SESEMMI FOR LINKEDMUSIC: DEMOCRATIZING ACCESS TO MUSICAL ARCHIVES VIA LARGE LANGUAGE MODELS

Anonymous Authors
Anonymous Affiliations
anonymous@ismir.net

ABSTRACT

Currently, there are over one hundred music metadata databases online; comprehensively answering even simple questions often means querying dozens of them separately. This fragmentation makes large-scale, cross-cultural, or longitudinal research difficult and time-consuming. The LinkedMusic initiative aims to solve this problem by combining these databases in one place. The ingested data are stored in RDF format and can be queried using SPARQL, a querying language.

However, SPARQL's complexity makes it prohibitively difficult for most users to use effectively. Our project, the Search Engine System for Enhancing Music Metadata Interoperability (SESEMMI), aims to overcome this barrier by providing a natural language interface for LinkedMusic. Using Large Language Models (LLMs), it translates the user's plain-language queries into SPARQL queries that retrieve results from all integrated databases.

In this paper, we conduct the first systematic study of the ability of LLMs in translating Natural Language Queries (NLQ) to SPARQL in the domain of music metadata research. We evaluate five models on twenty music-domain NLQ-to-SPARQL pairs with manually prepared ground-truth outputs. Results indicate that Claude Sonnet 4 achieves the highest accuracy of 100.0% on single-database queries in both zero- and one-shot contexts and 46.7% for complex zero-shot cross-database queries.

1. INTRODUCTION

The abundance of specialized online music metadata repositories, ranging from folk-music archives to vast cross-genre music encyclopedias, has created a wealth of scholarly and cultural resources. However, the heterogeneity of their data schemas poses a fundamental barrier to cross-collection search and analysis. For example, to answer a question like "Find all works commissioned by Isabella d'Este that have a surviving manuscript and a recording made after 1980", today's musicologists must navigate multiple disparate platforms, reconcile inconsistent identifiers, and stitch together results by hand. Furthermore,

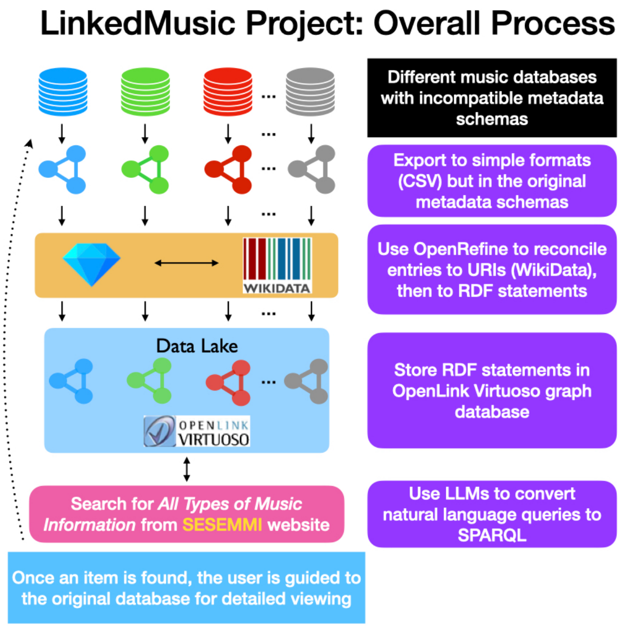

Figure 1. LinkedMusic overall process.

these websites often lack the nuanced search functionality required to answer precise questions.

The LinkedMusic project aims to address these issues by merging these music metadata databases into a single data lake, which is a vast network of raw data with inconsistent schemas. We begin this process by obtaining a dump of the databases, usually in tabular format (e.g., CSV), then use OpenRefine[1] with its Reconciliation API to match and link as many values as possible to their corresponding Wikidata [2] Uniform Resource Identifiers (URIs). For instance, the string "*Charlie Parker*" would be replaced with <http://www.wikidata.org/entity/Q103767>. After reconciliation is complete, we convert each dataset to Resource Description Framework (RDF) format[3] and merge them into a single knowledge graph, hosted in an OpenLink Virtuoso graph database,[4] which is queryable using SPARQL Protocol and RDF Query Language (SPARQL), the W3C standard for RDF data. This repository will be searchable online via the Search Engine System for Enhancing Music Metadata Interoperability (SESEMMI). The overall process is illustrated in Figure 1.

