Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОЙ РАБОТЕ

НА ТЕМУ:

Разработка базы данных для хранения и обработки результатов проведения тестов Тьюринга

Студент ИУ7-65Б (Группа) _____ (Подпись, дата) **С. К. Романов** (И.О.Фамилия)

Руководитель курсового проекта _____ (Подпись, дата) **К.А. Кивва** (И.О.Фамилия)

Консультант _____ (Подпись, дата) **К.А. Кивва** (И.О.Фамилия)

2023 г.

СОДЕРЖАНИЕ

ОПРЕДЕЛЕНИЯ

В настоящей расчетно-пояснительной записке применяют следующие термины с соответствующими определениями.

Natural Language Processing — «Обработка текстов на естественном языке» относится к области компьютерных наук, а точнее к области искусственного интеллекта или ИИ, связанной с предоставлением компьютерам возможность понимать текст и произносимые слова почти так же, как люди. НЛП сочетает в себе вычислительную лингвистику — моделирование человеческого языка на основе правил — со статистическими моделями, машинным обучением и моделями глубокого обучения. Вместе эти технологии позволяют компьютерам обрабатывать человеческий язык в виде текстовых или голосовых данных и «понимать» его полное значение, включая намерения и чувства говорящего или пишущего. [1]

ACID — В контексте обработки транзакций аббревиатура ACID относится к четырем ключевым свойствам транзакции: атомарность (Atomicity), непротиворечивость (Consistency), изоляция (Isolation) и устойчивость (Durability). [2].

Graph Database — база данных, использующая структуры графов для семантических запросов с узлами, ребрами и свойствами для представления и хранения данных. [2].

ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

В настоящей расчетно-пояснительной записке применяют следующие сокращения и обозначения.

NLP — «Natural Language Processing».

ACID — «Atomicity, Consistency, Isolation, Durability».

SQL — «Structured Query Language».

GDB — «Graph Database»

ИИ — «Искусственный Интеллект»

ВВЕДЕНИЕ

Тест Тьюринга — это способ определения способности машины производить интеллектуальные действия, неотличимые от действий человека. В современном мире тесты Тьюринга стали одним из ключевых инструментов для определения искусственного интеллекта. Эти тесты позволяют определить, насколько хорошо компьютер может имитировать разговор человека. Для проведения теста Тьюринга используется программное обеспечение, которое имитирует человеческое поведение и должно убедить эксперта в том, что он общается с живым человеком. Результаты проведения тестов могут быть использованы для разработки и улучшения алгоритмов искусственного интеллекта. Это позволяет разработчикам алгоритмов искусственного интеллекта улучшать свои продукты и технологии.

Примерами таких алгоритмов являются алгоритмы обработки естественных языков (Natural Language Processing — NLP). В общих словах — это совокупность методов и техник, которые позволяют компьютерам анализировать, понимать и генерировать естественный язык. NLP используется в ряде приложений, включая автоматический перевод, распознавание речи и анализ текста. Анализ результатов тестирования поможет в будущем улучшить данные модели, позволяя избегать различные грамматические, орфографические и смысловые ошибки.

Модели от команды OpenAI и многие другие играют важную роль в развитии искусственного интеллекта. GPT-3 неплохо справляется с созданием художественной литературы, поэзии, пресс-релизов, кода, музыки, шуток, технических руководств и новостных статей. Возможно, как предполагает Чалмерс (2020, Other Internet Resources), GPT-3 «предлагает потенциальный бездумный путь к общему искусственному интеллекту». Но, конечно, GPT-3 даже не близок к прохождению теста Тьюринга: на глобальном уровне — учитывая значения нескольких предложений, абзацев или двустороннего диалога — становится очевидным, что GPT-3 не понимает, о чем говорит. У него нет здравого смысла (Common Sense) или способности отслеживать объекты во время обсуждения. Разработанная база данных для хранения и обработки результатов тестов Тьюринга — это хороший инструмент для создания более развитых систем искусственного интеллекта.

Цель данной работы — разработать базу данных для хранения результатов проведения тестов Тьюринга. База данных должна содержать информацию о тестируемых программах, экспертах, результатах проведения тестов и другие данные, необходимые для анализа результатов.

