IBM Research

GraphQL Generation for Querying Data Lakehouse

Balaji Ganesan, Avirup Saha, Manish Kesarwani, Nitin Gupta, Sambit Ghosh,
Renuka Sindhgatta, Carlos Eberhardt, Dan Debrunner, Sameep Mehta

September 10, 2024

bganesa1@in.ibm.com

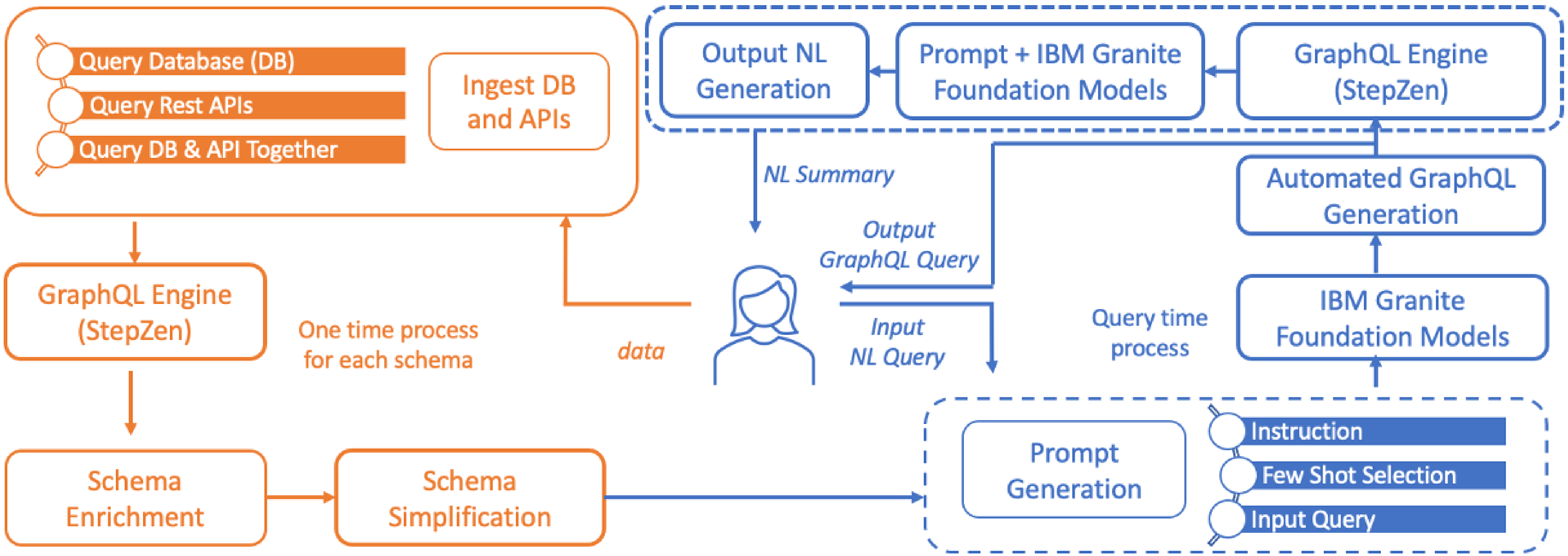
Why not SQL and Rest API?

SQL	REST API	GraphQL
Databases - Designed for relational databases.	**Databases -** Designed for variety of databases, including relational and non-relational databases.	**Databases -** Designed for variety of databases, including relational and non-relational databases (like REST API). But GraphQL's flexibility can simplify integration with different data sources.
Data Fetching - Direct queries to a database.	**Data Fetching -** Predefined endpoints for resources.	**Data Fetching -** Single flexible endpoint, custom data fetching.
SELECT first_name, last_name FROM employees WHERE department_id = 101; Directly queries a relational database for specific fields from the 'employees' table based on a condition.	GET /api/employees?departmentId=101 Predefined endpoint for retrieving employee data in a specific department.	query { employees(departmentId: 101) { firstName lastName } } Single flexible endpoint allows clients to request only the required data for employees in a specific department.
Performance Considerations - Efficient for database operations.	**Performance Considerations -** Multiple requests, potential over-fetching.	**Performance Considerations -** Single request, precise data retrieval.

SELECT
e.firstName,
e.lastName,
e.salary,
d.name
FROM employee e JOIN department d
ON e.department_id = d.id
WHERE e.department_id = 101
A SQL query fetching detailed information about employees and their departments.
	GET /api/employees
	GET /api/departments
	
	Requires multiple requests to fetch both employee and department data.
		query {
		employees(departmentId: 101)
		{
		firstName
		lastName
		salary
		department { name }
		} }
		A GraphQL query fetching detailed information about employees and their departments.

Over-fetching/Under-fetching Single request for precise data retrieval reduces network overhead. However, it only works for relational databases, and a little more complex in logic as compared to GraphQL.	**Over-fetching/Under-fetching** The response may include more data than needed, impacting bandwidth.	**Over-fetching/Under-fetching** • Single request for precise data retrieval reduces network overhead. • Allows clients to specify exactly the data they need, preventing over-fetching.

LLM-powered GraphQL Generator for Data Retrieval



Ganesan, Balaji, Sambit Ghosh, Nitin Gupta, Manish Kesarwani, Sameep Mehta, and Renuka Sindhgatta. "LLM-powered GraphQL Generator for Data Retrieval." In *International Joint Conference on Artificial Intelligence*. 2024.

LLM-powered GraphQL Generator for Data Retrieval

IBM **Research**

Natural Language to GraphQL

Setup Schema

Select a Test Scenario

- ○ Query Databases
- ○ Query REST APIs
- ● Query Databases and REST APIs together
- ○ Custom sources

Scenario Name: Query Databases and REST APIs together

Schema

```
type People {
  Birth_Date: String
  Birth_Place: String
  Height: Float
  Name: String
  People_ID: Int!
  Weight: Float
  body_builder: [Body_builder]
   @materializer(query: "body_builderUsingBody_builder_People_ID_fkey")
}
```

Schema

```
type Query {
  bb_stats: Body_builder_stats
   @rest(endpoint: "http://host.docker.internal:6003/")
}
```

Add Custom Source

Design Prompts

Natural Language Query:

Just give me the snatch of all the body builders and the average snatch

Choose a model:

- ● Granite-20B-Code Base Model
- ○ Granite-20B-Code Instruct Model

Few shot strategy

5

Generate Prompt

Instruction:

Your task is to write an API request for a custom database schema based on the API reference provided. For guidance on how to correctly format this API request, consult the API reference here: Note: Please

Few shots Prompt:

```
Training Example 0:

CUSTOM SCHEMA:
```

```
type Query {
 """
 return list of persons
 """
```

Test Query:

```
Test Example:

CUSTOM SCHEMA:
```

```
type Body_builder {
  Body_Builder_ID: Int!
  Clean_Jerk: Float
  Snatch: Float
```

Generate GraphQL

Natural Language Query:

Generated GraphQL:

```
graphql {
operationName = ""
query =
 query {
  body_builder(Body_Builder_ID: 1) {
   Body_Builder_ID
```

Execute GraphQL

```
{...}
```

Generate NL Summary

NL Summary: The name of the body builder is Jack Campbell. The height of the body builder is 182 cm. The weight of the body builder is 80 kg. The total time of the body builder is 317.5 seconds. The clean jerk of the body builder is 175.0 seconds. The snatch of the body builder is 142.5 seconds. The birth date of the body builder is January 1, 1992. The birth place of the body builder is Port Huron, Michigan.