However, querying the LinkedMusic data lake presents a significant challenge since crafting SPARQL queries is

[1] https://openrefine.org Accessed 29 July 2025.
[2] https://www.wikidata.org Accessed 9 August 2025.
[3] https://www.w3.org/wiki/RDF Accessed 9 August 2025.
[4] https://virtuoso.openlinksw.com Accessed 9 August 2025.

an intricate process that is impractically complex for most end users. Our solution is to utilize Large Language Models (LLMs), which show significant promise in translating musical Natural Language Queries (NLQ) to SPARQL [1–2]. Furthermore, LLMs closely align with LinkedMusic's goal for accessibility, not only by lowering the technical barriers to use, but also in their multilingual nature.

Our primary contributions are threefold. First, while NLQ to Structured Query Language (SQL) and general-domain NLQ to SPARQL have grown in interest, translating NLQ to SPARQL over heterogeneous music-metadata graphs remains unexplored. Currently, and to the best of our knowledge, we present the first systematic study of NLQ to SPARQL in the music domain, demonstrating its feasibility and identifying specific hurdles such as Wikidata "Q" identifier retrieval and effectively communicating a complex ontology (a structured representation of concepts and their relationships). Second, we empirically evaluate and benchmark five LLMs across NLQ to SPARQL tasks, divided into four challenge types, ranging from simple single database queries to cross-database federated queries with Wikidata. Last, we provide practical insights by analyzing our use of prompt-engineering strategies that aim to maximize SPARQL accuracy and offer guidelines for implementing NLQ search tools over complex datasets.

The remainder of the paper is organized as follows. Section 2 reviews related work in NLQ to SPARQL and NLQ to SQL tasks. Section 3 details our methods, including prompt design, dataset construction, and evaluation. Section 4 presents experimental results and discusses possibilities for methodology refinement. Lastly, Section 5 concludes and outlines steps for future work.

2. BACKGROUND

While research translating natural language into formal database queries dates back to the 1970s [3], interest in this area has grown significantly since the emergence of LLMs [4]. This section surveys the conversion of natural language to both SPARQL and SQL, with the latter being more established, but methodologically similar.

2.1 Natural Language Query to SPARQL

Historically, approaches to SPARQL query generation have included neural networks [5–6], Markov models [7], and rule- or template-based methods [8–11]. However, the field has become increasingly dominated by LLMs, which continue to improve at an astonishing rate [12–14].

At the same time, while LLMs show incredible potential, they often yield inconsistent results. LLMs frequently return correct templates with critical errors that prevent the SPARQL output from retrieving the desired entities, particularly in zero-shot contexts (contexts with no examples given) [15]. This often stems from incorrect entity linking, which is the ability to match natural language terms to entities within a graph [15–16]. LLMs may also incorrectly retrieve external identifiers [2] or misunderstand the underlying knowledge graph, which itself might be of poor quality [16].

A common strategy for mitigating these issues is in-context learning, more specifically, few-shot chain-of-thought prompting, where the provided examples guide LLMs through intermediate reasoning steps [1, 17].

The reverse problem has also been investigated, where SPARQL queries are explained by converting them to natural language [18].

2.2 Natural Language Query to SQL

Although SQL and SPARQL are different database query languages, many of the strategies used in NLQ to SQL can be applied effectively to NLQ to SPARQL. While approaches in NLQ to SQL were initially predominantly rule-based methods, they were overtaken by pre-trained language models (PLMs), and later LLMs around 2023 [4].

On the other hand, while LLM-based methods for NLQ to SQL show significant promise, they still have many limitations. Firstly, they are often trained and tested on just one database, meaning that they generalize poorly and struggle to query over multiple databases, especially if the schemas differ for each one. In addition, many of the best-performing approaches, like agents, have high token costs that can make them prohibitively expensive to implement at scale. Finally, and perhaps most importantly, while LLM-based methods are rapidly improving, they are still outperformed by human experts [4].

