Создание ClickHouse - SQL агента

Введение

Обычные LLM модели плохо справляются с числовыми данными - придумывают несуществующие цифры, неточно считают и не могут обрабатывать большие объемы информации. Для решения этой проблемы мы соединим возможности ИИ с профессиональными инструментами работы с данными.

В этом руководстве мы создадим специализированного агента для работы с базой данных ClickHouse, который сможет:

- Точно отвечать на вопросы с числовыми данными
- Выполнять сложные расчеты и сравнения
- Искать минимальные и максимальные значения
- Анализировать тарифы и условия

Пример использования: Банк собрал данные о конкурентах в одну таблицу и хочет, чтобы бот мог отвечать на вопросы типа "У какого банка самая низкая комиссия за перевод 25 000 долларов?"

Этап 1: Подключение к ClickHouse

Настройка подключения к базе данных

Первым шагом необходимо развернуть собственный кластер ClickHouse. После развертывания:

1. **Переходим в атрибуты бота**
2. **Создаем атрибут** `CLICK_HOUSE_CREDENTIALS`
3. **Заполняем данные подключения** в формате JSON:

```
{
    "host": "http://111.1111.11.101:8000",
    "user": "admin",
    "password": "12345",
    "database": "test"
}
```

Тестирование соединения

Создайте новый скрипт в боте для проверки подключения:

```
const ClickHouse = require('Common.Integrations.ClickHouse')
const CLICK_HOUSE_CREDENTIALS = bot.getJsonAttr("CLICK_HOUSE_CREDENTIALS")

const ch = new ClickHouse(CLICK_HOUSE_CREDENTIALS)

// 1. ПРОВЕРКА СОЕДИНЕНИЯ
const ping = ch.ping()
bot.sendMessage('PING → ' + ping.result)

// 2. ПРОСМОТР СУЩЕСТВУЮЩИХ ТАБЛИЦ
const tablesList = ch.select('SHOW TABLES')
bot.sendMessage('TABLES → ' + JSON.stringify(tablesList))
```

Запустите скрипт и проверьте результат. Если возникают ошибки - проверьте доступы к кластеру и правильность данных подключения.

Этап 2: Проектирование структуры таблицы

Подготовка исходных данных

Для примера используем таблицу сравнения банковских тарифов:

Банк	Мин. сумма	Сроки	До 20k USD	20–50k USD	50–100k USD	Валютный контроль	Мин. руб.	Макс. руб.				
Кофейный	нет	3–4 дня	3%	1,2%	0,6%	0,36%	0,36%	нет	н/д	0,15%	750	нет
Жёлтый	нет	1–3 дня	3%	2,5%	2,5%	2%	2,5%	нет	курс ЦБ	0,16%	360	29000
Салатовый	нет	до 5 дней	3,5%	3,5%	3,5%	2,8%	2,5%	нет	курс ЦБ	0,15%	2000	нет
Зелёный	нет	3–5 дней	3%	3%	2,7%	2,3–2,5%	2,2%	130 USD за SWIFT до 50k	банк/ЦБ/Investing	0,24%	2000	—
Розовый	нет	5 дней	3,5–4,5%	3,3–3,9%	3–3,5%	2,3–2,5%	2,2%	70–150 USD за SWIFT до 15k	Investing/ЦБ (макс ставка)	0,15%	750	нет
Охра	нет	3 дня	55000 руб.	2,65%	2,2%	2,1%	2,1%	нет	курс ЦБ	0,1%	700	20000
Малиновый	20 тыс.	3 дня	—	2–3%	2–3%	2–3%	2–3%	нет	ЦБ/Investing	0,12%	500	60000
Синий	50 тыс.	2–4 дня	Китай: 2,5%	Иные: 2–4%	—	—	—	нет	Investing	0,15%	1000	20000
Бирюзовый	10 тыс.	3–4 дня	2–4%	2–4%	2–4%	2–4%	2–4%	нет	ЦБ/profinance	0,12%	—	60000
Оранжевый	—	1–6 дней	4,5% мин. 80k руб.	4,5% мин. 80k руб.	2,5–4%	2,5–4%	2,5–4%	нет	курс ЦБ	0,15%	1000	20000
Бурый	50 тыс.	3 дня	нет	нет	2–3,5%	2–3,5%	2–3,5%	нет	Investing	0,15%	600	50000
Среднее	30 тыс.	3 дня	3,7%	3,1%	3,1%	2,7%	2,4%	нет	нет	0,14%	750	45000