API Sequencing for GraphQL Schema Generation

```
type Query {

location(ip: String!, lang: String! = "en"): IpApi_Location
        @rest(
          endpoint: "http://ip-
api.com/json/$ip?fields=64745471&lang=$lang"
          setters: [{ field: "ip", path: "query" }]
          )

weatherReport(openweather_appid: Secret!
     lang: String! = "en"
     lat: Float!
     lon: Float!):  Openweather_WeatherForecast
       @rest(
          endpoint:
"https://api.openweathermap.org/data/2.5/onecall?appid=$o
penweather_appid&lang=$lang&lat=$lat&lon=$lon&exclude=
minutely%2Chourly"
          setters: [{ field: "clouds", path: "current.clouds" },{
field: "temp", path: "current.temp" } ]
          )

# many other type queries

  }
```

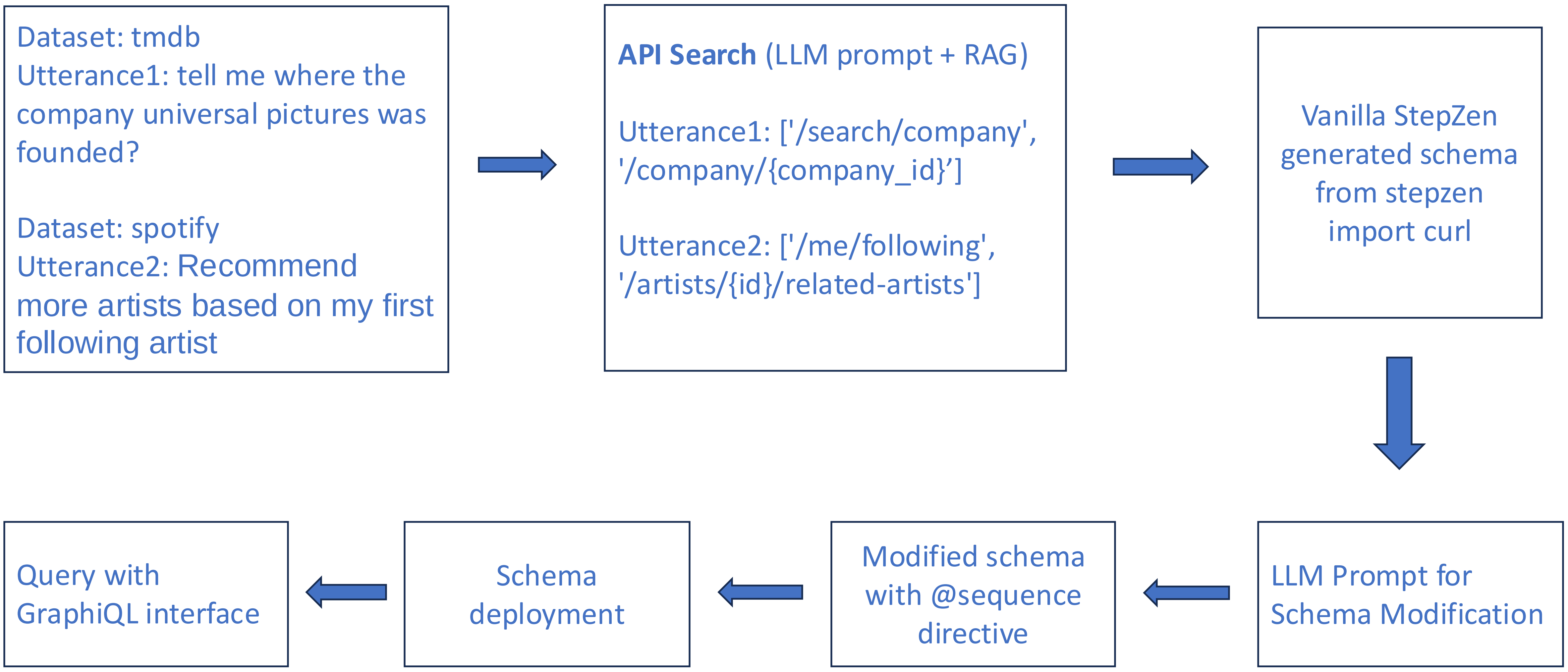
```
 weather(ip: String! openweather_appid: Secret! lang: String! = "en"):
Openweather_WeatherForecast
     @sequence(
       steps: [
         { query: "location"}
         { query: "weatherReport", arguments: [{name: "openweather_appid", argument:
"openweather_appid"}] }
         ]

   )
```

```
query MyQuery {
 weather(
  ip: "72.188.196.163"
  openweather_appid:
"b4548ecd778518b766619e797744de85"
 ) {
    clouds
    temp
  }
}
```

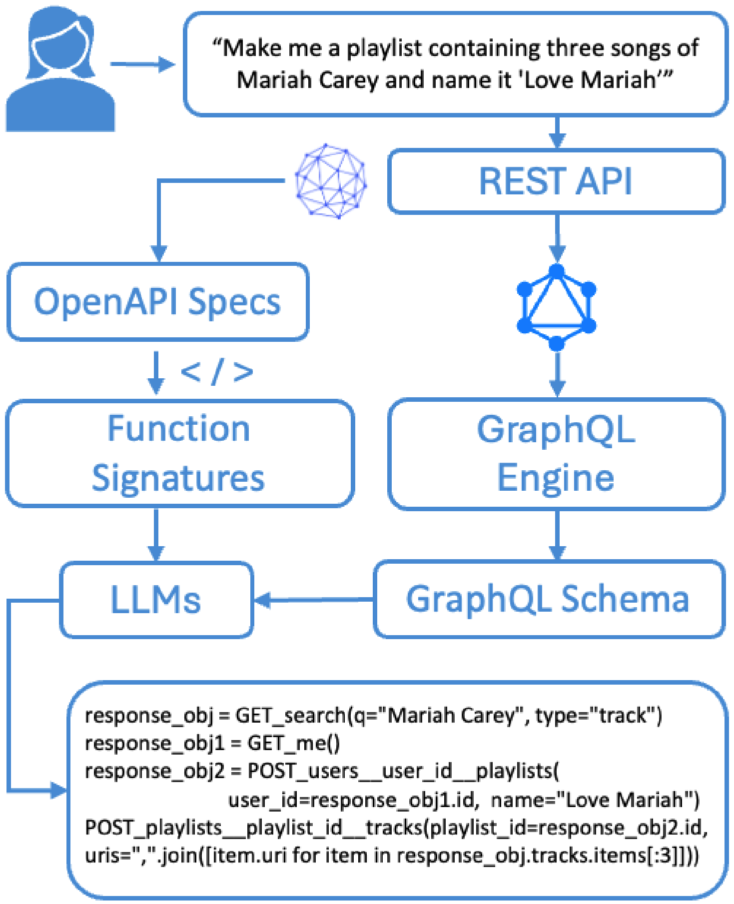

```
{
 "data": {
  "weather": {
   "clouds": 40,
   "temp": 303.5
  }
 }
}
```


The task is to identify and order location and weatherReport type queries to use with the @sequence directive

API Sequencing and Schema Modification



Sequential API/Function Calling Using GraphQL Schema



Spotify Example

Add the first song of The Dark Side of the Moon in my playback queue

```
response_obj = GET_search(q="The Dark Side of the Moon", type="album")
response_obj1 = GET_albums__id__tracks(id=response_obj.tracks.items[0].id)
POST_me_player_queue(uri=response_obj1.items[0].uri)
```

TMDB Example

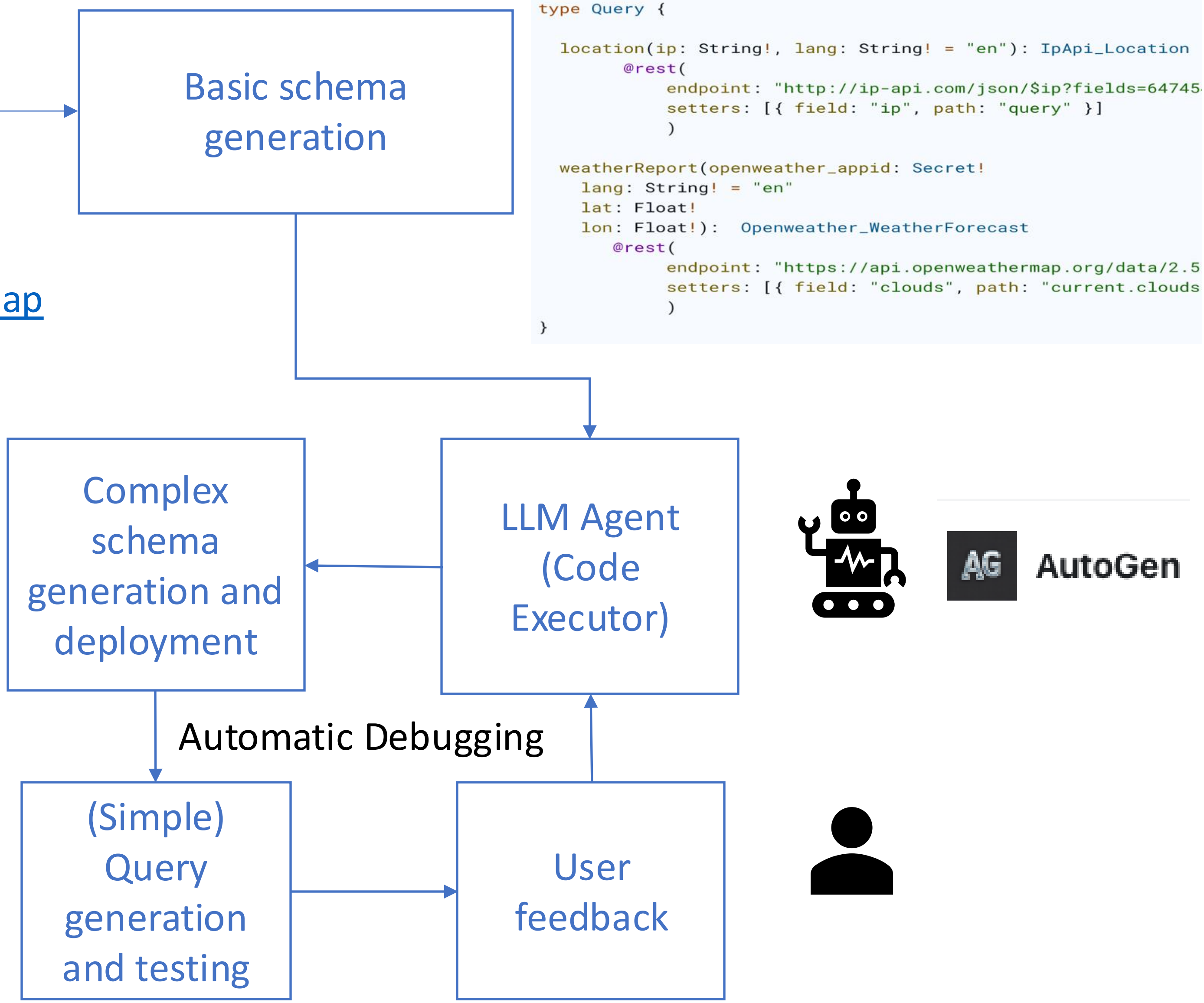
give me the number of movies directed by Sofia Coppola

```
response_obj = GET_search_person(query="Sofia Coppola")
GET_person__person_id__movie_credits(person_id=response_obj.results[0].id)
```

