе СУБД ArangoDB, которая интегрирует документную, графовую и ключ-значение модели данных. Архитектура решения охватывает ключевые компоненты RPG-игр: управление игроками и инвентарём, систему квестов, диалогов, рецептов крафта, таблиц добычи, клановых взаимоотношений, а также полнотекстовый поиск по внутриигровой энциклопедии с использованием ArangoSearch. Экспериментальная часть включает подробное сравнение производительности разработанного мультимодельного хранилища с реляционной СУБД PostgreSQL и документной MongoDB на реалистичных наборах данных и запросах. Результаты демонстрируют значительное преимущество мультимодельного подхода при выполнении операций, требующих обхода сложных связей: например, поиск враждебных игроков через граф клановых отношений в ArangoDB выполняется в среднем в 11 раз быстрее, чем аналогичный JOIN-запрос в PostgreSQL. В то же время, для сценариев с частыми модификациями линейно организованных данных (например, обновление статуса квестов) мультимодельное хранилище показывает несколько более низкую производительность по сравнению с реляционной моделью, что однако является допустимым в контексте общей архитектуры игрового проекта. Исследование подтверждает, что мультимодельные СУБД, в частности ArangoDB, представляют собой перспективное решение для игровой индустрии, позволяя в рамках единой платформы эффективно комбинировать различные модели данных, упрощать разработку и достигать высокой производительности на сложносвязанных данных, что является критически важным для современных многопользовательских игр.

Мультимодельные хранилища данных; NoSQL модели данных; схемы баз данных; документная модель; графовая модель; реляционная модель; хранилища данных; ArangoDB; разработка игр.

A.A. Koblov, O.M. Romakina, A.S. Klemesheva, A.Z. Arseneva

INVESTIGATION OF APPLICABILITY OF MULTIMODEL DATA WAREHOUSES IN GAMING INDUSTRY

This paper examines the feasibility and effectiveness of using multi-model databases for storing and processing data in the gaming industry. Modern gaming projects are characterized by highly complex and heterogeneous data: from strictly structured information about players, items, and quests to semi-structured and tightly coupled data, such as recipe systems, dialog trees, clan relationships, and in-game encyclopedias. Existing approaches based on relational or single-model NoSQL storage systems often fail to provide the necessary flexibility, performance, and development ease for such complex scenarios. The aim of this study is to design and comparatively analyze the performance of a multi-model solution for typical gaming mechanics. The authors developed a multi-model storage structure based on the ArangoDB DBMS that integrates document, graph, and key-value data models. The solution architecture encompasses key RPG game components: player and inventory management, quest systems, dialogue, crafting recipes, loot tables, clan relationships, and full-text search of the in-game encyclopedia using ArangoSearch. The experimental section includes a detailed performance comparison of the developed multi-model storage system with the PostgreSQL relational DBMS and the MongoDB document DBMS on realistic datasets and queries. The results demonstrate a significant advantage of the multi-model approach when performing operations that require traversing complex relationships: for example, searching for hostile players through a clan relationship graph in ArangoDB is, on average, 11 times faster than a similar JOIN query in PostgreSQL. However, for scenarios with frequent modifications to linearly organized data (e.g., updating quest status), the multi-model storage system exhibits slightly lower performance compared to the relational model, which, however, is acceptable within the context of the overall game project architecture. The study confirms that multi-model DBMSs, particularly ArangoDB, represent a promising solution for the gaming industry, enabling efficient combination of different data models within a single platform, simplifying development, and achieving high performance on complex data, which is critical for modern multiplayer games.

Multi-model data warehouses; NoSQL data models; database schemas; document model; graph model; relational model; data warehouses; ArangoDB; game development.

Введение. Игровая индустрия за последние десятилетия стала значимой частью мировой экономики с многомиллиардным бюджетом. Текущие тенденции роста экономических показателей видеоигровой индустрии демонстрируют перспективность и востребованность этой сферы на мировом рынке программного обеспечения [1].

Одной из задач, с которыми сталкиваются компании при разработке игровых проектов, является проектирование систем хранения игровых данных. Поскольку собственные хранилища, разработанные под конкретные игровые проекты, требуют трудоемкой разработки и дальнейшей поддержки, в проектах с постоянно растущим объемом данных, как правило, предпочтение отдается уже готовым решениям. Ввиду особенностей платформ для обеспечения большей части рынка требуется классическая клиент-серверная архитектура, а значит, необходимо сопровождение огромных баз клиентских данных даже без учета самой игровой архитектуры [2]. Помимо этого, растёт также и разнообразие самих данных, и использование только одного типа СУБД создаёт ограничения и сложности для реализации и поддержки определенных функций.

Данные можно разделить на три вида: структурированные, полуструктурированные и неструктурированные. Для хранения и обработки данных каждого вида требуются отдельные, наиболее эффективные для конкретных задач способы.

В настоящее время существуют два способа хранения вариативных данных: «многовариантное хранение» (polyglot persistence) и мультимодельные системы. Многовариантное хранение представляет собой идею одновременного использования нескольких СУБД для решения задач, где данные будут храниться в подходящем для них месте – отдельных СУБД.

Одним из способов реализации такой идеи является разделение доменного уровня приложения на несколько подуровней таким образом, чтобы каждый подуровень обращался к собственной базе данных. При этом на уровне верхнего домена возникает задача о соблюдении согласованности транзакций [3]. У такого способа организации есть боль-

шое количество существенных недостатков: во-первых, сложность реализации единого способа обращения к хранящимся данным и их дальнейшей обработки, поскольку за разные части данных отвечают разные системы. Ситуация усложняется, если данные сильно связаны. Во-вторых, невозможно в полной мере обеспечить поддержку транзакций и все требования ACID. В-третьих, при увеличении объёма используемых СУБД растёт нагрузка на поддержание стабильной работы БД, т.е. усложняется её масштабирование, затраты на обеспечение оптимальной скорости доступа к данным и т.д. Сложность подобной архитектуры сказывается на производительности, постоянстве и отказоустойчивости системы.

Вторым подходом к хранению разнородных данных являются мультимодельные СУБД, их основное преимущество заключается в возможности поддержки множества различных представлений данных в пределах одной системы. Авторы статьи [4] отмечают, что применению мультимодельных баз данных сопутствует некоторая сложность их поддержки и развития, однако, как утверждается в статье [5], конвергенция разных типов моделей данных необходима из-за постоянно растущего объема данных различных видов, которые необходимо обрабатывать в рамках одной системы. Единая СУБД для многомодельных данных удобна тем, что предоставляет не только унифицированный интерфейс запросов, но и единую платформу для уменьшения сложности интеграции и устранения проблем с миграцией [6].

На основе анализа источников [7–10] были выделены несколько основных типов моделей данных, на основе которых создаются мультимодельные СУБД.

- Иерархические. Данные организованы в виде древовидной структуры, узлы которой связаны между собой отношениями «родитель-ребёнок»: у каждого родителя может быть множество узлов-детей, но у каждого узла-ребенка может быть только один родитель.
- Сетевые. Здесь связи между узлами представляют собой произвольный граф, что позволяет переходить от одного узла к другому не только вертикально, но и в пределах одного уровня.
- Реляционные. В этой модели данные упорядочены по строкам и столбцам в таблицах, которые могут быть связаны с другими с помощью первичных и внешних ключей.
- Графовые. В таких моделях данные организованы в виде связного графа, где узел хранит саму сущность, а ребра – связи между ними.
- Документные. Здесь данные хранятся в виде документов – сущностей, каждая из которых может иметь уникальную структуру и содержать разное количество полей.
- Key-value. Данные представлены в виде пары ключ-значение, где каждый ключ является уникальным идентификатором, связанным с конкретной информацией.
- Колоночные. Схожи по принципу с key-value СУБД, однако каждое «значение» может содержать несколько столбцов и соответствующие им значения, что позволяет удобно хранить связанную информацию. При этом у каждого «значения» может быть свой собственный набор столбцов.

