

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ**

**ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ» (ДГТУ)**

Авиационно-технологический колледж

Директор Авиационно- технологического колледжа

В.А. Зибров

« » 2024 г.

**ИНДИВИДУАЛЬНЫЙ ПРОЕКТ**

Тема: «Исследование баз данных в разработке и управлении игровыми мирами»

Учебный предмет МДК.11.01 «Технология разработки и защиты баз данных»

Специальность:09.02.07 «Информационные системы и программирование» Обозначение индивидуального проекта Т.880000.000 Группа ИСП11-Kh21

Обучающийся

подпись

А. Е. Бабкин

Индивидуальный проект защищен

дата оценка

Руководитель работы

подпись

преподаватель М.С. Фудулей

Ростов-на-Дону 2024



МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ**

**ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ» (ДГТУ)**

Авиационно-технологический колледж

**ПАСПОРТ КУРСОВОГО ПРОЕКТА**

|  |  |
| --- | --- |
| Название проекта | Исследование баз данных в разработке и управлении игровыми мирами |
| Учебный предмет | МДК 11.01 «Технология разработки и защиты баз данных» |
| Автор проекта | Бабкин Александр Евгеньевич |
| Руководитель проекта | Фудулей Мария Сергеевна |
| Тип проекта | Исследовательская |
| Возраст пользователей, на  который рассчитан проект | Возраст пользователей, на которых рассчитан проект варьируется от 18 до 35 лет. В основном это начинающие разработчики, тестировщики, которые погружаются в иследование принципов работы игровых механик. |
| Год разработки | 2024 |
| Актуальность проекта (описание  проблемы) | Актуальность данного проекта обусловлена стремительным развитием разработки видеоигры и увеличением сложности проектов. |
| Цель и задачи проекта | Анализ структуры и технологий баз данных, используемых в игровой индустрии. |
| **Результаты**  (личностные,  метапредметные, предметные) | Личностные: навыки эффективного общения с коллегами и клиентами, что нужно для успешного выполнения проекта;  Критическое мышление: умение анализировать и оценивать информацию выявлять проблемы и находить решения в области защиты данных;  Ответственность: развитие чувства ответственности за качество выполненной работы и соблюдение стандартов безопасности данных |
|  | Метапредметные: улучшению навыков работы в команде способствует, участие в проектах, требующих совместных усилий;  Проектное управление: освоение основ управления проектами, включая планирование, организацию и контроль выполняемых задач. |
|  | сформировать умения применять полученные знания для объяснения условий протекания физических явлений в природе, профессиональной сфере и для принятия практических решений в повседневной жизни;  сформировать собственной позиции по отношению к физической информации, получаемой из разных источников |
| Предметные: знания в области безопасности данных: понимание принципов и технологий защиты информации включая шифрование, аутентификацию и управление доступом  Технические навыки: освоение языков программирования и технологий, используемых для разработки и защиты баз данных  Практические навыки разработки: умение создавать и оптимизировать базы данных, а также разрабатывать программные решения для их защиты |
| Продукт | Структуры баз данных для игровой индустрии |
| Аннотация (краткое описание  работы) | Исследовательская работа, посвященная анализу структуры и технологий баз данных, используемых в игровой индустрии. Работа охватывает ключевые аспекты, такие как архитектура баз данных для хранения игровых данных, производительность баз данных, кэширование, технологии NoSQL и облачные решения. |

Оглавление

Введение 5

1. Структура баз данных для хранения игровых данных 6

2.Отношение между производительностью и игровыми процессом 8

3. Кэширование и оптимизация запросов в играх 11

4. Технологии кэширования 13

5.Игра как сервис (GaaS) и хранение данных в облаке 14

6. Практическая часть 17

Заключение 23

Список литературы 25

# Введение

Современные видеоигры становятся все более сложными и многофункциональными. Это связано с ростом требований пользователей к качеству графики, интерактивности и многопользовательскому взаимодействию. Важным аспектом успешной игры является эффективная организация и управление данными. В данной исследовательской работе рассматриваются ключевые темы, связанные с архитектурой баз данных для хранения игровых данных, производительностью, кэшированием, технологиями NoSQL и облачными решениями для хранения данных.