Model	Prompt Style	Test split	Arg Match (full)	Arg Match (functions)	Seq Match (full)	Seq Match (conn. subseq.)
codellama-34b-instruct	CoT	overall	0.6875	0.8051	0.9062	0.9375
deepseek-coder-33b-instruct	CoT	overall	0.7500	**0.8701**	**0.9687**	**1.0000**
granite-34b-code-instruct	CoT	overall	**0.7812**	**0.8701**	0.9375	0.9687
codellama-34b-instruct	ReAct	overall	0.7188	0.8182	0.9062	0.8750
deepseek-coder-33b-instruct	ReAct	overall	0.7500	0.8312	0.9375	0.8438
granite-34b-code-instruct	ReAct	overall	**0.7812**	0.8571	0.8750	0.8750
codellama-34b-instruct	CoT	spotify	**0.5833**	**0.7741**	0.9166	0.9166
deepseek-coder-33b-instruct	CoT	spotify	**0.5833**	**0.7741**	**1.0000**	**1.0000**
granite-34b-code-instruct	CoT	spotify	0.5000	0.7096	0.9166	0.9166
codellama-34b-instruct	ReAct	spotify	0.4167	0.7097	0.8333	0.7500
deepseek-coder-33b-instruct	ReAct	spotify	0.5000	0.7419	**1.0000**	0.7500
granite-34b-code-instruct	ReAct	spotify	0.5000	0.6774	0.8333	0.8333
codellama-34b-instruct	CoT	tmdb	0.7500	0.8260	0.9000	0.9500
deepseek-coder-33b-instruct	CoT	tmdb	0.8500	0.9347	0.9500	**1.0000**
granite-34b-code-instruct	CoT	tmdb	**1.0000**	**1.0000**	**1.0000**	**1.0000**
codellama-34b-instruct	ReAct	tmdb	0.9000	0.8913	0.9500	0.9500
deepseek-coder-33b-instruct	ReAct	tmdb	0.9000	0.8913	0.9000	0.9000
granite-34b-code-instruct	ReAct	tmdb	0.9500	0.9783	0.9000	0.9000

Table 2: Few-shot Chain-of-Thought (CoT) and ReAct prompting results on the test split of GraphQLRestBench.

Agents for Schema Generation



Generate a weather report for the IP address 72.188.196.163

http://ip-api.com/json/
https://api.openweathermap.org/data/

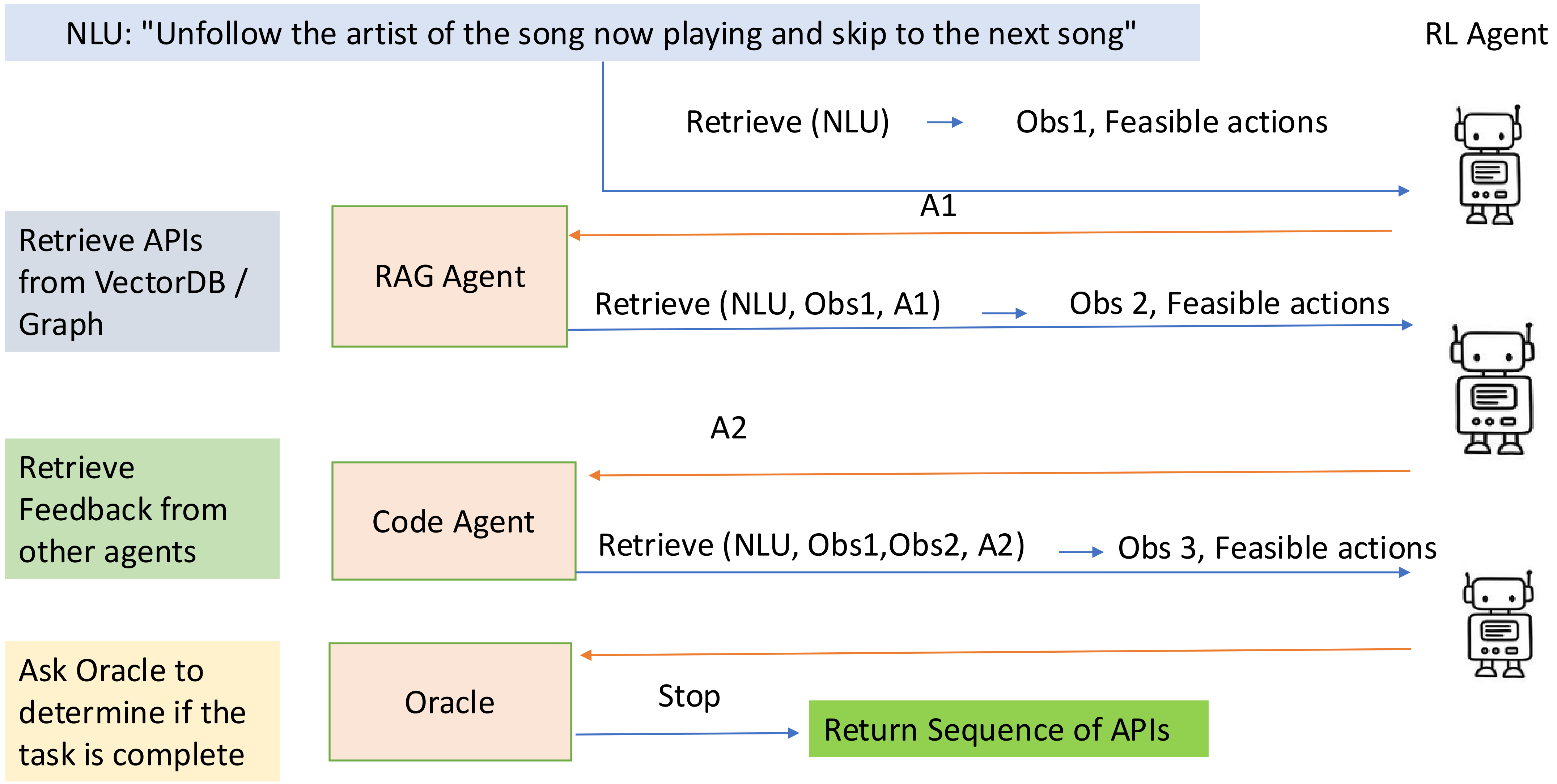
```
type Query {

  location(ip: String!, lang: String! = "en"): IpApi_Location
        @rest(
            endpoint: "http://ip-api.com/json/$ip?fields=64745
            setters: [{ field: "ip", path: "query" }]
            )

  weatherReport(openweather_appid: Secret!
    lang: String! = "en"
    lat: Float!
    lon: Float!):  Openweather_WeatherForecast
       @rest(
            endpoint: "https://api.openweathermap.org/data/2.5
            setters: [{ field: "clouds", path: "current.clouds
            )
}
```

```
weather(ip: String! openweather_appid: Secret! lang: String!
    @sequence(
        steps: [
            { query: "location"}
            { query: "weatherReport", arguments: [{name: "ope
            ]

)
```

