ЛИДЕРЫ ЦИФРОВОЙ ТРАНСФОРМАЦИИ

ПРОЕКТ МЭРА МОСКВЫ

ДЕПАРТАМЕНТ ПРЕДПРИНИМАТЕЛЬСТВА И ИННОВАЦИОННОГО РАЗВИТИЯ ГОРОДА МОСКВЫ

РАЗВИТИЕ ЧЕЛОВЕЧЕСКОГО КАПИТАЛА

tech

ТЕХНИЧЕСКОЕ ЗАДАНИЕ

ЗАДАЧА 2

Система рекомендаций по оптимизации производительности Data Lakehouse

1. Актуальность задачи

Разработка невизуального сервиса на основе больших языковых моделей (LLM) для оптимизации структуры данных и SQL-запросов становится особенно актуальной в современных условиях из-за нескольких ключевых факторов:

- Экспоненциальный рост данных

Объемы данных: количество хранимых данных растет ежегодно на 25–40% (по прогнозам IDC). Это приводит к увеличению нагрузки на базы данных и необходимости постоянной оптимизации производительности.

Сложность запросов: современные приложения требуют выполнения сложных аналитических запросов (например, OLAP), которые часто включают множественные JOIN-операции, GROUP BY и агрегации. Это создает узкие места в производительности.

- Дефицит квалифицированных специалистов

Ручная оптимизация: традиционный подход к оптимизации БД требует глубоких знаний у DBA и разработчиков (анализ планов выполнения, настройка индексов, денормализация). Однако рынок испытывает нехватку таких специалистов.

Автоматизация: LLM могут автоматизировать рутинные задачи, предоставляя рекомендации по изменению структуры данных и запросов без прямого участия человека.

- Возможности больших языковых моделей

Анализ паттернов: LLM способны выявлять скрытые закономерности в структуре данных и запросах. Например, они могут обнаружить, что 90% запросов выполняют JOIN трех конкретных таблиц, и предложить их денормализацию.

Генерация рекомендаций: модели могут генерировать варианты оптимизации (например, партиционирование, изменение типов данных) на основе исторических данных и метрик производительности.

- Оптимизация затрат

Стоимость инфраструктуры: улучшение производительности позволяет сократить расходы на серверы, хранилища и лицензии СУБД. Например, оптимизация запросов может снизить время выполнения на 50%, что эквивалентно уменьшению числа серверов.

Экономия времени: автоматизация ускоряет внедрение изменений в структуру БД и запросы, что критично для Agile-процессов.

- Масштабируемость

Распределенные системы: в распределенных архитектурах (например, микросервисы) сложно координировать оптимизацию данных. LLM могут

анализировать данные в реальном времени и предлагать изменения для улучшения работы кластеров.

Облака: Облачные платформы (AWS, Azure) требуют автоматической настройки ресурсов. Такой сервис может интегрироваться с ними для динамической оптимизации.

- Современные требования к скорости

Онлайн-сервисы: пользователи ожидают мгновенного ответа. Даже 100-миллисекундная задержка может привести к потере клиентов.

Интерактивный анализ: BI-системы требуют быстрого выполнения сложных запросов к большим данным. Оптимизация структуры и запросов здесь критична.

2. Постановка задачи

Разработать невизуальный сервис с применением больших языковых моделей (LLM), который будет анализировать логическую модель данных, статистику данных, а также статистику и структуру SQL-запросов для выработки рекомендации по изменению структуры данных и запросов с целью оптимизации производительности.

Пример: база данных построена по принципам 3-й нормальной формы и 95% запросов содержат в себе слияние (JOIN) трех основных таблиц. Исходя из структуры, выглядит так, что денормализация этих трех таблиц позволит улучшить быстродействие, так как будет легче применить разбиение единой таблицы на блоки (партиционирование).