On the BIRD-SQL benchmark [19], which contains over 12,751 unique question-SQL pairs with 95 databases across 37 professional domains, the best model, Long-Data-SQL, [5] achieves 77.53% accuracy compared to 92.96% for humans. Meanwhile, WindAgent + Claude-4-Sonnet[6] has achieved the top score of 58.32% accuracy on Spider 2.0-Snow, a dataset where correct responses for the 632 NLQ-to-SQL problems often require more than 100 lines of code and the ability to parse sub-databases with over 1,000 columns [20].

3. METHODOLOGY

While there are plans to add dozens more databases, the LinkedMusic data lake currently contains five sub-databases totalling over 352 million RDF triples: MusicBrainz [21], Digital Image Archive of Medieval Music (DIAMM) [22], The Global Jukebox [23], Dig That Lick 1000 [24], and The Session.[7] Each sub-database was reconciled to Wikidata using OpenRefine with human verification for edge cases that could not be reconciled automatically. They were then converted to RDF format, uploaded to an OpenLink Virtuoso graph database, and queried via Virtuoso's built-in SPARQL endpoint.

[5] https://bird-bench.github.io Accessed 29 July 2025.
[6] https://spider2-sql.github.io Accessed 29 July 2025.
[7] https://thesession.org Accessed 29 July 2025.

Challenge	Description	Example Query
Challenge 1	Retrieve information that can be found on a single sub-database's website.	Find all compositions by William Byrd in DIAMM.
Challenge 2	Retrieve information that is stored in a single sub-database but cannot be found through the website.	Find all different time signatures for jigs in The Session.
Challenge 3	Retrieve anything that can be found on a single sub-database plus Wikidata.	Find the average number of record labels that female singers in MusicBrainz have signed with.
Challenge 4	Retrieve any information in the entire LinkedMusic data lake and Wikidata.	Find all works commissioned by Isabella d'Este that have a surviving manuscript and a recording made after 1980.

Table 1. List of query challenge types, descriptions, and example queries.

To evaluate the models' performance, a custom test dataset of twenty NLQ/SPARQL pairs with ground-truth SPARQL was manually built. These questions were grouped into four challenge types of increasing difficulty (see Table 1), with five questions per challenge (one question per sub-database for Challenges 1–3).

During exploratory testing, we investigated methods such as prompt chaining, which breaks complex tasks into smaller, linked prompts. We also experimented with the deep-research feature (e.g., multi-step reasoning and chain-of-thought exploration) and even attempted emotional appeals, like begging or threatening the model. These approaches did not appear to improve results.

Difficulties encountered during the exploratory phase were numerous and often unexpected, including an issue where the SPARQL output was syntactically correct, but conflicted with Virtuoso's SPARQL query optimizer. To teach the LLM how to better cooperate with Virtuoso, we needed to add two lines to our prompt.

After extensive exploratory testing, a general all-purpose system prompt (see Appendix) was designed, which wraps around the natural language input and asks the LLM to return the corresponding SPARQL as output.

In total, five LLMs were tested: Claude Sonnet 4,[8] Gemini 2.5 Flash,[9] Gemini 2.5 Pro[10] GPT-4o,[11] and OpenAI o4-mini.[12] Each model was evaluated three times through the browser in a zero- or one-shot context (i.e., zero or one example provided) and was given the full ontology of the LinkedMusic data lake in Turtle format,[13] a compact and human-readable RDF format. For example, Figure 2 shows a diagram of the ontology for the Dig That Lick 1000 database.

For one-shot tests, the provided sample NLQ/SPARQL pair (see Appendix) was a separate Challenge 4 cross-database query that was designed to maximize the likelihood that the model would be shown the most relevant SPARQL query-building strategies. For each test, a new chat window was opened with the memory feature disabled. Browser-based tools like ChatGPT's web search feature were enabled. Evaluation was conducted on a pass/fail basis, with models passing if the generated SPARQL returned the exact same number of results as the handwritten ground-truth SPARQL. The full queries, prompts, provided ontology, and results can be accessed via our GitHub repository.[14]

