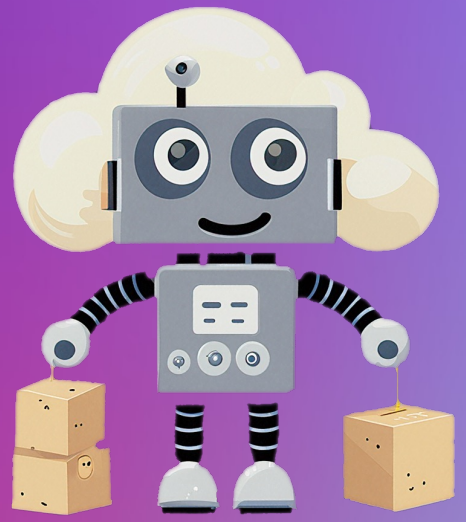
LIG105

Natural Language to SQL: The good, the bad and the ugly.

Philipp Kaindl

(he/him)
Sr. AI/ML Specialist Solutions Architect
Amazon Web Services



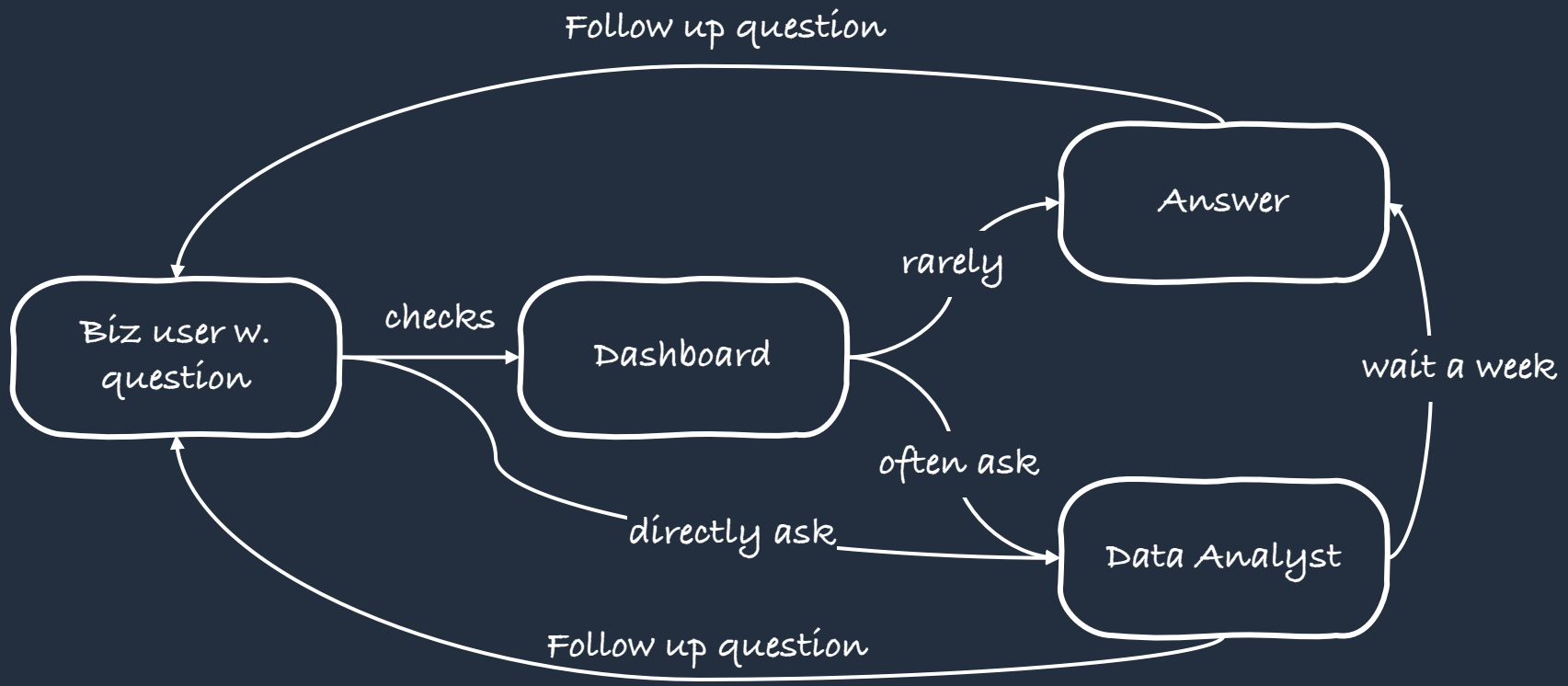
Agenda

- Why Natural Language 2 SQL
- Levels of autonomy
- Why things fail
- How we can make it better
- Wrap-up

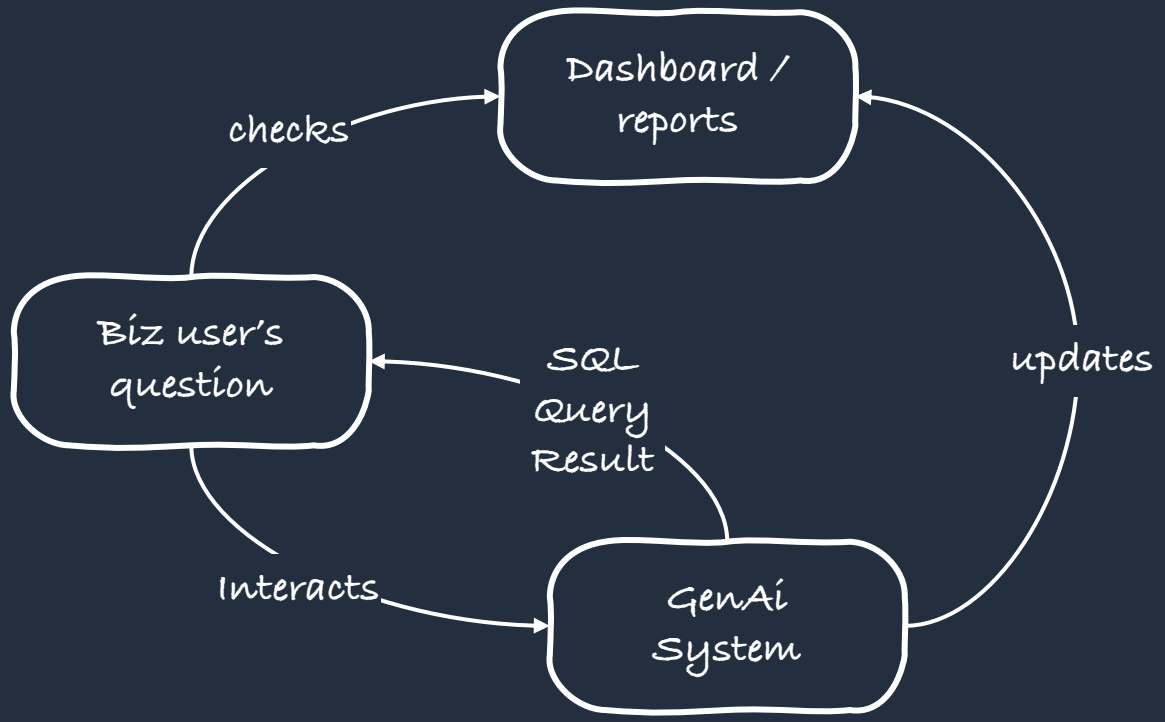
This image was generated by Titan Image Generator G1 v1 on Amazon Bedrock.

Problem Statement

Finding data to answer your business questions in SQL Systems is painful…



How would an GenAI powered experience look like?



Levels of autonomy

How much do we hand over to the GenAI system?

Level 1 – Human supported by GenAI system

- GenAI generates SQL statements
- GenAI helps navigate the business data catalog
- Executing queries on behalf of user

Level 2 – Human / GenAI collaboration

- GenAI generates SQL statements and makes suggestions for improvements, follow up questions, etc.
- If stuck, human unblocks GenAI

What is possible today

Level 3 – Full autonomy* of GenAI system

- Non technical users are enabled to answer diverse and complex queries, based on their companies data.
- System interacts efficiently with other systems to paint complete picture regarding questions.

