

TRAPecista

TRAP Expert Chatbot for Integrated SQL and Technical Assistance

Stefano Sorrentino | TE-MS-C-TM | Università di Bologna

Supervisor: Franco Julio Mangiarotti

09/09/2024

The SM18 test facility

- **Tests** on the LHC magnets;
- Need to **keep track** of test procedures, activities, results, reports, steps, items, responsibilities;
- Expanding and dynamic environment: data structures have to be **flexible**;
- Allowing a precise and efficient **data storage and retrieval** is crucial;
- Need to **visualise** specific information from a large knowledge base.



The SM18 data ecosystem

- **Carvings DB (SM18 Test Database):**

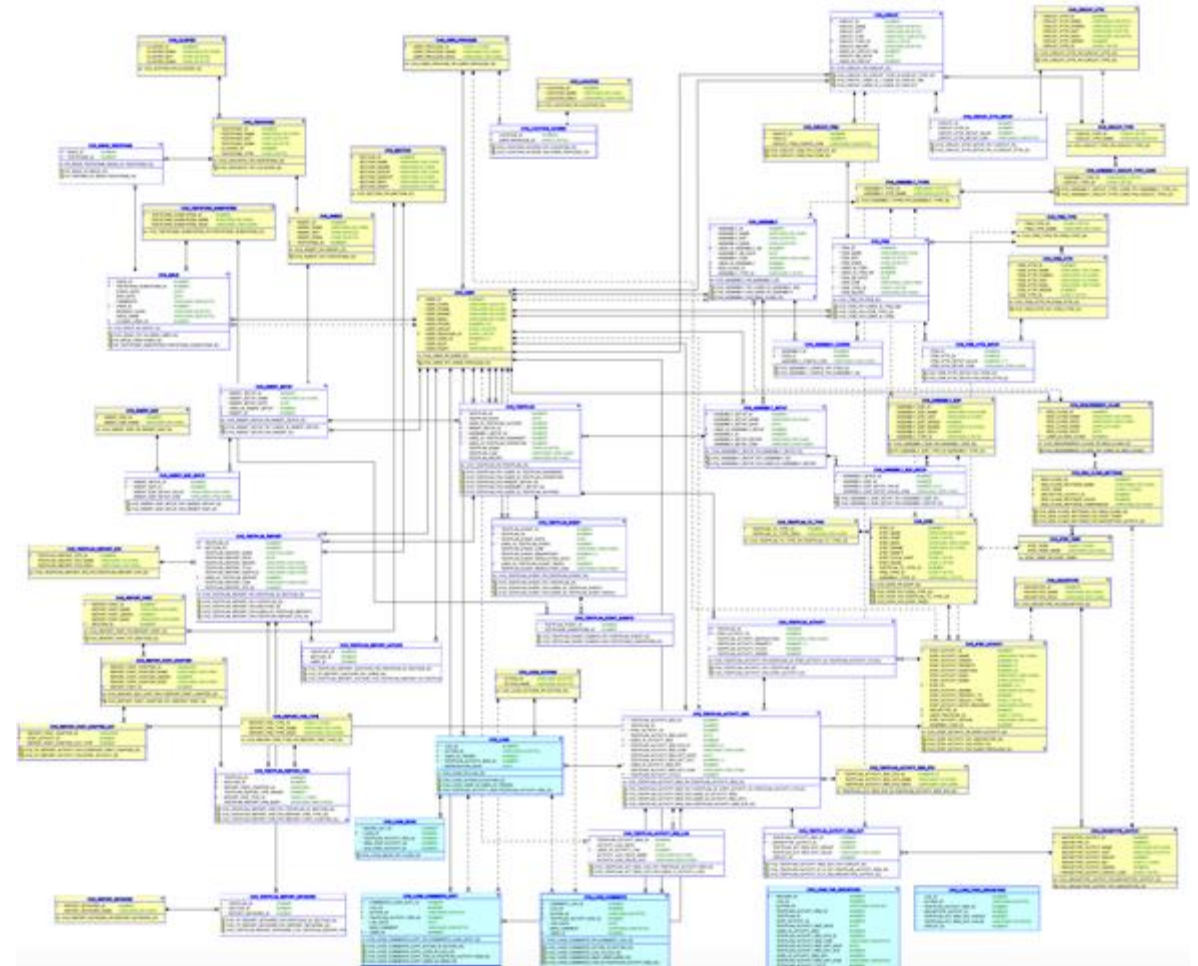
- Evolving through the years, now ~60 tables. Manages detailed test plans, activities, results, and configurations for magnets and related equipment at SM18;
- Extensive records, such as:
 - cvg_testplan_activity_res_out: **843,345** rows (single-valued test data entries)
 - cvg_testplan_activity_res: **18,638** rows (activities reports);
- Modular and complex schema.

- **Carpenter (DB web interface):**

- User interaction with the database;
- Data visualisation.