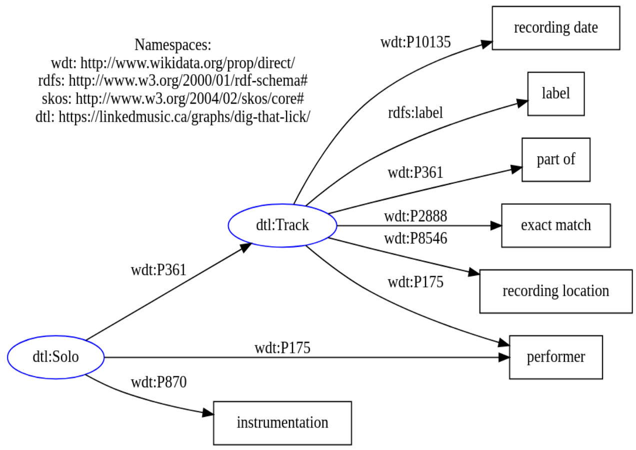

Figure 2. Dig That Lick 1000 ontology. Dig That Lick 1000 entities were reconciled to Wikidata using the properties shown above.

4. RESULTS AND DISCUSSION

The experimental results, shown in Figure 3, reveal several notable patterns. As expected, our data show a pronounced inverse correlation between query complexity and model accuracy across all evaluated systems. Challenge 1 queries, representing straightforward single-database retrieval, yielded the highest aggregate accuracy of 63.3%, while Challenge 4 queries dropped down to 26.0% on average. This highlights the substantial difficulties inherent in generating accurate SPARQL queries across large knowledge graphs.

The implementation of one-shot prompting did not yield consistent performance improvements, with some models exhibiting marginal performance decreases in one-

[8] https://www-cdn.anthropic.com/07b2a3f9902ee19fe39a36ca638e5ae987bc64dd.pdf Accessed 9 August 2025.
[9] https://storage.googleapis.com/model-cards/documents/gemini-2.5-flash.pdf Accessed 9 August 2025.
[10] https://storage.googleapis.com/model-cards/documents/gemini-2.5-pro.pdf Accessed 9 August 2025.
[11] https://cdn.openai.com/gpt-4o-system-card.pdf Accessed 9 August 2025.
[12] https://cdn.openai.com/pdf/2221c875-02dc-4789-800b-e7758f3722c1/o3-and-o4-mini-system-card.pdf Accessed 9 August 2025.
[13] https://www.w3.org/TR/turtle Accessed 29 July 2025.
[14] https://github.com/ANONYMOUS

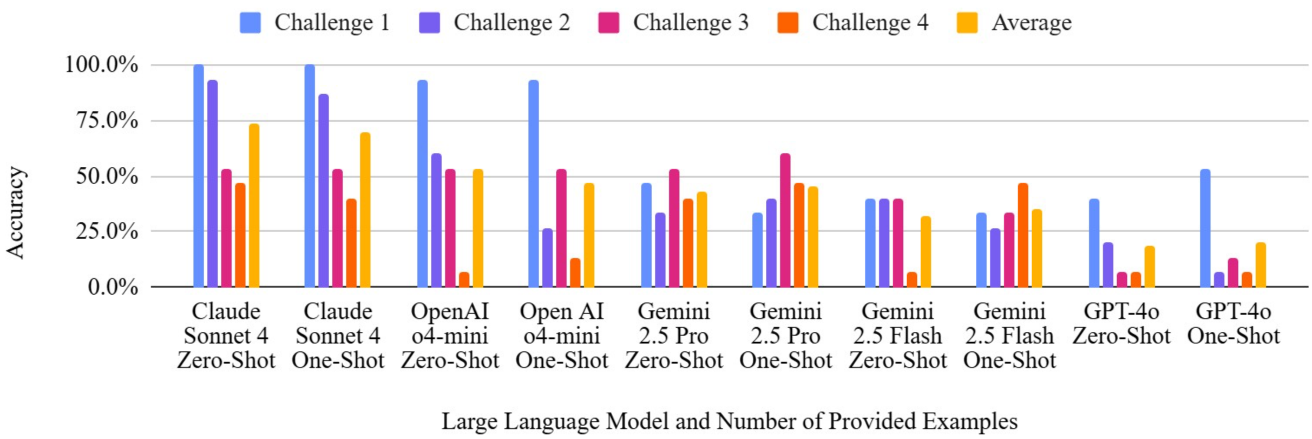

Figure 3: Average performance for various LLMs in zero- and one-shot contexts across four natural language query challenges of increasing difficulty. Each test was performed three times, and the results were averaged.

shot versus zero-shot conditions, suggesting potential interference from the provided exemplar.