What we will focus on today!

* GenAI system has ***Read Only*** rights DB and on other systems

Why things fail – Academic datasets are not the real world!

Why is NL2SQL hard in real-world databases (DB)?

Academic datasets	Real-world databases
Pre-organized, often pointing to the correct table	Not organized, no indication of right table
Maximum 100s of columns	Span 10s to 100s of tables; 100s to 1000s of columns
Column names are indicative in the context of the question	Column names can be misleading, oftentimes duplicated and outdated
If available, a business data catalog is defined	Often business data catalogue is inhomogeneous across different entities, and data derivatives differ
The DB is static	DB is ever-evolving
DB query latency is not of importance	Heavy read / join operations can have significant impact on the DBs performance

Benchmarks and bencmark model performance

Public Benchmarks:

- WikiSQL
- Spider
- BIRD (**BI**g Bench for La**R**ge-scale **D**atabase Grounded Text-to-SQL Evaluation)

Dataset	# Example	# DB	# Table/DB	# Row/DB
WikiSQL [58]	80,654	26,521	1	17
SPIDER [53]	10,181	200	5.1	2K
KaggleDBQA [24]	272	8	2.3	280K
BIRD	12,751	95	7.3	549K

https://arxiv.org/abs/2305.03111

In general, we can expect fine-tuned models will outperform base model for Text-to-SQL generation tasks.

LLMs Text-to-SQL capability evaluation before 20231104

the follow our experiment execution accuracy of Spider is base on the database which is download from the Spider official website ,size only 95M.

name	Execution Accuracy	reference
GPT-4	**0.762**	numbersstation-eval-res
ChatGPT	0.728	numbersstation-eval-res
CodeLlama-13b-Instruct-hf_lora	**0.789**	sft train by our this project,only used spider train dataset, the same eval way in this project with lora SFT. The weights has pulished .
CodeLlama-13b-Instruct-hf_qlora	**0.825**	sft train by our this project, used around 50 thousand pieces of text-to-sql data. The same eval way in this project with lora SFT, and we make sure the training set has filtered the spider eval dataset.

Fine tuned CodeLlama-13B outperforms GPT-4 base model for Text-2-SQL

https://github.com/eosphoros-ai/DB-GPT-Hub/blob/main/docs/eval_llm_result.md

How can we make things better?