Чтобы достигнуть поставленной цели, требуется решить следующие задачи:

- Определить структуру базы данных и ее таблиц.
- Разработать модели данных для каждой таблицы.
- Написать запросы для вставки, обновления и удаления данных в таблицах.
- Реализовать функционал для получения результатов проведенных тестов, с возможностью фильтрации и сортировки по различным полям.
- Разработать интерфейс пользователя для удобного использования базы данных.

1 Аналитический раздел

В данном разделе описана структура теста Тьюринга. Представлен анализ способов хранения данных и систем управления базами данных, оптимальных для решения поставленной задачи. Описаны проблемы кэшированных данных и представлены методы их решения.

1.1 Формализация задачи

Тест Тьюринга — это метод оценки способности машины производить интеллектуальные действия, сравнивая ее поведение с поведением человека в решении задач. Тест заключается в том, что человек задает вопросы человеку и компьютеру, а затем пытается определить, от кого пришли ответы. Если компьютер может убедительно имитировать поведение человека, то он считается способным производить интеллектуальные действия.

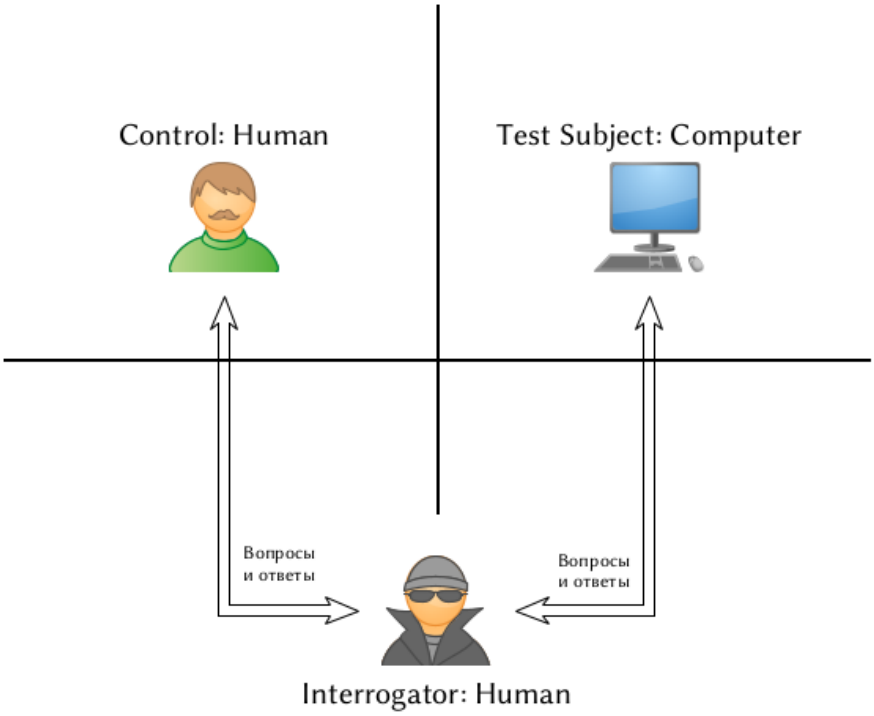

Рис. 1: Тест Тьюринга

Тьюринг в своей работе [3] описывает следующий вид игры. Предположим, что у нас есть человек, машина и эксперт. Эксперт находится в комнате,

отделенной от другого человека и машины. Цель игры состоит в том, чтобы эксперт определил, кто из двух является человеком, а кто машиной. Эксперт знает человека и машину по меткам «X» и «Y» — но, по крайней мере в начале игры, не знает, кто из них человек и кто – машина — и в конце игры он должен сказать либо «X — это человек, а Y — машина», либо «X — это машина, а Y — человек». Эксперту разрешается задавать человеку и машине вопросы следующего вида: «Скажите, пожалуйста, X, играет ли X в шахматы?» Кто бы из машины и другого человека ни был X, он должен отвечать на вопросы, адресованные X. Цель машины состоит в том, чтобы попытаться заставить эксперта ошибочно заключить, что машина — это другой человек; цель другого человека состоит в том, чтобы попытаться помочь эксперту правильно идентифицировать машину. [4]

Следует отметить, что во времена Тьюринга, было ограничение, что ответы поступали через ограниченные врменные рамки, поскольку время ответа компьютера было гораздо больше, чем у человека. Сегодня это ограничение сохраняется, однако из-за обратного: реакция компьютера быстрее, чем реакция человека.

