[Слайд 1] Здравствуйте, уважаемые председатель и члены государственной комиссии! Тема выпускной квалификационной работы — «Веб-сервис для изучения SQL с применением больших языковых моделей».

[Слайд 2] SQL — это язык запросов к базам данных, один из базовых навыков в анализе данных. Осваивают его прежде всего на практике — решая задачи на реальных базах. Но такие задачи преподаватель составляет вручную: сам придумывает базу данных, формулировки вопросов и проверяет ответы. Это долго и ограничивает разнообразие заданий.

Готовые онлайн-тренажёры, например HackerRank или LeetCode, эту проблему не решают: преподаватель не может загрузить туда свою базу данных, не может получить задания автоматически и не видит, как справилась его группа.

Поэтому цель работы — сделать сервис, в котором преподаватель загружает свою базу данных и получает по ней готовые задания с автоматической проверкой, а составлять их помогает llm.

[Слайд 3] Сервис устроен как обычное веб-приложение: преподаватель и студент работают через браузер. Интерфейс сделан на фреймворке Vue, серверная часть — на языке Python. Для хранения данных — самой базы, входа по логину и загруженных файлов — использована готовая облачная платформа Supabase, которая всё это берёт на себя.

Ключевое решение — задания генерирует llm, которая запускается локально, на собственном компьютере, а не в облаке вроде ChatGPT. Для учебного заведения это важно: данные студентов и загруженные базы не уходят за пределы организации.

[Слайд 4] Ключевая инженерная проблема здесь — заставить модель понимать, что для человека значит «уровень сложности 3»: сама по себе цифра ей ничего не говорит. Поэтому пять уровней были явно определены через конкретные конструкции SQL — от простой выборки без фильтрации до оконных функций и рекурсивных запросов — и это определение зашито прямо в промпт.

Чтобы проверить, какая модель и какой способ построения промпта дают лучший результат, был поставлен сравнительный эксперимент: три локальные модели от 9 до 20 миллиардов параметров, три стратегии составления промпта — от минимальной до промпта с явным описанием каждого уровня и примерами — и три предметные базы данных, по всем пяти уровням сложности. В сумме — 135

сгенерированных заданий, и различия между вариантами проверялись статистически, а не «на глаз», чтобы не принять случайный разброс за закономерность.

Каждое задание оценивалось двумя разными способами. Автоматически проверялось, исполняется ли SQL-запрос без ошибок и возвращает ли он данные. А качество самой формулировки оценивала отдельная, более сильная языковая модель в роли независимого эксперта — и, что важно, эта модель-судья намеренно не входила в число трёх сравниваемых моделей, чтобы исключить необъективность модели при оценке самой себя. Судья выставлял оценки от 1 до 5 по четырём критериям: понятен ли вопрос студенту, действительно ли предложенный SQL-запрос отвечает на этот вопрос, соответствует ли фактическая сложность запроса запрошенному уровню и есть ли в задании учебная ценность.

Получилось два результата. Первый — неожиданный: самая компактная модель, на 9 миллиардов параметров, обошла обе более крупные, включая вдвое большую, на 20 миллиардов. Лишние параметры не дали преимущества для этой задачи, а лёгкая модель к тому же дешевле в эксплуатации и быстрее отвечает. Второй результат — про промптинг: когда в промпт добавили явное описание каждого уровня сложности, соответствие фактической сложности запрошенной выросло заметно и стабильно для всех моделей — то есть модели действительно не хватало не мощности, а самого определения уровня.

[Слайд 5] Далее — о том, как работает сервис. Вот главная страница со списком тестов. Преподаватель создаёт тест и загружает к нему свой файл базы данных.

[Слайд 6] Дальше он наполняет тест вопросами. Для каждого вопроса есть два пути: написать формулировку и правильный SQL-запрос вручную — или нажать «Сгенерировать», выбрать ползунком уровень сложности, и llm сама придумает вопрос и правильный ответ к нему по структуре загруженной базы.