3. Требования к сервису

3.1. Функциональные требования

Сервис должен представлять собой REST API, который получает на вход задачу анализа структуры базы данных (далее БД). Такой анализ не может быть выполнен синхронно, поэтому в ответ ожидается некоторый номер задачи и статус готовности, который можно запрашивать асинхронно. По готовности задача должна иметь возможность получить результат.

Такие требования приводят к следующему набору REST API-запросов:

Запрос	Описание
POST https://<endpoint>/new	Запуск задачи (формат запроса ниже)
GET https://<endpoint>/status?<task_id>	Запрос статуса задачи
GET https://<endpoint>/getresult?<task_id>	Запрос на получение результатов.

3.2. Форматы запросов.

Запрос на запуск задачи.

Данные для запуска представляют собой три элемента:

- DDL создания таблиц. Скрипт разбивается на отдельные запросы для создания каждой отдельной таблицы.
- Набор запросов со статистикой по количеству вызовов каждого. Каждый запрос имеет идентификатор, который позволит в дальнейшем оценить изменение скорости работы.
- Строка подключения в формате jdbc с логином и паролем для оценки самих данных.

Формат запроса:

```
{
   "url":"jdbc://some-endpoint/database?login=xxx&password=yyyy",
   "ddl":[
      {
         "statement": "CREATE TABLE Table1.....",
      },
      {
         "statement": "CREATE TABLE SecondTable.....",
      },
    ],
   "queries": [
      {
         "queryid":"0197a0b2-2284-7af8-9012-fcb21e1a9785",
         "query":"SELECT a.ID, b.XXX....",
         "runquantity": 123
      },
      {
         "queryid":"c8ed3309-1acb-439a-b32b-f802ba41db3e",
         "query":"WITH (...",
         "runquantity":112233
      }
   ]
}
```

Ответ на такой запрос должен содержать идентификатор, по которому задача однозначно определяется сервисом. Пример ответа:

```
{
   "taskid":"6c12bd3f-80b1-4c0a-84ab-d3160d2e8f7a"
}
```

Запрос статуса задачи.

Ответом на такой запрос должно быть одно поле со статусом выполнения RUNNING, если запрос еще не закончен, DONE, если можно забирать результаты и FAILED, если что-то пошло не так, и запуск не удался. Предельное время ожидания ответа — 20 минут. После этого запрос считается неудавшимся.

Запрос на получение результатов.

В ответ на этот запрос ваш сервис должен вернуть ответ, содержащий следующую информацию:

- Новый набор DDL-запросов для модификации структуры таблиц.
- Набор запросов для миграции данных.
- Набор запросов с их идентификаторами, которые используют новую структуру таблиц.

Пример ответа:

```
{
   "ddl":[
      {
         "statement":"CREATE TABLE t1..."
      },
      {
         "statement":"CREATE TABLE T2..."
      }
   ],
   "migrations":[
      {
         "statement":"INSERT INTO T1 SELECT * FROM OldT1 LEFT JOIN ..."
      },
      {
         "statement":"INSERT INTO T2 SELECT..."
      }
   ],
   "queries":[
      {
         "queryid":"0197a0b2-2284-7af8-9012-fcb21e1a9785",
         "query":"WITH (..."
      },
      {
         "queryid":"c8ed3309-1acb-439a-b32b-f802ba41db3e",
         "query":"WITH (..."
      }
   ]
}
```

ВНИМАНИЕ!

Все команды работы с таблицами должны использовать полный путь к таблице в формате <каталог>.<схема>.<таблица>. В вашем ответе первой DDL командой должна идти команда создания новой схемы в этом же каталоге!

Пример:

CREATE SCHEMA data.NewSchema

Все SQL запросы, которые переносят данные в новую структуру также должны придерживаться этого правила полной идентификации таблиц, пример:

INSERT INTO catalog.myschema.h_authors

SELECT * FROM catalog.public.h_authors

Все запросы к новой структуре данных должны также указывать полный путь в новой схеме, пример:

SELECT a.Col1, a.Col2, b.Col4

FROM catalog.myschema.MyTable1 as a

JOIN catalog.myschema.MyTable2 as b on a.ID=b.ID

4. Возможный пользовательский путь

1. Осуществляется сбор информации по количеству запросов, среднему времени их выполнения и объема данных в хранилище.