1.2 Способы хранения данных

Для решения задачи теста Тьюринга необходимо хранить данные о человеке, машине и эксперте. Также необходимо хранить данные о заданных вопросах и полученных ответах. Хранить данные, на какие вопросы были даны какие ответы. Поскольку в конце теста выносится вердикт о том, является ли отвечающий машиной или человеком, необходимо также хранить какие ответы были даны в каком порядке и каким актором. Эти данные должны быть доступны для обработки и сравнения в процессе игры.

Один из способов хранения данных — это использование реляционных баз данных. В этом случае можно создать таблицы для каждого объекта (люди, машины и эксперты, вопросы и ответы, и т.д.) и связать их отношениями. Например, таблицы «Person», «Machine» и «Interrogator» могут быть связаны через внешние ключи. Это является надежным и проверенным способом хранения данных.

Однако более эффективный способ хранения данных — это использова-

ние графовых баз данных (GDB), таких как SurrealDB или Neo4j. В этом случае каждый объект может быть представлен узлом графа, а отношения между объектами — ребрами графа. 2

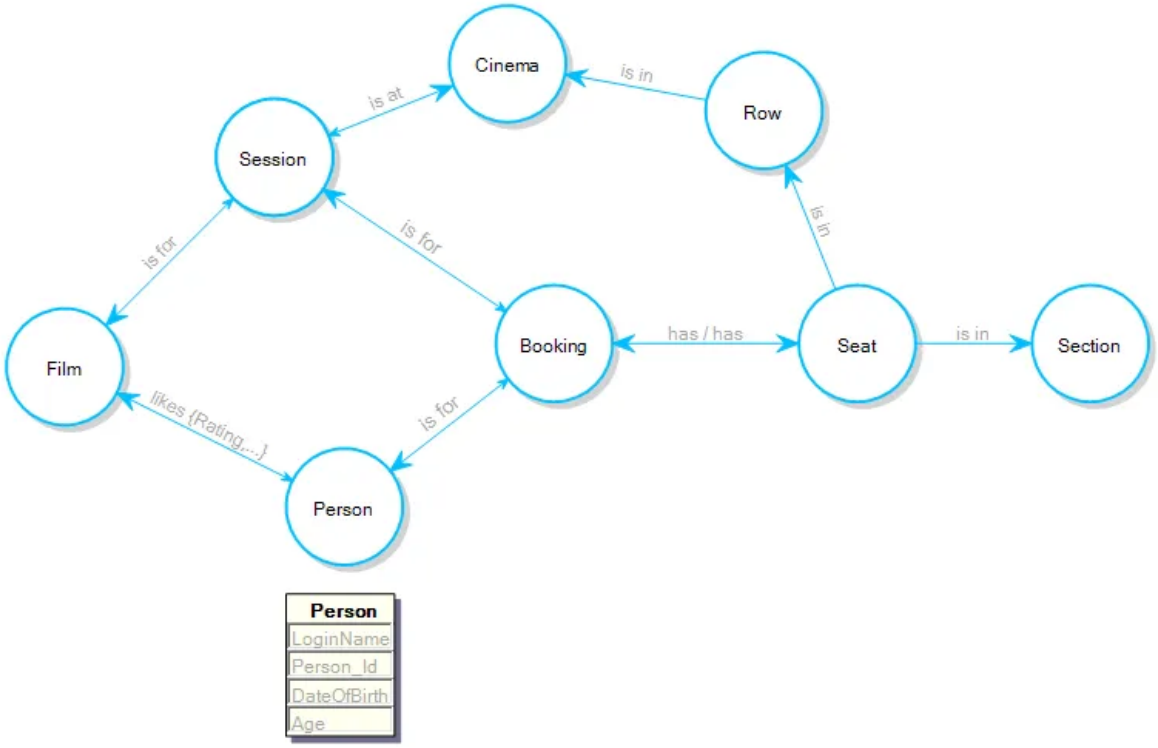

Рис. 2: Представление базы данных в виде графа

Ключевым понятием системы является граф (или ребро, или взаимосвязь). Граф связывает элементы данных в хранилище с набором узлов и ребер, причем ребра представляют отношения между узлами. Отношения позволяют напрямую связывать данные в хранилище и во многих случаях извлекать их с помощью одной операции. Базы данных графов удерживают отношения между данными в качестве приоритета. Запрашивать отношения быстро, потому что они постоянно хранятся в базе данных. Отношения можно интуитивно визуализировать с помощью баз данных графов, что делает их полезными для сильно взаимосвязанных данных. [5]. Поскольку графовая модель данных более естественным образом отображает связи между объектами, это делает ее более подходящей для задач, связанных с анализом связей и отношений между данными. В графовых базах данных нет необходимости использовать сложные JOIN-запросы, что может существенно упростить запросы к данным.