Банк	Мин. сумма	Сроки	До 20k USD	20–50 kUSD	50–100k USD	Валютный контроль	Мин. руб.	Макс. руб.				
Красный	—	3 дня	3% мин. 500 USD / 1,5–2 % мин. 400 USD	то же	то же	2,5% / 1,5–2 % мин. 400 USD	2% / 1,5–2 % мин. 400 USD	нет	банк/ЦБ	0,22%	1000	нет

Генерация оптимальной структуры с помощью ИИ

Чтобы создать правильную структуру базы данных, воспользуемся специальным промптом:

1. **Переходим на** https://console.anthropic.com
2. **Используем проверенный промпт**:

```
You are an AI agent (Data Engineer) tasked with building ClickHouse database tables for a bank's calculation purposes. Your goal is to analyze JSON data and create an optimal table structure that allows for easy querying in ClickHouse.

Here are some important guidelines:

- Always split percentages into minimum and maximum values.

- Never store percentages as STRING or in quotes like "0.3-0.5".

- Use appropriate data types: FLOAT for percentages and decimals, INT for whole numbers.

- Design the structure carefully, thinking like a database engineer.

Analyze the following JSON data:

<json_data>

{{JSON_DATA}}
```

</json_data>

Your task is to:

1. Carefully examine the JSON structure and its contents.

2. Design an appropriate database structure that will allow for efficient querying in ClickHouse. Consider:

- What fields should be created?

- What data types should be used for each field?

- How to handle nested structures or arrays if present?

- How to split percentage ranges into separate minimum and maximum fields?

3. Write a SQL query to create the database table(s) based on your designed structure. Ensure that:

- All field names are clear and descriptive

- Appropriate data types are used (FLOAT for percentages and decimals, INT for whole numbers, etc.)

- The structure allows for easy and efficient querying

Remember: The goal is to create a structure that will enable straightforward and performant queries in ClickHouse. Think carefully about how the data will be used and queried in the future.

Provide your response in the following format:

<database_design>

[Explain your thought process and the rationale behind your database design here]

</database_design>

```
<sql_query>

[Write the SQL query to create the database table(s) here]

</sql_query>

If you need to make any assumptions or have any questions about the data structure, state them clearly in your explanation.
```

3. **В параметр JSON_DATA** подставляете несколько строк из вашей таблицы
4. **Получаете готовый SQL** для создания оптимальной структуры

Создание таблицы в ClickHouse

Создайте скрипт с полученным SQL-запросом:

```
const ClickHouse = require('Common.Integrations.ClickHouse')
const CLICK_HOUSE_CREDENTIALS = bot.getJsonAttr("CLICK_HOUSE_CREDENTIALS")

const ch = new ClickHouse(CLICK_HOUSE_CREDENTIALS)

// 1. ПРОВЕРКА СОЕДИНЕНИЯ
const ping = ch.ping()
bot.sendMessage('PING → ' + ping.result)

// 2. УДАЛЕНИЕ СТАРОЙ ТАБЛИЦЫ (если существует)
const dropResult = ch.query(`DROP TABLE IF EXISTS bank_commissions`)
bot.sendMessage('DROP → DONE')

// 3. СОЗДАНИЕ НОВОЙ ТАБЛИЦЫ (Сюда подставляем SQL из прошлого шага)
const createResult = ch.createTable(`
CREATE TABLE bank_commissions
(
    bank_name String,
    min_payment Nullable(Int32),
    delivery_days_min UInt8,
    delivery_days_max UInt8,
    commission_under_20k_min Float32,
    commission_under_20k_max Float32,
    commission_20k_50k_min Float32,
```

```
    commission_20k_50k_max Float32,
    commission_50k_100k_min Float32,
    commission_50k_100k_max Float32,
    currency_control_min Float32,
    currency_control_max Float32,
    min_rub Nullable(Int32),
    max_rub Nullable(Int32)
)
ENGINE = MergeTree
ORDER BY bank_name`)
bot.sendMessage('CREATE → DONE')
```

