



# Опыт внедрения искусственного интеллекта (AI) в PostgreSQL и Postgres Pro

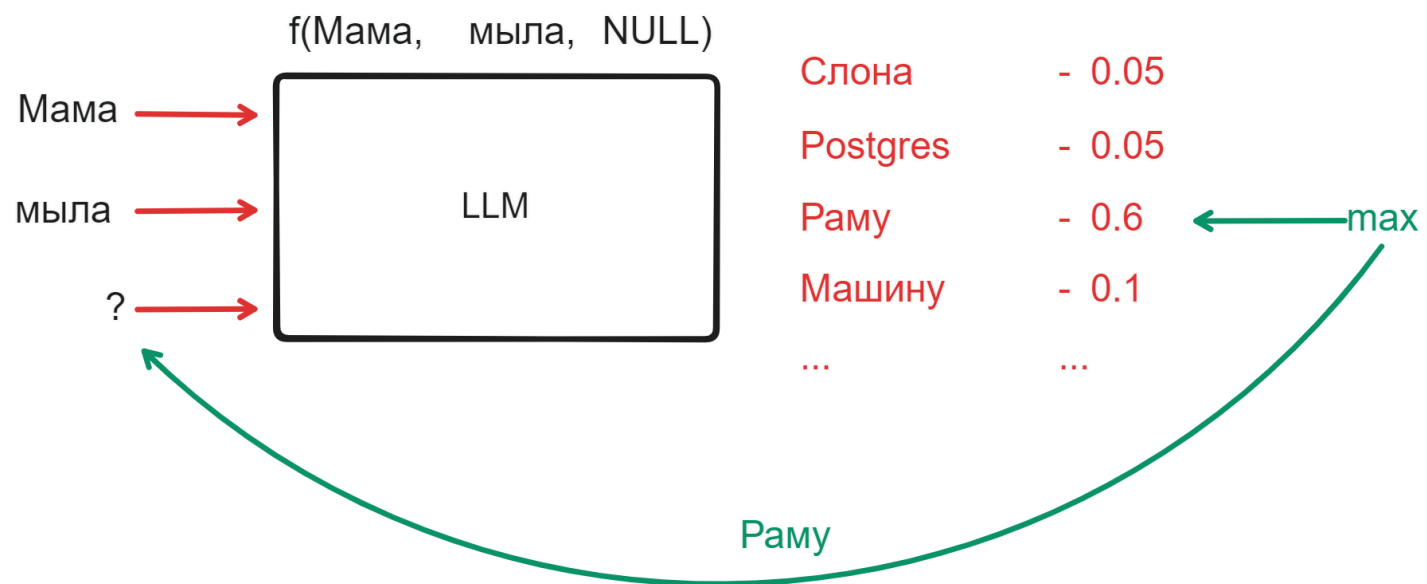
Савелий Батурин,  
28 января 2025 года

*PGProDay*



# Что такое LLM?

- ИИ?
- Вероятностная модель языка
- Нечёткий слепок интернета
- Стохастический попугай
- ...



# Применение LLM в PostgreSQL

- Генерация SQL
- Генерация Cypher
- RAG

# Генерация SQL



Какова общая сумма проданных билетов для каждой категории тарифа (например, эконом, бизнес)?

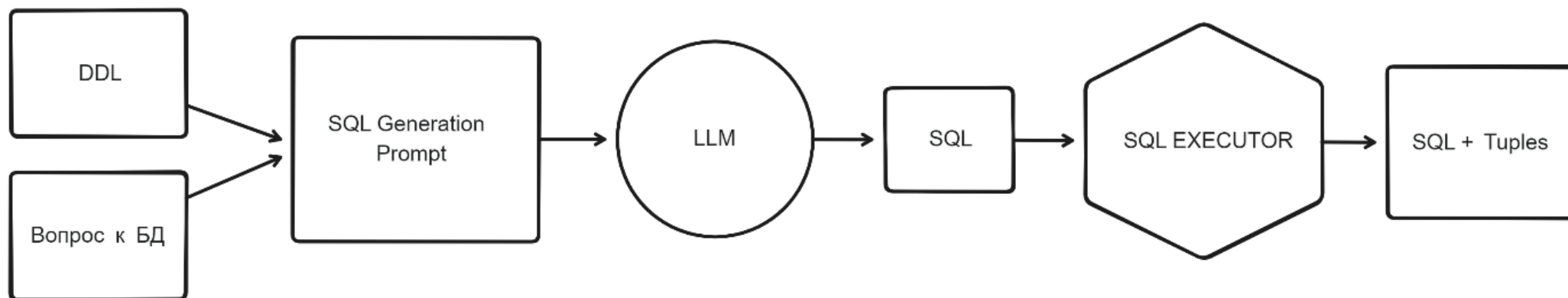


```
SELECT tf.fare_conditions AS fare_category,  
       SUM(tf.amount) AS total_sales  
FROM tickets t  
JOIN ticket_flights tf ON t.ticket_no = tf.ticket_no  
GROUP BY tf.fare_conditions;
```