As illustrated in Figure 4, which shows the combined results for zero- and one-shot tests aggregated by LLM, Claude Sonnet 4 achieved the highest overall accuracy, scoring 100.0% on Challenge 1 questions for both zero- and one-shot contexts and an average of 73.3% across all challenges in zero-shot contexts.

Across all models and challenges, the most common issues involved misunderstanding the underlying graph content, with LLMs often failing to parse the representation of Wikidata-reconciled items within the graph database and entity links between sub-graphs. Wikidata “Q” identifiers were often incorrect, and in rare cases, no SPARQL was produced. We also observed moderate variance between attempts, with queries frequently being successful for one or two out of three attempts.

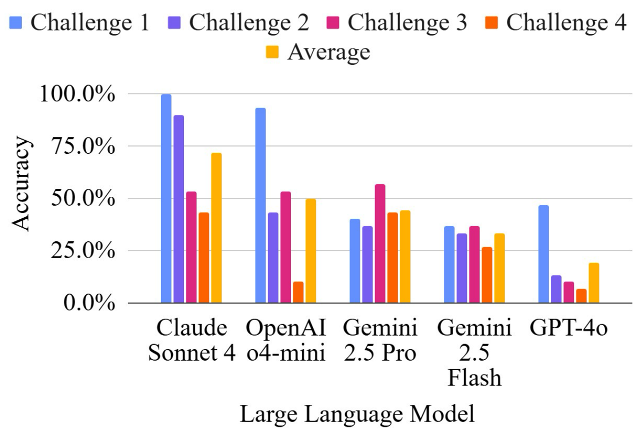

Figure 4: Average performance by LLM for zero- and one-shot contexts combined across four natural language query challenges of increasing difficulty. Each test was performed three times, and the results were averaged.

5. CONCLUSION AND FUTURE WORK

This study presents the first systematic investigation of NLQ to SPARQL translation using LLMs in the music domain and introduces SESEMMI as a solution for querying heterogeneous music metadata within the LinkedMusic data lake. Through our evaluation of five LLMs across twenty test cases, we demonstrate both the feasibility and current difficulties in this approach.

Results indicate that performance decreases significantly with query complexity, ranging from simple single-database retrieval to cross-database federated queries with Wikidata. Claude Sonnet 4 achieved the highest average accuracy of 73.3% for zero-shot contexts, with 100.0% accuracy on both zero- and one-shot single database queries and 46.7% accuracy on zero-shot cross-database queries.

Our findings contribute to the growing field of semantic query generation, particularly highlighting domain-specific issues such as entity linking, Wikidata “Q” identifier retrieval, and the complexities of communicating heterogeneous ontologies to LLMs. While the results show promise for making specialized music repositories more accessible to researchers and scholars, they also underscore the need for refinement in prompt engineering strategies and model architectures.

As LinkedMusic expands with the addition of more music databases, we will investigate the capacity of these methods to scale to a larger dataset with a greater variety of sub-databases. Future work should also systematically test different prompting approaches, including using chain-of-thought prompting and providing more examples. Furthermore, more advanced LLM implementations, such as an agent that performs API calls, should be tested. This could allow the model to break down the problem into smaller steps like retrieving the relevant ontology or performing a function call for Wikidata Q identifier retrieval. Moreover, the effect of changing parameters like temperature should also be investigated.

As LLMs improve, these methods offer a promising path toward making specialized music databases more accessible to researchers and practitioners who lack technical query expertise.

6. REFERENCES

[1] J. D’Abramo, A. Zugarini, and P. Torroni, “Investigating Large Language Models for Text-to-SPARQL Generation,” in *Proc. 4th Int. Workshop Knowl.-Augmented Methods Natural Lang. Process.*, W. Shi, W.