2. С помощью сервиса генерируется предложение по изменению структуры БД и запросов.

3. Пользователь тестирует данное предложение и разрабатывает план миграции на его основе.

5. Целевая аудитория

Целевая аудитория такого LLM-сервиса по аудиту и оптимизации SQL и хранилищ данных включает специалистов, отвечающих за проектирование, эксплуатацию и эффективность аналитических систем. Вот основные группы:

1. Data Engineers

- Основная аудитория.
- Занимаются построением и поддержкой ETL/ELT-пайплайнов
- Оптимизируют структуру данных в Iceberg, S3, Spark
- Часто сталкиваются с медленными запросами и ростом стоимости
- Нуждаются в автоматизированной диагностике и конкретных рекомендациях

Зачем им сервис:

«Как быстро понять, почему пайплайн стал тормозить, и как исправить схему или запросы без ручного разбора логов?»

2. Data Architects

- Отвечают за общую архитектуру хранилища и модели данных
- Принимают решения по партиционированию, форматам хранения, распределению данных
- Хотят стандартизировать подходы и избежать технического долга

Зачем им сервис:

«Как проверить, соблюдаются ли best practices по моделированию в сотнях таблиц?»

3. Аналитики данных / BI-разработчики

- Пишут SQL для отчётов и дашбордов
- Не всегда понимают, как их запросы влияют на производительность
- Получают «медленный запрос» от коллег или систем мониторинга

Зачем им сервис:

«Как понять, почему мой запрос работает 10 минут, и как его исправить без глубоких знаний Spark или Iceberg?»

4. Администраторы данных / Data Platform Teams

- Управляют общей платформой аналитики (Trino, Spark, S3)
- Контролируют стоимость, нагрузку, SLA
- Ищут способы снизить потребление ресурсов

Зачем им сервис:

«Как автоматически находить "тяжёлые" запросы и "плохие" модели данных до того, как они сломают кластер?»

Основные пользователи:

- Data EngineersData
- ArchitectsData
- Platform Teams

Вторичные пользователи:

- Аналитики
- BI-разработчики

Сервис особенно ценен в компаниях с большими данными на базе data lakehouse (S3 + Iceberg + Trino/Spark), где рост объёмов приводит к росту сложности и стоимости.

6. Источники данных

На каждом из этапов организатор предоставляет структуру БД, запросы и статистику их выполнения (частота и время на выполнение каждого запроса), а

также ссылку на подключение к кластеру Trino для оценки состава и структуры данных.

На каждом из этапов предлагается 2 варианта структуры БД:

1. Звезда / снежинка
2. Широкая плоская таблица

Для каждого из вариантов будет предложен собственный набор запросов.

На каждом этапе проверки будет использована новая структура данных, ранее не известная сервису, и запросы к ней.

7. Требования к решению

Участники должны предоставить URL к REST-сервису, которому отправляются задания на оптимизацию. Устанавливается предельное время ожидания ответа, равное 15 минутам.

8. Образ финального результата

Специализированный REST-сервис на базе LLM-модели, который позволяет оптимизировать структуру БД и запросов к ней.

9.1. Порядок проверки результата.

Мероприятие проводится в два этапа. На предварительном этапе определяются 10 финалистов. 8 участников с наилучшими показателями. 1 команда, не вошедшая в топ-8 и показавшая наилучшее сокращение времени, требуемого для выполнения всех операций. 1 команда, не вошедшая в топ-8 и показавшая наилучший результат по сокращению объема ресурсов, требующихся для хранения данных.

9.2. Порядок проведения предварительной экспертизы

После получения корректного ответа от сервиса, система приступает к оценке времени работы запросов с новой структурой данных.