Согласно данным опроса разработчиков Stack Overflow [11], реляционные СУБД являются наиболее популярной моделью организации данных. Иерархические и сетевые модели представления данных со временем были признаны устаревшими и почти не используются [12]. Отдельно стоит отметить обобщающий термин «базы данных NoSQL» или нереляционные СУБД. Такие базы данных не обладают характеристиками, свойственными реляционным СУБД, а также в них не используется стандартный подход с созданием таблиц [13]. У них может не быть жесткой структуры. К этой категории относятся графовые, документные, key-value, колоночные и ряд прочих, менее популярных СУБД.

Мультимодельные СУБД могут поддерживать несколько моделей в различных сочетаниях, например, реляционную и key-value или колоночную, документную и графовую. Некоторые из таких СУБД являются мультимодельными изначально, некоторые имеют основную модель, но обеспечивают поддержку дополнительных [6, 14]. Наиболее подходящие для конкретных задач комбинации моделей в мультимодельных СУБД определяются преимуществами и недостатками каждой модели данных.

Реляционная модель имеет проблемы с производительностью и требует оптимизации при высокой нагрузке сложными операциями, состоящими из нескольких подзапросов. Кроме того, изменение таблиц и добавление новых столбцов проблематично, поскольку требует изменения данных в приложениях, которые обращаются к БД и миграции существующих данных [15].

Графовая модель незаменима в хранении и обработке высокосвязных данных, она демонстрирует высокую производительность при локальном чтении во время перемещения по графу [16]. Из недостатков графовой модели можно выделить отсутствие стандартизированного языка запросов.

Документные модели представления данных обладают очень высокой производительностью и возможностью масштабирования, поскольку формат хранимых данных спроектирован таким образом, чтобы данные можно было найти и извлечь за минимально возможное время. Недостатком этих моделей является ограниченная поддержка транзакций. Документные модели также неэффективны, если данные приходится часто обновлять, поскольку для этого потребуется заново сохранять всю структуру [17].

Key-value модель способна хранить любую информацию, будь то JSON, бинарные или любые другие данные. Key-value модель имеет высокую производительность при поиске конкретных ключей, однако эта производительность сильно падает, если необходим поиск по нескольким ключам [18]. Также модель этого типа имеет крайне ограниченный набор операций для работы с хранимыми данными.

Некоторые игровые проекты успешно работают с NoSQL-хранилищами, например, компания Epic Games для хранения данных в игре Fortnite использует MongoDB [19]. Другим вариантом хранения информации в игровой индустрии является использование частично мультимодельного подхода, за счёт возможностей одномодельных баз данных. Разработчики Аллоды и Skyforge отмечали, что реляционная модель имеет ряд значимых недостатков, которые можно преодолеть только используя связку реляционных возможностей с нереляционными. Разработчиками было решено создать несколько столбцов, работающих в качестве хранилищ для данных в документном стиле, поскольку эти данные сильно связаны друг с другом и редко меняются по отдельности [20].

Несмотря на потенциальные преимущества мультимодельных СУБД над одномодельными, не было найдено публичных примеров их использования в игровых проектах. Компании в своих продуктах, как правило, используют одномодельные СУБД. Большая часть проектов работает с реляционной моделью [21], например, такие проекты как Eve Online [22] или World of Warcraft [23]. С точки зрения Abdullah Alqwbani и др. [24] использование реляционной модели будет наиболее подходящим решением для хранения такой информации, как, например, данные аккаунтов, поскольку таких данных не так много, все они должны храниться наиболее безопасным образом и не требуют сложной аналитики. Однако, как подчёркивает автор статьи [25], реляционные СУБД в их стандартном виде неудобны при хранении игровых объектов, поскольку для выстраивания связей необходимо будет создавать транзакцию на несколько таблиц.

Таким образом, можно заметить, что в игровой отрасли наблюдается потребность в одновременном использовании нескольких моделей данных в рамках одного проекта. Целью настоящей работы является исследование применимости мультимодельных СУБД в игровой индустрии и оценка эффективности такого решения.

Материалы и методы. Игровые проекты делятся на однопользовательские и многопользовательские. Согласно отчёту, собранного компанией Unity в 2024 году [26], 77% всех игроков предпочитают многопользовательские игры. Согласно отчёту компании Newzoo [27], топ жанров по доходу возглавляют шутеры, адвенчуры и RPG. За последние 20 лет в рамках этих жанров было создано большое количество игр, что позволяет на их основе выделить наиболее часто используемые компоненты и механики, данные которых требуется хранить и отправлять игрокам по необходимости.

Такими компонентами в большинстве игр являются:

♦ Неигровые персонажи (NPC). Их данные включают характеристики здоровья, урона, типа (дружественный, нейтральный или враждебный), а также таблицы добычи, определяющие, какую награду игрок получит за победу над ними. Таблица состоит из идентификаторов предметов, которые могут выпасть, и частоты их выпадения.

♦ Система квестов. Квесты могут содержать как одну, так и несколько задач, например, «выковать меч у кузнеца» и «купить яблоко на рынке». Квесты могут быть повторяющимися или неповторяющимися, как правило, имеют название, описание, а их выполнение может давать игроку награду и/или опыт. Квесты могут быть объединены в цепочки, таким образом по завершении одного квеста игроку выдаётся новый.

♦ Система диалогов. Диалоги состоят из инструкций – запрограммированных последовательностей из действий, реплик NPC, внутриигрового видео и т.д. В случае учета и анализа ответных фраз игрока, система диалогов будет иметь нелинейный характер.

♦ Предметы. Содержат такую информацию как название, описание, характеристики и прочие параметры, набор которых одинаков для всех экземпляров данного предмета.

♦ Инвентарь. Список экземпляров предметов, принадлежащих конкретному игроку, существу или объекту в мире. В экземплярах, помимо ссылки на предмет, могут быть ещё и уникальные данные – пользовательское название или редкость.

♦ Система рецептов. Часто в игровые проекты добавляют возможность создавать предметы по рецепту или улучшать их с помощью некоторых компонентов, превращая в другие.

♦ Внутриигровая энциклопедия. Как правило, разделена на категории и имеет возможности поиска по ключевым словам.

Помимо основных сюжетных компонентов, игровые проекты зачастую включают в себя системы, повторяющиеся в играх вне зависимости от жанра, поскольку носят либо утилитарный характер, либо необходимы для внутриигрового социального взаимодействия. К ним относятся системы сбора статистических данных, доски лидерства, списки друзей, кланы и другие внутриигровые сообщества. В играх с подобными компонентами также может быть реализована система подбора игроков схожего уровня для совместных испытаний или сражений друг против друга.

Для реализации мультимодельной БД и оценке эффективности данного подхода применительно к наиболее популярным игровым механикам необходимо выбрать СУБД для реализации баз данных в различных моделях и последующего анализа времени выполнения наиболее популярных запросов.

В качестве реляционной СУБД для проведения экспериментов выбрана PostgreSQL, как одна из наиболее популярных среди реляционных СУБД [28]. PostgreSQL обладает высокой мощностью и рассчитана на работу с огромными массивами информации, а также имеет библиотеки для подключения из различных языков программирования.

Для тестирования документной модели была выбрана MongoDB, поскольку она также имеет набор библиотек для подключения и является кроссплатформенной БД.

Эксперименты на графовых СУБД и сравнение быстродействия выполнения запросов к графовой БД и мультимодельной БД не проводились, что обусловлено следующими обстоятельствами. Во-первых, в силу существенного различия архитектур графовых СУБД (Property Graph и RDF-граф), отсутствия унифицированного языка запросов и единой концепции поддержания целостности, необходимо проведение отдельного исследования принципиальной возможности реализации различных типов запросов в существующих графовых СУБД, а также скорости выполнения реализованных запросов. Во-вторых, с ростом объема графа, а также увеличением количества взаимосвязей в нем, скорость выполнения различных запросов на графе нелинейно снижается, причем снижается различным образом в графах с разной архитектурой, что также требует дополнительных исследований. Очевидно, перечисленные исследования выходят за рамки данной работы.