Графовые базы данных также обеспечивают быстрый доступ к данным по отношениям, что делает их эффективными при работе с глубоко связан-

ными данными. Они также позволяют легко добавлять новые данные в граф без необходимости изменения схемы базы данных. Однако, реляционные базы данных обладают более высокой надежностью и могут обеспечивать лучшую производительность при выполнении сложных запросов, особенно если используются правильно настроенные индексы.

Таким образом, выбор между реляционными базами данных и графовыми зависит от конкретных требований проекта. В контексте данной работы, графовая модель подходит больше, чем реляционная, потому что тест Тьюринга включает в себя множество связей между объектами (человек, машина, эксперт, вопросы и ответы и т.д.), которые можно представить в виде графа.

Графовая модель также становится еще более привлекательной, если вспомнить какая идея была обозначена в начале данной работы: создание инструмента для улучшения искусственного интеллекта. В большинстве случаев ИИ работает не со стандартными «табличными» данными, а данными, представленных в виде графа. Таким образом, схожая структура данных внутри СУБД поможет разработать более гибкую и быстродействующую систему при меньших затраченных ресурсах

На рис. 3 можно увидеть результаты сравнения 3 различных баз данных: реляционной (PostgreSQL), графовой (Neo4j) и мультимодельной (ArangoDB).

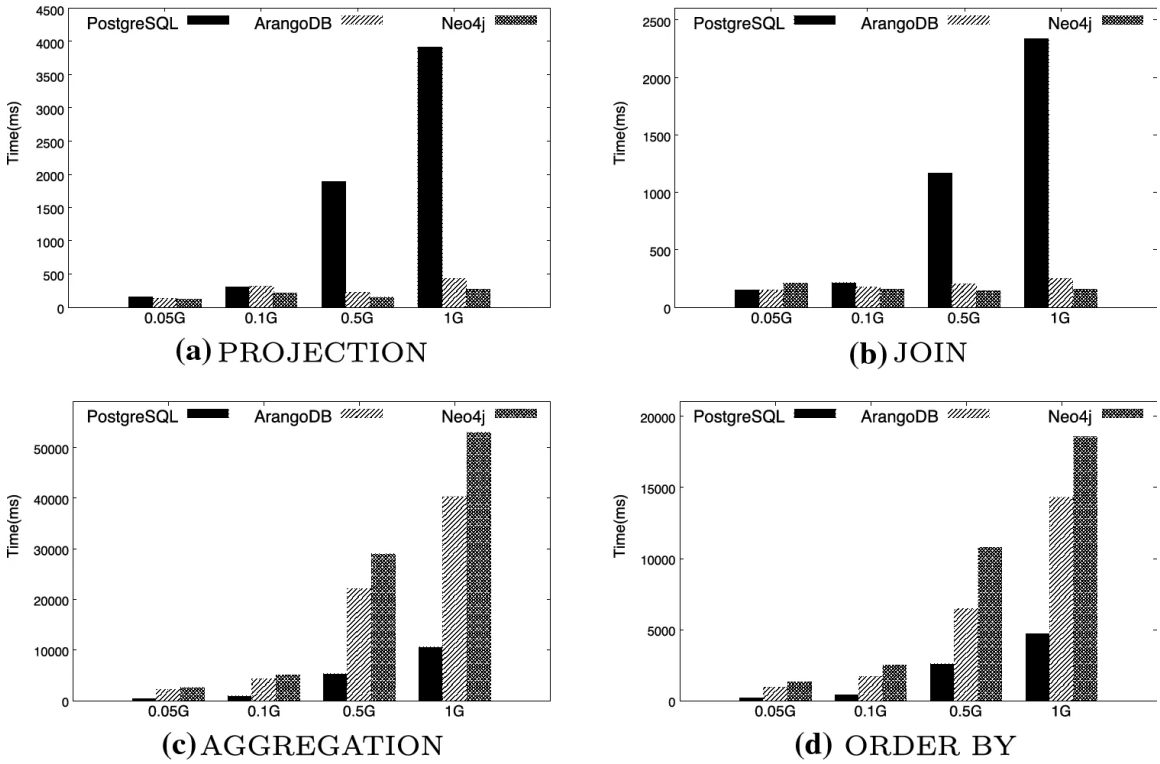

Рис. 3: Сравнение времени работы различных баз данных над атомарными операциями.