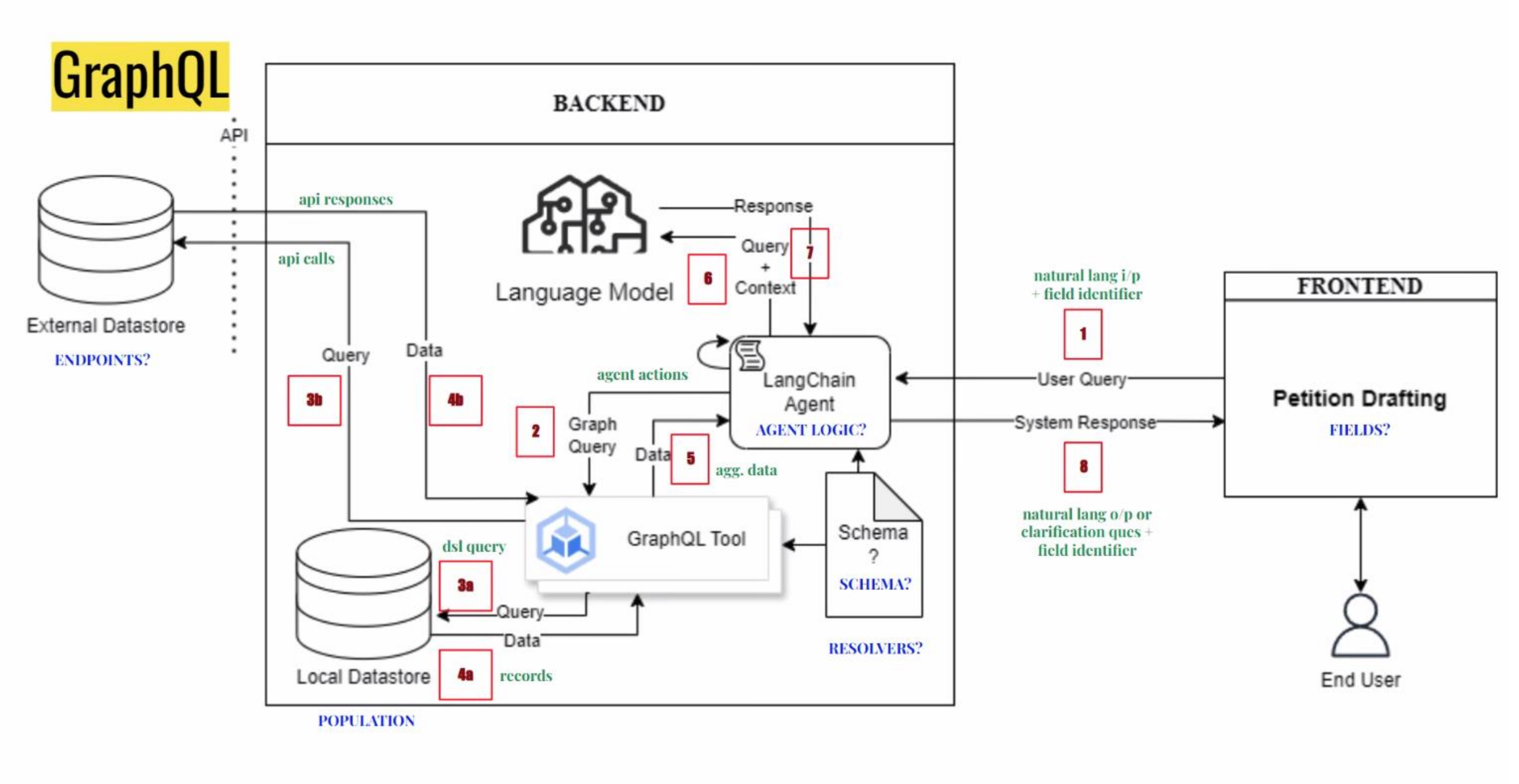
```
query MyQuery {
  weather(
    ip: "72.188.196.163"
    openweather_appid: "b4548ecd778518b766619e797744de85"
  ) {
      clouds
      temp
    }
}
```