Этап 3: Загрузка данных

Подготовка данных в JSON формате

Преобразуйте табличные данные в структурированный JSON с полями соответствующими полученной таблице:

```
[{
    "bank_name": "Кофейный",
    "min_payment": null,
    "delivery_days_min": 3,
    "delivery_days_max": 4,
    "commission_under_20k_min": 3.0,
    "commission_under_20k_max": 3.0,
    "commission_20k_50k_min": 1.2,
    "commission_20k_50k_max": 1.2,
    "commission_50k_100k_min": 0.6,
    "commission_50k_100k_max": 0.6,
    "currency_control_min": 0.15,
    "currency_control_max": 0.15,
    "min_rub": 750,
    "max_rub": null
}, {
    "bank_name": "Жёлтый",
```

```
    "min_payment": null,
    "delivery_days_min": 1,
    "delivery_days_max": 3,
    "commission_under_20k_min": 3.0,
    "commission_under_20k_max": 3.0,
    "commission_20k_50k_min": 2.5,
    "commission_20k_50k_max": 2.5,
    "commission_50k_100k_min": 2.5,
    "commission_50k_100k_max": 2.5,
    "currency_control_min": 0.16,
    "currency_control_max": 0.16,
    "min_rub": 360,
    "max_rub": 29000
}]
```

Загрузка через Метабот

Создайте скрипт для загрузки данных:

```
const ClickHouse = require('Common.Integrations.ClickHouse')
const CLICK_HOUSE_CREDENTIALS = bot.getJsonAttr("CLICK_HOUSE_CREDENTIALS")

const ch = new ClickHouse(CLICK_HOUSE_CREDENTIALS)

// 1. ПРОВЕРКА СОЕДИНЕНИЯ
const ping = ch.ping()
bot.sendMessage('PING → ' + ping.result)

// 2. ОЧИСТКА СТАРЫХ ДАННЫХ
const truncateResult = ch.query('TRUNCATE TABLE bank_commissions')
bot.sendMessage('TRUNCATE → DONE')

// 3. ПОДГОТОВКА ДАННЫХ
const banksData = [
    // Вставьте здесь ваш JSON массив с данными банков
]

// 4. ЗАГРУЗКА В БАЗУ
const insertResult = ch.insert('bank_commissions', banksData)
```

```
bot.sendMessage('INSERT → DONE')
```

Этап 4: Интеграция SQL-агента в систему

Теперь создадим специализированного агента, который будет генерировать SQL-запросы и возвращать точные ответы на основе данных из ClickHouse.

Шаг 1: Обновление конфигурации

Откройте плагин `AgentsParams.MainConfig` и дополните конфигурацию:

А) Добавить новый инструмент в маршрутизатор

```
detectRoute: {
    // ... остальные параметры остаются без изменений
    routerTools: [{
        tools: "RAG_Tools",
        route: "RAG: DetectIntent_and_FindChunks"
    }, {
        tools: "INFO_Tools",
        route: "CompanyInfo"
    }, {
        tools: "TABLE1_Tools", // Новый инструмент для работы с данными
        route: "ClickHouse: GenerateSQL"
    }]
    // ... остальные параметры остаются без изменений
}
```

