

Проверка качества данных

- Модуль интеллектуального управления и проверки качества данных (ML)
- Модуль интеллектуального управления и проверки качества данных (Фронт)
- Архитектура взаимодействия с моделью

Модуль интеллектуального управления и проверки качества данных (ML)

Необходимо создание и внедрение в редактор модели данных модуля проверки качества данных с использованием языковой модели нейронной сети для обработки текстовых сообщений от пользователя.

Концепт:

Языковая модель для обработки текстовых сообщений от пользователя. Языковая модель обрабатывает 2 вида запросов (и приводит к ним иные): задачи редактирования и задачи поиска. Превращает запрос от пользователя в SQL-код, который используется для создания демонстрационной таблицы или окна с ответом. Демонстрационная таблица хранится временно до тех пор, пока пользователь не подтвердит её использование, не отменит выбор или не выйдет из модели данных.

Скрипт загрузки Загрузить данные **Модели данных** Задания

Модель 1 **Модель 2** Модель 3 +

Измени формат данных для всех полей с датой на ДД/ММ/ГГГГ Применить Сохранить

Rate_detail 2 Предварительный просмотр

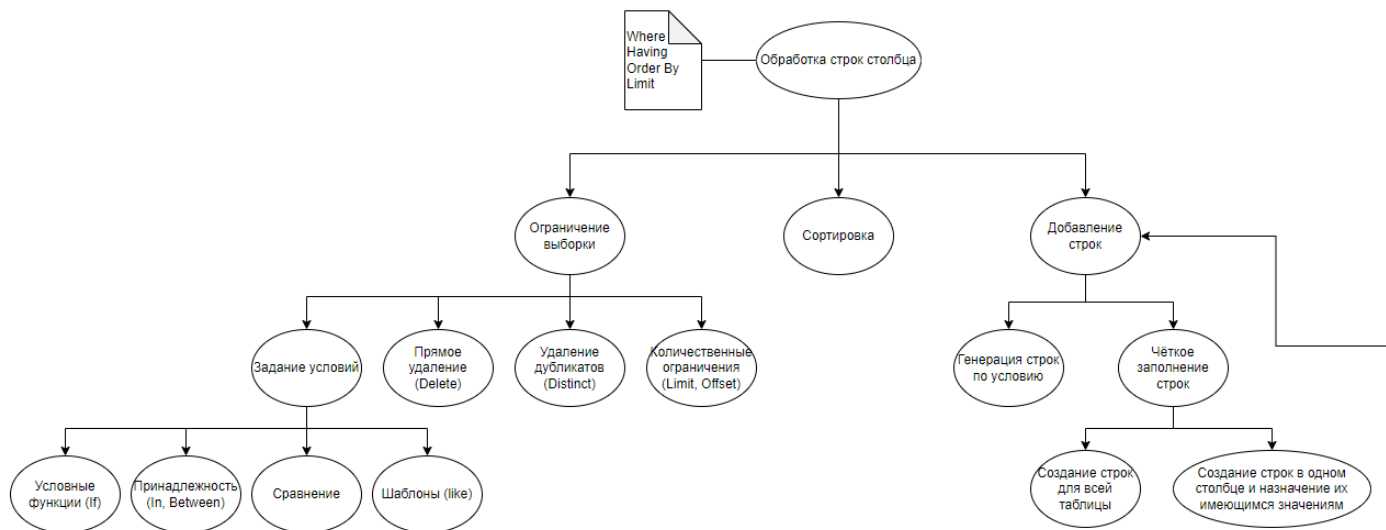
Rate_detail	INNER	Products_ids	T	ov	\$	12				
Key	=	Id	data_type	data_type	data_type	data_type	data_type	data_type	data_type	data_type
Key		Id	Тип показателя	Детализация	Key	Тип показателя	Голос	Интернет	Date	
Строка	Строка	Строка	Строка	Строка	Строка	Строка	Строка	Строка	Строка	10/07
Строка	Строка	Строка	Строка	Строка	Строка	Строка	Строка	Строка	Строка	10 июля
Строка	Строка	Строка	Строка	Строка	Строка	Строка	Строка	Строка	Строка	10 июля 2024
Строка	Строка	Строка	Строка	Строка	Строка	Строка	Строка	Строка	Строка	10.07.24
Строка	Строка	Строка	Строка	Строка	Строка	Строка	Строка	Строка	Строка	10.07.2024
Строка	Строка	Строка	Строка	Строка	Строка	Строка	Строка	Строка	Строка	10.07.24 г.
Строка	Строка	Строка	Строка	Строка	Строка	Строка	Строка	Строка	Строка	10/07/24
Строка	Строка	Строка	Строка	Строка	Строка	Строка	Строка	Строка	Строка	10.07
Строка	Строка	Строка	Строка	Строка	Строка	Строка	Строка	Строка	Строка	10 июля 24 г
Строка	Строка	Строка	Строка	Строка	Строка	Строка	Строка	Строка	Строка	10 июля 2024
Строка	Строка	Строка	Строка	Строка	Строка	Строка	Строка	Строка	Строка	пт, 10.07