Наиболее популярными СУБД, которые изначально проектировались как мультимодельные, являются CosmosDB, OrientDB и ArangoDB.

CosmosDB - это облачная мультимодельная СУБД с автоматической масштабируемостью. Поддержка различных моделей данных реализуется здесь с помощью набора API для наиболее популярных СУБД. С помощью этих API можно представлять данные требуемым способом, однако CosmosDB является скорее единой интегрированной оболочкой различных API, а не полноценной мультимодельной СУБД [29].

OrientDB и ArangoDB по своей структуре сильно отличаются от CosmosDB. Они позволяют хранить данные в документной и графовой модели данных, а также поддерживают принципы ACID и транзакции. OrientDB отличается возможностью поддержки классов с свойствами и наследованием.

ArangoDB для запросов использует свой собственный язык – AQL (ArangoDB Query Language). За счёт того, что этот язык создан специально для мультимодельных задач, он является достаточно гибким при работе данными, принадлежащими разным моделям. Отдельно стоит отметить анализ производительности, проведённый разработчиками ArangoDB [30]. Тестирование на чтение, запись, агрегацию, занимаемую память, а также на специфичные для поиска по графам запросы показали значительное преимущество по отношению к OrientDB. Это также подтверждается в статье [31]. Для анализа применимости мультимодельных СУБД в игровой отрасли было принято решение использовать ArangoDB.

Для демонстрации возможностей мультимодельной СУБД в рамках игровой отрасли, был разработан пример базы данных для хранения информации об основных игровых компонентах и механиках типовой RPG, а также подготовлен набор наиболее часто используемых запросов. К сожалению, возможности создания схемы данных по аналогии с привычной реляционной схемой данных в ArangoDB не предусмотрено по причине отсутствия нотации для моделирования мультимодельных БД.

Поскольку стандартное общепринятое наименование объекта, именуемого «сущностью» в реляционных БД или «коллекцией» в документных БД, в мультимодельных БД также отсутствует, то далее, во избежание двусмысленностей и неоднозначного понимания, будем именовать такой объект «коллекцией».

Основными коллекциями разработанного хранилища являются clans (кланы), instructions (инструкции), items (предметы), loot_tables (таблицы добычи), players (игроки), players_clans (принадлежность игроков к кланам), quests (квесты), recipes (рецепты), wiki (игровая энциклопедия). Их назначение и структура далее будут рассмотрены подробно.

Проектирование основных компонентов хранилища.

Основные коллекции в разработанном хранилище – items и players.

Коллекция items необходима для хранения описаний предметов. Описание предмета состоит из уникального идентификатора, его названия, а также любой метаинформации, которая будет полезна для создания экземпляров этого предмета, например, качество предмета или прочность у инструментов и оружия. Экземпляр предмета – это конкретный объект, который имеет все свойства, указанные в определении предмета, но представляет собой его конкретное воплощение.

Коллекция players хранит информацию об игроке, являющемся центральным элементом таких игровых направлений, как шутеры, адвенчуры и RPG. Ключом является никнейм, поскольку он уникален, а экземпляр игрока (или документ) в этой коллекции может включать в себя инвентарь (список предметов, который носит игрок и использует по мере надобности), параметры игрока (уровень, количество опыта, здоровья, позиция и т.д.), открытые навыки, активные эффекты и прочее.

Рассмотрим следующую популярную игровую механику: систему рецептов. Для ее реализации создадим коллекцию рёбер recipes, которая будет содержать связи между предметами из коллекции items с направлением от ингредиентов к результату рецепта. Поскольку некоторые рецепты могут требовать несколько экземпляров предмета, необходимо сохранять требуемое количество как атрибут ребра. ArangoDB имеет уникальные возможности визуального представления объектов БД в различных форматах. Способ визуализации объекта можно выбрать в зависимости от текущей задачи, стоящей перед разработчиком. Так, например, объект recipes можно представить как в виде графа, так и в виде, характерном для документной модели (рис. 1 и 2).

Наиболее частыми запросами к системе рецептов будут следующие: «Получить все предметы, которые можно создать из имеющегося набора ингредиентов», «Получить все рецепты, содержащие указанный ингредиент», «Получить все ингредиенты, требуемые для создания конкретного предмета». Наилучшей моделью данных для оптимального выполнения подобных запросов является графовая модель, для ее реализации был создан

граф recipes. Тогда для выполнения запроса «Получить все предметы, которые можно создать из имеющегося набора ингредиентов», код которого приведен в [32], в качестве входных данных необходимо передать список объектов «item» с атрибутами «id» предмета и «count» (количество предметов). Результатом выполнения запроса будет список рецептов, которые можно сделать из всех указанных ингредиентов. Пример параметров запроса, а также его результат представлены на рис. 3 и 4 соответственно. Т.е. в параметрах было указано, что имеется 7 досок и 5 кремней, в результате выполнения запроса получено, что из данных ингредиентов можно изготовить верстак и плиты.

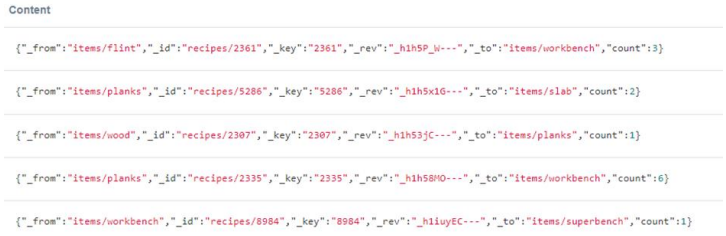


Рис. 1. Документное представление коллекции recipes

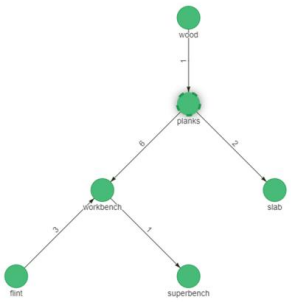
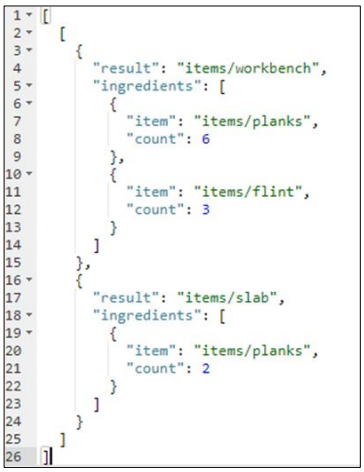


Рис. 2. Графовое представление коллекции recipes

```
{
    "items": [
        {
            "item": "items/planks",
            "count": 7
        },
        {
            "item": "items/flint",
            "count": 5
        }
    ]
}
```

Рис. 3. Параметры запроса на поиск подходящих рецептов



```
[
  [
    {
      "result": "items/workbench",
      "ingredients": [
        {
          "item": "items/planks",
          "count": 6
        },
        {
          "item": "items/flint",
          "count": 3
        }
      ]
    },
    {
      "result": "items/slab",
      "ingredients": [
        {
          "item": "items/planks",
          "count": 2
        }
      ]
    }
  ]
]
```

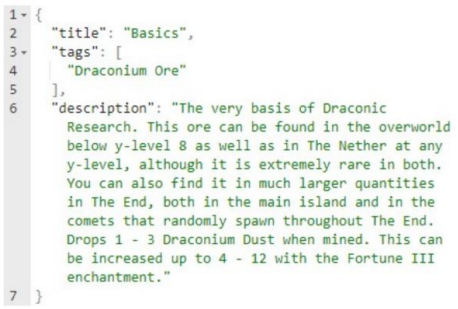
Рис. 4. Пример результата запроса на поиск подходящих рецептов

Второй и третий запрос из перечня выше аналогично используют поиск по графу, только в качестве параметра запроса необходимо предоставить идентификатор конкретного предмета.