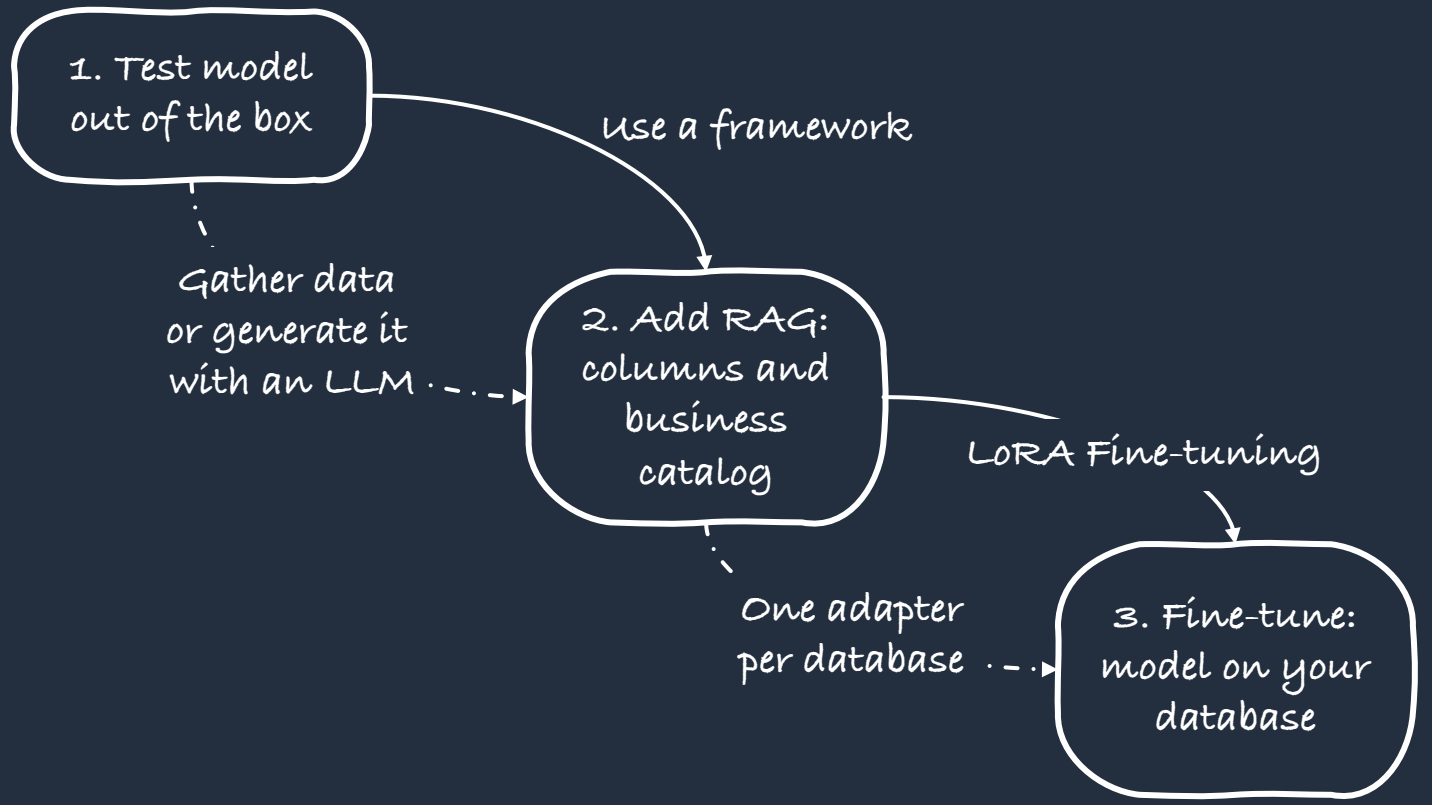
How to improve upon our challenges from real-world DBs?

Real-world databases	Improval technique	
Not organized, no indication of right table based on query	FT	RAG
Column names can be misleading, oftentimes duplicated and outdated	FT	RAG
Ofen business data catalogue is inhomogeneous across different entities, and data derivatives differ	FT	RAG
DB is ever-evolving		RAG
Heavy read / join operations can have significant impact on the DBs performance	FT	