Функциональные требования:

Языковая модель принимает текстовый запрос и исходный SQL-код. Возвращает измененный SQL-код или текстовое сообщение. Решает задачи редактирования строк столбцов, столбцов таблицы, таблиц модели, а также задачи поиска.

При удачном срабатывании модели при решении задач редактирования должен отправляться только SQL-запрос. При решении задач поиска или при неудачном срабатывании модели должны отправляться текстовые сообщения с результатом поиска или ошибкой.

Ошибка: неправильно указано правило для решения поставленной задачи. Текст ошибки: "Неверно указано правило для обработки. Пожалуйста, измените запрос"



1. Обработка строк:

1.1. Ограничение выборки – уменьшение числа строк в столбце по одному из следующих принципов:

1.1.1. Прямое удаление.

Пример: "Удали все строки, где Расходы пустые".

`ALTER TABLE Таблица DELETE WHERE Расходы is not null`

1.1.2. Удаление дубликатов.

Пример: "Оставь только неповторяющиеся значения в столбце Доходы".

`SELECT DISTINCT ... FROM Таблица ИЛИ`

`OPTIMIZE TABLE Таблица DEDUPLICATE`

1.1.3. Количественные ограничения на число значений с начала / конца .

Пример: "Оставь только первые 100 строк в таблице".

`SELECT ... FROM Таблица LIMIT / OFFSET Значение`

1.1.4. Задание условий - ограничение выборки одним из следующих способов:

1.1.4.1. Условные функции.

Пример: "Если в столбце нет числовых значений, то оставь его без изменений. Иначе умножь все числовые значения на 100".

`SELECT IF (Условие, Результат, Иначе)`

1.1.4.2. Принадлежность – проверка на наличие значения в массиве / множестве / диапазоне. Использует операторы **IN, EXIST, BETWEEN**.

Пример: "Оставь только даты в диапазоне от января 2023 до апреля 2024".

1.1.4.3. Сравнение. Использует операторы для сравнения как с фиксированными значениями, так и для сравнения столбцов между собой.

Пример: "Оставь только положительные значения в столбце Прибыль";
"Оставь только строки, в которых Доход больше Расхода"

1.1.4.4. Условия по шаблонам – поиск по значениям строк в соответствии с заданным шаблоном.

Пример: "Оставь строки, где в названии проекта содержатся кавычки"

1.2. Сортировка по столбцу.

Пример: "Отсортируй столбец Количество по убыванию, пустые строки в конце".

ORDER BY Количество desc

Пример 2: "Отсортируй: задачи с большим отклонением факта от плана в начале".

ORDER BY Fact - Plan desc

1.3. Добавление / изменение строк – расширение или заполнение таблицы одним из следующих способов:

1.3.1. Генерация строк по условию – добавление новых строк с использованием генератора значений. Неуказанные столбцы должны заполняться пустыми значениями.

Пример: "Добавь в таблицу 50 строк, где Название компании – ООО "Домик", Доходы варьируются от 20000 до 30000 с равномерными шагами между значениями".

Должен формироваться массив строк, которые после будут добавлены с помощью INSERT INTO.

ВАЖНО! Задача нуждается в оценке затрат времени: если генерация строк моделью с последующей вставкой в БД выполняется быстрее, чем в КХ, то выполнять таким образом. Иначе модель должна отдавать код SQL для генерации строк в КХ.