1.3 Системы управления базами данных

Для выбора системы управления базами данных необходимо учитывать требования к производительности и масштабируемости приложения.

Реляционные базы данных имеют высокую надежность и поддерживают ACID-свойства транзакций. Они также обеспечивают хорошую производительность при выполнении сложных запросов. Однако, они требуют дополнительного управления индексами и ключевыми полями.

Графовые базы данных обеспечивают высокую производительность при работе с глубоко связанными данными. Граф связывает элементы данных в хранилище с набором узлов и ребер, причем сами ребра представляют отношения между узлами. Отношения позволяют напрямую связывать данные в хранилище и во многих случаях извлекать их с помощью одной операции. Отношения между данными в подобных системах имеют приоритет над самими данными, поэтому запрос по отношениям является крайне быстрой операцией, поскольку они постоянно хранятся в базе данных. Также отношения можно интуитивно визуализировать с помощью баз данных графов, что делает их полезными для сильно взаимосвязанных данных.

1.3.1 SurrealDB

SurrealDB — мультимодельная NewSQL база данных, которая работает в режиме полной схемы или без схемы, с таблицами, ссылками на записи между документами (без JOIN) и функциями моделирования базы данных на основе графов.

Благодаря использованию SurrealDB особых методов сегментирования и репликации, становится возможным повысить производительность за счет распределения нагрузки между несколькими компьютерами.

Также особая архитектура базы данных позволяет работать как in-memory, on-disk или как распределенная база данных, используя TiKV.

Поскольку SurrealDB — мультимодельная база данных, становится также возможным классические реляционные методики проектирования баз данных, что повышает гибкость итоговой системы.

1.3.2 Neo4j

Neo4j — это графовая база данных, которая позволяет хранить, управлять и анализировать связанные данные. Она была разработана с учетом графовой модели данных, в которой данные представлены в виде узлов (вершин) и связей (ребер).

Одним из преимуществ Neo4j является то, что она позволяет эффективно моделировать и анализировать сложные связи между данными, такие как социальные сети, географические карты и сети предприятий. Это делает ее очень полезной для приложений, которые требуют быстрого доступа к сложным данным и быстрой обработки запросов.

Однако поскольку Neo4j — исключительно графовая база данных, хранение и получение данных без каких-либо связей друг с другом может вызвать проблемы с производительностью, вне зависимости от размера запроса.

1.4 Выбор СУБД для решения задачи

Для решения задачи теста Тьюринга необходимо выбрать графовую базу данных, поскольку графовая модель данных более естественным образом отображает связи между объектами.

Среди графовых баз данных можно выделить две наиболее подходящие системы: SurrealDB и Neo4j. Обе СУБД обеспечивают быстрый доступ к данным по отношениям, что делает их эффективными при работе с глубоко связанными данными.

Однако SurrealDB имеет дополнительные преимущества перед Neo4j. Она является мультимодельной базой данных, что позволяет эффективное хранение и получение несвязанных данных, где Neo4j может испытывать определенные проблемы.

1.5 Проблемы кэширования данных и их решение

Одной из проблем, связанных с хранением и обработкой данных в базах данных, является производительность. Возможны ситуации, когда приложение должно быстро получать данные из базы данных, но поиск этих данных

может занять много времени. Кэширование может помочь решить эту проблему.

Кэширование — это метод хранения данных в памяти для быстрого доступа к ним. Кэш представляет собой временное хранилище данных, которые часто запрашиваются из базы данных. Если данные уже находятся в кэше, приложение может получить доступ к ним намного быстрее, чем если они были запрошены непосредственно из базы данных.