Б) Создать конфигурацию для ClickHouse агента

```
else if (activeAgent === "ClickHouse") { // Агент для работы с SQL таблицами ClickHouse
    agentCFG = {
        common: {
            title: "ClickHouse Query Agent",
            agentName: "ClickHouse",
```

```
            promptTable: "gpt_prompts",
            userQueryAttibName: "user_query",
            historyMaxLength: 2,
            exitScript: "MainFlow:FollowUp"
        },
        generateSQL: {
            provider: "OpenAI",
            model: "gpt-4o",
            modelParams: {
                "temperature": 1
            },
            prompt: "$clickhouse_sql_prompt",
            errorScript: "RAG:ErrorFallback",
            exitScript: "RAG:UserReply",
            useHistory: 1,
            addUserQuery: 1,
            sendBotAnwser: 1
        },
        userReply: {
            provider: "OpenAI",
            model: "gpt-4o",
            modelParams: {
                "temperature": 1
            },
            errorScript: "RAG:ErrorFallback",
            useHistory: 1,
            addUserQuery: 1,
            sendBotAnwser: 1,
            systemPrompts: {
                start: [
                    ["$clickhouse_result_prompt"]
                ],
                final: []
            },
        }
    }
}
```

Шаг 2: Создание скрипта агента

Создайте новый скрипт с именем `ClickHouse: GenerateSQL`:

> Код создавайте в блоке "Выполнить асинхронный API-запрос"

```
const LLMClient = require("Common.MetabotAI.LLMClient")

// Установка активного агента ClickHouse
lead.setAttr('activeAgent', 'ClickHouse')
snippet("Business.AgentsParams.MainConfig")
const llm = new LLMClient("UserReply", agentCFG.common.agentName)

// Настройка истории диалога
if (!agentCFG.generateSQL.useHistory) {
  llm.disableHistory()
}

if (isFirstImmediateCall) {
  // Настройка LLM клиента
  llm.setProvider(agentCFG.generateSQL.provider)
  llm.setModel(agentCFG.generateSQL.model)
  llm.setModelParams(agentCFG.generateSQL.modelParams)
  llm.addSystemPrompt(agentCFG.generateSQL.prompt)

  llm.addUserQuery(lead.getAttr(agentCFG.common.userQueryAttibName))

  llm.prepareRequest()
  return llm.sendRequest()
}

// Обработка ответа LLM
llm.handleResponse()

// Извлечение SQL-запроса из ответа
let sqlQuery = extractContentBetweenTags("sql_query")

// Проверка корректности SQL
if (!sqlQuery) {
  return llm.nextFlow(agentCFG.generateSQL.errorScript)
```

```
}

// Подключение к ClickHouse
const ClickHouse = require('Common.Integrations.ClickHouse')
const CLICK_HOUSE_CREDENTIALS = bot.getJsonAttr("CLICK_HOUSE_CREDENTIALS")
const ch = new ClickHouse(CLICK_HOUSE_CREDENTIALS)

// Выполнение SQL-запроса
const queryResult = ch.query(sqlQuery)

// Добавление результата в историю
llm.addToHistory({
  role: "assistant",
  content: JSON.stringify(queryResult)
})

// Переход к формированию ответа пользователю
return llm.nextFlow(agentCFG.generateSQL.exitScript)

// Функция извлечения контента между тегами
function extractContentBetweenTags(tagName, text = null) {
  text = text || llm.getResponseText()
  const openTag = `<${tagName}>`
  const closeTag = `</${tagName}>`

  const startIndex = text.indexOf(openTag)
  if (startIndex === -1) return null

  const endIndex = text.indexOf(closeTag, startIndex + openTag.length)
  if (endIndex === -1) return null

  return text.substring(startIndex + openTag.length, endIndex).trim()
}
```