Аналогичным образом была спроектирована система таблиц добычи ресурсов. Для хранения таблиц добычи была задействована документная модель данных, входящая в состав мультимодельной СУБД ArangoDB. Благодаря языку запросов AQL, расчёт таблиц добычи был произведен встроенными средствами СУБД без использования сторонних языков программирования. Наиболее популярными запросами здесь будут следующие: «Получить случайный набор ресурсов по настроенной таблице добычи» и «Показать шанс выпадения определённого предмета и его количества». Код этих запросов и структура коллекций системы добычи ресурсов приведены в [32].

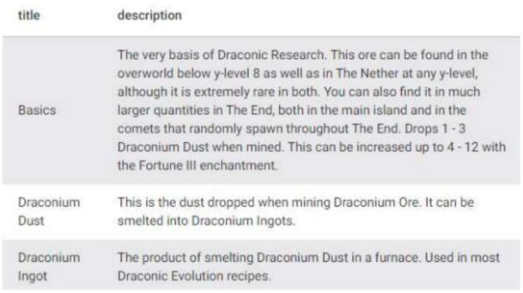
Для проектирования системы внутриигровой энциклопедии была использована встроенная система анализа текстов ArangoSearch, являющаяся одним из преимуществ ArangoDB по отношению к прочим СУБД данного класса. Хранение справочной информации будет осуществляться с использованием документной модели данных в коллекции wiki, состоящей из документов, представляющих собой статьи со свойствами «title» (название), «description» (описание) и «tags» (теги). Поиск будет выполняться как по названию, так и по тегам. Теги не являются обязательными, однако позволяют связывать определённые слова или фразы со статьями, которые не содержат указанных слов или фраз непосредственно, но являются единственными источниками информации по искомой теме. Пример статьи приведен на рис. 5.

Для корректной работы ArangoSearch необходимо создать и настроить для последующего поиска представление коллекции wiki, после чего можно строить запросы с помощью фразового анализатора. Код запроса на поиск информации во внутриигровой энциклопедии приведен в [32]. Запрос принимает на вход любое словосочетание, и если совпадение по названию или тегу является полным, то найденная статья отображается в качестве результата. Если же полные совпадения отсутствуют, то осуществляется поиск подстроки, где подстрокой является искомое словосочетание. Например, по названию «Draconium» показываются все статьи, так или иначе с ним связанные (рис. 6), однако если ввести «Ore», то результатом будет статья с рис. 5.



```
{
  "title": "Basics",
  "tags": [
    "Draconium Ore"
  ],
  "description": "The very basis of Draconic
    Research. This ore can be found in the overworld
    below y-level 8 as well as in The Nether at any
    y-level, although it is extremely rare in both.
    You can also find it in much larger quantities
    in The End, both in the main island and in the
    comets that randomly spawn throughout The End.
    Drops 1 - 3 Draconium Dust when mined. This can
    be increased up to 4 - 12 with the Fortune III
    enchantment."
}
```

Рис. 5.Пример статьи внутриигровой энциклопедии



title	description
Basics	The very basis of Draconic Research. This ore can be found in the overworld below y-level 8 as well as in The Nether at any y-level, although it is extremely rare in both. You can also find it in much larger quantities in The End, both in the main island and in the comets that randomly spawn throughout The End. Drops 1 - 3 Draconium Dust when mined. This can be increased up to 4 - 12 with the Fortune III enchantment.
Draconium Dust	This is the dust dropped when mining Draconium Ore. It can be smelted into Draconium Ingots.
Draconium Ingot	The product of smelting Draconium Dust in a furnace. Used in most Draconic Evolution recipes.

Рис. 6. Вывод статей по запросу ArangoSearch

Аналогично системе внутриигровой энциклопедии, документная модель данных может быть применена для реализации системы подбора игроков. Такая система пригодится для поиска соперников в матчах или союзников для похода в подземелье. Главная цель такой системы – получить ближайших по уровню игроков или игроков, уровень которых находится в заданном диапазоне, при этом игрок должен быть «онлайн», то есть присутствовать в игре в данный момент. Как и ранее, необходимо создать и настроить для последующего поиска представление коллекции players, после чего можно формировать запросы, используя команду search фразового анализатора. На вход разработанного запроса [32] в ArangoSearch подаётся никнейм игрока, для которого нужно произвести подбор, результатом запроса является перечень подходящих игроков. Так, например, для игрока, чьи параметры приведены на рис. 7, ближайшим по уровню игроком, находящимся онлайн, будет игрок, имеющий данные, приведенные на рис. 8.

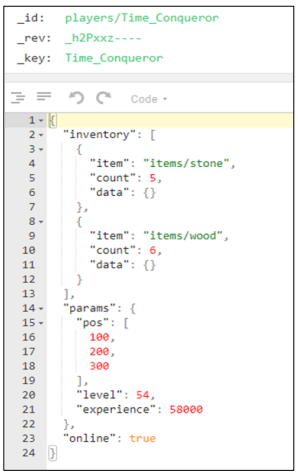


Рис. 7. Пример документа коллекции players

Рис. 8. Пример данных игрока, подобранного системой

Чтобы выяснить преимущества мультимодельных СУБД по отношению к одномодельным, необходимо рассмотреть примеры, в которых имеются отличающиеся друг от друга критерии работы с данными.

Первый эксперимент был проведён на квестовой системе. Типовые запросы здесь – это добавление игроку нового активного квеста, завершение квеста, получение всех активных и завершённых квестов игрока для отображения на его стороне, а также проверка, выполнен ли квест у игрока.

Выбранные для тестирования структуры одномодельных СУБД представлены на рис. 9 и 10. Для MongoDB и ArangoDB были созданы коллекции players (users) и quests. В случае с документной моделью было принято решение о хранении активных и завершенных квестов внутри документа игрока (рис. 11). Такой же подход можно было бы использовать и в мультимодельной БД, однако было принято решение провести эксперимент со связующей коллекцией players_quests. Эксперимент проводился на выборке из 20000 игроков, 50000 квестов, у каждого игрока в среднем 3000 завершенных квестов, 50 активных. Результаты эксперимента приведены в табл. 1 раздела Результаты.

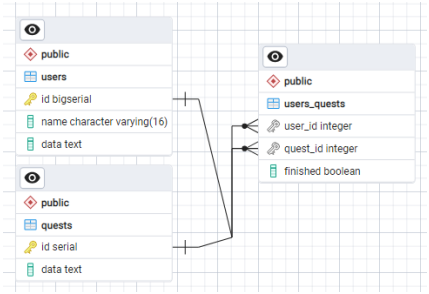


Рис. 9. Система квестов в реляционной БД (PostgreSQL)

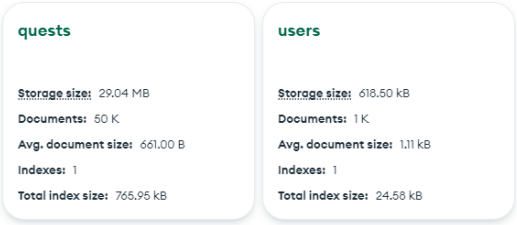


Рис. 10. Система квестов в документной БД

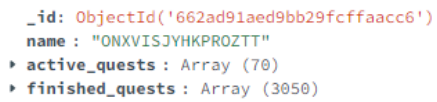


Рис. 11. Пример документа players

Следующий компонент игровых проектов – возможность создавать кланы. Кланом владеет какой-либо игрок, который может приглашать в него других игроков, а также выдавать им специальные роли. Помимо этого, кланы могут враждовать друг с другом. Игроки в таком случае могут атаковать игроков из враждебного клана.

Для хранения таких данных в реляционной БД потребуется четыре таблицы: players, clans, players_clans, clans_relations (рис. 13), SQL-запрос о враждебности конкретного игрока приведен в [32].

Чтобы хранить такую информацию в мультимодельной СУБД, также необходимо иметь коллекции players и clans, а также коллекцию связей players_clans, в которой будут храниться принадлежность игроков к кланам, а также отношения между самими кланами. Чтобы найти, враждуют ли два игрока, нужно пройти по графу на 3 вершины вперёд (игрок принадлежит клану, клан относится к клану, клану принадлежит игрок, см. рис. 13). AQL-запрос о враждебности конкретного игрока также приведен в [32].