Yu, A. Asai, M. Jiang, G. Durrett, H. Hajishirzi, and L. Zettlemoyer, Eds., Albuquerque, NM, USA, May 2025, pp. 66–80. doi: 10.18653/v1/2025.knowledgenlp-1.5.

[2] V. Emonet, J. Bolleman, S. Duvaud, T. Mendes de Farias, and A. C. Sima, "LLM-based SPARQL Query Generation from Natural Language over Federated Knowledge Graphs," in *Proc. Special Session Harmonising Generative AI Semantic Web Technol. (HGAIS 2024)*, R. Alharbi, J. de Berardinis, P. Groth, A. Meroño Peñuela, E. Simperl, and V. Tamma, Eds., Baltimore, MD, USA, Nov. 2024. Accessed: July 18, 2025. [Online]. Available: https://ceur-ws.org/Vol-3953/355.pdf

[3] D. Waltz, "Natural Language Access to a Large Data Base: An Engineering Approach," in *Proc. 4th Int. Joint Conf. Artificial Intell. (IJCAI)*, Tbilisi, USSR, Sept. 1975, pp. 868–872. [Online]. Available: https://www.ijcai.org/Proceedings/75/Papers/129.pdf

[4] X. Liu et al., "A Survey of Text-to-SQL in the Era of LLMs: Where are We, and Where are We Going?," *IEEE Trans. Knowl. Data Eng., early access,* July 24, 2025. doi: 10.1109/TKDE.2025.3592032.

[5] F. F. Luz and M. Finger, "Semantic Parsing Natural Language into SPARQL: Improving Target Language Representation with Neural Attention," Mar. 12, 2018, arXiv:1803.04329. doi: 10.48550/arXiv.1803.04329.

[6] X. Yin, D. Gromann, and S. Rudolph, "Neural Machine Translating from Natural Language to SPARQL," *Future Gener. Comput. Syst.*, vol. 117, pp. 510–519, Apr. 2021, doi: 10.1016/j.future.2020.12.013.

[7] Y.-H. Chen, E. J.-L. Lu, and T.-A. Ou, "Intelligent SPARQL Query Generation for Natural Language Processing Systems," *IEEE Access*, vol. 9, pp. 158638–158650, Nov. 2021, doi: 10.1109/ACCESS.2021.3130667.

[8] C. Pradel, O. Haemmerlé, and N. J. Hernandez, "Natural Language Query Interpretation into SPARQL Using Patterns," in *Proc. 4th Int. Workshop Consum. Linked Data (COLD 2013)*, Sydney, Australia, Oct. 2013, pp. 1–12. Accessed: July 25, 2025. [Online]. Available: https://hal.science/hal-01143219

[9] M. Dubey, S. Dasgupta, A. Sharma, K. Höffner, and J. Lehmann, "AskNow: A Framework for Natural Language Query Formalization in SPARQL," in *Semantic Web. Latest Adv. New Domains*, H. Sack, E. Blomqvist, M. d'Aquin, C. Ghidini, S. P. Ponzetto, and C. Lange, Eds., Heraklion, Greece: Springer International Publishing, June 2016, pp. 300–316. doi: 10.1007/978-3-319-34129-3_19.

[10] N. Steinmetz, A.-K. Arning, and K.-U. Sattler, "From Natural Language Questions to SPARQL Queries: A Pattern-based Approach," in *Proc. Datenbanksysteme Business, Technol. Web (BTW 2019)*, T. Grust et al., Eds., in Gesellschaft für Informatik, vol. P-289. Rostock, Germany, Mar. 2019, pp. 289–308. doi: 10.18420/btw2019-18.

[11] H. Jung and W. Kim, "Automated Conversion from Natural Language Query to SPARQL Query," *J. Intell. Inf. Syst.*, vol. 55, pp. 501–520, Dec. 2020, doi: 10.1007/s10844-019-00589-2.