[Слайд 7] Так выглядит страница теста со стороны преподавателя: список вопросов с их уровнем сложности, а внизу — статистика прохождения по группе: кто из студентов сколько заданий решил верно, в том числе в виде наглядной диаграммы.

[Слайд 8] А это интерфейс студента. Слева — список вопросов, в центре — редактор для SQL-запроса с подсветкой синтаксиса, справа — схема базы данных в виде наглядной диаграммы. Студент пишет запрос, нажимает «Выполнить» и сразу видит результат. При отправке сервис проверяет ответ автоматически —

сравнивает результат запроса студента с эталонным. И про безопасность: каждый запрос выполняется на одноразовой копии базы, поэтому даже если студент попробует удалить данные, пострадает только копия, а оригинал останется нетронутым.

[Слайд 9] Работа сервиса проверена по десяти сквозным сценариям — от регистрации до сохранения результата, для роли и преподавателя, и студента; отдельно проверена защита от опасных запросов. Сервис не остался прототипом: он упакован для развёртывания, работает по защищённому протоколу и уже доступен в интернете — по адресу queryquest.serveremey.ru.

[Слайд 10] В итоге создан рабочий сервис: преподаватель загружает базу данных и получает по ней задания — вручную или с помощью llm, — а студент решает их с автоматической проверкой и наглядной схемой базы. Экспериментально подобраны модель и способ генерации заданий. Все задачи работы выполнены, сервис протестирован и развёрнут. В дальнейшем его можно расширить — другими типами заданий и связкой с учебными платформами.

[Слайд 11] Доклад окончен. Спасибо за внимание!

Ответы на возможные вопросы комиссии

1. Почему LLM локальная, а не облачная — не жертвуете ли качеством ради конфиденциальности? Да, это осознанный компромисс: небольшая локальная модель уступает топовым облачным по общему уровню, но данные студентов и загруженная база не покидают организацию. Потеря частично компенсируется тем, что явное описание уровня сложности в промпте, как показал эксперимент, заметно повышает точность генерации даже для компактной модели.

2. Почему Supabase, а не собственный бэкенд для авторизации и хранения? Supabase — open-source платформа и может быть развёрнута на собственной инфраструктуре учебного заведения, то есть требование «данные не уходят наружу» не нарушается. Использование готового решения для авторизации и хранения файлов вместо написания своего — сознательный выбор в пользу простоты (KISS): не изобретать то, что уже есть готовым и проверенным.

3. Какие требования к железу для локальной LLM? Модели уровня 9–20 млрд параметров запускаются на одной потребительской видеокарте (12–24 ГБ видеопамяти, например RTX 3090/4090). Победившая в эксперименте модель — самая маленькая, 9 млрд параметров — требует меньше всего ресурсов, что является дополнительным практическим плюсом результата, а не только академическим.

4. Почему не сравнивали с более крупными моделями (70B и выше)? Цель эксперимента — не найти абсолютно лучшую модель, а найти минимально достаточную для локального инференса на доступном железе. Модели 70B+ требуют серверных GPU-кластеров, что противоречит самой идее локального развёртывания в учебном заведении; 20B было выбрано как верхняя граница того, что практично на одной видеокарте.

5. Кто проверял, что LLM-судья вообще оценивает адекватно? Судья — отдельная модель (deepseek-v4-pro), не входящая в тестируемую тройку, что исключает self-evaluation bias. Согласованность её оценок дополнительно проверена вручную на подвыборке из 10–15 заданий. Это признанное ограничение методологии: полная статистическая валидация судьи на большой выборке — предмет дальнейшей работы.

6. Насколько велика выборка эксперимента? Полный прогон — 135 заданий: 3 модели × 3 стратегии промптинга × 3 базы данных × 5 уровней слож-

ности, по одному повтору на комбинацию. Для анализа использованы непараметрические блочные тесты (Friedman, Wilcoxon с поправкой Холма), подходящие для порядковой шкалы и малой выборки. Ограничение N=1 на ячейку зафиксировано явно; для уровней 4–5, где эффект тоньше, дополнительные повторы — следующий шаг.