API Sequencing with Reinforcement Learning

Lakshmi Mandal (IISc), Balaji Ganesan, Avirup Saha, Renuka Sindhgatta

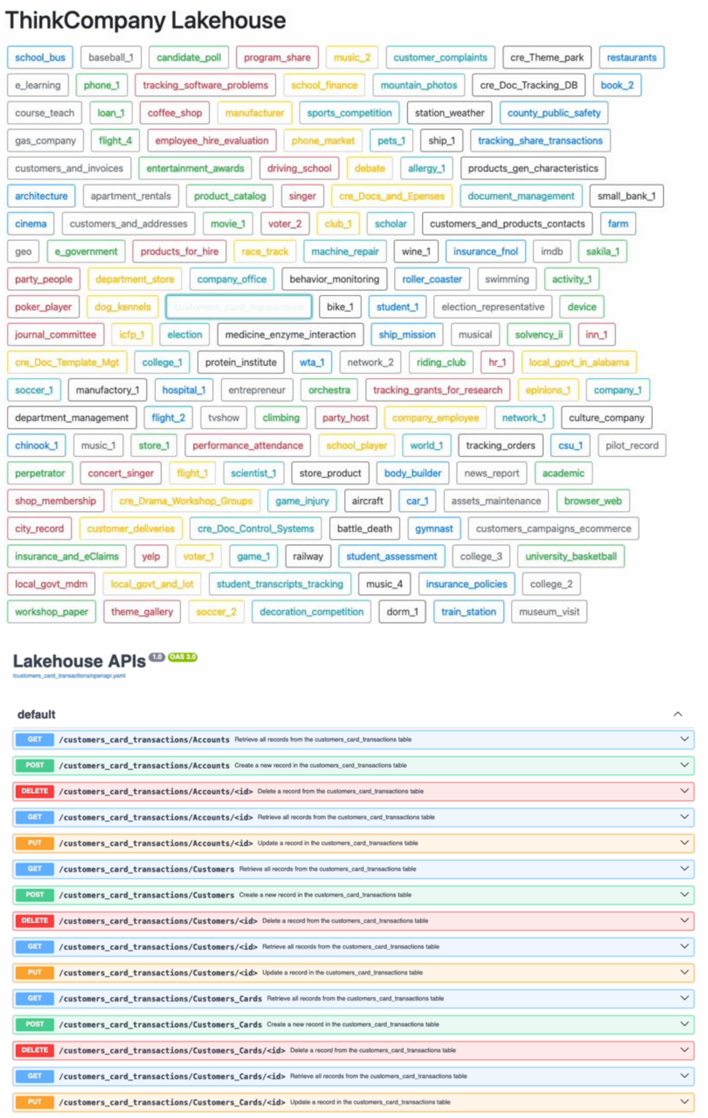


GraphQL for RAG in Legal Petition Drafting

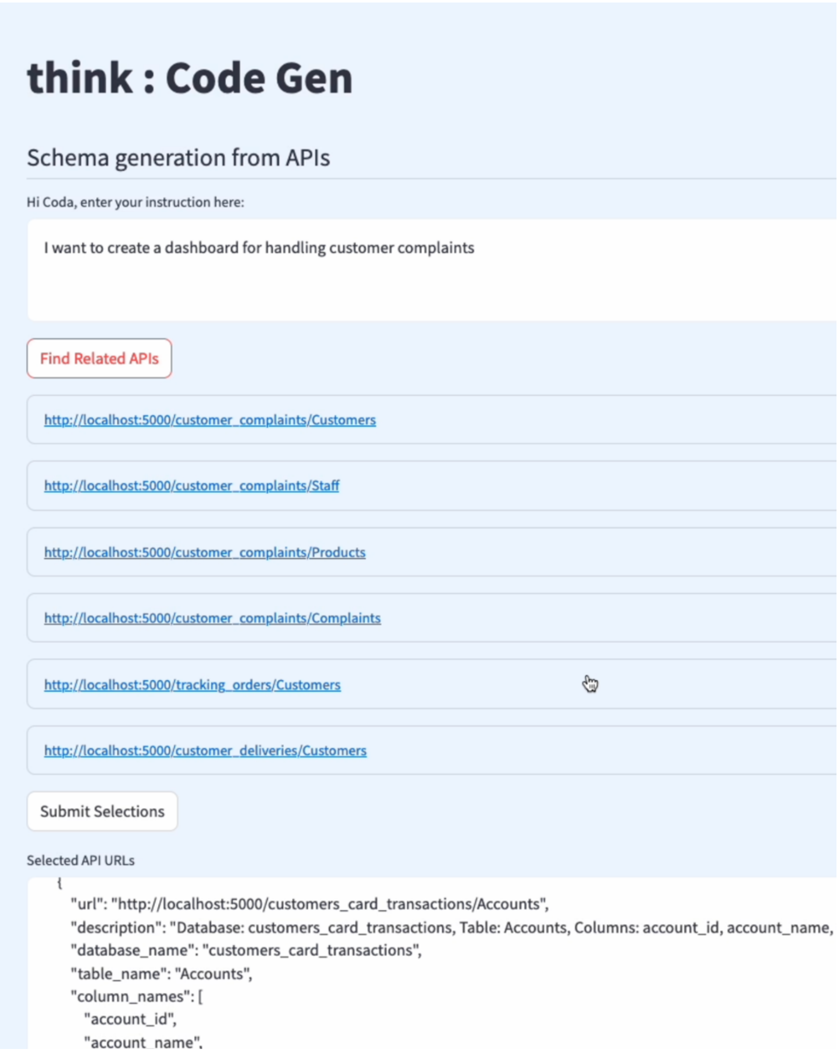


Sudipto Ghosh, Devanshu Verma, Balaji Ganesan, Purnima Bindal, Vikas Kumar, Vasudha Bhatnagar. InLegalLLaMA: Indian Legal Knowledge Enhanced Large Language Models. OpenKG Workshop at IJCAI 2024.

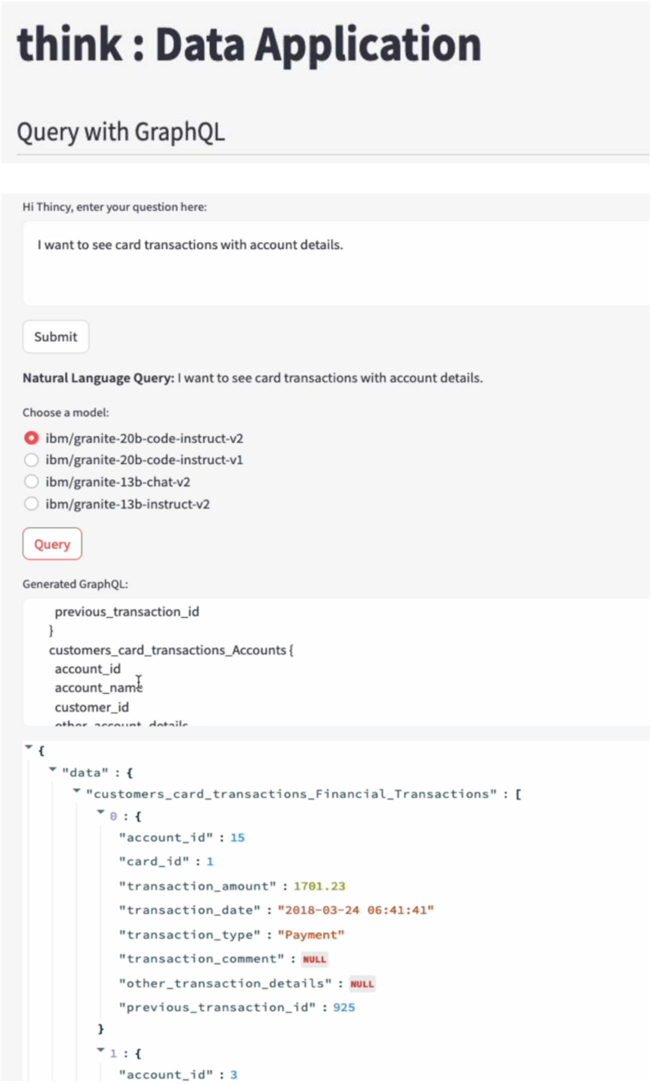
Schema Generation for BFF (Backend for Frontend) in Lakehouse