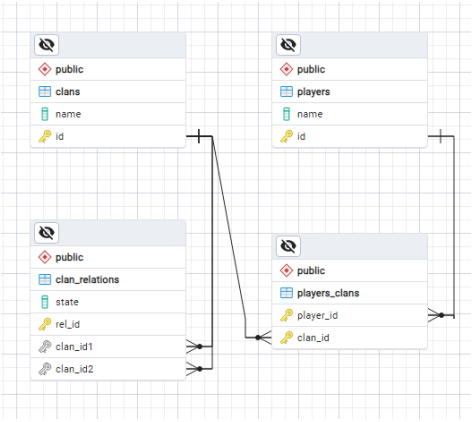


Рис. 12. Схема в PostgreSQL для системы кланов

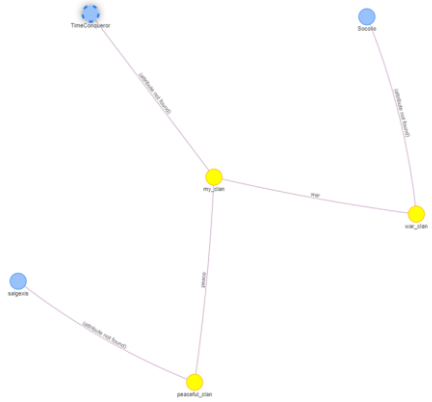


Рис. 13. Пример графа для поиска враждебных игроков. Голубым цветом отмечены игроки, желтым – кланы

Эксперимент по сравнению средней скорости ответа проведён на выборке из 230000 игроков и 7000 кланов. Сравнение средних скоростей запросов для поиска враждебных игроков в реляционной и мультимодельной БД приведены в табл. 2 раздела Результаты.

Далее была рассмотрена система диалогов, которая встречается в большинстве игр. Диалоги – это последовательность инструкций, каждая из которых обладает уникальным идентификатором и набором ссылок на идентификаторы следующих инструкций вместе с предикатами, указывающими, может ли следующая инструкция быть запущена, учитывая некоторые параметры. Инструкция может принадлежать только одному диалогу.

Если рассматривать структуру диалогов в рамках реляционной СУБД, то необходимо создать две таблицы: instructions, которая содержит в себе сами инструкции и instruction_links, реализующую связи между инструкциями (рис. 14).

Рассмотрим два варианта того, как запрашивать данные диалога из реляционной БД, приведенной на рис. 14. Первый из них – отправить в запросе идентификатор стартовой инструкции и от неё найти все следующие инструкции, принадлежащие одной последовательности до тех пор, пока ссылка на следующую инструкцию не будет равна null. Для реляционной базы данных задача извлечения информации из таблицы, структура которой имеет иерархическую рекурсию – крайне сложная и трудоёмкая. Следовательно, в данном случае все

данные в рамках одного диалога проще хранить в запакованном виде в одном столбце. Однако этот способ предполагает накладки по производительности, так как частая выгрузка объемных данных будет довольно медленной. В таких случаях данные можно закешировать и хранить в памяти вызывающего приложения, однако и это приведёт к большому количеству занимаемой памяти в случае большого количества игроков.

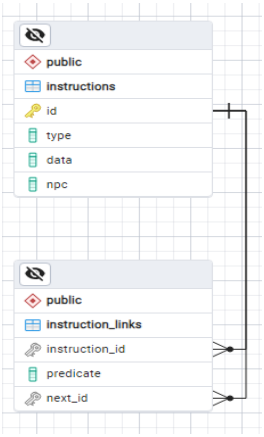


Рис. 14. Структура реляционной БД для диалоговой системы

Если диалоговую систему проектировать в мультимодельной СУБД, то имеет смысл хранить её в рамках документной модели, где каждый документ содержит в себе все инструкции (рис. 15). При этом легко решается вопрос с тем, что реплики NPC ограничиваются единственной строкой текста, в то время как ответов игрока может быть несколько – компонент data можно хранить как в виде строки, так и в виде массива строк. Компонент next, отвечающий за выбор следующих инструкций, также является массивом, где каждый элемент может хранить в себе условие для перехода – predicate. Если поле условия отсутствует, то следующий идентификатор инструкции рассматривается как путь по умолчанию. Такой формат хранения данных вместе с использованием мультимодельной СУБД позволяет удобно и в тоже время эффективно извлекать нужную информацию, а также устраняет необходимость дополнительного кеширования.

```
{
  "instructions": [
    {
      "id": 0,
      "type": "npc_text",
      "data": "Привет, путник!",
      "next": [
        {
          "id": 1
        }
      ]
    },
    {
      "id": 1,
      "type": "player_answer",
      "data": ["И тебе привет!", "Промолчать"],
      "next": [
        {
          "id": 2
        },
        {
          "predicate": ["silence_potion", "choosen_2"],
          "id": null
        }
      ]
    },
    {
      "id": 2,
      "type": "action",
      "data": "npc_go_away",
      "next": [
        {
          "id": null
        }
      ]
    }
  ]
}
```

Рис. 15. Пример хранения в документном стиле для диалоговой системы

Результаты. Рассмотрим подробнее результаты проведенных экспериментов.

Исследование быстродействия типовых запросов к квестовой системе показало, что реляционная база данных справляется с задачей извлечения квестов для конкретного игрока приблизительно с той же скоростью, что и документная база данных, однако добавление и изменение информации в случае использования реляционной БД происходит значительно быстрее (табл. 1). Это также является следствием того, что при выполнении основных вышеперечисленных запросов нет необходимости извлекать информацию о квестах частями, так как для отображения результатов запросов на стороне клиента нужна полная информация о квестах.

Несмотря на возможность ArangoDB работать с графовыми представлениями, при проведении тестов возникла проблема с отбором квестов конкретного игрока, причём чем больше было квестов у игрока, тем дольше длился запрос. По сравнению с реляционной и документной базами данных разница во времени выполнения запросов существенная. Для решения проблемы быстродействия были предприняты попытки добавления индексов в графовое представление, однако ввиду того, что квесты, хранимые в связях, по статистике почти всегда будут существенно преобладать над активными квестами, индексы не исправят проблему.

С точки зрения игровой логики представляется разумным запрашивать информацию о квестах игрока в момент входа пользователя в игру. Пока квест активен, системе нужна вся информация о квесте, при этом часть данных нужна для отображения на стороне пользователя, часть – для серверной логики. Таким образом, в квестовой системе не требуется запрашивать какие-либо характеристики отдельно, а, следовательно, реляционная БД в этом случае будет иметь преимущество. Мультимодельная же СУБД (при структуре документной базы данных) демонстрирует преимущество в скорости при добавлении и изменении данных, то есть при необходимости миграции на мультимодельную СУБД или изначальной реализации БД в мультимодельной СУБД, сильного ухудшения производительности в данном случае не ожидается.

Таблица 1

Сравнение средней скорости запросов для разных моделей (ms)

	Реляционная	Документная	Мультимодельная	Мультимодельная (ArangoDB, документная модель)
Добавление активного квеста игроку	0,56	2,76	1,4	1,4
Перевод квеста в статус «Завершён»	0,66 (с индексом по user_id и quest_id)	4,084	4	1,8
Получить все активные квесты игрока	1,09	1,874	4,69	1,4
Получить все завершённые квесты игрока	1,917	1,76	8,3	1,4
Завершён ли квест у игрока	0,5	1,573	1,45	1,3

Результаты сравнения производительности системы кланов представлены в табл. 2. Как можно заметить, реляционная СУБД многократно уступает мультимодельной в случае наличия нескольких соединений между объемными таблицами. Графовая же модель, являющаяся частью мультимодельной СУБД, позволяет эффективно решать такие задачи. Необходимо отметить, что ArangoDB реализует архитектурную модель Property Graph.