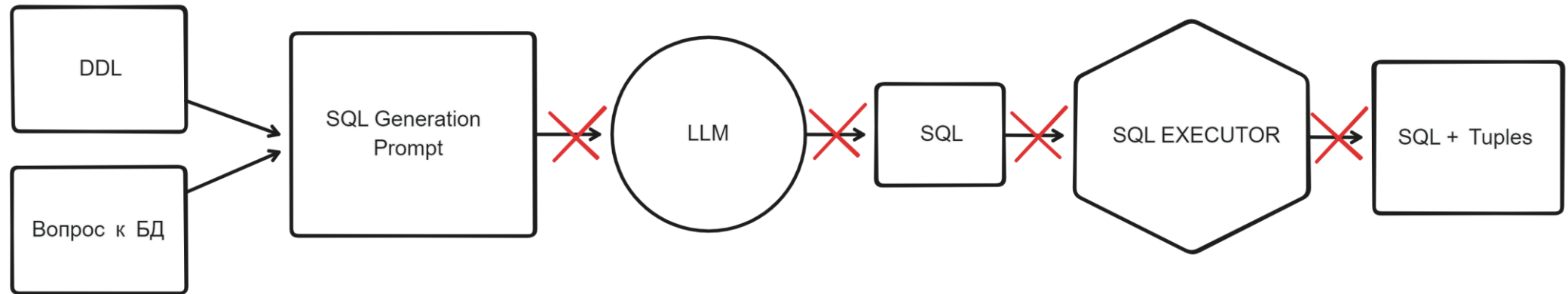
fare_category	total_sales
Business	5505179600.0
Comfort	566116900.0
Economy	14695684400.0