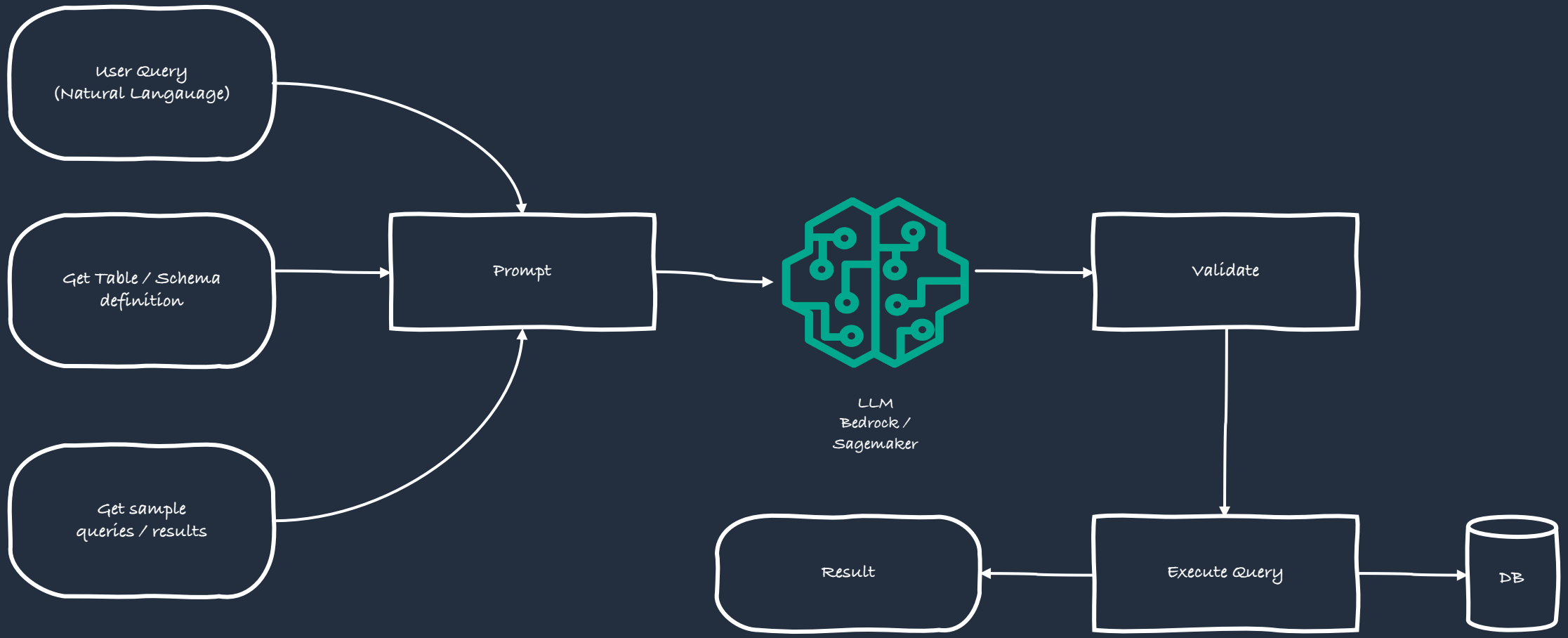
RAG = Retrieval Augmented Generation

FT = Fine-tuning; both FT of Text2SQL model, as well as FT of RAG model (e.g. cross-encoder model)