1.3.2. Чёткое заполнение строк – пользователь сам вводит все нужные ему значения:

1.3.2.1. Создание строк для всей таблицы. Не указанные поля заполняются пустыми значениями (NULL):

Пример: "Добавь две строки: Вася, апельсины, 5000, Краснодар. Миша, бананы, 10000, Сочи".

INSERT INTO TABLE Таблица VALUES (Значение столбца 1, Значение столбца 2, ..., Значение столбца n)

Пример 2: "Вставь новые значения: Пользователь Вася, Товар Апельсины, Количество 5000, Город Краснодар"

INSERT INTO TABLE Таблица (Пользователь, Товар, Количество, Город)
VALUES ('Вася', 'Апельсины', 5000, 'Краснодар')

Ошибка: в таблице нет столбцов для некоторых из указанных значений.
Текст ошибки: "В таблице нет столбцов для некоторых из указанных значений. Пожалуйста, проверьте список доступных столбцов и измените запрос"

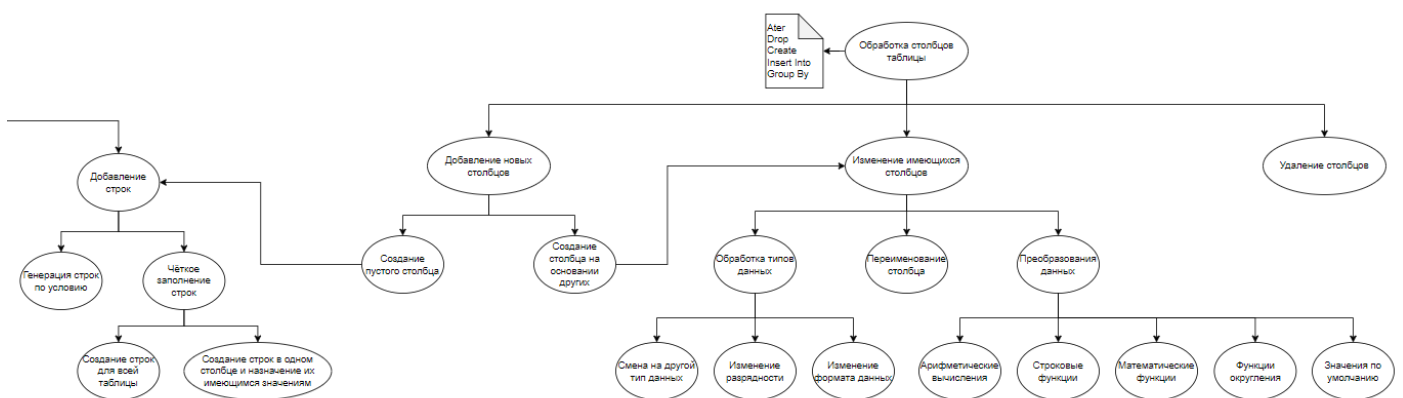
Ошибка: попытка подставить данные неподходящего типа. Текст ошибки: "Для данных {данные} выбран столбец с неподходящим типом. Пожалуйста, измените запрос"

1.3.2.2. Создание строк в одном столбце и назначение их имеющимся значениям. Аналогично предыдущему пункту, но заполняется только одно поле для заполненной строки.

Пример запроса: "Добавь название города Краснодар ко всем строкам, где цена больше 5000". Пример до выполнения: "Москва, 3000, яблоки", "NULL, 6000, бананы", "Ростов, 7000, апельсины". Пример после выполнения: "Москва, 3000, яблоки", "Краснодар, 6000, бананы", "Ростов, 7000, апельсины".

Альтернативный запрос: "Замени название города на Краснодар во всех строках, где цена больше 5000". Пример до выполнения: "Москва, 3000, яблоки", "NULL, 6000, бананы", "Ростов, 7000, апельсины". Пример после выполнения: "Москва, 3000, яблоки", "Краснодар, 6000, бананы", "Краснодар, 7000, апельсины".

Ошибки аналогичны предыдущему пункту.



2. **Обработка столбцов:**

2.1. Добавление новых столбцов (ADD COLUMN). Модель должна не только создавать столбец, но и самостоятельно определять его тип данных (при отсутствии запроса от пользователя) и выполнять проверку на существование столбца с указанным названием:

2.1.1. Создание пустого столбца.