# Наивная схема SQL-генератора

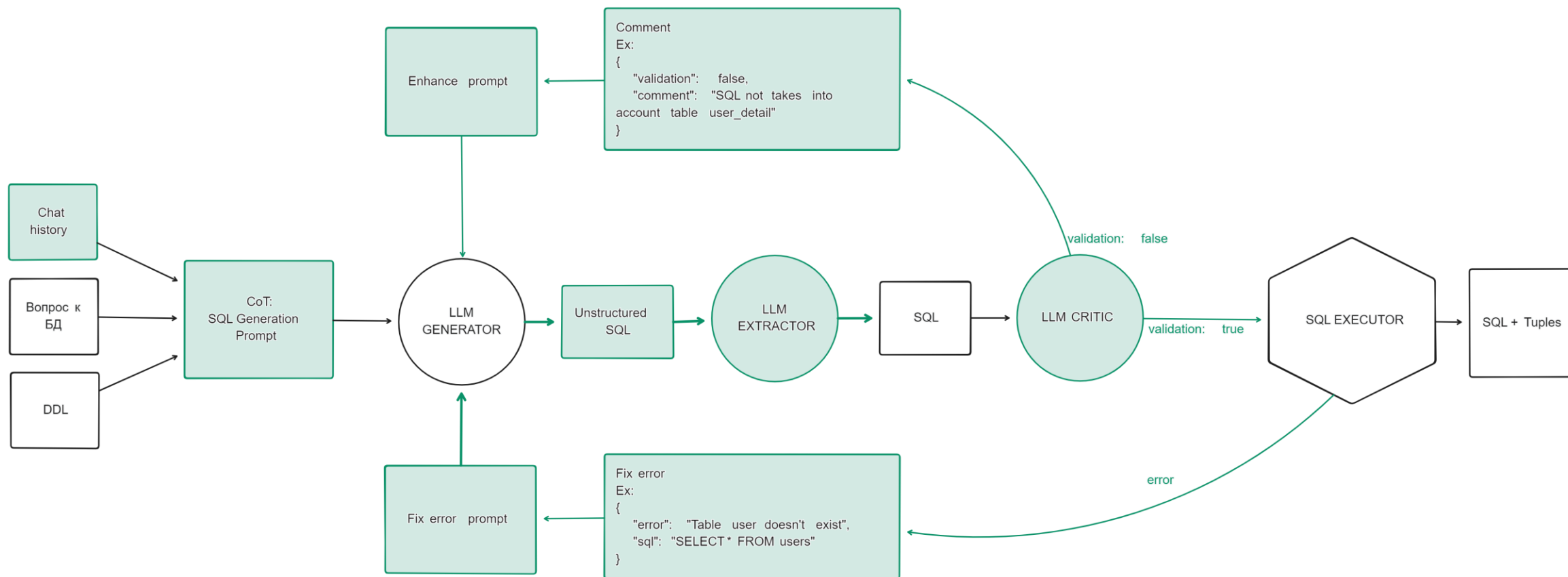


# Проблемы наивной схемы SQL-генератора

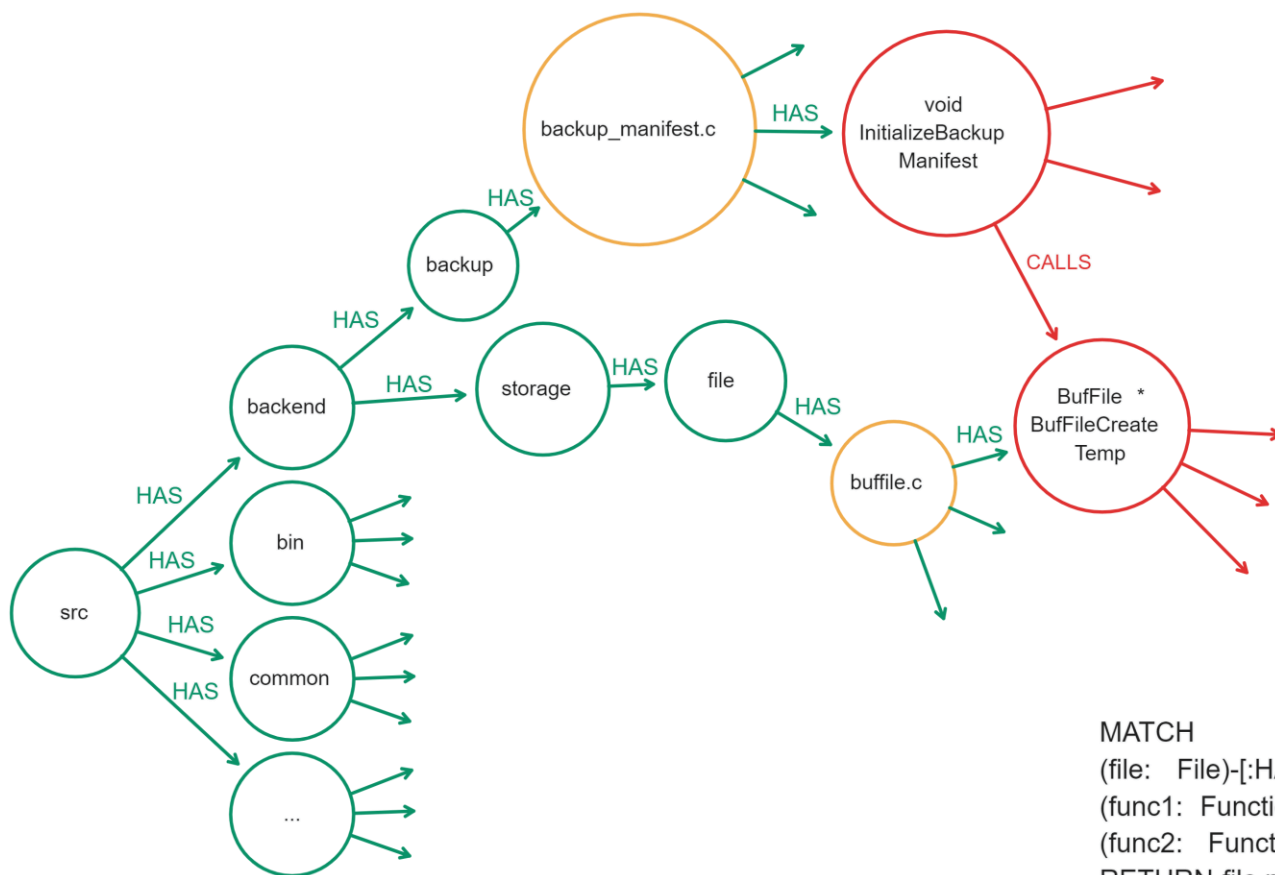


- Слишком абстрактный вопрос пользователя
- Возможность логической/семантической ошибки при генерации SQL
- Нечёткий формат генерации SQL (JSON / Блок кода / Чистый SQL / SQL с пояснениями/...)
- Возможность ошибки при исполнении