- **TRAP (Test Results Analysis Platform):**

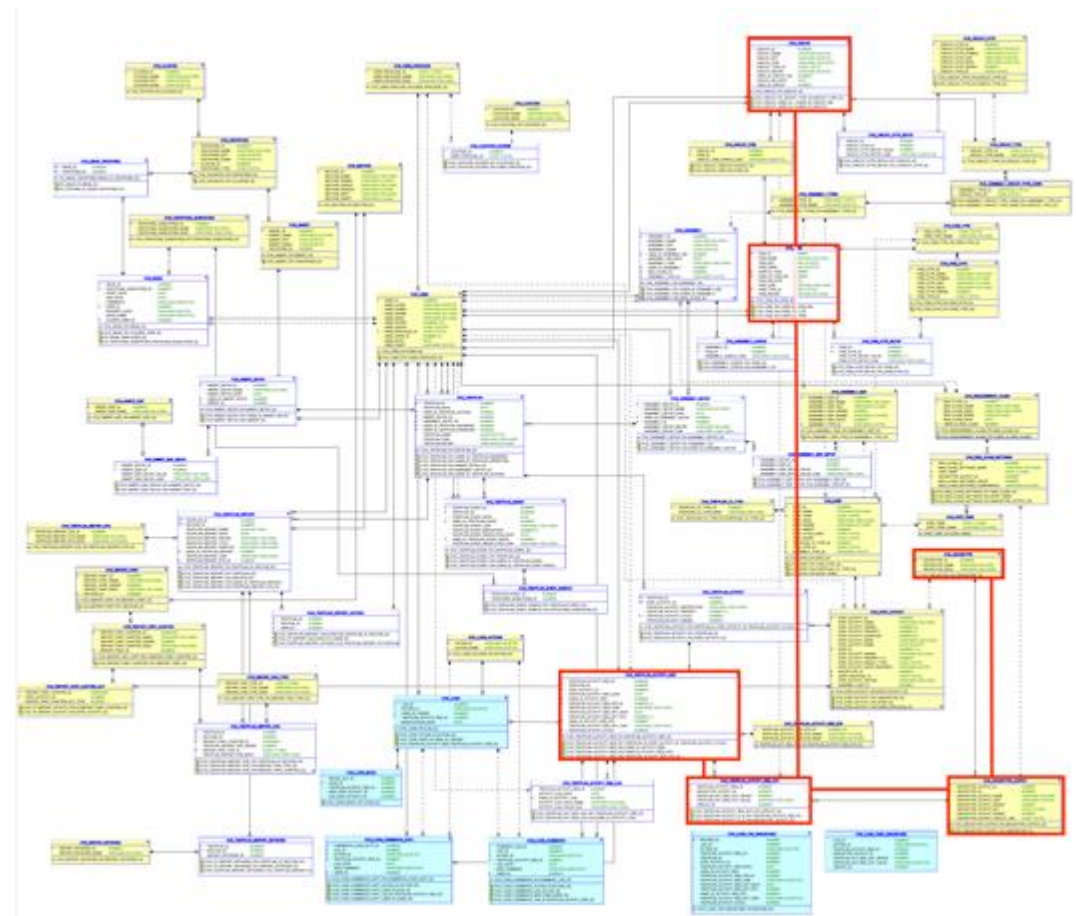
- Web application to run scripts interacting with Carvings;
- Upload, run and publish scripts on an isolated and containerized environment.



Retrieving nested information

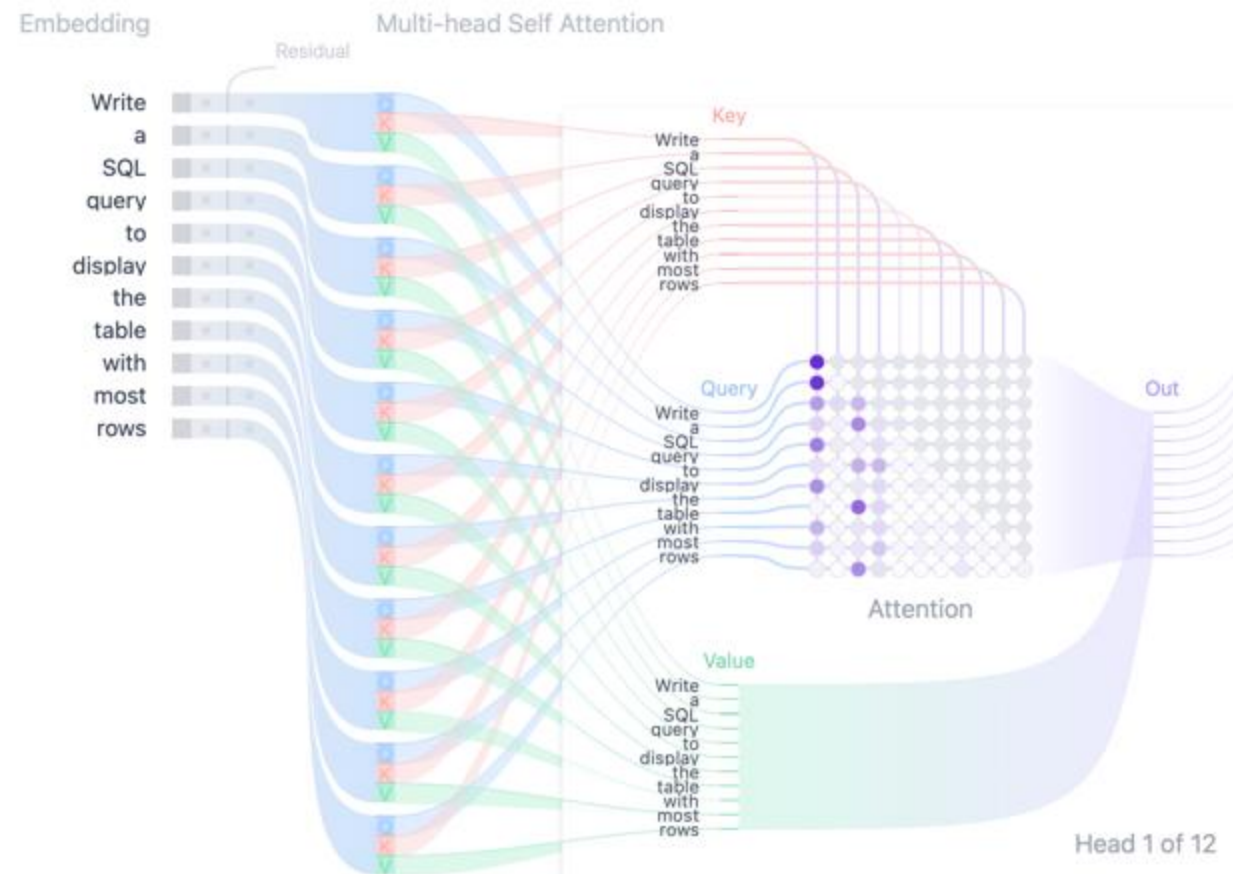
e.g. «write a query to retrieve for every hv test performed on magnets that start with mcb the ambient humidity and the test date» + TRAP script to plot the results

- **Identify Relevant Tables and Fields**
 - cvg_archetype: Contains archetypes (e.g., "HV test")
 - cvg_archetype_output: Maps outputs to archetypes (e.g., ambient humidity)
 - cvg_testplan_activity_res: Stores test results, including dates
 - cvg_testplan_activity_res_out: Stores output values from test results
 - cvg_circuit: Stores circuit details (e.g., names of magnets)
- **Explore relationships and cardinalities between tables**
- **Understand constraints and requirements, data types, characters, univoque names**
- **Construct and optimize the query (subquery, joins, filters)**
- **Understand TRAP application + python requirements**



Large Language Models (LLMs): new tools for information management at CERN

- **Transformer** DL architecture, «**attention**» mechanism¹: **context** is key to shape semantics of word embeddings;
- Scaled up **pre-trained** LLMs perform well when it comes to task-oriented **retrieval** and application of vast amounts of **knowledge that is provided as context**²;
- ~ **20 LLMs use cases** at the moment. E.g:
 - chATLAS: QnA over ATLAS collaboration knowledge base;
 - IT-GPT: QnA over IT Dept documentation.
- **IT-GPT** Collaboration:
 - ~ 40 collaborators;
 - Weekly meetings;
 - LLM hosted through vLLM service (Meta-Llama-3.1-8B-Instruct).