Пример: "Добавь в таблицу Фрукты столбец Количество"

ALTER TABLE Таблица ADD COLUMN Столбец

2.1.2 Создание столбца на основании других - модель должна считать метаданные со столбца-основания, создать новый столбец, заполнить согласно запросу (подробно в п.2.2.). Использует MATERIALIZED - Материализованное вычисляемое выражение. Такое значение не передаётся при вставке, а вычисляется и сохраняется как физический столбец.

Пример: "Создай столбец Стоимость, в котором будет считаться произведение столбцов Цена и Количество"

ALTER TABLE Таблица ADD COLUMN Стоимость MATERIALIZED Цена * Количество

Альтернативный вариант – использовать ALIAS (В отличие от MATERIALIZED, результат вычисления не сохраняется как физический столбец, а вычисляется на лету при выборках из данного поля.):

ALTER TABLE Таблица ADD COLUMN Стоимость ALIAS Цена * Количество

2.2. Изменение столбцов:

2.2.1. Обработка типов данных (MODIFY COLUMN):

2.2.1.1. Смена на другой тип данных – модель должна выполнять проверку на возможность изменения типа данных столбца на указанный.

Пример: "Измени тип данных для столбца Количество с текстового на числовой"

Ошибка: пользователь предлагает неподходящий тип данных. Текст ошибки: "Невозможно изменить тип данных столбца на указанный. Пожалуйста, выберите подходящий тип данных"

ALTER TABLE Таблица MODIFY COLUMN Столбец Тип данных

2.2.1.2. Изменение разрядности – модель должна проверять наличие свойства «разрядность» в типе данных столбца таблицы.

Пример: "Добавь два знака после запятой в столбце Расходы"

Ошибка: у типа данных выбранного столбца нет разрядности. Текст ошибки: "Изменить разрядность указанного столбца невозможно – это свойство отсутствует"

2.2.1.3. (???) Изменение формата данных – сложная функция, которая преобразует представление исходных данных в формат, удобный пользователю. Модель должна определять положение исходных данных по столбцам, точки разбиения, функцию разбиения и сборки нужного формата. При этом результат сложного запроса всегда будет строкой.

Пример: "Измени записи названий региона с Регион №n на n-й регион, где n – число после номера № в названии"

```
select id, concat (arr[2], '-й регион') as name from (select *,  
regexp_split_to_array(name, '№')  
as arr from region r) f*
```

*запрос составлен на PostgreSQL и является тестовым

Запрос в ClickHouse:

```
ALTER TABLE Таблица UPDATE Регион = concat(regexp_substr(Регион, '\d+'), '-й  
регион')
```

Пример неудачной обработки запроса языковой моделью (установлен 1-й регион во всех названиях):

напиши код на PostgreSQL для изменения записи названий
региона с Регион №n на n-й регион, где n – число после номера
№ в названии

Для того чтобы изменить записи в таблице, где названия региона содержат текст формата

Регион №n на формат n-й регион, можно использовать следующий SQL-запрос в PostgreSQL:

```
sql Копировать код  
  
UPDATE your_table  
SET region_name = regexp_replace(region_name, 'Регион №(\d+)', '\1-й регион')  
WHERE region_name ~ 'Регион №\d+';
```

2.2.2. Переименование столбца (RENAME COLUMN). Используется и для изменения регистра в названии.

Пример: "Измени название столбца Фрукты на Овощи".

```
ALTER TABLE Таблица RENAME COLUMN Столбец TO Новое название
```

2.2.3. Преобразования данных:

2.2.3.1. **Арифметические вычисления.** Вычисление суммы, разности, произведения данных и т.д.

Пример: "Добавь ко всем числам, меньшим 5000, в столбце Доходы число 500"

Ошибка: над указанным типом данных невозможно проводить арифметические вычисления. Текст ошибки: "Над указанным типом данных невозможно проводить арифметические вычисления. Пожалуйста, измените запрос"

2.2.3.2. **Строковые функции.** Соединение и разъединение строк, поиск символов, определение длины и т.д.

Пример: "Соедини значения в столбце Название региона с его порядковым номером в столбце Id"

Ошибка: для указанного типа данных неприменимы строковые функции. Текст ошибки: "Для указанного типа данных неприменимы строковые функции. Пожалуйста, измените запрос"

2.2.3.3. **Математические функции.** Вычисление экспонент, логарифмов, поиск корней и т.д.