Spider REST API as proxy for Lakehouse

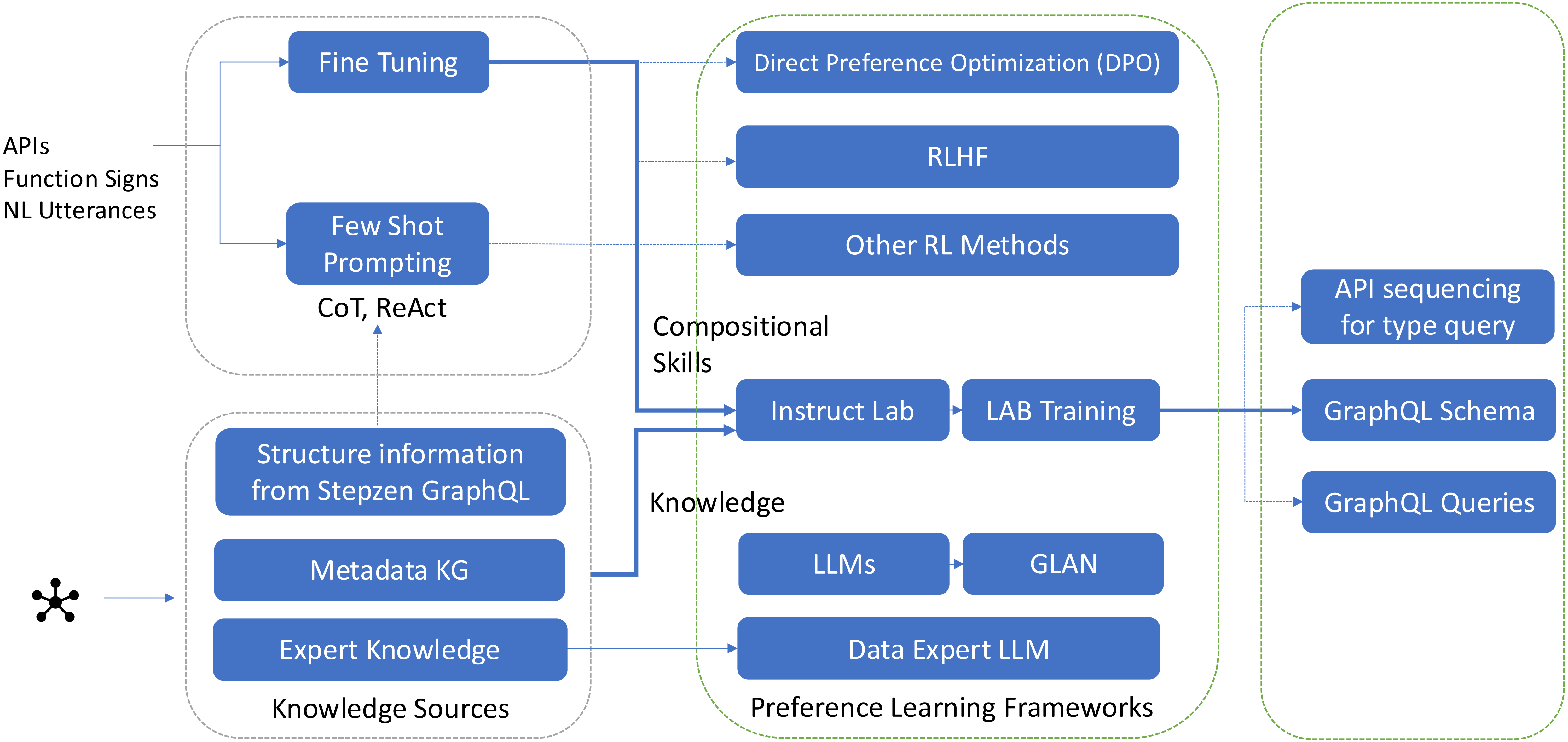


Data Steward Persona to SchemaGen



Data Analyst Persona for QueryGen

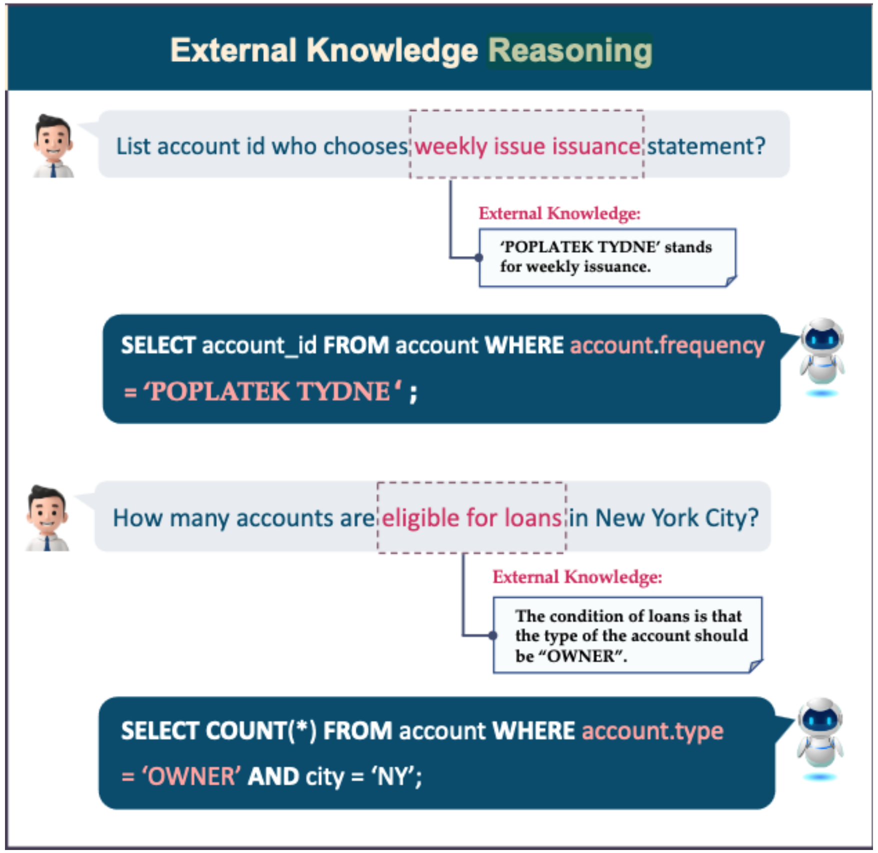
GraphQL Schema Generation – Potential Directions



Querying Lakehouse

A brief discussion of reasoning in benchmarks

BIRD



Models must handle that only "OWNER" accounts are eligible for loans.

Reasoning Type			
	Domain Knowledge	Name the ID and age of patient with two or more laboratory examinations which show their hematoclit level exceeded the normal range. `SELECT T1.ID, STRFTIME('%Y', CURRENT_TIMESTAMP) - STRFTIME('%Y', T1.Birthday) FROM Patient AS T1 INNER JOIN Laboratory AS T2 ON T1.ID = T2.ID WHERE T1.ID IN ( SELECT ID FROM Laboratory WHERE HCT > 52 GROUP BY ID HAVING COUNT(ID) >= 2 )`	23.6 %
	Numeric Computation	Among the posts with a score of over 20, what is the percentage of them being owned by an elder user? `SELECT CAST(SUM(IIF(T2.Age > 65, 1, 0)) AS REAL) * 100 / count(T1.Id) FROM posts AS T1 INNER JOIN users AS T2 ON T1.OwnerUserId = T2.Id WHERE T1.Score > 20`	24.5 %
	Synonym	How many clients opened their accounts in Jesenik branch were women ? (female) `SELECT COUNT(T1.client_id) FROM client AS T1 INNER JOIN district AS T2 ON T1.district_id = T2.district_id WHERE T1.gender = 'F' AND T2.A2 = 'Jesenik'`	7.2 %
	Value Illustration	Among the weekly issuance accounts, how many have a loan of under 200000? `SELECT COUNT(T1.account_id) FROM loan AS T1 INNER JOIN account AS T2 ON T1.account_id = T2.account_id WHERE T2.frequency = 'POPLATEK TYDNE' AND T1.amount < 200000`	70.1 %

Li, Jinyang, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang et al. "Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls." *Advances in Neural Information Processing Systems* 36 (2024).

BIRD

- We hypothesize that the internal multi-step knowledge reasoning of LLMs is not compatible with the way of external knowledge (evidence) in this situation. Therefore, the development of methods that effectively combine the strong multi-step self-reasoning capabilities of LLMs with external knowledge reasoning coherently presents a promising future direction [See Mialon et al].

Table 1: An overview comparison between BIRD and other cross-domain text-to-SQL benchmarks. In SQL, `Function` pertains to the SQL functions (Appendix B.11). `Knowledge` refers to whether or not this dataset necessitates external knowledge reasoning from the model. `Efficiency` refers to whether or not this dataset takes into consideration execution efficiency.

Dataset	# Example	# DB	# Table/DB	# Row/DB	Function	Knowledge	Efficiency
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	✓	✓	✓

Li, Jinyang, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang et al. "Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls." *Advances in Neural Information Processing Systems* 36 (2024).

Archer

Dataset	Scale						Complexity							Reasoning Distribution							Lang
	#Q	#SQL	#DB	#Dom	T/DB	C/DB	QL	SQLL	VS	TM	NL	GB	OB	A(+)	A(-)	A(*)	A(/)	H	C	C+H	
ATIS	5280	947	1	1	25	131	10.53	**99.75**	3.14	**4.66**	0.39	0.01	0.00	✗	✗	✗	✗	✗	✗	✗	en
GeoQuery	877	246	1	1	8	31	7.48	26.76	0.82	1.46	1.04	0.18	0.07	✗	✗	✗	0.2%	✗	✗	✗	en
Scholar	817	193	1	1	12	28	6.59	38.03	1.36	3.26	0.02	0.37	0.28	✗	0.5%	✗	✗	✗	✗	✗	en
Academic	196	185	1	1	15	42	13.33	36.85	1.30	3.23	0.04	0.21	0.12	✗	✗	✗	✗	✗	✗	✗	en
IMDB	131	89	1	1	16	65	10.23	29.51	1.20	2.84	0.01	0.07	0.11	✗	✗	✗	✗	✗	✗	✗	en
Yelp	128	120	1	1	7	38	9.87	28.33	1.68	2.25	0.00	0.10	0.08	✗	✗	✗	✗	✗	✗	✗	en
Advising	4387	205	1	1	18	124	10.90	48.08	3.06	3.13	0.17	0.03	0.07	3.4%	✗	✗	✗	✗	✗	✗	en
Restaurant	378	23	1	1	3	12	10.13	29.57	2.26	2.26	0.17	0.00	0.00	✗	✗	✗	✗	✗	✗	✗	en
WikiSQL	80654	51159	26531	-	1	6.33	12.46	13.32	0.53	1.00	0.00	0.00	0.00	✗	✗	✗	✗	✗	✗	✗	en
DuSQL	25003	20308	208	-	4.04	21.38	19.20	20.63	1.16	1.33	0.20	0.42	0.30	2.4%	9.5%	1.0%	4.4%	✗	-	✗	zh
BIRD	10962	10841	80	-	7.68	54.71	15.81	23.85	1.16	2.20	0.08	0.10	0.19	0.8%	5.0%	7.9%	10.0%	✗	-	✗	en
Cspider	9693	5275	166	99	5.28	27.13	11.90	24.37	0.93	1.69	0.10	0.23	0.21	0.1%	0.1%	✗	0.0%	✗	✗	✗	<u>zh</u>
Spider	9693	5275	166	99	5.28	27.13	13.29	24.37	0.93	1.69	0.10	0.23	0.21	0.1%	0.1%	✗	0.0%	✗	✗	✗	en
KaggleDBQA	272	249	8	8	2.13	22.38	9.83	13.80	0.54	1.18	0.00	0.44	**0.50**	0.0%	0.0%	✗	0.0%	✗	✗	✗	en
Archer (Ours)	1042	521	20	20	7.55	45.25	en-**29.94** zh-**25.99**	79.71	**6.21**	2.17	**1.08**	**0.59**	0.26	34.0%	47.8%	62.0%	40.7%	44.0%	51.4%	22.1%	en <u>zh</u>

Table 1: Comparison of public text-to-SQL datasets. The abbreviations used are as follows: #Q for the number of unique questions, #SQL for the number of unique SQLs, #DB for the number of databases, #Dom for the number of domains, T/DB for the number of tables per database, C/DB for the number of columns per database, QL for the average question length, SQLL for the average SQL length, VS for the average number of value slots per question, TM for the average number of tables mentioned in each SQL, NL for the average nested level per SQL, GB and OB for the average number of `GROUP BY` and `ORDER BY` clauses per SQL respectively. A, H, C, and Lang represent arithmetic, hypothetical, commonsense, and language, respectively. The cross mark, - denote absence and presence respectively. The statistics for BIRD, CSpider, and Spider is based on training and dev sets as their test sets are unavailable. Language is represented as en for English databases and questions, zh for Chinese databases and questions, and <u>zh</u> for English databases and Chinese questions.