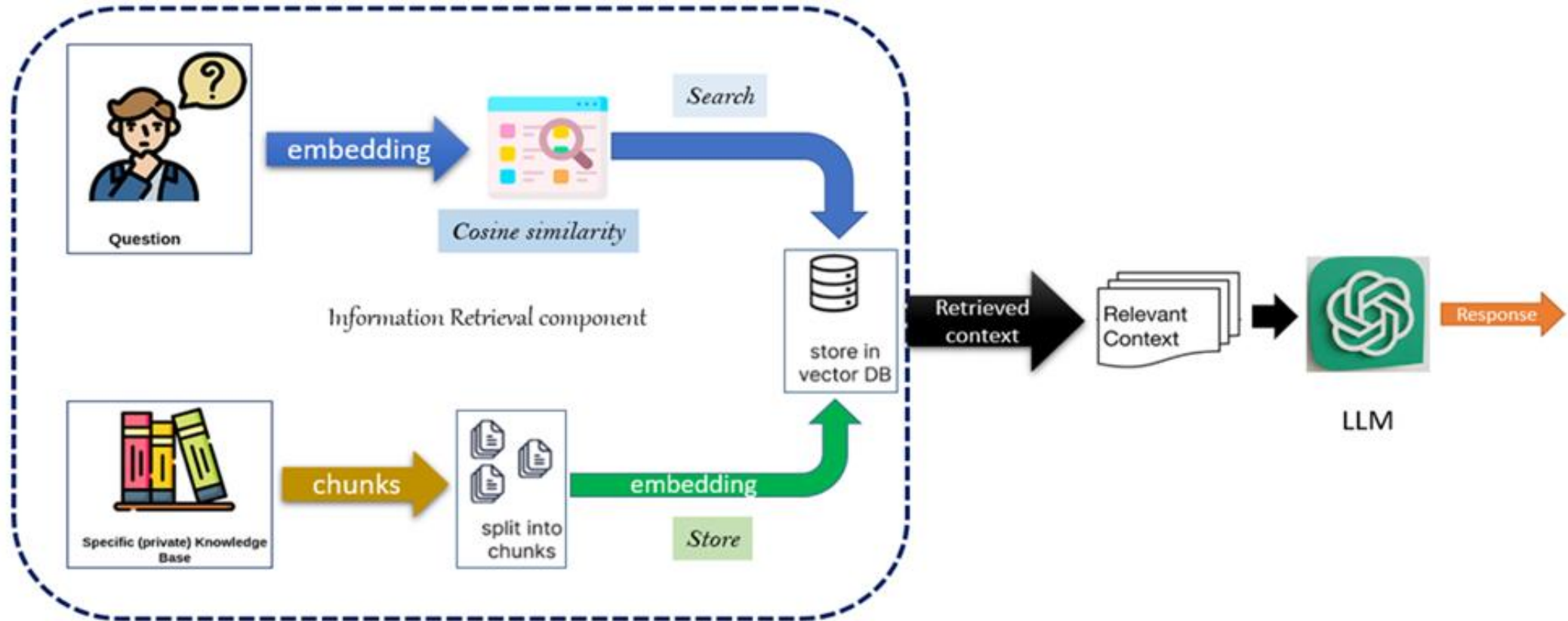


¹ Vaswani et al. «Attention is all you need». Google 2017

² Brown et al. «Language Models are Few-Shot Learners ». OpenAI 2020

Cho et al. «TRANSFORMER EXPLAINER: Interactive Learning of Text-Generative Models ». 2024

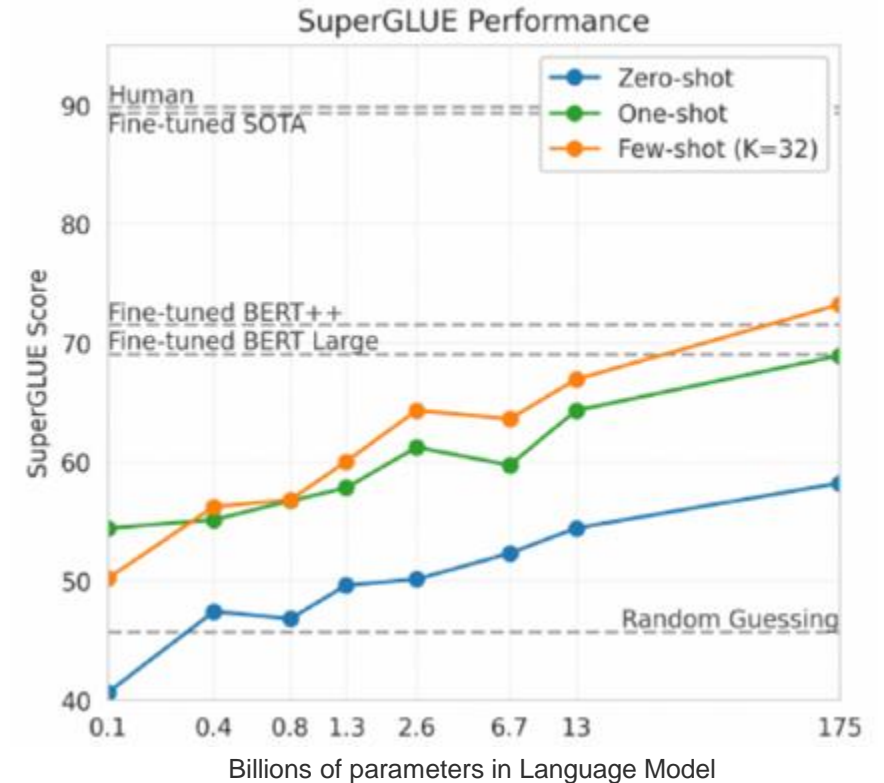
RAG: Retrieval Augmented Generation



Abdelazim et al. «Semantic embeddings for arabic Retrieval Augemented Generation». 2023