Для кэширования данных можно использовать NoSQL in-memory базы данных. Такие базы данных хранят данные в оперативной памяти, что обеспечивает более быстрый доступ к данным. Примерами таких баз данных могут считаться Redis и Tarantool. Однако использование кэширования может столкнуться со следующими проблемами:

1. Неконсистентность данных. Если данные изменились в базе данных, но не обновлены в кэше, то будет получен неверный результат. Решением этой проблемы является использование механизма инвалидации кэша, который обновляет данные в кэше при изменении в базе данных.

2. Ограничение объема памяти. Кэш может занять значительное количество оперативной памяти, что может привести к ограничениям по объему хранимых данных. Решением этой проблемы может быть использование LRU-алгоритма (Least Recently Used), который удаляет менее используемые данные из кэша для освобождения памяти.

3. Потеря данных при сбое системы. Если происходит сбой системы или выключение сервера, то данные в кэше могут быть потеряны. Решением этой проблемы может быть использование репликации и бекапов для сохранения данных.

4. Производительность записи в базу данных. Использование кэша может повысить производительность чтения данных, но при записи данных в базу производительность может снижаться из-за необходимости обновления как в базе данных, так и в кэше. Решением этой проблемы является оптимизация процесса записи данных и использование асинхронных запросов.

1.6 Обзор in-memory NoSQL СУБД

1.6.1 Tarantool

Tarantool [6] — это платформа in-memory вычислений с гибкой схемой хранения данных для эффективного создания высоконагруженных приложений. Включает себя базу данных и сервер приложений на языке программирования Lua [7].

Tarantool обладает высокой скоростью работы по сравнению с традиционными СУБД. При этом, в рассматриваемой платформе для транзакций реализованы свойства ACID, репликация master-slave [8] и master-master [9], как и в традиционных СУБД.

Для хранения данных используется кортежи (англ. tuple) данных. Кортеж – это массив не типизированных данных. Кортежи объединяются в спейсы (англ. space), аналоги таблицы из реляционной модели хранения данных. Спейс – коллекция кортежей, кортеж – коллекция полей.

В Tarantool реализован механизм «снимков» текущего состояния хранилища и журналирования всех операций, что позволяет восстановить состояние базы данных после ее перезагрузки.

1.6.2 Redis

Redis [10] — резидентная система управлениями базами данных класса NoSQL с открытым исходным кодом. Основной структурой данных, с которой работает Redis является структура типа «ключ-значение». Данная СУБД используется как для хранения данных, так и для реализации кэшей и брокеров сообщений.

Redis хранит данные в оперативной памяти и снабжена механизмом «снимков» и журналирования, что обеспечивает постоянное хранение данных. Предоставляются операции для реализации механизма обмена сообщениями в шаблоне «издатель-подписчик»: с его помощью приложения могут создавать программные каналы, подписываться на них и помещать в эти каналы сообщения, которые будут получены всеми подписчиками. Существует поддержка репликации данных типа master-slave, транзакций и пакетной обработки ко-

манд.

Все данные Redis хранит в виде словаря, в котором ключи связаны со своими значениями. Ключевое отличие Redis от других хранилищ данных заключается в том, что значения этих ключей не ограничиваются строками. Поддерживаются следующие абстрактные типы данных:

- строки;
- списки;
- множества;
- хеш-таблицы;
- упорядоченные множества.

Тип данных значения определяет, какие операции доступные для него; поддерживаются высокоуровневые операции: например, объединение, разность или сортировка наборов.

1.7 Выбор СУБД для кэширования данных

Для кэширования данных можно использовать любую in-memory NoSQL базу данных. Обе системы, Tarantool и Redis, обеспечивают быстрый доступ к данным в памяти и имеют механизмы инвалидации кэша.

Однако, Tarantool обладает дополнительными преимуществами перед Redis. Она предоставляет возможность хранения данных в гибком формате кортежей, что позволяет более эффективно организовывать данные в базе данных. Также в Tarantool реализованы свойства ACID для транзакций и механизм репликации данных, что делает ее более надежной для использования в высоконагруженных приложениях.

Поэтому для решения задачи кэширования данных лучше выбрать Tarantool.

Вывод

В данном разделе были рассмотрены различные типы баз данных: реляционные и графовые, а также обзор in-memory NoSQL СУБД. Было выявлено,

что для решения задачи теста Тьюринга наиболее подходящей является графовая база данных, а конкретно SurrealDB. Для кэширования данных можно использовать любую in-memory NoSQL базу данных, но наилучшим выбором для этой задачи является Tarantool.