Проверка будет производиться на основе ответа вашего сервиса в следующем порядке:

- Последовательное выполнение запросов секции DDL. Запросы будут исполнены в том порядке, в котором они выдаются в ответе.
- Последовательное выполнение запросов секции migrations. Запросы должны перелить данные из оригинальных таблиц в новые. Запросы будут выполнены в том порядке, в котором они выдаются в ответе.
- Измерение времени выполнения запросов к данным. Сохранение оригинального queryid обязательно, без него такой запрос не будет проверен, что уменьшит итоговый балл!

9.3. Порядок вычисления итоговой оценки.

Мероприятие проводится в два этапа: отборочный и финальный.

В процессе оценки происходит измерение динамики изменения показателей:

- время, которое требуется для выполнения всех операций
- объем ресурсов, требующихся для хранения данных.

Каждый показатель имеет свой коэффициент значимости в итоговом зачете.

Для расчёта изменения времени, требующегося на выполнение всех операций, применяется формула:

$$x = \frac{\sum_{i=1}^{n} \frac{Q_i A_i}{T_i}}{\sum_{i=1}^{n} Q_i}$$

где
A — исходное время, затраченное на выполнение операции
T — время, затраченное на выполнение операции после изменения запроса
Q — частота применения операции
n — количество операций

При проверке устанавливается таймаут на выполнение запроса, равный 10 минутам. Если запрос не может быть выполнен из-за какой-либо ошибки, его время считается равным 10 минутам.

Для расчёта изменения требуемого объема ресурсов применяется формула:

$$y = \frac{S}{C}$$

где
S — объем ресурсов, требующийся системе, до выполнения оптимизации
C — объем ресурсов, требующийся системе, после выполнения оптимизации

Итоговый расчёт производится по формуле:

$$z = x\sqrt[3]{y}$$

где
x — результат изменения среднего времени, требующегося для выполнения всех операций
y — результат изменения среднего объема ресурсов, требующегося для выполнения всех операций

10. Требования к сдаче решений

1. Требования для промежуточной сдачи решения

1. Ссылка на endpoint (например, https://team1-solution.ru/api/predict) и доступ, если API закрыто: токен, ключ авторизации или тестовый логин/пароль.
2. Ссылка на репозиторий с исходным кодом (должен быть размещен на GitHub/GitLab или аналогичной платформе с доступом для организаторов)
3. Ссылка на презентацию
4. Контакт кого-либо из участников (телеграм), если endpoint не отвечает.

2. Требования для финальной сдачи решения

1. Ссылка на endpoint (например, https://team1-solution.ru/api/predict) и доступ, если API закрыто: токен, ключ авторизации или тестовый логин/пароль.
2. Ссылка на репозиторий с исходным кодом (должен быть размещен на GitHub/GitLab или аналогичной платформе с доступом для организаторов)
3. Ссылка на презентацию
4. Контакт кого-либо из участников (телеграм), если endpoint не отвечает.

11. Критерии оценки

1. Подход коллектива к решению задачи

- Ясность и логичность идеи (насколько команда четко понимает задачу и сформулировала стратегию ее решения)
- Оригинальность подхода (наличие нестандартных идей)
- Обоснованность архитектуры решения
- Выбор технологий и инструментов

2. Техническая проработка решения

- Качество кода
- Скорость работы решения (время выполнения предсказания при отправке через API)
- Корректность работы сервиса

3. Соответствие решения поставленной задачи

- Полнота выполнения требований ТЗ
- Согласованность результатов с этапами решения
- Качество описания решения

4. Эффективность решения в рамках поставленной задачи

- Описана в п. 9.2

5. Выступление коллектива на питч-сессии (только для финальной экспертизы)

- Четкое изложение идеи (команда ясно формулирует проблему, которую решает сервис, и ключевые преимущества своего подхода)
- Демонстрация работы сервиса
- Структурированность презентации
- Умение отвечать на вопросы