The value for SM18: from text documents to tabular data as context

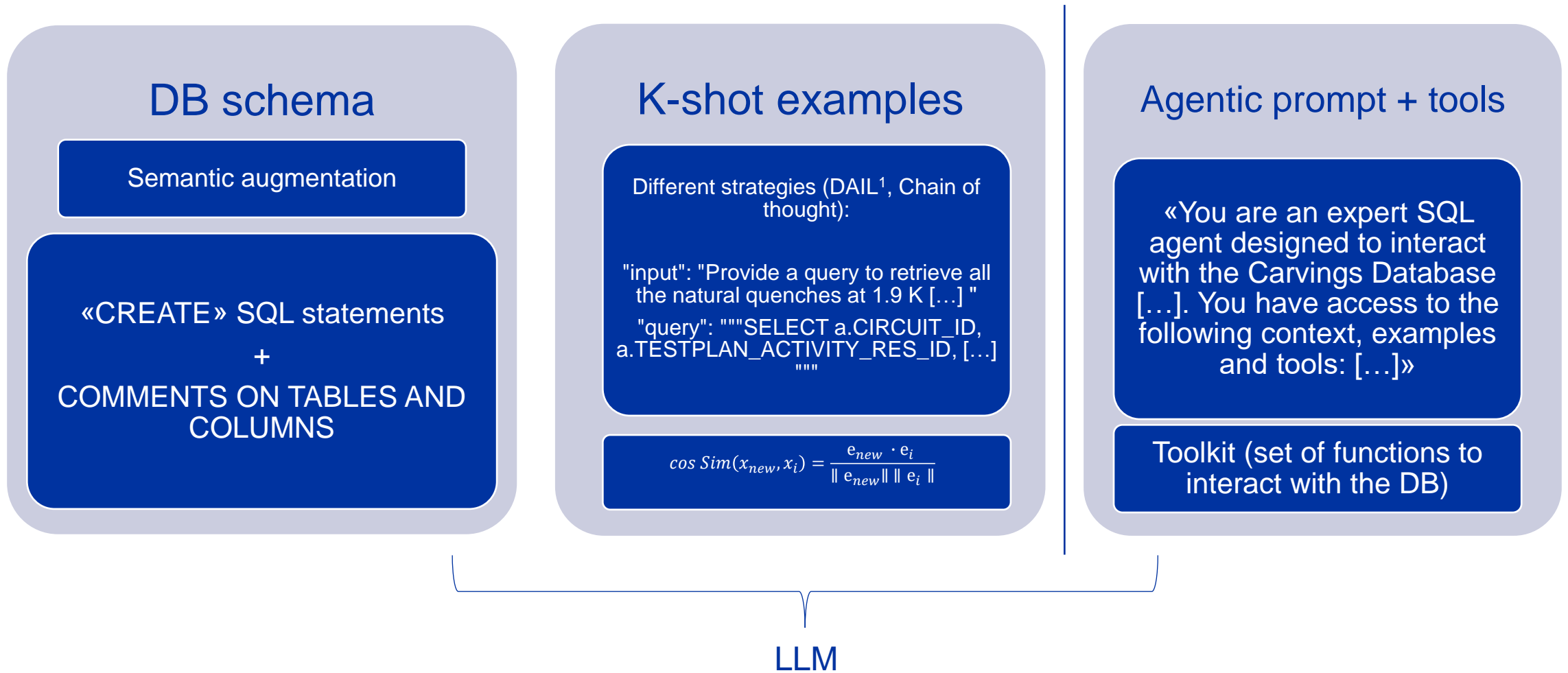
- A **Text-To-SQL** assistant;
- Leveraging the power of pre-trained LLMs as «SQL experts» through prompt-design¹. **Scalable** efficiency;
- Providing the Carvings **Database schema and few-shot example queries** as context;
- **Help TRAP application users** in writing complex queries, retrieve data and interact with the database;
- Without any fine tuning (for now):
 - Resource efficient;
 - Flexible;
 - Rapid prototyping;
 - Comparable efficiency scores relying on built-in SQL knowledge.



Lester et al. «The Power of Scale for Parameter-Efficient Prompt Tuning». Google Research, 2021

¹Anonymous submission. «Exploring Example Selection for Few-shot Text-to-SQL Semantic Parsing». Association for Computational Linguistics 2022

Building an efficient prompt (and agent)