Целью данной исследовательской работы является анализ структуры и технологий баз данных в игровой индустрии. Исследование направлено на изучение различных типов баз данных, их архитектуры и применения в контексте хранения игровых данных, оценку производительности реляционных и NoSQL баз данных в многопользовательских играх, анализ технологий кэширования и оптимизации запросов для повышения отзывчивости игр, а также исследование роли облачных решений и модели "Игра как сервис" в управлении игровыми данными.

Актуальность темы заключается в том, что в современном мире видеоигры стали неотъемлемой частью культуры и развлечений, привлекая миллионы пользователей по всему миру. С увеличением популярности многопользовательских онлайн-игр и сложных игровых механик, требования к качеству и производительности игр значительно возросли.

Объекты исследования включают структуры баз данных, такие как реляционные и NoSQL базы данных, игровые данные, используемые в играх (персонажи, предметы, уровни), производительность баз данных, влияние производительности на отзывчивость и качество игрового процесса, технологии

кэширования, такие как Redis и Memcached, методы оптимизации SQL-запросов, а также облачные решения и модель "Игра как сервис".

**1 Структура баз данных для хранения игровых данных**

Правильная модель данных является основой эффективной работы игры [7]. Ниже описаны основные элементы модели:

Персонажи. Данные о персонажах представляют одну из самых сложных и объемных структур:

1. Идентификатор персонажа: уникальный ключ для идентификации в базе данных.
2. Основные параметры: имя, уровень, текущие очки здоровья (HP), мана (MP), опыт и прочие показатели.
3. Характеристики: сила, ловкость, интеллект, выносливость, скорость и другие параметры.
4. Инвентарь: список предметов, закрепленных за персонажем, включая их количество и состояние.
5. История достижений: хронология выполненных заданий, собранных наград и участвовавших событий.

Предметы в игре могут быть как стандартными, так и уникальными:

1. Уникальный идентификатор: для отслеживания конкретного предмета.
2. Название и описание: отображение для игрока.
3. Тип предмета: оружие, броня, расходуемые материалы (например, зелья).
4. Статистики: урон, защита, дополнительные эффекты.
5. Стоимость: как внутриигровая валюта, так и реальные деньги (для микротранзакций).
6. Уровни

Данные уровней используются для хранения информации об игровом окружении:

1. Идентификатор уровня: уникальный ключ.
2. Название и описание: текстовые данные для отображения.
3. Сложность: характеристика, влияющая на игровой процесс.
4. Связанные элементы: враги, задания, доступные предметы.

Пример модели в реляционной базе данных для MySQL это может выглядеть следующим образом:

В игре World of Warcraft используется реляционная модель данных. Каждый элемент игры связан с другими: персонажи, их снаряжение, выполненные квесты и взаимодействия. Это позволяет быстро извлекать нужные данные и гарантировать их целостность.

**2 Отношение между производительностью   
и игровым процессом**

Производительность базы данных – один из ключевых факторов, определяющих успех многопользовательских игр. Высокая скорость обработки данных напрямую влияет на отзывчивость игрового процесса, минимизацию задержек и плавность взаимодействия пользователей [1]. Реляционные против NoSQL баз данных [3].

Реляционные базы данных (RDBMS), такие как MySQL, PostgreSQL и Microsoft SQL Server, используют строго структурированные таблицы и обеспечивают надежность данных благодаря поддержке ACID-транзакций (атомарность, согласованность, изолированность и долговечность).[9]

Преимущества:

1. Целостность данных: подходит для приложений, где требуется строгая логика и сложные взаимосвязи между данными (например, MMORPG).
2. Гибкость запросов: использование SQL позволяет выполнять сложные запросы и объединять таблицы.
3. Расширенная аналитика: поддержка аналитических запросов для анализа игрового поведения.

Ограничения:

1. Сложность масштабирования: масштабирование реляционных баз данных горизонтально (добавлением новых серверов) требует значительных усилий.
2. Падение производительности при больших объемах данных: в условиях высоконагруженных игр обработка миллионов запросов одновременно может вызывать задержки.

NoSQL базы данных, такие как MongoDB, Redis, Cassandra и DynamoDB, предназначены для работы с гибкими структурами данных и высокой масштабируемостью.

Преимущества:

1. Горизонтальное масштабирование: легко масштабируются, добавляя новые узлы.
2. Высокая производительность: оптимизированы для быстрого чтения и записи данных.
3. Поддержка неструктурированных данных: подходит для игр с постоянно меняющимися данными, таких как инвентари, параметры персонажей и динамические события.