Пример: "Замени данные в столбце Наклон на синус от имеющихся значений в этом столбце"

Ошибка: для указанного типа данных неприменимы математические функции. Текст ошибки: "Для указанного типа данных неприменимы математические функции. Пожалуйста, измените запрос"

2.2.3.4. **Функции округления.**

Пример: "Округли значения в столбце Прибыль до 2 знаков после запятой"

Ошибка: для указанного типа данных неприменимы функции округления. Текст ошибки: "Для указанного типа данных неприменимы функции округления. Пожалуйста, измените запрос"

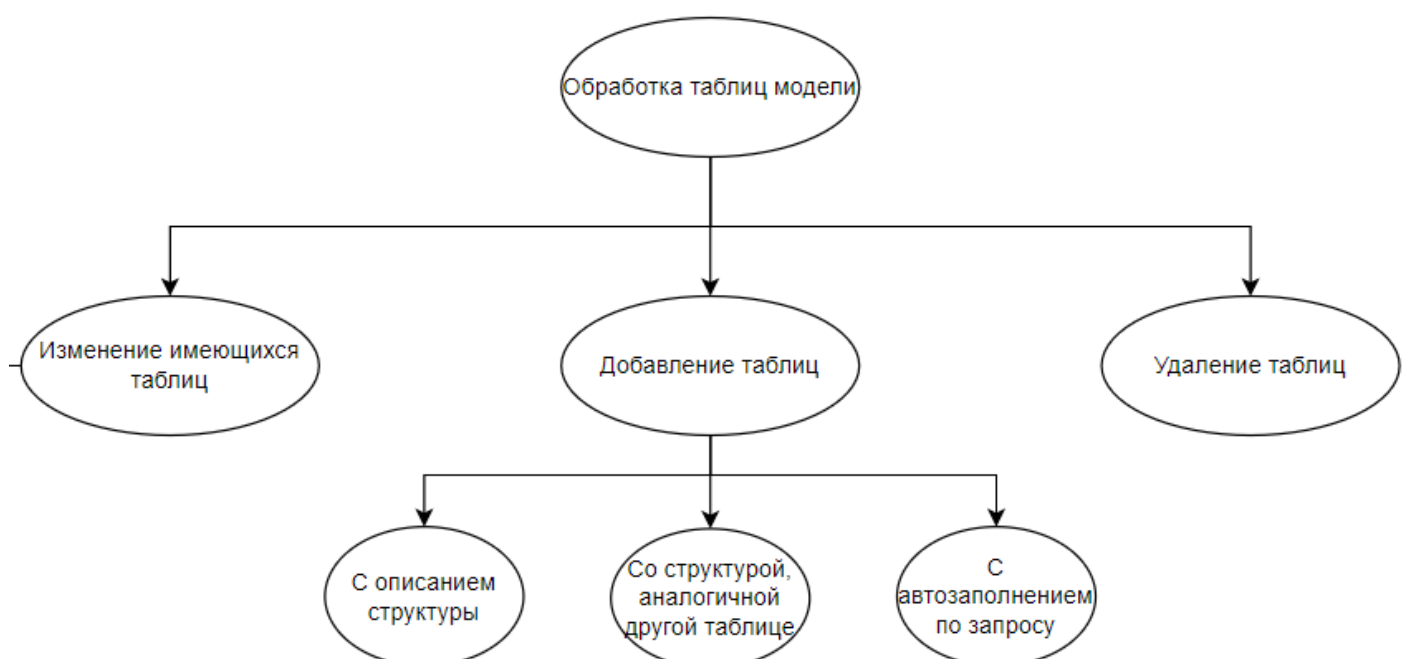
2.2.3.5. Установка **значений по умолчанию.**

Пример: "Установи в качестве базового значения в столбце Город значение Краснодар".

2.3. **Удаление столбцов** (DROP COLUMN).

Пример: "Удали столбец Фрукты"

Ошибка: столбца с указанным именем не существует. Текст ошибки: "Столбца с указанным именем не существует"



3. Обработка таблиц:

3.1. Изменение таблиц – любой запрос, содержащий задание на изменение таблицы, работает либо с её столбцами (п.2.), либо с её строками (п.1.)