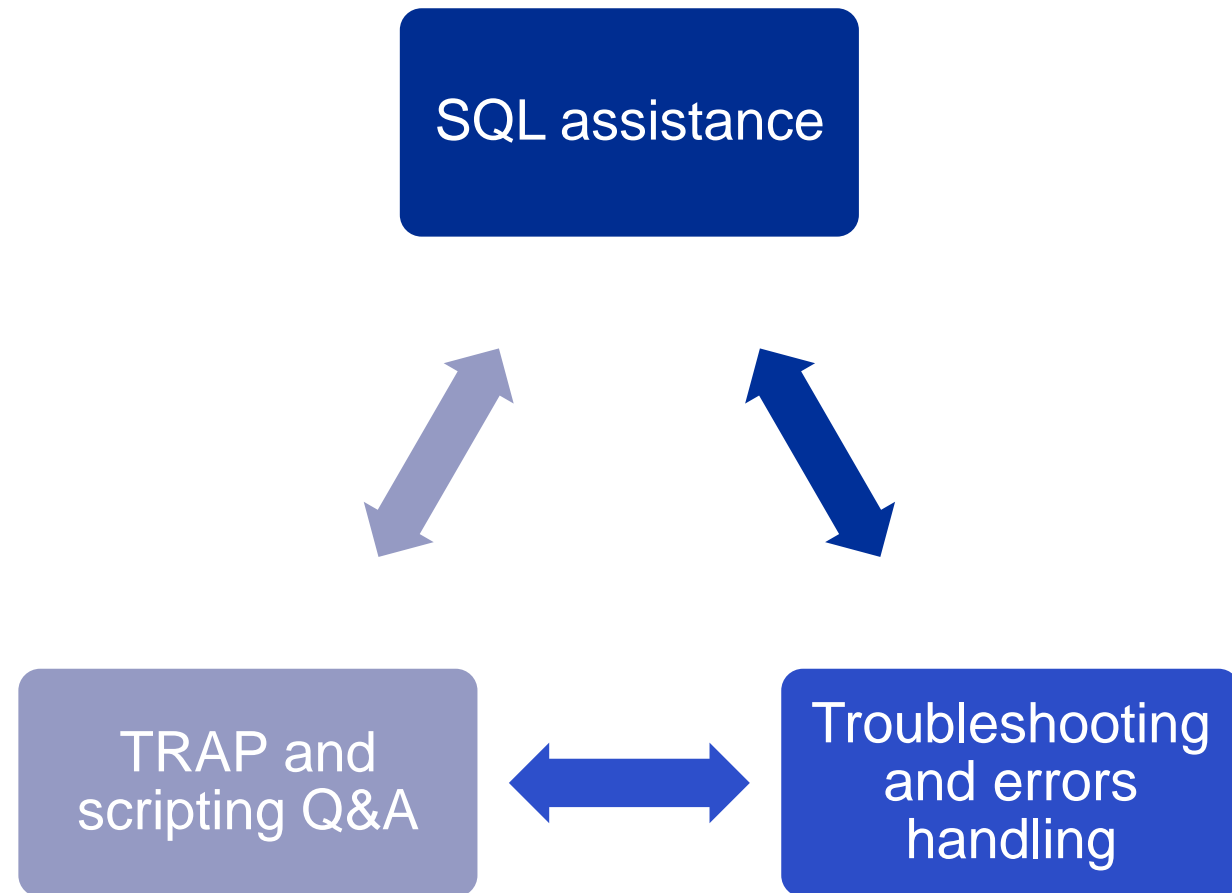


¹ Gao et al. «Text-to-SQL Empowered by Large Language Models: A Benchmark Evaluation». AlibabaGroup 2020