Ограничения: ограниченная транзакционность, некоторые NoSQL базы данных не гарантируют строгую изолированность транзакций.

Примеры использования:

1. Реляционные базы данных: в World of Warcraft используется реляционная модель для хранения информации о персонажах, их экипировке, квестах и связях между игроками.
2. NoSQL базы данных: в Fortnite данные о пользователях и игровом процессе хранятся в NoSQL Cassandra и Redis, что позволяет обрабатывать миллионы операций записи и чтения в секунду, сохраняя высокую скорость.

Производительность базы данных может определять ключевые аспекты игрового опыта:

1. Обработка игровых событий в реальном времени. В играх типа Battle Royale, таких как PUBG или Fortnite, требуется мгновенная обработка событий, таких как движения игроков, попадания пуль или смена зон. Использование NoSQL решений позволяет обрабатывать миллионы операций в реальном времени.

2. Хранение данных игроков. Сложные системы инвентаря, персонализации и статистики требуют надежной и масштабируемой архитектуры. Например, Destiny 2 использует гибридный подход, комбинируя реляционные и NoSQL решения для управления данными персонажей.

3. Масштабирование при пиковых нагрузках. В многопользовательских играх нагрузка может резко увеличиваться во время выхода новых обновлений. Облачные NoSQL базы, такие как Amazon DynamoDB, автоматически масштабируются, чтобы избежать падения серверов.

**3 Кэширование и оптимизация запросов в играх**

Кэширование – ключевой метод повышения производительности за счет хранения часто запрашиваемых данных в оперативной памяти. Оно используется для сокращения задержек и уменьшения нагрузки на основную базу данных [4].

Redis – это key-value хранилище, которое активно используется в игровом кэшировании:

1. Хранение данных о сессиях игроков.
2. Кэширование информации о матчах и игровых объектах.
3. Быстрое обновление состояния игры.

Пример: В League of Legends Redis используется для хранения данных о текущем состоянии матчей, что позволяет сократить время на запросы к основной базе данных.

Memcached оптимален для кэширования больших объемов простых данных. Используется для хранения:

1. Авторизационных токенов.
2. Информации о рейтингах игроков.

Пример: В Clash Royale Memcached используется для хранения данных о лидербордах.

CDN для статических данных

Контентные сети доставки (CDN), такие как Cloudflare, используются для хранения и доставки статических данных, например:

1. Текстур.
2. Анимаций.
3. Звуковых эффектов.
4. Оптимизация запросов.

Оптимизация запросов позволяет снизить нагрузку на базу данных и увеличить скорость выполнения операций.

Методы оптимизации:

1. Индексация данных. Индексы на полях, часто используемых в WHERE и JOIN, ускоряют поиск. Например:
2. Индекс на PlayerID позволяет быстро находить данные конкретного игрока.
3. Пакетная обработка запросов. Объединение нескольких операций в одну транзакцию уменьшает количество взаимодействий с базой данных.
4. Денормализация данных. Часто используемые данные копируются в одной таблице для ускорения запросов. Например, информация о текущем состоянии персонажа может храниться в одном документе в MongoDB.
5. Шардинг базы данных. Разделение базы на сегменты позволяет распределить нагрузку между серверами. Пример: В Genshin Impact шардинг используется для разделения данных игроков по регионам.

Пример оптимизации в играх: в EVE Online база данных оптимизирована для обработки миллионов транзакций одновременно благодаря активной индексации и денормализации часто запрашиваемых данных.

**4 Технологии NoSQL в игровых приложениях**

NoSQL базы данных становятся все более популярными в игровой индустрии благодаря своей способности обрабатывать большие объемы данных и обеспечивать высокую производительность.[10]

Документные базы данных, такие как MongoDB, позволяют хранить данные в формате JSON, что упрощает работу с гибкими структурами данных. Это особенно полезно для хранения информации о предметах и персонажах.

Графовые базы данных, такие как Neo4j, идеально подходят для управления сложными взаимосвязями между объектами в игре. Это может быть полезно для создания социальных взаимодействий между игроками.

**5 Игра как сервис (GaaS) и хранение данных в облаке**

Модель "Игра как сервис" (Game as a Service, GaaS) представляет собой стратегию, при которой игры предоставляются пользователям как постоянный сервис. Это предполагает регулярные обновления, управление игровыми профилями, поддержание онлайн-сообщества и проведение событий. Для реализации GaaS ключевым является эффективное использование облачных технологий для хранения и обработки игровых данных.[6]

Облачные технологии предлагают разработчикам множество преимуществ, делая их неотъемлемой частью современной игровой индустрии:

Возможность легко увеличивать или уменьшать ресурсы в зависимости от объема нагрузки.