3.2. Добавление таблиц (CREATE TABLE):

3.2.1. С описанием структуры – модель должна создавать пустую таблицу, в которой должны быть описаны столбцы, могут быть описаны первичный ключ и движок. Если движок не описан, то создаётся такой же, как и в редакторе скрипта по умолчанию (MergeTree).

Пример: "Создай таблицу Регионы с полями ID (ключевое) и Название".

Ошибка: не указано ни одно поле в новой таблице. Текст ошибки: "При создании новой таблицы необходимо указать хотя бы один столбец для создания"

3.2.2. Со структурой, аналогичной другой таблице:

Пример: "Создай таблицу Доходы на основании таблицы Расходы".

Ошибка: таблицы-основания не существует. Текст ошибки: "Вы указали несуществующую таблицу в качестве основания для создания новой. Пожалуйста, выберите существующую таблицу или проверьте корректность указанного названия таблицы"

3.2.3. С автозаполнением по запросу – модель должна формировать названия столбцов и их тип данных исходя из запроса на создание выборки:

Пример: "Создай таблицу Адреса, заполнив её названиями городов из таблицы Города и названиями улиц из таблицы Улицы".

Ошибка: таблицы-основания или столбцов из таблицы-основания не существует. Текст ошибки: "Вы указали несуществующую таблицу / несуществующие столбцы в качестве основания для создания новой. Пожалуйста, выберите существующую таблицу / существующие столбцы или проверьте корректность указанного названия таблицы / столбцов"

3.3. Удаление таблиц (DROP TABLE).

Пример: "Удали таблицу Прибыль".

Ошибка: таблицы для удаления не существует. Текст ошибки: "Вы указали несуществующую таблицу для удаления. Пожалуйста, выберите существующую таблицу или проверьте корректность указанного названия таблицы"

4. Поисковые функции. На выходе модель возвращает текстовое описание элемента модели данных по запросу от пользователя (без SQL-кода):

4.1. Сколько раз встречается значение, указанное пользователем, в столбце / таблице.

4.2. Наиболее часто / редко встречающееся значение, указанное пользователем, в столбце / таблице.

4.3. Наибольшее / наименьшее значение (самая длинная / короткая строка), указанное пользователем, в столбце.

4.4. Порядковый номер в столбце значения, указанного пользователем.

4.5. Количество значений указанного формата в столбце / таблице, или значений указанного типа в таблице.

Пример: "Сколько раз встречается слово Краснодар в столбце Город?".

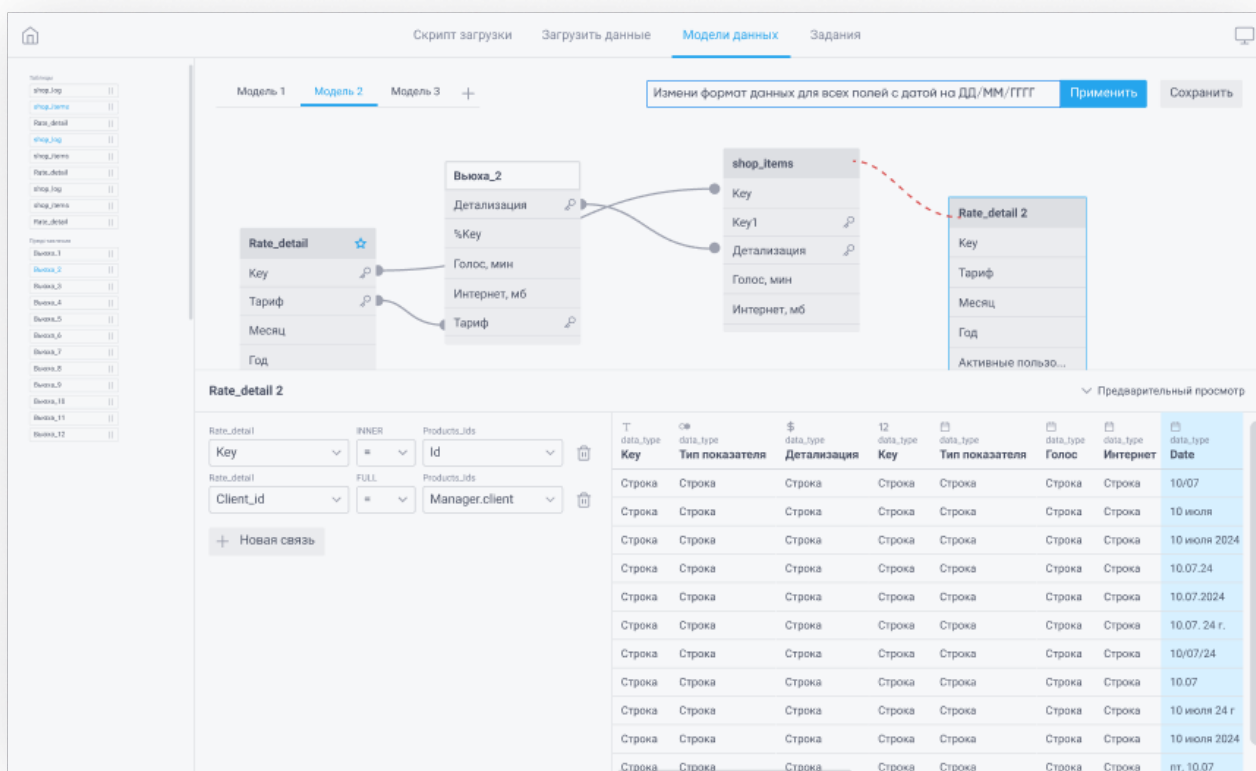
Пример: "Сколько раз встречаются столбцы со строковым типом данных в таблице Продажи?".