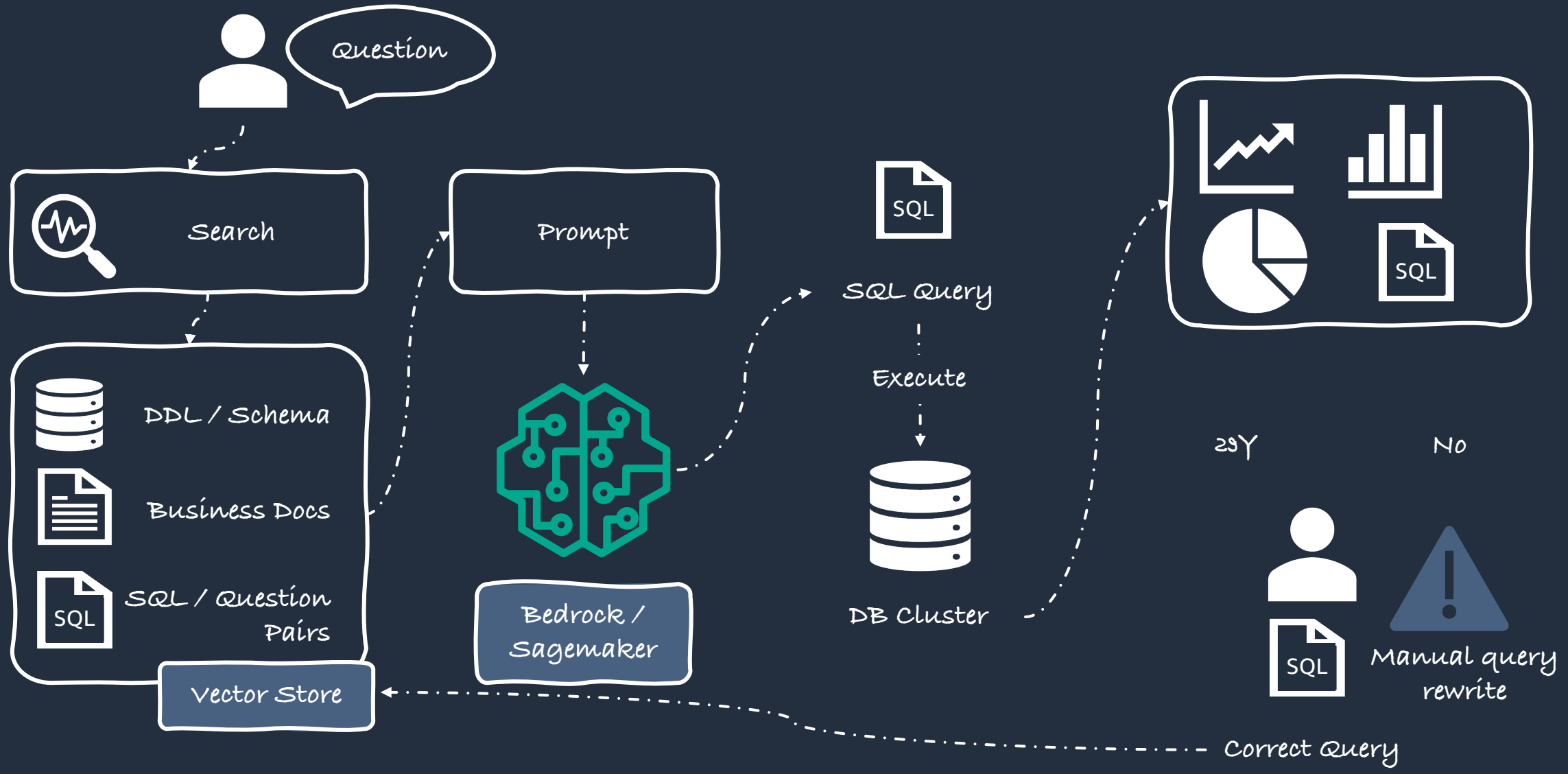
The NL2SQL journey of improvement



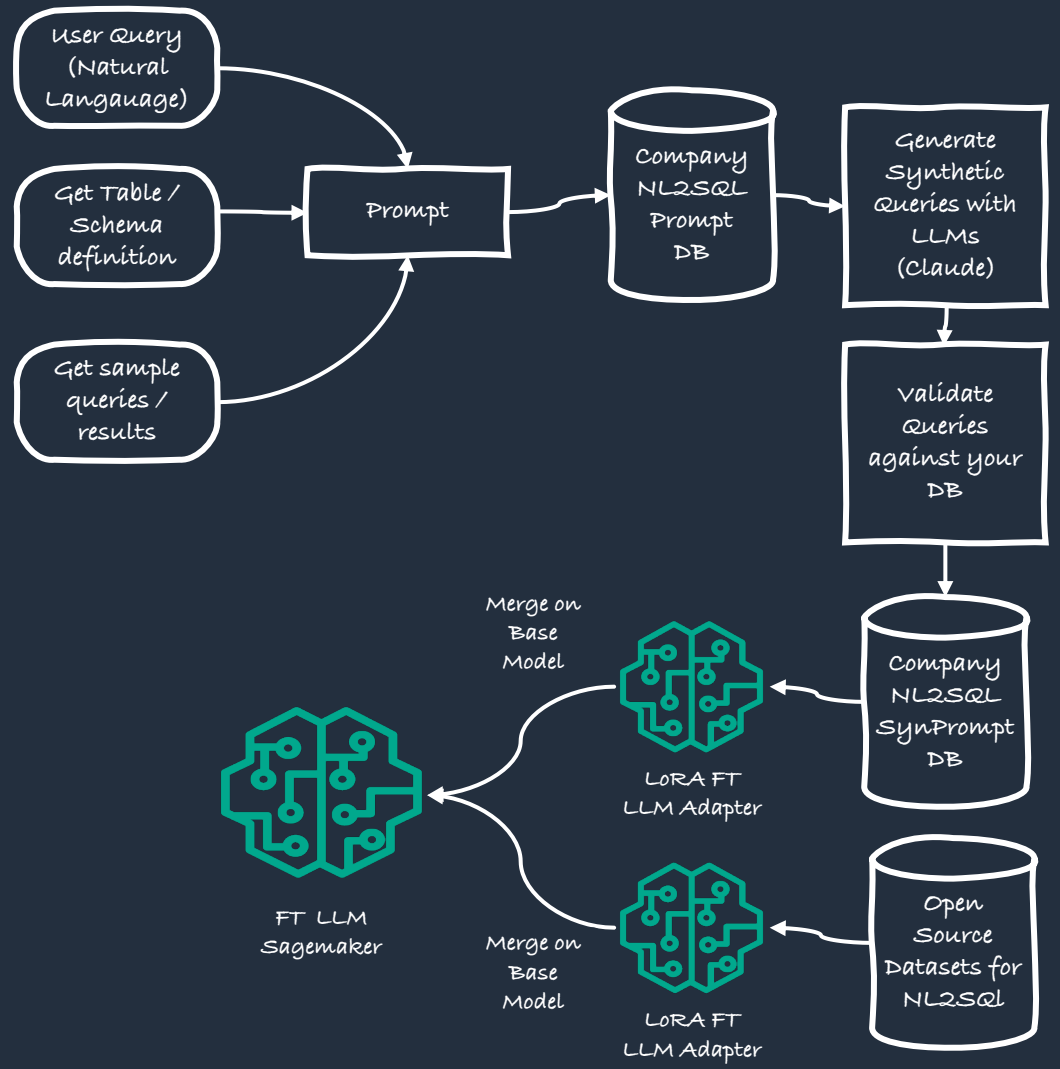
Journey stop 1 – prompt engineering



Journey stop 2 – RAG with Continous Improvement



Journey stop 3 – fine-tuning



Things to consider:

- Utilize LoRA Fine-Tuning
- Create a good base adapter for NL2SQL on your tables
- Fine tune one LoRA Adapter per DB
- Merge base NL2SQL adapter, load DB specific adapter

Results of different approaches

ON SPIDER EVAL DATASET – 20 DATABASES

Step 1 - Prompt Engineering *

- Execution Match Accuracy (EMA) improves from 39% > 88% (Claude 2)
- Few-shot with relevant examples is the highest driver of improvement

Cond: DB Schema + RAG-Few-Shot supplied based on question – Claude 2.0

Step 2 - RAG

- EMA improves from 5% > 58%
- RAG on Docs, Synth. Question + Query pairs, DB Schema's

Cond: All 20 DBs in Vector store; System selects and finds right DB – Claude Instant

Step 3 – Fine-tuning

- EMA improves from 38% > 69% with CodeLlama (zero-shot)
- Synthetic Query + Question pairs generated based on 20 target DBs.
- Training with 2000 examples 10 Epochs

Cond: DB Schema supplied based on question
- CodeLlama-13b and Titan Express

Changes have been made in isolation to judge fair impact on performance. Test results come from my own testing, no outside reference

* https://medium.com/@machangsha/a-comprehensive-guidance-of-prompt-engineering-for-natural-language-to-sql-20d4c9dea080

Wrap up

Natural Language 2 SQL

The good news – everyone wants it!

The bad news - It will take effort.

The ugly truth - Every DB is different and the devil is in the details!

Please complete the session survey.

Thank you!

Philipp Kaindl

philikai@amazon.de

linkedin.com/in/philipp-kaindl/

Medium: @philippkai

Further References

- SPIDER Dataset

- BIRD Dataset

- Open Source RAG NL2SQL Framework Vanna AI

- Fine-tuning CodeLlama 7b on NL2SQL Part 1