7. Как устроена «одноразовая копия» базы технически и не тормозит ли это? Здесь формулировка слайда обновилась: файл базы теста скачивается из Supabase Storage один раз и кэшируется на диске, а каждое SQL-соединение к нему открывается в режиме только для чтения (`mode=ro`). Изменяющий запрос отклоняется на уровне самого SQLite раньше, чем успеет что-то испортить, — это даже надёжнее и дешевле, чем создавать новую копию файла на каждый запрос.

8. Кто проверяет корректность самого эталонного SQL-ответа, сгенерированного LLM? Автоматически: до публикации вопроса проверяется, что эталонный запрос выполняется без ошибки и возвращает данные. Кроме того, любой сгенерированный вопрос и SQL-ответ преподаватель видит и может отредактировать перед тем, как опубликовать тест студентам — человек остаётся последней инстанцией.

9. Что мешает студенту положить сервис тяжёлым или бесконечным запросом? Сейчас — только то, что запрос выполняется в отдельном потоке и не блокирует остальных пользователей, плюс режим «только чтение» исключает порчу данных. Жёсткого лимита по времени выполнения или объёму результата пока нет — это открытое ограничение, естественное развитие: таймаут на запрос или ограничение размера результата.

10. Чем это отличается от того, что преподаватель просто попросит ChatGPT сгенерировать задания вручную? Сервис избавляет от ручного копирования схемы в чат и разбора ответа: он сам подставляет схему загруженной базы в промпт с нужным уровнем сложности, сразу проверяет, что сгенерированный SQL исполним и возвращает данные, и автоматически проверяет ответы студентов по этому эталону — то есть автоматизирует весь цикл, а не только генерацию текста вопроса.

11. Как проверяется правильность ответа, если для вопроса возможно несколько разных, но верных SQL-запросов? Сравнивается результат выполнения запроса студента с результатом эталонного запроса (как есть, без изменений). Это значит, что разные по тексту, но эквивалентные запросы проходят

проверку, но сравнение сейчас чувствительно к порядку строк и столбцов — если формулировка вопроса требует сортировку, это нужно закладывать явно. Более мягкое сравнение (без учёта порядка) — известное направление для доработки.

12. Почему именно пять уровней сложности? Таксономия построена не произвольно, а по конкретным SQL-конструкциям: простая выборка, фильтрация, соединение/агрегация, составные запросы, продвинутые конструкции (оконные функции, CTE). Это естественная прогрессия, соответствующая типичному порядку изучения SQL, а не абстрактная шкала «от 1 до 5».

13. Тестирование по десяти сценариям — это автоматические тесты или ручная проверка? Ручное тестирование по заранее описанным сценариям, не автоматизированный набор регрессионных тестов. Для дипломного прототипа такого объёма это разумный масштаб; перевод сценариев в автоматические end-to-end тесты — следующий шаг перед промышленной эксплуатацией.

14. В чём измеримая выгода для преподавателя по времени? Отдельного количественного замера в рамках работы не проводилось — это качественная, а не количественная оценка выгоды. Честная граница результата: показано, что генерация вопроса занимает секунды вместо ручного придумывания базы и вопросов, но замер трудозатрат преподавателя на реальных группах — предмет отдельного (уже прикладного, не инженерного) исследования.

15. Какие «другие типы заданий» имелись в виду и почему не сделаны сейчас? Например, задания на исправление ошибки в чужом SQL-запросе или на объяснение того, что делает запрос, — форматы, которые проверяют понимание, а не только написание кода. Они не реализованы в этой работе, потому что требуют отдельной логики проверки ответа (сравнение результата запроса здесь уже не подходит), а рамки дипломной работы ограничены одним, но полностью доведённым до продакшена сценарием.