Модуль интеллектуального управления и проверки качества данных (Фронт)

Необходимо создание и внедрение в редактор модели данных модуля проверки качества данных с использованием языковой модели нейронной сети для обработки текстовых сообщений от пользователя.

Концепт:

Текстовая строка для ввода сообщения от пользователя с кнопками «Применить», «Сохранить» и «Отменить». При нажатии на кнопку «Применить» запускается обработка текстового сообщения пользователя языковой моделью. После обработки сообщения и внесения моделью изменений в имеющуюся таблицу в секции «Предварительный просмотр» отображается измененная таблица. При нажатии на кнопку «Сохранить» внесенные изменения вносятся в скрипт и меняют модель данных. При нажатии на кнопку «Отменить» (при условии, что не была нажата кнопка «Сохранить») все предложенные моделью изменения сбрасываются, таблица для предварительного просмотра возвращается в первоначальное состояние.



Функциональные требования:

Интеграция в модуль работы с моделями данных текстовой строки с 3 управляющими кнопками:

1. **Текстовое поле** имеет неограниченную длину по числу символов. При достижении видимой границы текстового поля справа оно должно автоматически расширяться вниз с переносом текста на следующую строку. Расширенное поле должно быть видно только при выборе строки запроса нажатием на него. Нажатие на любое пустое место на странице должно приводить к автоматическому скрыванию поля до размеров первоначальной строки.

2. **Кнопка «Применить»** отправляет POST-запрос через API на бэкенд. В теле запроса находится текст из текстового поля и SQL-код из фронта. После нажатия на кнопку «Применить» другие кнопки (в т.ч. для перехода в скрипт загрузки или возвращения на дашборд) остаются недоступными до окончания работы модели. Из бэкенда возвращается SQL-запрос, который формирует временную таблицу для просмотра. Временная таблица добавляется в список таблиц, выносится на модель, курсор устанавливается на эту таблицу.

3. **Кнопка «Сохранить»**. По умолчанию недоступна (некликабельна). Активируется после формирования таблицы предварительного просмотра. При нажатии на кнопку SQL-код интегрируется во фронт и происходит перезапись модели: сохранение названия старой таблицы - удаление старой таблицы - переименование временной таблицы - сохранение

временной таблицы как обычной.

При нажатии на кнопку «Сохранить» должно появляться системное сообщение с кнопками для продолжения или отмены и текстом: "Применение изменений приведёт к перезаписи имеющейся таблицы. Все существующие связи между таблицами будут потеряны, их необходимо создать заново. Вы уверены, что хотите продолжить?"

4. **Кнопка «Отменить»** - удаляет временную таблицу, возвращает в таблицу предварительного просмотра результат работы скрипта без изменений от языковой модели.

Архитектура взаимодействия с моделью