[12] J. Lehmann, P. Gattogi, D. Bhandiwad, S. Ferré, and S. Vahdati, "Language Models as Controlled Natural Language Semantic Parsers for Knowledge Graph Question Answering," in *Proc. 26th Eur. Conf. Artif. Intell. (ECAI 2023)*, K. Gal, A. Nowé, G. J. Nalepa, R. Fairstein, and R. Rădulescu, Eds., in Frontiers in Artificial Intelligence and Applications, vol. 372. Kraków, Poland: IOS Press, Oct. 2023, pp. 1348–1356. doi: 10.3233/FAIA230411.

[13] S. Yang, M. Teng, X. Dong, and F. Bo, "LLM-Based SPARQL Generation with Selected Schema from Large Scale Knowledge Base," in *Knowl. Graph Semantic Comput.: Knowl. Graph Empowers Artif. General Intell. (CCKS 2023)*, H. Wang, X. Han, M. Liu, C. Gong, Y. Liu, and N. Zhang, Eds., Shenyang, Springer Singapore, Aug. 2023, pp. 304–316. doi: 10.1007/978-981-99-7224-1.

[14] M. R. A. H. Rony, U. Kumar, R. Teucher, L. Kovriguina, and J. Lehmann, "SGPT: A Generative Approach for SPARQL Query Generation from Natural Language Questions," *IEEE Access*, vol. 10, pp. 70712–70723, July 2022, doi: 10.1109/ACCESS.2022.3188714.

[15] D. Bustamante and H. Takeda, "SPARQL Generation with Entity Pre-trained GPT for KG Question Answering," Feb. 01, 2024, arXiv:2402.00969. doi: 10.48550/arXiv.2402.00969.

[16] I.-V. Hernandez-Camero, E. Garcia-Lopez, A. Garcia-Cabot, and S. Caro-Alvaro, "Context-Aware Few-Shot Learning SPARQL Query Generation from Natural Language on an Aviation Knowledge Graph," *Mach. Learn. Knowl. Extr.*, vol. 7, no. 2, p. 52, June 2025, doi: 10.3390/make7020052.

[17] H. M. Zahera, M. Ali, M. A. Sherif, D. Moussallem, and A.-C. Ngonga Ngomo, "Generating SPARQL from Natural Language Using Chain-of-Thoughts Prompting," in *Proc. 20th Int. Conf. Semantic Syst. (SEMANTiCS 2024)*, A. Salatino et al., Eds., in Studies on the Semantic Web, vol. 60. Amsterdam, Netherlands: IOS Press, Sept. 2024, pp. 353–368. doi: 10.3233/SSW240028.

[18] A.-C. Ngonga Ngomo, L. Bühmann, C. Unger, J. Lehmann, and D. Gerber, "Sorry, i don't speak SPARQL: Translating SPARQL Queries into Natural Language," in *Proc. 22nd Int. Conf. World Wide Web (WWW '13)*, in WWW '13. Rio de Janeiro, Brazil: Association for Computing Machinery, May 2013, pp. 977–988. doi: 10.1145/2488388.2488473.

[19] J. Li et al., "Can LLM Already Serve as a Database Interface? A Big Bench for Large-scale Database

Grounded Text-to-SQLs," in *Proc. 37th Int. Conf. Neural Inf. Process. Syst. (NeurIPS 2023)*, New Orleans, LA, USA: Curran Associates Inc., Dec. 2023, pp. 42330–42357. Accessed: July 29, 2025. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2023/file/83fc8fab1710363050bbd1d4b8cc0021-Paper-Datasets_and_Benchmarks.pdf

[20] F. Lei et al., "Spider 2.0: Evaluating Language Models on Real-World Enterprise Text-to-SQL Workflows," in *Proc. Int. Conf. Learn. Representations (ICLR)*, Singapore: Curran Associates, Inc., Apr. 2025. Accessed: July 31, 2025. [Online]. Available: https://openreview.net/forum?id=XmProj9cPs

[21] A. Swartz, "MusicBrainz: A Semantic Web Service," *IEEE Intell. Syst.*, vol. 17, no. 1, pp. 76–77, Feb. 2002, doi: 10.1109/5254.988466.