Игры, такие как Apex Legends, испытывают значительное увеличение количества пользователей во время событий или релизов обновлений. Облачные провайдеры, такие как Amazon Web Services (AWS), позволяют автоматизировать добавление серверов для предотвращения задержек.

Облачные хранилища, такие как Google Cloud Storage и Microsoft Azure, обеспечивают высокую доступность данных за счет распределения по дата-центрам в разных регионах. Это минимизирует риск потери данных даже при сбоях оборудования.

Разработчикам не нужно покупать и поддерживать собственное серверное оборудование. Вместо этого они платят только за использованные ресурсы.

Например, малые студии могут быстро запустить проект на облачной платформе, начиная с минимальных затрат, а затем масштабировать инфраструктуру по мере роста аудитории.

Облачные решения позволяют разместить игровые серверы ближе к конечным пользователям, минимизируя задержки.

Microsoft Azure PlayFab предоставляет разработчикам возможность выбора серверов в разных регионах для улучшения игрового опыта.

Большинство облачных платформ предлагают встроенные инструменты аналитики (например, Google BigQuery или AWS QuickSight). Это помогает анализировать игровые данные, выявлять тенденции и оптимизировать игровой процесс.

Облачные технологии позволяют автоматизировать развертывание обновлений, обеспечивая их одновременное применение ко всем серверам.

Примеры облачных технологий в GaaS:

1. Apex Legends. Платформа: Amazon Web Services (AWS).

Применение: хранение данных о пользователях, таких как статистика матчей, прогресс и инвентарь. Автоматическое масштабирование серверов во время пиковых нагрузок. Облачные базы данных, такие как DynamoDB, для быстрого доступа к пользовательским данным.

1. Fortnite. Платформа: AWS и собственные серверы Epic Games.

Применение: хранение данных о покупках и прогрессе игроков в облачных базах. Реализация глобального сервиса с низкими задержками благодаря географически распределенным серверам.

1. Minecraft Realms. Платформа: Azure (Microsoft).

Применение: Обеспечение персональных игровых серверов для игроков. Поддержка глобального соединения между игроками и сохранение данных о игровых мирах.

1. Genshin Impact. Платформа: использует гибридную архитектуру облачных серверов (включая Alibaba Cloud).

Применение: обеспечение глобального охвата игроков благодаря региональным серверам. Управление событиями и регулярными обновлениями с минимальными задержками.

1. World of Warcraft. Платформа: использует гибридное облачное решение.

Применение: хранение огромных массивов данных о персонажах и мирах. Облачная обработка аналитики игровых данных для анализа пользовательского опыта.

Дополнительные аспекты облачных технологий в GaaS:

Безопасность данных. Для GaaS-игр крайне важно защитить данные пользователей, включая информацию об аккаунтах и покупках. Большинство облачных провайдеров предлагают инструменты для шифрования данных, многофакторной аутентификации и мониторинга безопасности.

Например, AWS Shield защищает от DDoS-атак, которые часто становятся угрозой для популярных игр.

Инфраструктура для машинного обучения. Облачные платформы, такие как Google Cloud и Azure, предоставляют инструменты для использования машинного обучения. Это может быть применимо для:

1. Анализа пользовательского поведения.
2. Определения токсичных игроков или предотвращения мошенничества.
3. Динамической настройки сложности игры на основе данных.
4. Интеграция с GaaS-платформами

Многие облачные решения предлагают специализированные инструменты для игр, такие как:

1. Google Cloud Game Servers: упрощает управление игровыми серверами.
2. Amazon GameLift: помогает быстро развертывать и масштабировать игровые серверы.
3. PlayFab: позволяет управлять игровыми данными, монетизацией и аналитикой.