2 Конструкторский раздел

В данном разделе представлены этапы проектирования выделенных в предыдущем разделе баз данных, нужных для решения задачи

2.1 Проектирование отношений сущностей

На рисунке представлена схема сущностей, необходимых для реализации конечного приложения

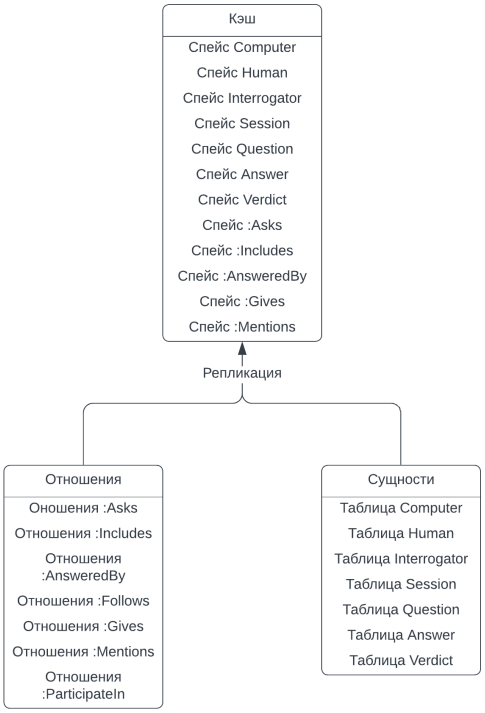

Рис. 4: Схема сущностей приложения.

2.2 Проектирование базы данных для хранения Тестов Тьюринга

База данных для хранения Тестов Тьюринга будет реализована с использованием СУБД SurrealDB. В базе данных будет существовать 7 сущностей и

7 типов отношений. ER-диаграма сущностей этой базы данных представлена на рисунке.

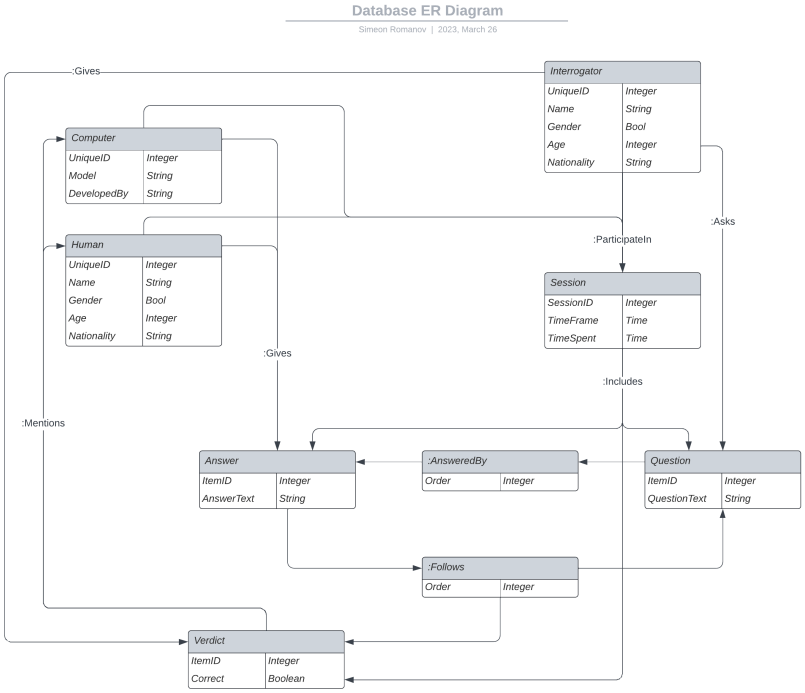

Рис. 5: Диаграмма сущностей и отношений базы данных.

Поля вершины Interrogator:

1. UniqueID — уникальный идентификатор, присваемый сущности-субъекту; будет использоваться чтобы однозначно идентифицировать сущности, способные принимать решения в системе.
2. Name — имя «дознавателя».
3. Gender — пол «дознавателя».
4. Age — возраст «дознавателя».
5. Nationality — национальность «дознавателя»;

Поля вершины Computer:

1. UniqueID — уникальный идентификатор, присваемый сущности-субъекту; будет использоваться чтобы однозначно идентифицировать сущности, способные принимать решения в системе.