Таблица 2

Средняя скорость запроса реляционной и мультимодельной СУБД для поиска враждебных игроков (ms)

	Реляционная	Мультимодельная (ArangoDB, графовая модель)
Поиск враждебных игроков	57.6	5

Как видно из приведенной таблицы, мультимодельная БД, используя поиск на графе, значительно выигрывает в производительности. Отметим, что ранее в примере с подбором подходящего игрока коллекция игроков была представлена с помощью документной модели. В случае использования двух одномодельных СУБД для реализации, было бы необходимо решать проблему синхронизации данных двух БД. В случае же использования мультимодельного подхода на одни и те же данные можно смотреть с разных точек зрения и выбирать наиболее подходящее представление данных для решения текущей задачи, что, в свою очередь, повышает эффективность выполнения запросов.

Обсуждение. Экспериментальная часть исследования показала, что в случае, когда данные необходимо часто добавлять и изменять, а схема данных имеет структуру, близкую к линейной и выполнение запросов не требует многочисленных операций соединения, мультимодельная СУБД проигрывает реляционной по производительности, в особенности это касается внешних запросов больших массивов данных. Тем не менее, указанное обстоятельство не является препятствием к применению мультимодельных СУБД в игровой индустрии в случае, если при этом они активно используются для решения иных задач, возникающих в данной предметной области и задействуют различные модели данных.

Так, например, игровым проектам для хранения информации зачастую необходимо использовать графовую модель. Реляционная СУБД частично может покрыть эту необходимость ценой резкого снижения производительности и значительного увеличения сложности запросов, однако мультимодельная СУБД показывает лучшие результаты и позволяет покрыть большее количество потребностей. При этом мультимодельные СУБД показывают хорошие результаты и при обращении к данным, хранящимся в документной модели, что является их несомненным преимуществом, поскольку часть игровой информации удобно хранить именно в документном формате.

Помимо прочего, в работе был проведён анализ производительности и удобства использования мультимодельных СУБД по сравнению с одномодельными. Анализ показал, что мультимодельный подход имеет значительные преимущества в игровой области на большем количестве компонентов, чем одномодельные аналоги и позволяет в рамках одной базы данных: а) создавать отношения между документами одного типа, б) эффективно работать с графовой моделью, в) строить комплексные запросы к документной модели. Также существует возможность использования модели «ключ-значение» для хранения, например, игровых логов, что реализуемо в рамках ArangoDB. Отдельно стоит отметить удобство и универсальность языка запросов AQL, являющегося неотъемлемым компонентом ArangoDB, а также специализированный поисковый движок Arangosearch, упрощающий написание запросов.

Из недостатков использования ArangoDB стоит отметить сложность просмотра графовых данных, поскольку встроенный визуализатор имеет довольно скудное количество возможностей: нельзя выделять и передвигать группы компонентов и связей, а также быстро повторно сгенерировать получившийся граф. Собственно связи в графе часто визуализируются неоптимальным образом, оставляя большое количество неиспользуемого пространства (см. рис. 13).

В ходе анализа были выявлены случаи, когда мультимодельная БД проигрывает в производительности другим моделям, например, при выстраивании связей между элементами коллекции разных типов через специальную коллекцию рёбер, что оказалось менее эффективным, чем использование изначально документного подхода.

В рамках работы было проведено исследование применимости мультимодельных СУБД в рамках игровой индустрии и разработана структура мультимодельного хранилища на базе ArangoDB для наиболее часто использующихся игровых механик, систем и

компонентов. Исследование продемонстрировало, что разные компоненты системы ввиду специфики рассматриваемой области зачастую требуют как запросов к графовой модели, так и хранения данных в виде документов.

Дальнейшие исследования могут быть направлены на сравнение нескольких мультимодельных СУБД друг с другом с учётом особенностей различных предметных областей и анализ эффективности выполнения графовых запросов в зависимости от используемой архитектурной модели, плотности графа и объема хранимых данных.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Листьев Д.С., Савина А.Г., Малявкина Л.И.* Тенденции развития игровой индустрии // Цифровые инструменты обеспечения устойчивого развития экономики и образования: новые подходы и актуальные проблемы. – 2022. – С. 69-75.
2. *Кишкевич Д.В., Нестеренков С.Н., Марков А.Н.* Применение технологии Big Data в игровой индустрии = Application of Big Data technology in the game industry // BIG DATA and Advanced Analytics = BIG DATA и анализ высокого уровня: материалы VIII Междунар. науч.-практ. конф., Минск, 11–12 мая 2022 г. Белорус. гос. ун-т информатики и радиоэлектроники; редкол.: В.А. Богуш [и др.]. – Минск, 2022. – С. 120-124.
3. *Khine P.P., Wang Z.* A review of polyglot persistence in the big data world // Information. – 2019. – Vol. 10, No. 4. – P. 141.
4. *Holubová I., Vavrek M., Scherzinger S.* Evolution management in multi-model databases // Data & Knowledge Engineering. – 2021. – Vol. 136. – Art. 101932.
5. *Mihai (Rizescu) G.* Multi-Model Database Systems: The State of Affairs // Annals of Dunarea de Jos University of Galati. Fascicle I. Economics and Applied Informatics. – 2020. – Vol. XXVI. – P. 211-215.
6. *Lu J., Holubova I.* Multi-model Databases: A New Journey to Handle the Variety of Data // ACM Computing Surveys. – 2019. – Vol. 52, No. 3. – P. 1-38.
7. *Лапуценко А.В., Клочкова М.В.* Особенности различных систем управления базами данных // Потенциал российской экономики и инновационные пути его реализации. – 2022. – С. 337-342.
8. *Jowan S.A., Faraj Swese R., Yousf Aldabrzi A., Saad Shertil M.* Traditional RDBMS to NoSQL Database: New Era of databases for Big Data // Journal of Humanities and Applied Science. – 2016. – Vol. 29.
9. *Алексеев А.М., Борозна В.С., Суружий Н.А.* Системы управления базами данных. Классификация систем управления базами данных. – 2023.
10. *Карпюк А.Д., Власенкова Д.Г.* Какие виды СУБД и их реализации существует и что подходит лучшим образом для разработки автоматизированных информационных систем? // Приоритетные направления инновационной деятельности в промышленности: сб. науч. ст. по итогам Четвертой междунар. науч. конф., Казань, 29–30 апреля 2020 года. В 2 ч. Ч. 2. – 2020. – С. 62-64. – EDN UPKNFL.
11. Stack Overflow Developer Survey 2025 // Stack Overflow. – URL: https://survey.stackoverflow.co/2025 (дата обращения: 01.08.2025).
12. *Rai P.K., Singh P.* International Journal of Computer Science and Mobile Computing Studies and Analysis of Popular Database Models // International Journal of Computer Science and Mobile Computing. – 2015. – Vol. 4, No. 5. – P. 834-838.
13. *Соколов К.К.* Сравнение реляционных и нереляционных моделей СУБД // Научное обеспечение технического и технологического прогресса. – 2019. – С. 39-41.
14. *Guo Q., Zhang C., Zhang S., Lu J.* Multi-model query languages: taming the variety of big data // Distributed and Parallel Databases. – 2023. – P. 1-41.
15. *Куприянчик Е.М.* Сравнительный анализ подходов к разработке приложений NoSQL и SQL СУБД // XI Конгресс молодых ученых. – 2022. – С. 72-75.
16. *Pokorný J.* Graph databases: their power and limitations // Computer Information Systems and Industrial Management: Proc. of the 14th IFIP TC 8 International Conference, CISIM 2015, Warsaw, Poland, September 24-26, 2015. – Springer International Publishing, 2015. – P. 58-69.
17. *Шарипова Н.Н.* Об использовании NOSQL-хранилищ данных // Восточно-Европейский научный журнал. – 2016. – Т. 9, № 3. – С. 73-76. – EDN XRXLFF.
18. *Галигузова Е.В., Илларионова Ю.Е.* Сравнение реляционных и нереляционных СУБД // Символ науки. – 2023. – № 1-2. – С. 14-17.
19. Postmortem of service outage at 3.4M CCU // Fortnite: Official Website. – URL: https://www.fortnite.com/news/postmortem-of-service-outage-at-3-4m-ccu (дата обращения: 01.08.2025).
20. Базы данных в онлайн-играх. От Аллодов Онлайн до Skyforge // Блог компании VK. – URL: https://habr.com/ru/companies/vk/articles/182088/ (дата обращения: 01.08.2025).
21. Making Our Backside Bigger // Eve Online: Official Website. – URL: https://www.eveonline.com/news/view/making-our-backside-bigger (дата обращения: 01.08.2025).