Шаг 3: Создание промптов

1. Откройте таблицу `gpt_prompts`, создайте промпт `clickhouse_sql_prompt` для агента `ClickHouse`:

```
You are a Data Engineer who communicates exclusively through SQL queries. Your task is to interpret requests from a senior agent and formulate appropriate SQL queries for a ClickHouse database. Here's how you should proceed:

First, familiarize yourself with the structure of the ClickHouse table:

<table_structure>
{{@click_house_tables}}
</table_structure>

When you receive a query from the senior agent, it will be in simple language. Your job is to interpret this query and create an SQL statement that will retrieve the requested information from the ClickHouse database.

The user's query will be provided in the following format:

<user_query>
{{$user_query}}
</user_query>

Based on this query and your knowledge of the table structure, formulate an SQL query that will retrieve the requested information. Your query should be as detailed as possible, providing comprehensive data for the senior agent to analyze and make decisions.

Remember:
1. You should only output SQL code. Do not provide any explanations, comments, or additional text.
2. Strive to create complex queries when necessary to calculate the required data.
3. Always aim to provide detailed data in your query results.
4. Do not use any functions or features that are not standard SQL or specific to ClickHouse.

Your response should consist solely of the SQL query, enclosed in <sql_query> tags. For example:

<sql_query>
SELECT column1, column2, COUNT(*) as count
FROM table_name
WHERE condition
GROUP BY column1, column2
```

```
ORDER BY count DESC
LIMIT 10
</sql_query>

Do not include any text before or after the SQL query. Your entire response should be contained within the <sql_query> tags.
```

2. Создайте промпт `clickhouse_result_prompt` для агента `ClickHouse`:

```
Вы являетесь Помощником банка. Ваша задача - проанализировать историю диалога, получить из неё запрос пользователя, проанализировать ответ SQL агента и дать понятный ответ, соответствующий запросу пользователя. Отвечайте четко и по существу, представляя числовые данные в удобном для восприятия формате.
```

3. Создайте атрибут бота `click_house_tables` с описанием структуры таблицы:

```
TABLE: bank_commissions

COLUMNS:
- bank_name (String): Название банка
- min_payment (Nullable Int32): Минимальная сумма платежа в рублях
- delivery_days_min (UInt8): Минимальное время доставки в днях
- delivery_days_max (UInt8): Максимальное время доставки в днях
- commission_under_20k_min (Float32): Минимальная комиссия до 20k USD в процентах
- commission_under_20k_max (Float32): Максимальная комиссия до 20k USD в процентах
- commission_20k_50k_min (Float32): Минимальная комиссия 20-50k USD в процентах
- commission_20k_50k_max (Float32): Максимальная комиссия 20-50k USD в процентах
- commission_50k_100k_min (Float32): Минимальная комиссия 50-100k USD в процентах
- commission_50k_100k_max (Float32): Максимальная комиссия 50-100k USD в процентах
- currency_control_min (Float32): Минимальная комиссия валютного контроля в процентах
- currency_control_max (Float32): Максимальная комиссия валютного контроля в процентах
- min_rub (Nullable Int32): Минимальная сумма в рублях
- max_rub (Nullable Int32): Максимальная сумма в рублях
```

Шаг 4: Обновление маршрутизации

Откройте таблицу `gpt_prompts`, найдите промпт `route_prompt` для агента `MainFlow` и обновите секцию с инструментами:

```
<tools>
**RAG_Tools** - Используется, когда нужно ответить на вопросы об услугах
и продукции компании, об условиях эксплуатации продукции, об используемых
инструментах для монтажа продукции и т.д.

**INFO_Tools** - Используется когда пользователь хочет узнать о филиалах
компании, контактных данных или времени их работы

**TABLE1_Tools** используется для поиска информации и ответов на вопросы о:
- Ценах и стоимости переводов
- Комиссиях за переводы и платежи
- Точных цифрах и числовых данных, хранящихся в локальной базе данных
- Сравнении различных параметров (цены, комиссии, характеристики)
- Поиске минимальных/максимальных значений (самая низкая комиссия, самая высокая цена и т.д.)
- Условиях и тарифах банков-партнеров
- Расчетах и вычислениях на основе имеющихся данных
</tools>
```

Этап 5: Тестирование и результат

Примеры работы системы

После настройки система сможет отвечать на сложные вопросы с числовыми данными:

Простые запросы:

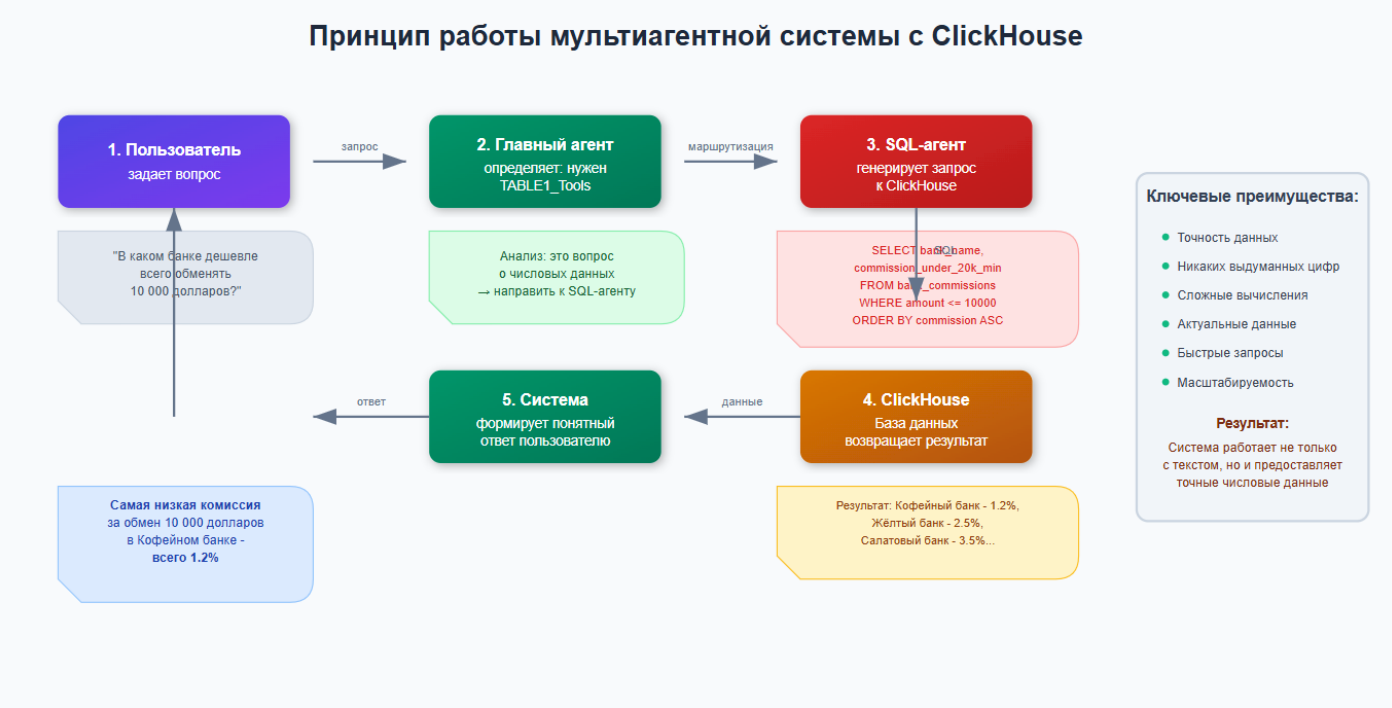
- "Какая комиссия у банка Кофейный?"
- "Сколько дней занимает перевод в Желтом банке?"

Сложные аналитические запросы:

- "В каком банке дешевле всего обменять 10 000 долларов?"
- "Покажи банки с минимальной комиссией валютного контроля"
- "Какой банк предлагает самые быстрые переводы?"

Принцип работы

1. **Пользователь задает вопрос** о числовых данных
2. **Главный агент определяет** - нужен TABLE1_Tools
3. **SQL-агент генерирует** соответствующий запрос к ClickHouse
4. **База данных возвращает** точные результаты
5. **Система формирует** понятный ответ для пользователя



Преимущества решения

1. **Точность данных** - никаких выдуманных цифр
2. **Сложные вычисления** - система может выполнять расчеты любой сложности
3. **Масштабируемость** - легко добавлять новые таблицы и данные
4. **Актуальность** - данные всегда свежие из базы
5. **Производительность** - ClickHouse обеспечивает быстрые запросы

Ваша мультиагентная система теперь может работать не только с текстовой информацией, но и предоставлять точные числовые данные и выполнять сложные аналитические запросы.

Версия #1
Павел Борисов создал 5 September 2025 14:04:41
Павел Борисов обновил 12 November 2025 14:03:47