2. Model — Модель ИИ, проходившая тест
3. DevelopedBy — Разработчики указанной ИИ

Поля вершины Human:

1. UniqueID — уникальный идентификатор, присваемый сущности-субъекту; будет использоваться чтобы однозначно идентифицировать сущности, способные принимать решения в системе;
2. Name — имя человека;
3. Gender — пол человека.
4. Age — возраст человека.
5. Nationality — национальность человека;

Поля вершины Answer:

1. ItemID — уникальный идентификатор, присваемый сущности-объекту; будет использоваться чтобы однозначно идентифицировать сущности, которые являются производными от объектов.
2. AnswerText — Текст Ответа.

Поля вершины Question:

1. ItemID — уникальный идентификатор, присваемый сущности-объекту; будет использоваться чтобы однозначно идентифицировать сущности, которые являются производными от объектов.
2. QuestionText — Текст Вопроса.

Поля вершины Verdict:

1. ItemID — уникальный идентификатор, присваемый сущности-объекту; будет использоваться чтобы однозначно идентифицировать сущности, которые являются производными от объектов.
2. Correct — Верен ли вердикт, выданный дознаватель.

Поля Вершины Session:

1. SessionID — уникальный идентификатор, присваемый сессии.
2. TimeFrame — период времени, отведенный на ответ на вопрос
3. TimeSpent — продолжительность сессии.

Поля ребер :AnsweredBy и :Follows:

1. Order — порядковый номер вопроса/ответа/вердикта в системе

2.3 Проектирование базы данных кэширования

База данных кэширования будет реализована с помощью использования СУБД Tarantool. В базе данных будут полностью продублированны таблицы (в виде спейсов) из хранилища тестов Тюринга. Первичным ключом будет являться поле с уникальным идентификатором этих таблиц (id). При запросе данных у приложения, будет проводиться проверка, присутствует ли запись в кэше. Если запись присутствует, запрос к базе данных производиться не будет и будут возвращены данные из кэша. В противном случае, будет произведен запрос к базе данных. Все спейсы будут созданы на основе движка memtx, хранящего все данные в оперативной памяти. Персистентность данных будет обеспечивается при помощи ведения журнала транзакция и системы «снимков» текущего состояния кэша. Эти технологии помогут решить проблему «холодного»старта базы данных кэширования.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] What is natural language processing? [Электронный ресурс]. Режим доступа: https://www.ibm.com/topics/natural-language-processing.

[2] Bourbakis Nikolaos G. Artificial Intelligence and Automation. World Scientific, 1998.

[3] TURING A. M. I.—COMPUTING MACHINERY AND INTELLIGENCE // Mind. 1950. 10. Т. LIX, № 236. С. 433–460. URL: https://doi.org/10.1093/mind/LIX.236.433.

[4] Oppy Graham, Dowe David. The Turing Test // The Stanford Encyclopedia of Philosophy / под ред. Edward N. Zalta. Metaphysics Research Lab, Stanford University, 2021.

[5] Yoon Byoung-Ha, Kim Seon-Kyu, Kim Seon-Young. Use of graph database for the integration of heterogeneous biological data // Genomics & Informatics. 2017. Mar. Т. 15, № 1. с. 19–27.

[6] Tarantool – Платформа In-memory вычислений [Электронный ресурс]. Режим доступа: https://www.tarantool.io/ru/ (дата обращения: 03.04.2023).

[7] The Programming Language Lua [Электронный ресурс]. Режим доступа: http://www.lua.org/ (дата обращения: 07.04.2023).

[8] Tech Confronts Its Use of the Labels «Master» and «Slave» [Электронный ресурс]. Режим доступа: https://www.wired.com/story/tech-confronts-use-labels-master-slave/ (дата обращения: 05.04.2023).

[9] How To Set Up MySQL Master-Master Replication [Электронный ресурс]. Режим доступа: https://www.digitalocean.com/community/tutorials/how-to-set-up-mysql-master-master-replication (дата обращения: 05.04.2023).

[10] Redis is an open source (BSD licensed), in-memory data structure store, used as a database, cache, and message broker [Электронный ресурс]. Режим доступа: `https://redis.io/` (дата обращения: 05.04.2023).