22. *Котенко В.Н., Елисеев В.О.* Инновационный метод хранения данных в играх в жанре Role-Playing Game // Донецкие чтения 2021: образование, наука, инновации, культура и вызовы современности. – 2021. – С. 243-246.
23. За чашкой кофе с разработчиками: классическая версия World of Warcraft // Официальный сайт компании Activision Blizzard. – URL: https://news.blizzard.com/ru-ru/world-of-warcraft/21881587/za-chashkoj-kofe-s-razrabotchikami-klassicheskaya-versiya-world-of-warcraft (дата обращения: 01.08.2025).
24. *Abdullah Alqwbani B., Zuping Z., Aqlan F., Alqwbani A., Zuping Z., Aqlan F.* Big Data Management for MMO Games and Integrated Website Implementation // Global Journals Inc. (USA). – 2014. – Vol. 14, No. 6. – Version 1.0.
25. MMORPG Data Storage (Part 1) // Plant Based Games. – URL: https://plantbasedgames.io/blog/posts/01-mmorpg-data-storage-part-one/ (дата обращения: 01.08.2025).
26. Unity: The Gaming Industry in 2025 // GameDev Reports. – 2025. – URL: https://gamedevreports.substack.com/p/unity-the-gaming-industry-in-2025 (дата обращения: 24.08.2025).
27. Newzoo Global Games Market Report 2024 // Newzoo. – URL: https://newzoo.com/resources/trend-reports/newzoo-global-games-market-report-2024-free-version (дата обращения: 24.08.2025).
28. DB-Engines Ranking // DB-Engines. – URL: https://db-engines.com/en/ranking (дата обращения: 01.08.2025).
29. Обзор мультимодельных баз данных // Big Data School. – 2023. – URL: https://bigdataschool.ru/blog/multimodel-databases-overview/ (дата обращения: 24.08.2025).
30. NoSQL Performance Benchmark 2018 – MongoDB, PostgreSQL, OrientDB, Neo4j and ArangoDB // ArangoDB. – URL: https://arangodb.com/2018/02/nosql-performance-benchmark-2018-mongodb-postgresql-orientdb-neo4j-arangodb/ (дата обращения: 01.08.2025).
31. *Ye F., Sheng X., Nedjah N., Sun J., Zhang P.* A Benchmark for Performance Evaluation of a Multi-Model Database vs. Polyglot Persistence // Journal of Database Management. – 2023. – Vol. 34, No. 1. – P. 1-20.
32. TimeConqueror/gamedev-multimodal-dbms // GitHub. – URL: https://github.com/TimeConqueror/gamedev-multimodal-dbms (дата обращения: 01.08.2025).

REFERENCES

1. *List'ev D.S., Savina A.G., Malyavkina L.I.* Tendentsii razvitiya igrovoy industrii [Trends in the development of the gaming industry], *Tsifrovye instrumenty obespecheniya ustoychivogo razvitiya ekonomiki i obrazovaniya: novye podkhody i aktual'nye problem* [Digital tools for ensuring sustainable development of economy and education: new approaches and current problems]. 2022. pp. 69-75.
2. *Kishkevich D.V., Nesterenkov S.N., Markov A.N.* Primenenie tekhnologii Big Data v igrovoy industrii = Application of Big Data technology in the game industry [Application of Big Data technology in the game industry], *BIG DATA and Advanced Analytics = BIG DATA i analiz vysokogo urovnya: materialy VIII Mezhdunar. nauch.-prakt. konf., Minsk, 11–12 maya 2022 g. Belorus. gos. un-t informatiki i radioelektroniki* [BIG DATA and Advanced Analytics = BIG DATA i analiz vysokogo urovnya: materials of the VIII International Scientific and Practical Conference, Minsk, May 11–12, 2022. Belarusian State University of Informatics and Radioelectronics]; editorial board: V.A. Bogush [i dr.]. Minsk, 2022, pp. 120-124.
3. *Khine P.P., Wang Z.* A review of polyglot persistence in the big data world, *Information,* 2019, Vol. 10, No. 4, pp. 141.
4. *Holubová I., Vavrek M., Scherzinger S.* Evolution management in multi-model databases, *Data & Knowledge Engineering,* 2021, Vol. 136, Art. 101932.
5. *Mihai (Rizescu) G.* Multi-Model Database Systems: The State of Affairs, *Annals of Dunarea de Jos University of Galati. Fascicle I. Economics and Applied Informatics,* 2020, Vol. XXVI, pp. 211-215.
6. *Lu J., Holubova I.* Multi-model Databases: A New Journey to Handle the Variety of Data, *ACM Computing Surveys,* 2019, Vol. 52, No. 3, pp. 1-38.
7. *Laputsenko A.V., Klochkova M.V.* Osobennosti razlichnykh sistem upravleniya bazami dannykh [Features of various database management systems], *Potentsial rossiyskoy ekonomiki i innovatsionnye puti ego realizatsii* [Potential of the Russian economy and innovative ways of its implementation], 2022, pp. 337-342.
8. *Jowan S.A., Faraj Swese R., Yousf Aldabrzi A., Saad Shertil M.* Traditional RDBMS to NoSQL Database: New Era of databases for Big Data, *Journal of Humanities and Applied Science,* 2016, Vol. 29.
9. *Alekseev A.M., Borozna V.S., Suruzhiy N.A.* Sistemy upravleniya bazami dannykh. Klassifikatsiya sistem upravleniya bazami dannykh [Database management systems. Classification of database management systems], 2023.