## 6 Практическая часть

Пример создания таблицы в SQL для хранения:

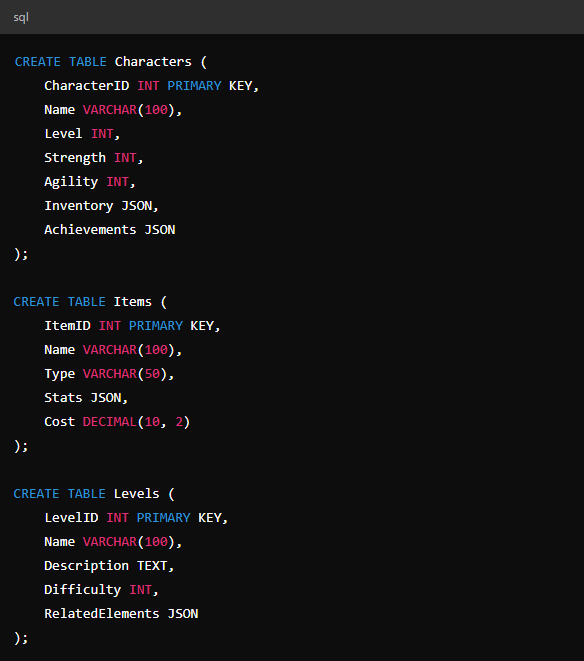


Рис. 1 «Информация о персонажах и их характеристиках»

Вариант заполнения таблицы, который позволяет гибко управлять информацией о персонажах, их характеристиках, экипировке и умениях, можете добавлять новые таблицы, например, для хранения достижений или журналов событий:

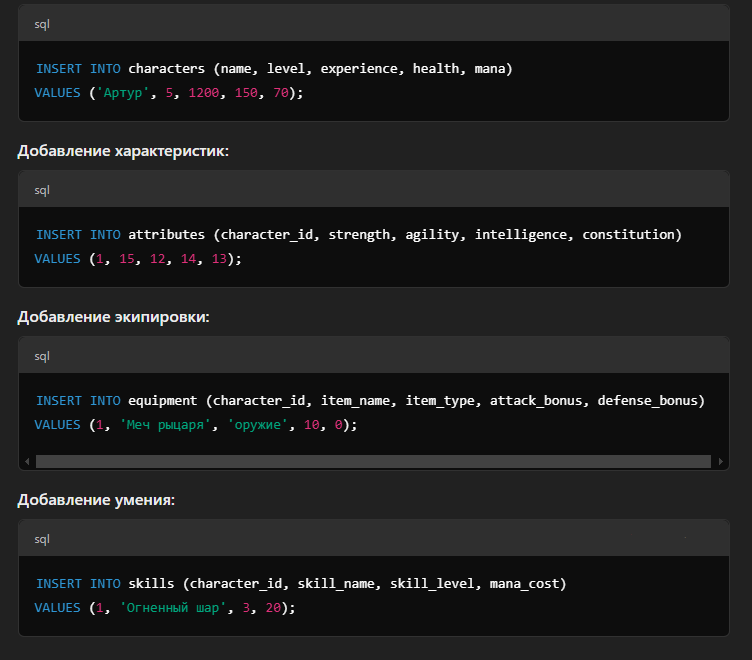


Рис. 2 «Запросы для заполнения»

Простые запросы, рассчитанные для получения общего онлайна на сервере:

## 

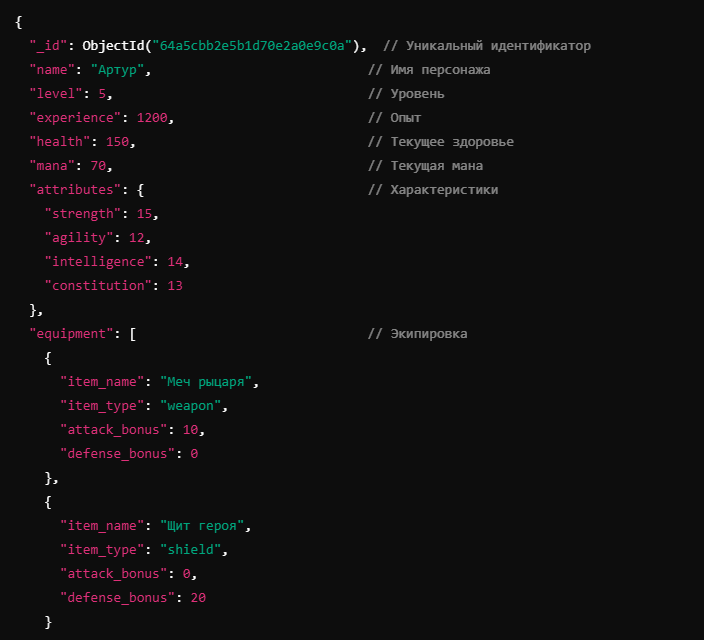
## Рис. 3 «Таблица и запрос кол-во игроков»