TRAPecista: a multi-agent approach

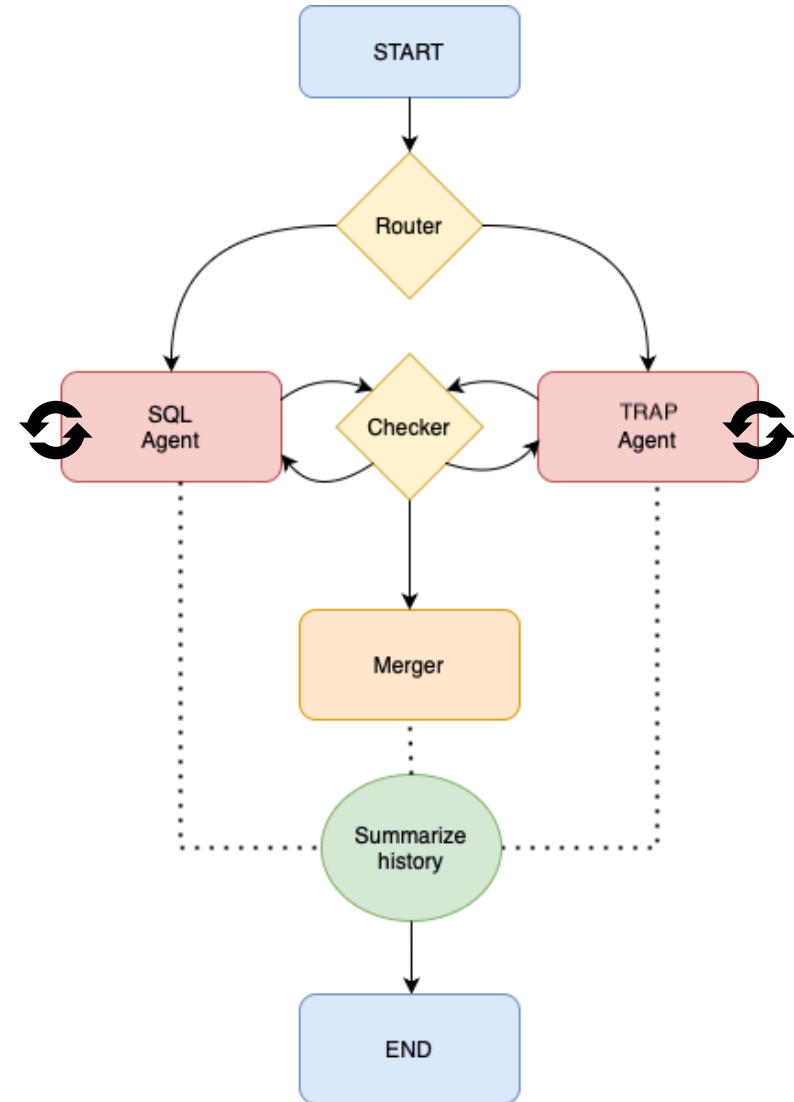


«Le trapèze volant», Henri de Toulouse-Lautrec. 1899



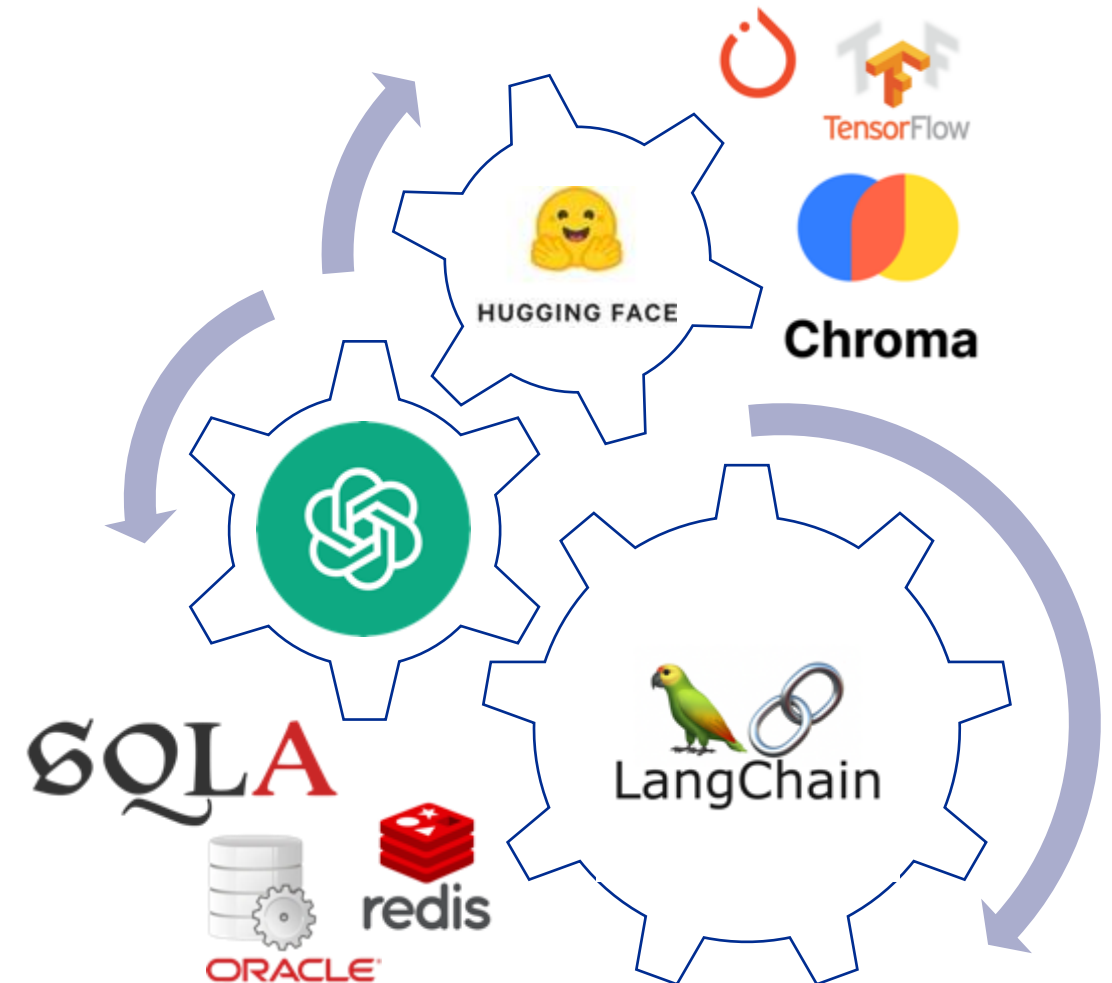
Architecture and graph flow

- **START:** handles conversion of plain input text into LLM processable message + user session + previous interaction history + graph state class initiation;
- **Router:** conditional edge function (prompt | LLM | JSONparser() chain);
- **SQL:** re-act agent to handle SQL-related tasks;
 - ❑ sql_db_query
 - ❑ sql_db_schema
 - ❑ sql_db_list_tables
 - ❑ sql_db_query_checker
 - ❑ search_proper_names
- **TRAP:** re-act agent to handle TRAP-related tasks;
 - ❑ retrieve_trap_documentation
- **Checker:** conditional edge function (prompt | LLM | JSONparser() chain), determines if the answer requires input from the other agent
- **Merger:** prompt | LLM chain, merges the SQL and RAG answers into a final one to be provided to the user
- **Summarize history:** summarizes message history into one message if messages list is too long.
- **END:** updates history, parses LLM outputs to be user readable



Main components under the hood

- **Langchain:** most used Open Source Python framework to build LLM-based applications. Supports chains, agents, graphs, performance monitoring and FASTAPI serving across a variety of models and output formats.
- **Models:**
 - Llama-3.1-8B-Instruct: 128k tokens c.w. (CERN hosted)
 - GPT-4o-mini: 128k tokens c.w. (Open AI API 0.150 \$ / 1M input tokens. 750 words = ~ 1000 tokens)
- **Embedding model:** Hugging Face all-MiniLM-L6-v2 («maps sentences & paragraphs to a 384 dimensional dense vector space and can be used for tasks like clustering or semantic search.»)
- **Vector databases:** Chroma (stores vector embeddings of documents and examples)
- **NoSQL databases:** redisDB (for history and session management)
- **SQLAlchemy:** DB interaction Python library
- **Carvings Oracle DB**