10. *Karpyuk A.D., Vlasenkova D.G.* Kakie vidy SUBD i ikh realizatsii sushchestvuet i chto podkhodit luchshim obrazom dlya razrabotki avtomatizirovannykh informatsionnykh sistem? [What types of DBMS and their implementations exist and what is best suited for the development of automated information systems?], *Prioritetnye napravleniya innovatsionnoy deyatel'nosti v promyshlennosti: sb. nauch. st. po itogam Chetvertoy mezhdunar. nauch. konf., Kazan', 29–30 aprelya 2020 goda* [Priority directions of innovative activity in industry: collection of scientific articles on the results of the Fourth International Scientific Conference, Kazan, April 29–30, 2020]. In 2 parts. Part 2, 2020, pp. 62-64. EDN UPKNFL.
11. Stack Overflow Developer Survey 2025, *Stack Overflow*. Available at: https://survey.stackoverflow.co/2025 (accessed 01 August 2025).
12. *Rai P.K., Singh P.* International Journal of Computer Science and Mobile Computing Studies and Analysis of Popular Database Models, *International Journal of Computer Science and Mobile Computing,* 2015, Vol. 4, No. 5, pp. 834-838.
13. *Sokolov K.K.* Sravnenie relyatsionnykh i nerelyatsionnykh modeley SUBD [Comparison of relational and non-relational DBMS models], *Nauchnoe obespechenie tekhnicheskogo i tekhnologicheskogo progressa* [Scientific support of technical and technological progress], 2019, pp. 39-41.
14. *Guo Q., Zhang C., Zhang S., Lu J.* Multi-model query languages: taming the variety of big data, *Distributed and Parallel Databases,* 2023, pp. 1-41.
15. *Kupriyanchik E.M.* Sravnitel'nyy analiz podkhodov k razrabotke prilozheniy NoSQL i SQL SUBD [Comparative analysis of approaches to application development for NoSQL and SQL DBMS], *XI Kongress molodykh uchenykh* [XI Congress of Young Scientists], 2022, pp. 72-75.
16. *Pokorný J.* Graph databases: their power and limitations, *Computer Information Systems and Industrial Management: Proc. of the 14th IFIP TC 8 International Conference, CISIM 2015, Warsaw, Poland, September 24-26, 2015*. Springer International Publishing, 2015, pp. 58-69.
17. *Sharipova N.N.* Ob ispol'zovanii NOSQL-khranilishch dannykh [On the use of NoSQL data storages], *Vostochno-Evropeyskiy nauchnyy zhurnal* [East-European Scientific Journal], 2016, Vol. 9, No. 3, pp. 73-76. EDN XRXLFF.
18. *Galiguzova E.V., Illarionova Yu.E.* Sravnenie relyatsionnykh i nerelyatsionnykh SUBD [Comparison of relational and non-relational DBMS], *Simvol nauki* [Symbol of Science], 2023, No. 1-2, pp. 14-17.
19. Postmortem of service outage at 3.4M CCU, *Fortnite: Official Website*. Available at: https://www.fortnite.com/news/postmortem-of-service-outage-at-3-4m-ccu (accessed 01 August 2025).
20. Bazy dannykh v onlayn-igrakh. Ot Allodov Onlayn do Skyforge [Databases in online games. From Allods Online to Skyforge], *Blog kompanii VK* [VK Company Blog]. Available at: https://habr.com/ru/companies/vk/articles/182088/ (accessed 01 August 2025).
21. Making Our Backside Bigger [Making Our Backside Bigger], *Eve Online: Official Website* [Eve Online: Official Website]. Available at: https://www.eveonline.com/news/view/making-our-backside-bigger (accessed 01 August 2025).
22. *Kotenko V.N., Eliseev V.O.* Innovatsionnyy metod khraneniya dannykh v igrakh v zhanre Role-Playing Game [Innovative method of data storage in Role-Playing Games], *Donetskie chteniya 2021: obrazovanie, nauka, innovatsii, kul'tura i vyzovy sovremennosti* [Donetsk readings 2021: education, science, innovation, culture and challenges of our time], 2021, pp. 243-246.
23. Za chashkoy kofe s razrabotchikami: klassicheskaya versiya World of Warcraft [Over a cup of coffee with the developers: the classic version of World of Warcraft], *Ofitsial'nyy sayt kompanii Activision Blizzard* [Official website of Activision Blizzard company]. Available at: https://news.blizzard.com/ru-ru/world-of-warcraft/21881587/za-chashkoj-kofe-s-razrabotchikami-klassicheskaya-versiya-world-of-warcraft (accessed 01 August 2025).
24. *Abdullah Alqwbani B., Zuping Z., Aqlan F., Alqwbani A., Zuping Z., Aqlan F.* Big Data Management for MMO Games and Integrated Website Implementation, *Global Journals Inc. (USA),* 2014, Vol. 14, No. 6, Version 1.0.
25. MMORPG Data Storage (Part 1), *Plant Based Games*. Available at: https://plantbasedgames.io/blog/posts/01-mmorpg-data-storage-part-one/ (accessed 01 August 2025).
26. Unity: The Gaming Industry in 2025, *GameDev Reports,* 2025. Available at: https://gamedevreports.substack.com/p/unity-the-gaming-industry-in-2025 (accessed 01 August 2025).
27. Newzoo Global Games Market Report 2024, *Newzoo*. Available at: https://newzoo.com/resources/trend-reports/newzoo-global-games-market-report-2024-free-version (accessed 24 August 2025).
28. DB-Engines Ranking, *DB-Engines*. Available at: https://db-engines.com/en/ranking (accessed 01 August 2025).
29. Obzor mul'timodel'nykh baz dannykh [Overview of multi-model databases], *Big Data School*, 2023. Available at: https://bigdataschool.ru/blog/multimodel-databases-overview/ (accessed 24 August 2025).

30. NoSQL Performance Benchmark 2018 – MongoDB, PostgreSQL, OrientDB, Neo4j and ArangoDB // ArangoDB. Available at: https://arangodb.com/2018/02/nosql-performance-benchmark-2018-mongodb-postgresql-orientdb-neo4j-arangodb/ (accessed 01 August 2025).
31. *Ye F., Sheng X., Nedjah N., Sun J., Zhang P.* A Benchmark for Performance Evaluation of a Multi-Model Database vs. Polyglot Persistence, *Journal of Database Management*, 2023, Vol. 34, No. 1, pp. 1-20.
32. TimeConqueror/gamedev-multimodal-dbms, *GitHub*. Available at: https://github.com/TimeConqueror/gamedev-multimodal-dbms (accessed 01 August 2025).

Коблов Андрей Александрович – ООО "Коммуникационная платформа"; e-mail: timeconqueror999@gmail.com; г. Москва, Россия; программист-разработчик; ORCID: 0009-0007-0075-1794.

Ромакина Оксана Михайловна – Национальный исследовательский университет ИТМО; e-mail: omromakina@itmo.ru; г. Санкт-Петербург, Россия; к.ф.-м.н.; доцент факультета Прикладной информатики; ORCID: 0000-0001-9468-4404.

Клемешева Анастасия Сергеевна – Национальный исследовательский университет ИТМО; e-mail: primell@mail.ru; г. Санкт-Петербург, Россия; магистрант факультета Прикладной информатики; ORCID: 0009-0008-1259-8886.

Арсеньева Анна Закировна – Национальный исследовательский университет ИТМО; e-mail: anna.z.arseneva@itmo.ru; г. Санкт-Петербург, Россия; старший преподаватель факультета Прикладной информатики; ORCID: 0000-0002-2606-1667.

Koblov Andrey Alexandrovich – OOO "Communication Platform"; e-mail: timeconqueror999@gmail.com; Moscow, Russia; programmer-developer; ORCID: 0009-0007-0075-1794.

Romakina Oksana Mikhaylovna – ITMO University; e-mail: omromakina@itmo.ru; St. Petersburg, Russia; cand. of phys. and math. sc.; associate professor of the Faculty of Applied Informatics; ORCID: 0000-0001-9468-4404.

Klemesheva Anastasiia Sergeevna – ITMO University; e-mail: primell@mail.ru; St. Petersburg, Russia; master student of the Faculty of Applied Informatics; ORCID: 0009-0008-1259-8886.

Arseneva Anna Zakirovna – ITMO University; e-mail: anna.z.arseneva@itmo.ru; St. Petersburg, Russia; senior tutor of the Faculty of Applied Informatics; ORCID: 0000-0002-2606-1667.

УДК 004.382.2 DOI 10.18522/2311-3103-2025-6-121-136

А.К. Мельников

ИССЛЕДОВАНИЕ ВОЗМОЖНОСТЕЙ ПРИМЕНЕНИЯ ФОТОННЫХ И КВАНТОВЫХ ВЫЧИСЛИТЕЛЬНЫХ ТЕХНОЛОГИЙ ДЛЯ РАСЧЕТА ТОЧНЫХ РАСПРЕДЕЛЕНИЙ ВЕРОЯТНОСТЕЙ ЗНАЧЕНИЙ СТАТИСТИК КОНЕЧНЫХ ДИСКРЕТНЫХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ

Проведено исследование возможности применения фотонных и квантовых вычислительных технологий для расчета точных распределений вероятностей значений статистик дискретных последовательностей в предположении о наличии работающих технических образцов вычислительных систем и создания требуемых квантовых алгоритмов. Оценка производительности вычислительных систем на базе фотонных вычислительных технологий базируется на материалах НЦФМ РАН г. Саров. Оценка производительности квантовой вычислительной системы проведена методом сравнения времени решения задачи отбора проб бозонов из заданного распределения на вычислительной системе с известной производительностью и времени её выполнения на квантовой вычислительной системе. Для оценки возможности применения фотонных и квантовых вычислительных технологий к расчету точных распределений рассмотрены современные методы их вычисления, основанные на решении уравнения кратности типов и системы линейных уравнений в неотрицательных целых числах. Приводятся аналитические выражения, определяющие вычислительную сложность этих методов. Проведено определения значений границ параметров точных распределений доступных для вычисления с помощью применения фотонных и квантовых вычислительных технологий. Приводится сравнение полученных результатов с результатами примене-