# Сложный пайплайн генерации SQL



# Генерация Cypher



MATCH  
 (file: File)-[:HAS]->  
 (func1: Function)-[:CALLS]->  
 (func2: Function {name: "BufFileCreateTemp"})  
 RETURN file.name

→ backup\_manifest.c



# Промпт генератора Cypher

```

generate_codebase_cypher_prompt = """
## DB Schema:
{scheme}

## Example Queries:
{examples}

## USER'S QUERY:
{query}

## INSTRUCTION:
You are a professional Cypher graph database analyst. The graph database contains PostgreSQL source code information. Your task is to write an optimized Cypher query to satisfy the user's request.

### Guidelines for Writing Efficient Queries:
1. Filter Early:
| - Apply `WHERE` conditions directly in the `MATCH` clause to limit unnecessary traversals.

2. Combine Paths:
| - Use combined paths (e.g., `(a)-[:REL]->(b)-[:REL]->(c)`) whenever relationships are closely related.

3. Avoid Redundancy:
| - Avoid `OPTIONAL MATCH` unless absolutely necessary. Use `MATCH` for expected relationships.

4. Limit Results:
| - Use `LIMIT` to control the result size and improve performance.

5. Return Only Necessary Fields:
| - Include only required properties in the `RETURN` clause. Avoid returning entire nodes unless needed.

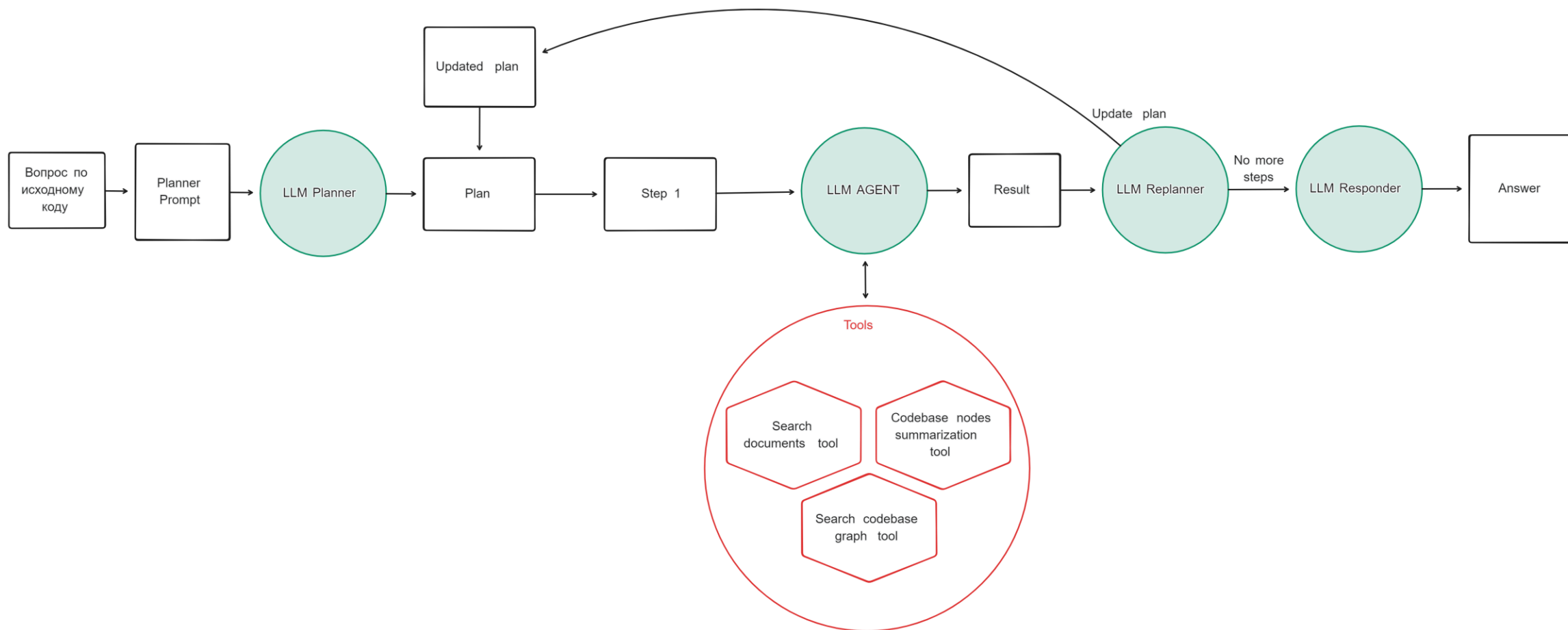
6. Optimize Traversals:
| - Traverse direct relationships where possible. Avoid patterns that may lead to Cartesian products.

7. Use Contextual Examples:
| - Refer to the schema and examples for clarity on the structure and relationships.

### STEP-BY-STEP QUERY GENERATION:
1. Identify the nodes, relationships, and properties relevant to the user's query.
2. Construct an efficient Cypher query by applying filters, combining paths, and minimizing traversals.
3. Validate the query for performance and correctness.
"""