Arithmetic Reasoning

How much higher is the maximum power of a BMW car than the maximum power of a Fiat car?
宝马汽车的最高功率比飞雅特汽车的最高功率高多少?

```
SELECT MAX(horsepower) - (SELECT MAX (horsepower)
FROM cars_data A JOIN car_names B ON A.id=B.makeid
WHERE B.model="fiat") AS diff FROM cars_data A JOIN
car_names B ON A.id=B.makeid WHERE B.model="bmw"
```

Commonsense Reasoning

Which 4-cylinder car needs the most fuel to drive 300 miles? List how many gallons it needs, and its make and model.
开300英里耗油最多的四缸车的品牌和型号分别是什么，它需要多少加仑的油?

Commonsense Knowledge: Fuel used is calculated by divding distance driven by fuel consumption.

```
SELECT B. Make, B.Model, 1.0 * 300 / mpg AS n_gallon
FROM cars_data A JOIN car_names B ON A.Id=B.MakeId
WHERE cylinders="4" ORDER BY mpg ASC LIMIT 1
```

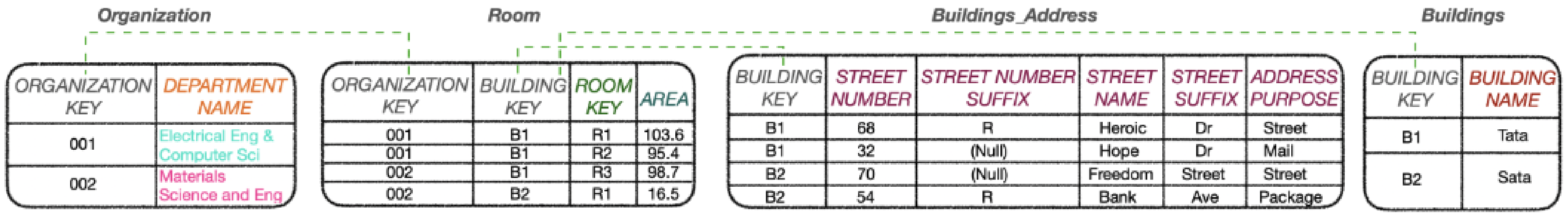
Hypothetical Reasoning

If all cars produced by the Daimler Benz company have 4-cylinders, then in all 4-cylinder cars, which one needs the most fuel to drive 300 miles? Please list how many gallons it needs, along with its make and model.
假如生产自奔驰公司的车都是四缸，开300英里耗油最多的四缸车的品牌和型号分别是什么，它需要多少加仑的油?

```
SELECT B.Make, B.Model, 1.0 * 300 / mpg AS n_gallon
FROM cars_data A JOIN car_names B ON A.id=B.makeid
JOIN model_list C ON B.model=C.model JOIN car_makers
D on C.maker=D.id WHERE D.fullname="Daimler Benz" or
A.cylinders="4" ORDER BY mpg ASC LIMIT 1
```

Zheng, Danna, Mirella Lapata, and Jeff Z. Pan. "Archer: A Human-Labeled Text-to-SQL Dataset with Arithmetic, Commonsense and Hypothetical Reasoning." *arXiv preprint arXiv:2402.12554* (2024).

Beaver



Organization

ORGANIZATION KEY	DEPARTMENT NAME
001	Electrical Eng & Computer Sci
002	Materials Science and Eng

Room

ORGANIZATION KEY	BUILDING KEY	ROOM KEY	AREA
001	B1	R1	103.6
001	B1	R2	95.4
002	B1	R3	98.7
002	B2	R1	16.5

Buildings_Address

BUILDING KEY	STREET NUMBER	STREET NUMBER SUFFIX	STREET NAME	STREET SUFFIX	ADDRESS PURPOSE
B1	68	R	Heroic	Dr	Street
B1	32	(Null)	Hope	Dr	Mail
B2	70	(Null)	Freedom	Street	Street
B2	54	R	Bank	Ave	Package

Buildings

BUILDING KEY	BUILDING NAME
B1	Tata
B2	Sata

Figure 2: Four tables in an enterprise data warehouse with their table names, schema, and example instances shown. Green lines connecting column ORGANIZATION_KEY in different tables and column BUILDING_KEY refer to join relationships among the four tables.

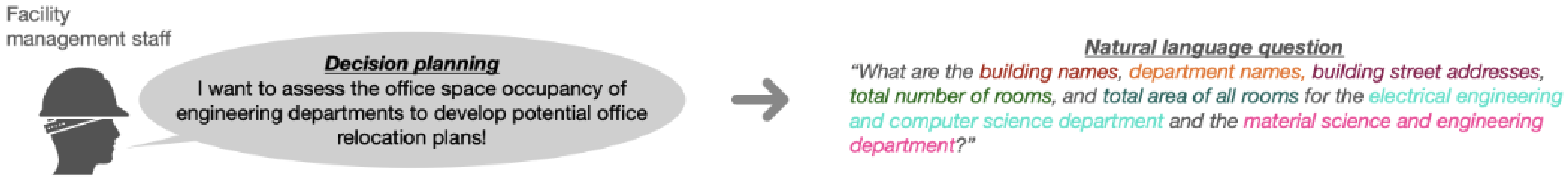


Figure 3: An example question posed by a facility management staff on the tables shown in Figure 2. Correct mappings between information/ constraints mentioned in the user question and table columns/ instances are illustrated with the same color.

Chen, Peter Baile, Fabian Wenz, Yi Zhang, Moe Kayali, Nesime Tatbul, Michael Cafarella, Çağatay Demiralp, and Michael Stonebraker. "BEAVER: An Enterprise Benchmark for Text-to-SQL." *arXiv preprint arXiv:2409.02038* (2024).

Beaver

__GPT-4o predicted SQL__

```sql
SELECT
  fb.BUILDING_NAME, fo.DEPARTMENT_NAME,
  fba.STREET_NUMBER || fba.STREET_NAME,
  COUNT(distinct fr.ROOM_KEY) AS total_rooms, SUM(fr.AREA) as
total_area
FROM
  Organization fo JOIN Room fr ON fo.ORGANIZATION_KEY =
fr.ORGANIZATION_KEY
  JOIN Buildings_Address fba on fb.BUILDING_KEY =
fba.BUILDING_KEY
  JOIN Buildings fb ON fb.BUILDING_KEY = fr.BUILDING_KEY
WHERE
  fo.DEPARTMENT_NAME in ('Materials Science and Engineering',
'Electrical Engineering & Computer Science')
GROUP BY
  fb.BUILDING_NAME, fo.DEPARTMENT_NAME,
  fba.STREET_NUMBER || fba.STREET_NAME;
```

__Gold SQL__

```sql
SELECT
  fb.BUILDING_NAME, fo.DEPARTMENT_NAME,
  fba.STREET_NUMBER || fba.STREET_NUMBER_SUFFIX || fba.STREET_NAME ||
fba.STREET_SUFFIX,
  COUNT(distinct fr.ROOM_KEY) AS total_rooms, SUM(fr.AREA) as total_area
FROM
  Organization fo JOIN Room fr ON fo.ORGANIZATION_KEY =
fr.ORGANIZATION_KEY
  JOIN Buildings_Address fba on fb.BUILDING_KEY = fba.BUILDING_KEY
  JOIN Buildings fb ON fb.BUILDING_KEY = fr.BUILDING_KEY
WHERE
  fba.ADDRESS_PURPOSE = 'STREET' and
  fo.DEPARTMENT_NAME in ('Materials Science and Eng', 'Electrical Eng &
Computer Sci')
GROUP BY
  fb.BUILDING_NAME, fo.DEPARTMENT_NAME,
  fba.STREET_NUMBER || fba.STREET_NUMBER_SUFFIX || fba.STREET_NAME ||
fba.STREET_SUFFIX;
```

Figure 4: GPT-4o predicted SQL and the gold SQL corresponding to the user question in Figure 3. Color-coded parts in SQL statements are mappings to the information/ constraints in the user question. Gold SQL includes correct mappings, but the predicted SQL might include incorrect mappings.