Safety and privacy notes

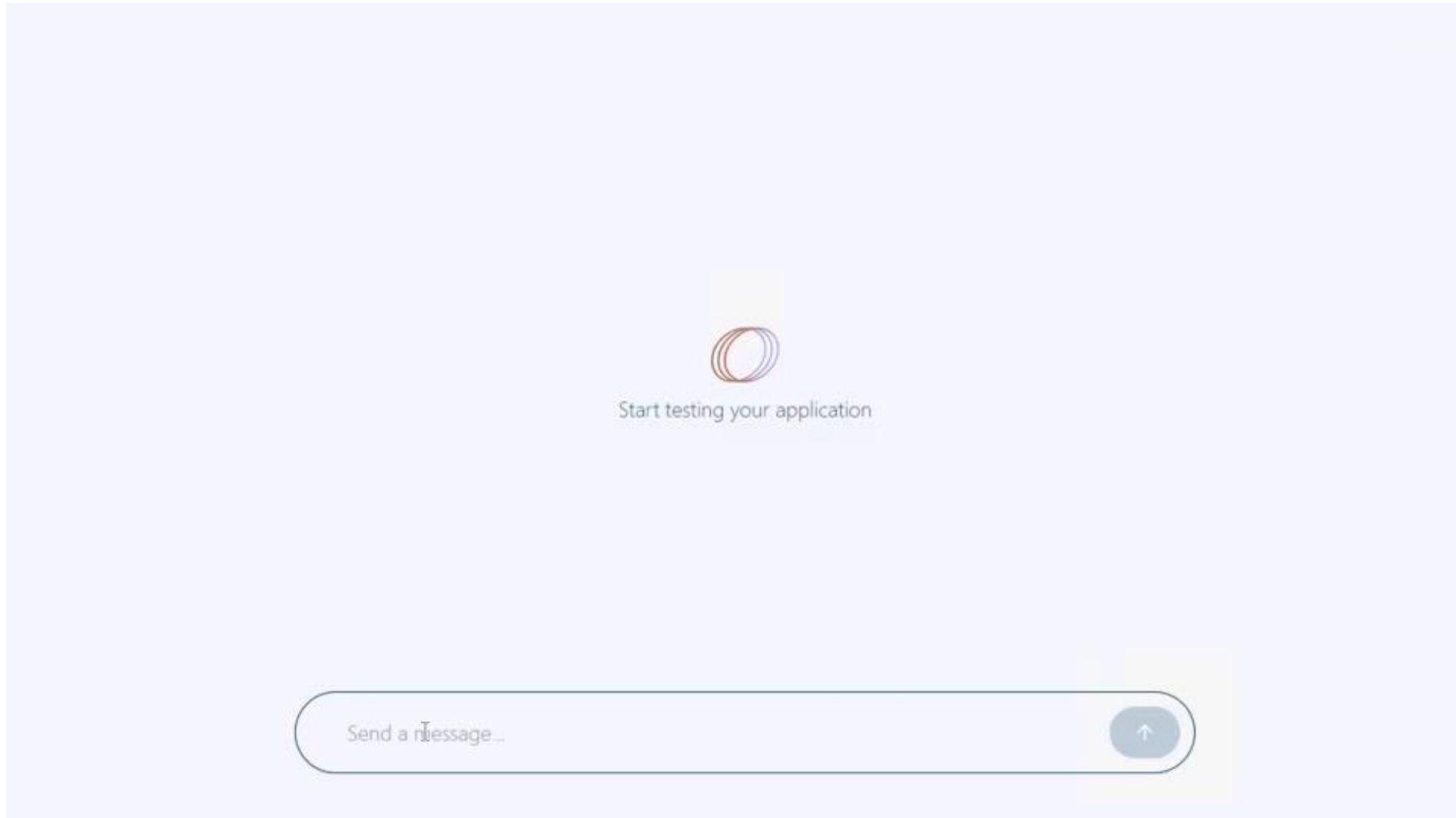
- TRAPecista cannot make or commit any change to the Carvings production database;
- TRAPecista runs on the same isolated and containerized environment as TRAP, using CERN's PaaS / OpenShift;
- TRAPecista is served by CERN SSO authentication;
- Each conversational history is associated to a user's https session ID, which is stored ephemerally as a key in the RedisDB running in the same environment.
- OPEN AI API (used for prototyping) terms of use are compliant with the european GDPR - General Data Protection Regulation (*apply in the European Economic Area (EEA), Switzerland, or UK*)¹;
- OpenAI encrypts all data at rest (AES-256) and in transit (TLS 1.2+)²;
- OpenAI API doesn't retain any data for training of future model. Data is permanently delated after 30 days¹.

¹ <https://openai.com/policies/eu-privacy-policy/>

² <https://openai.com/enterprise-privacy/>

TRAPecista – Demo

(«write a query to retrieve every hv test performed on magnets that start with mcb» + «add ambient humidity and the test date» + TRAP script to plot the results)



TRAPecista – Demo (results)

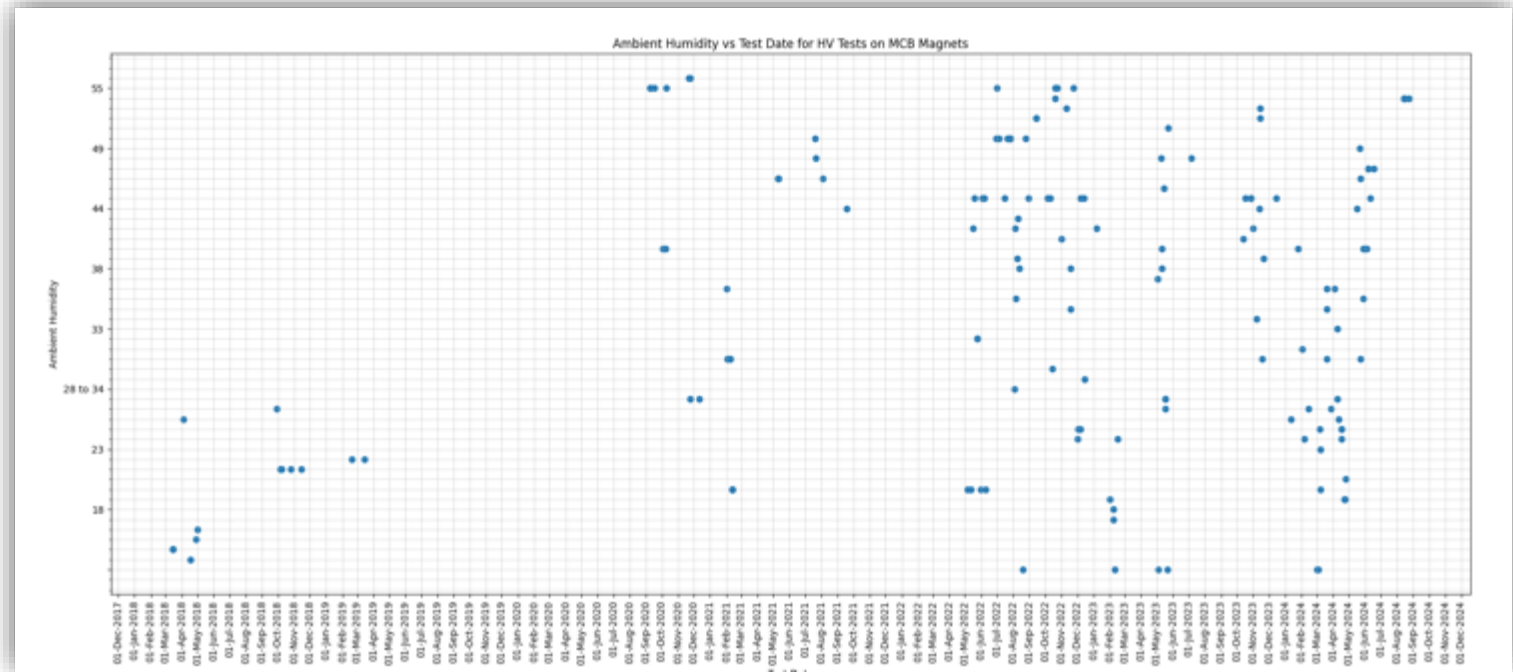
```
SELECT ARCHETYPE_ID
FROM cvg_archetype
WHERE archetype_name = 'HV test'

SELECT b.TESTPLAN_ACTIVITY_RES_DATE, a.TESTPLAN_ACT_RES_OUT_VAL
FROM cvg_testplan_activity_res_out a
JOIN cvg_testplan_activity_res b ON a.TESTPLAN_ACTIVITY_RES_ID
JOIN cvg_archetype_output e ON a.ARCHETYPE_OUTPUT_ID = e.ARCHET
WHERE b.TESTPLAN_ACTIVITY_RES_ID IN (
  SELECT DISTINCT al.TESTPLAN_ACTIVITY_RES_ID
  FROM cvg_testplan_activity_res_out al
  JOIN cvg_circuit d ON al.CIRCUIT_ID = d.CIRCUIT_ID
  WHERE d.CIRCUIT_NAME LIKE 'MCBA'
)
AND e.ARCHETYPE_OUTPUT_NAME = 'Ambient Humidity'
AND e.ARCHETYPE_ID = (SELECT ARCHETYPE_ID FROM hv_test_archetyp
ORDER BY b.TESTPLAN_ACTIVITY_RES_DATE ASC;
```