```

# Пайплайн генерации ответа по исходникам



# Пример использования пайплайна поиска по исходникам




Return a full list of names of all recurrent functions called directly from itself




The list below contains the full set of names for all recurrent functions that call themselves directly as identified by the `search_codebase_graph` function with the query "find functions that call themselves":

1. MemoryContextStatsInternal
2. dsa\_free
3. sum\_free\_pages\_recurse
4. FreePageManagerDumpBtree
5. getObjectDescription
6. getObjectIdentityParts
7. RenameTypeInternal
8. CheckAttributeType
9. ProcessStartupPacket
10. UtilityContainsQuery
11. CreateCommandTag
12. GetCommandLogLevel
13. deltraverse
14. removetraverse
15. cleartraverse
16. findconstraintloop
17. clonesuccessorstates
18. checkmatchall\_recurse
19. zaptreesubs
20. traverse\_lacons
21. CreateTriggerFiringOn
22. renametrig\_partition
23. EnableDisableTrigger
24. ExecAlterExtensionContentsRecurse
25. DefineIndex

# RAG по документации PostgreSQL

 у меня есть сервер с бд и ещё 2 резервных сервера , как построить кластер BiHA

 Чтобы построить кластер BiHA с одним сервером базы данных и двумя резервными серверами, следуйте этим шагам:

## 1. Установка PostgreSQL Pro:

- Установите `postgrespro-ent-16-contrib` и `pg-probackup-ent-16` на всех узлах кластера.

## 2. Инициализация Кластера:

- На первом сервере (лидер) выполните команду для инициализации кластера с использованием `bihactl init`. Эта команда создаст базовую конфигурацию и подготовит узел-лидера.

```
bihactl init \  
  --biha-node-id=1 \  
  --host=<узел_1> \  
  --port=5432 \  
  --biha-port=5433 \  
  --nquorum=2 \  
  --minnodes=2 \  
  --pgdata=<локальный_каталог_pgdata_лидера>
```

- После выполнения команды `bihactl` запросит пароль для роли `biha_replication_user`, который будет использоваться для подключения узлов-последователей к лидеру.

## 3. Запуск Лидера:

- Запустите базовый сервер на лидере.

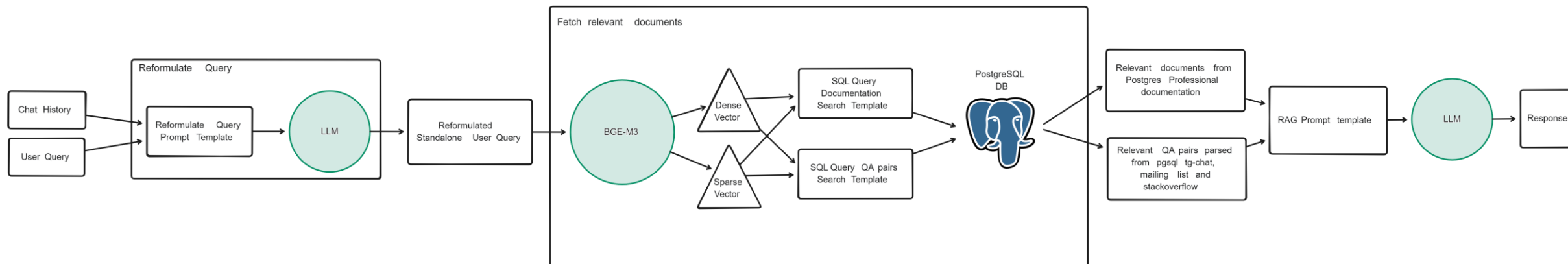
```
pg_ctl start -D <локальный_каталог_pgdata_лидера> -l <файл_журнала_лидера>
```

## 4. Добавление Узлов-Последователей:

- На каждом из двух резервных серверов выполните команду `bihactl add` для добавления их в кластер как узлы-последователи.

```
bihactl add \  
  --biha-node-id=<ID_узла> \  
  --host=<узел_X> \  
  --port=5432 \  
  --biha-port=5433 \  
  --pgdata=<локальный_каталог_pgdata_лидера>
```

# Пайплайн RAG



PosgresPro

Спасибо  
за внимание!