Chen, Peter Baile, Fabian Wenz, Yi Zhang, Moe Kayali, Nesime Tatbul, Michael Cafarella, Çağatay Demiralp, and Michael Stonebraker. "BEAVER: An Enterprise Benchmark for Text-to-SQL." *arXiv preprint arXiv:2409.02038* (2024).

Infusing Knowledge into Large Language Models with Contextual Prompts

Kinshuk Vashist, Balaji Ganesan, Vikas Kumar, Vasudha Bhatnagar

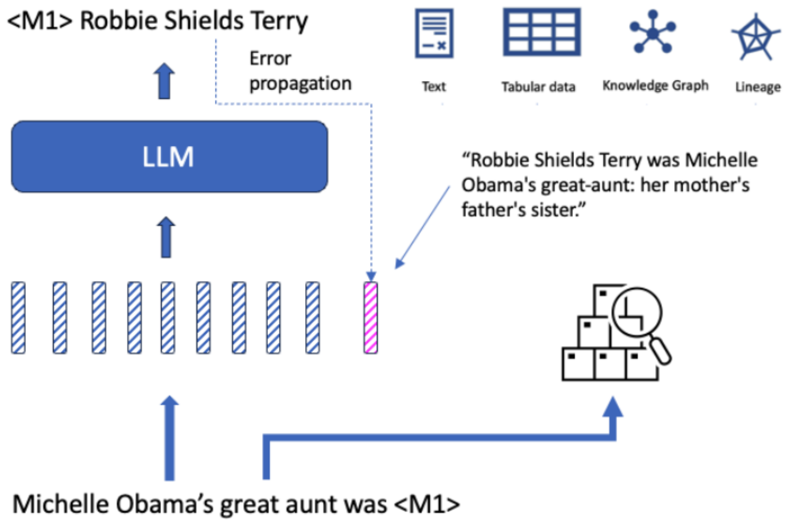


Figure 1: Contextual prompts to infuse knowledge about entities into Large Language Models

Dataset	Model	Hits@1 ↑	Hits@5 ↑	Hits@10 ↑	AED ↓	MRR ↑
KELM-TEKGEN	google/flan-t5-small	0.019	0.036	0.045	18.75	0.024
	google/flan-t5-base	0.047	0.063	0.095	85.5	0.055
	google/flan-t5-large	0.082	0.102	0.138	109.5	0.088
	flan-t5-small-fine-tuned	0.528	0.535	0.541	96.75	0.538
	flan-t5-base-fine-tuned	0.514	0.520	0.539	83.25	0.525
	flan-t5-small-fine-tuned-w-context	0.800	0.801	0.804	2.75	0.805
	flan-t5-base-fine-tuned-w-context	**0.825**	**0.825**	**0.833**	**0.75**	**0.827**
TACRED	google/flan-t5-small	0.004	0.006	0.006	84.75	0.005
	google/flan-t5-base	0.004	0.014	0.018	9.75	0.008
	google/flan-t5-large	0.034	0.044	0.060	22.50	0.039
	flan-t5-small-fine-tuned	0.366	0.368	0.39	50.25	0.376
	flan-t5-small-fine-tuned-w-context	0.782	0.782	0.784	**3.75**	0.788
	flan-t5-base-fine-tuned-w-context	**0.818**	**0.820**	**0.824**	5.25	**0.823**
Re-TACRED	google/flan-t5-small	0.000	0.010	0.016	66.00	0.005
	google/flan-t5-base	0.006	0.016	0.028	28.50	0.010
	google/flan-t5-large	0.052	0.070	0.084	5.25	0.060
	flan-t5-small-fine-tuned	0.352	0.366	0.406	15.75	0.370
	flan-t5-small-fine-tuned-w-context	0.798	0.798	0.800	6.00	0.805
	flan-t5-base-fine-tuned-w-context	**0.846**	**0.846**	**0.850**	**0.00**	**0.852**

Table 1: flan-T5 performance on relation prediction task on KELM-TEKGEN, TACRED and Re-TACRED datasets.

Kinshuk Vasisht, Balaji Ganesan, Vikas Kumar, and Vasudha Bhatnagar. 2023. Infusing Knowledge into Large Language Models with Contextual Prompts. In *Proceedings of the 20th International Conference on Natural Language Processing (ICON)*

Sherpas Framework

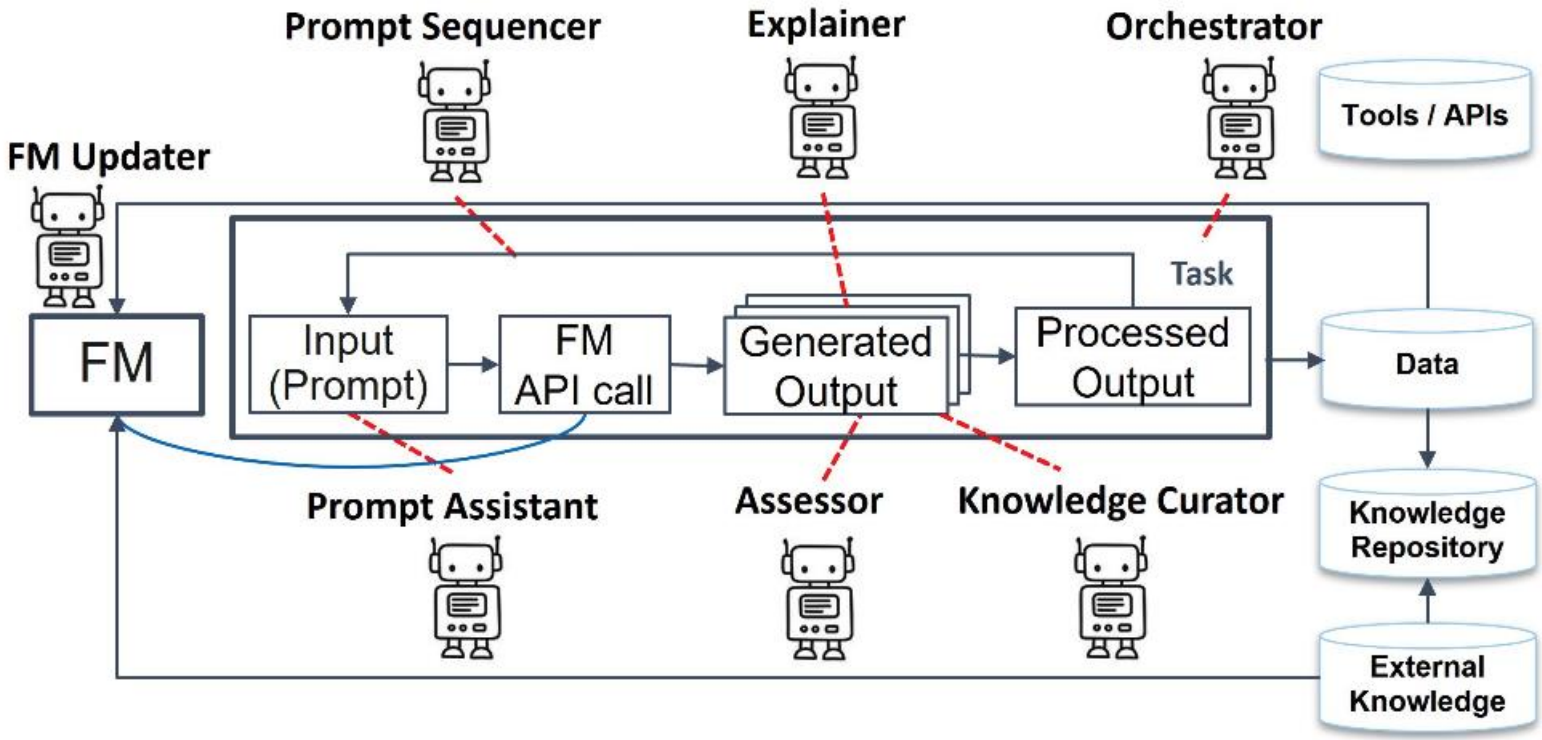


Figure 1: The Sherpas framework for guiding Foundation Models, showing various agent categories and their interaction with the FM as it executes or assists completion of a set of tasks.

Bhattacharjya, Debarun, Junkyu Lee, Don Joven Ravoy Agravante, Balaji Ganesan, and Radu Marinescu. "A Framework for Agents Guiding Foundation Models through Knowledge and Reasoning." In Trustworthy AI Workshop at *International Joint Conference on Artificial Intelligence*. 2024.