Query Result x

SQL | Fetched 50 rows in 0.505 seconds

TESTPLAN_ACTIVITY_RES_DATE	TESTPLAN_ACT_RES_OUT_VALUE
1 14-MAR-18	15
2 14-MAR-18	15
3 15-MAR-18	15
4 03-APR-18	26
5 16-APR-18	13
6 27-APR-18	16
7 30-APR-18	17
8 28-SEP-18	27
9 05-OCT-18	21
10 08-OCT-18	21
11 25-OCT-18	21
12 14-NOV-18	21
13 19-FEB-19	22
14 14-MAR-19	22



↑ TRAP generated plot using the script provided by TRAPecista

← Direct SQL Query execution over Caervings DB using ORACLE SQL Developer environment

TRAPecista – Demo (Q&A)

HUMAN

how do I write a TRAP script that plots the activities of testplan 1141 vs their date?
Do that using TRAP builtin module for interacting with carvings' tables rather than using the raw_query

AI

To write a TRAP script that plots the activities of test plan 1141 against their dates using the built-in modules for interacting with the Carvings database, you can follow this structure. Note that you'll need to import the appropriate modules from

```
import matplotlib.pyplot as plt
from datetime import datetime

# Fetch activities for test plan 1141
activities = cvg.CvgTestplanActivity.query().filter(cvg.CvgTestplanActivity.testpla

# Extract dates and activity details
dates = [activity.testplan_activity_res_date for activity in activities] # Ensure
activity_details = [activity.step_activity_name for activity in activities] # Ensu

# Convert dates to datetime objects if necessary
dates = [datetime.strptime(date, '%Y-%m-%d %H:%M:%S') for date in dates] # Adjust

# Plotting the results
plt.figure(figsize=(10, 5))
plt.plot(dates, activity_details, marker='o')
plt.title('Activities of Test Plan 1141 Over Time')
plt.xlabel('Date')
plt.ylabel('Activity Details')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

↑ Asking TRAPecista to write a TRAP script using built-in functions

HUMAN

How do I manage permissions in trap?

AI

To manage permissions in TRAP, you can follow these guidelines based on the structures available in the TRAP database:

1. **Viewing Permissions**:

You can check the permissions that have been granted to you by looking at the **TRAP_USER_PERMISSIONS** table. This table contains a list of all available permissions for users and scripts, including their names and descriptions.

2. **Requesting Permissions**:

If you need additional permissions, you can request them through the **TRAP_USER_PERMISSION_REQUEST** table. Here, you can submit a request for specific permissions, which will need to be approved or denied by an administrator. When making a request, you will need to specify:

QnA over TRAP functionalities ↑

TRAPecista – Demo (fails)

HUMAN

are you sure I can display the results like that in trap?

AI

You're right; I apologize for the confusion. In the TRAP environment, to display results, you would typically use the 'print' function rather than the `'trap_displaydisplay'` function as I previously suggested.

Here's the corrected way to display the results in TRAP:

↑ *RAG hallucination example: wrong information about displaying results in TRAP is provided*

HUMAN

ok, I see that every activity result output in which there is a trigger type output is repeated multiple times, and it creates redundancy in the results. why is that?

AI

The redundancy in your results, where every activity result output with a trigger type appears multiple times, can be attributed to the nature of the joins in your SQL query. Here are a few common reasons that might cause such duplication:

1. **Multiple Joins**: If you are joining tables that have multiple matching rows for

↑ *Graph flow problem: the RAG agent is invoked instead of the SQL one to handle a query optimization task*

I get this error: ORA-00904: "B"."STEP_ID": invalid identifier
00904. 00000 - "%s: invalid identifier"

*Cause:

*Action:

Error at Line: 10 Column: 5

AI

The error you're encountering, "ORA-00904: 'B'.STEP_ID: invalid identifier," suggests that the `'step_id'` field does not exist in the `'cvg_testplan_activity_res'` table, which is likely the source of the problem.

The SQL agent's analysis indicates that instead of using `'step_id'`, you should use the `'step_activity_id'` from the `'cvg_testplan_activity_res'` table and join it with the `'cvg_step_activity'` table to retrieve the `'step_id'`.

Here's a revised version of your query that addresses the issue:

↑ *SQL hallucination example: a wrong query is provided, resulting in an Oracle Error*

Error

GraphRecursionError('Recursion limit of 25 reached without hitting a stop condition.')

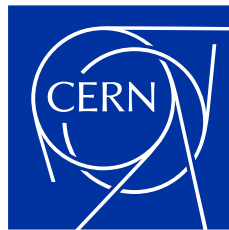
↑ *Agent recursion error: the SQL agent fell in a tool-execution loop when checking the accuracy of a query*

Challenges and future deployments

- Avoiding **hallucinations** / maintaining **interaction among agents and nodes**;
- Finding the best **hyper-parameters** (temperature, chunks sizes, k-examples, history size) for optimal performance;
- Handling **database updates**;
- Explore and evaluate performance of other approaches (**fine-tuning**, «**plan & execute**» examples);
- Design a better **UI**;
- Definitive migration to CERN hosted model.

Thank you for your attention!

Stefano Sorrentino | TE-MS-C-TM | Università di Bologna
Supervisor: Franco Julio Mangiarotti



home.cern