## Кэширование данных персонажа может быть полезным, чтобы ускорить доступ к часто используемой информации.

## 

## Рис. 4 «Запросы для кэширования данных о персонаже»

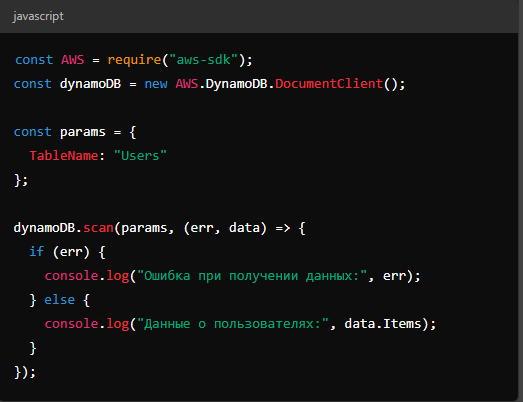
Примеры запросов при помощи Amazon DynamoDB (NoSQL), запросы выполняются с помощью AWS SDK для JavaScript:



## Рис. 5 «Пример запроса получение информации о персонаже»



## Рис. 6 «Пример запроса помощью AWS SDK»



## Рис. 7 «Пример запроса AWS SDK для всех пользователей»

## Заключение

Современная игровая индустрия представляет собой высокотехнологичную сферу, где использование баз данных играет ключевую роль в создании и поддержании сложных игровых систем. Тщательно продуманная структура баз данных обеспечивает эффективное управление игровыми данными, начиная с характеристик персонажей и заканчивая уровнями сложности и инвентарем. Использование реляционных и NoSQL решений позволяет разработчикам выбирать наиболее подходящие инструменты в зависимости от сложности данных и требований производительности. Такие игры, как World of Warcraft, демонстрируют мощь реляционных баз, тогда как Fortnite показывает, как NoSQL технологии справляются с высоконагруженными системами.

Производительность баз данных напрямую влияет на качество игрового процесса. От скорости обработки запросов зависит отзывчивость игры, минимизация задержек и плавность взаимодействия. Примеры, такие как **Destiny 2** и EVEOnline, иллюстрируют важность оптимизации запросов, кэширования данных и использования распределенных систем для достижения высоких стандартов в игровом опыте. Комбинация реляционных и NoSQL баз данных часто оказывается наиболее эффективным подходом, позволяя учитывать особенности различных типов данных.

Ключевым аспектом успешной работы многопользовательских игр стала интеграция облачных технологий. Такие платформы, как AWS, GoogleCloud и Azure, предлагают не только масштабируемость и надежность, но и инструменты для глобального охвата, анализа данных и защиты от угроз. Игры в формате "Game as a Service" (GaaS) используют эти технологии для обеспечения непрерывной работы серверов, минимизации задержек и автоматизации обновлений, что позволяет разрабатывать игры, способные адаптироваться к растущим запросам пользователей.

В итоге, инновации в области баз данных и облачных технологий открывают перед разработчиками огромные возможности. Эти решения позволяют улучшить производительность, обеспечить гибкость, безопасность и надежность игровых систем. Исследования и примеры из реальной практики демонстрируют, как грамотный выбор и применение технологий делают возможным создание масштабируемых, интерактивных и захватывающих игровых проектов, способных удовлетворить потребности современных игроков.

## Список используемых информационных ресурсов

1. Г. Домбровская, Б. Новиков, А. Бейликова «Оптимизация запросов PostgreSQL», 2022. – С 138-141
2. Н. Копилэнд «MongoDB Applied», 2013. – С 77-85
3. Р. Фабиан «Data-Oriented Design», 2018. – С 33-45
4. С. Ботрос, Д. Тинлиу «High-Perfimance MySQL», 2013. – С 246-261
5. М. Клиппманн «Designing Data-Intensive Applications», 2024. – С 58-87
6. Р. Нусторм «Game Programming Patterns», 2014. – С 99-132
7. И. А. Липаев «Базы данных: Теория и практика», 2016. – С 25-89
8. С. Тарасов «СУБД для программиста Базы данных изнутри»,2015. – С 44-79
9. Дж. Аллен «SQL для чайников», 2020. – С 88-108