[22] A. Wathey, M. Bent, and J. Craig-McFeely, "The Art of Virtual Restoration: Creating the Digital Image Archive of Medieval Music (DIAMM)," in *The Virtual Score, Volume 12: Representation, Retrieval, Restoration*, W. Hewlett B. and E. Selfridge-Field, Eds. Cambridge, MA, USA: The MIT Press, 2001, ch. 15, pp. 227–240. Accessed: July 29, 2025. [Online]. Available: https://doi.org/10.7551/mitpress/2058.003.0019

[23] A. L. C. Wood et al., "The Global Jukebox: A Public Database of Performing Arts and Culture," *PLoS ONE*, vol. 17, no. 11, Nov. 2022, Art. no. e0275469. doi: 10.1371/journal.pone.0275469.

[24] C. Stover, "Dig That Lick (DTL): Analyzing Large-Scale Data for Melodic Patterns in Jazz Performances," *J. Am. Musicol. Soc.*, vol. 74, no. 1, pp. 195–214, Apr. 2021, doi: 10.1525/jams.2021.74.1.195.

7. APPENDIX

The full prompt used in testing is provided below. Note that <<USER INPUT>> would be replaced with the natural language query being tested. For brevity, the ontology has been replaced with <<ONTOLOGY>>. However, it can be accessed via our GitHub repository.[15]

```
I have a graph database containing musical linked
data from various databases. As much of the in-
formation as possible is reconciled against Wik-
idata.

Please write me a SPARQL query to perform the
following query:

<<USER_INPUT>>

When an entity is reconciled against Wikidata,
wdt:P2888 is used to point to the reconciled Wik-
idata entity.
When an entity has a wdt:P31 triple, it contains
information about the subclass that the entity is
a part of (e.g. for mb:Artist, the wdt:P31 can
point to human, musical group, etc).

The steps you should follow are:
1. Examine the ontology and extract the relevant
parts.
2. Using that ontology, figure out which Q-IDs
you need and perform web searches to find them.
3. Using the ontology and the Q-IDs, build the
final SPARQL query.

Please follow these instructions:
- When asked to return a list of entities, please
always return both the label (when available) and
the URI for the entities.
- When finding Q-IDs to match against, search the
web to get the best and most accurate results.
- Ensure that the Q-IDs that you've found are
correct by performing another web search.
- Please scan all entities across all databases to
find out which one(s) correspond to the query, and
only select the relevant databases and entities.
- For any entity you search for within the
LinkedMusic graph (not in Wikidata), please add a
triple that uses the rdf:type property to explic-
itly verify its type.
- Do not use Wikidata to verify the type of enti-
ties, please instead use the LinkedMusic types,
using the rdf:type property.
    - The only exception to this is when local
entities have a wdt:P31 triple (like mb:Artist),
then it is fine to check that triple using wdt:P31
in the local LinkedMusic graph, but never in a
federated query.
- If you need data that is not located in the
LinkedMusic graph, i.e. when there is no property
for the information you need directly present in
the ontology I give you, please use a federated
query with Wikidata using the <https://query.wik-
idata.org/sparql> endpoint, but only do so if the
information doesn't appear at all in the
LinkedMusic graph ontology.
- Please ensure that you've fully reviewed the
LinkedMusic ontology and extracted the relevant
parts before performing federated queries.
- Please also double-check that you're not trying
to use properties that do not appear in the on-
tology, unless they are a part of a federated
query.
- When performing a federated query, ensure that
the SPARQL query is efficient and will not create
an unnecessarily high amount of requests.
- When resolving a Wikidata Q-ID, you must use
the provided ontology to determine the linking
path.
    - If a property's object is another defined
class in the ontology (e.g., diamm:City wdt:P17
diamm:Country), your query must first navigate to
that class and then use its wdt:P2888 property to
get the Q-ID.
    - If a property's object is described by a
literal string (e.g., ts:Session wdt:P17 "coun-
try"@en), you should assume the property links
directly to the Wikidata URI.
- Once the SPARQL query is finalized, please re-
read it and double-check that all QIDs are cor-
rect.
- For MusicBrainz, very few mb:Recording entities
are reconciled against Wikidata since Wikidata
does not carry information about specific record-
ings, only about the actual songs, so it's better
to match reconciled data against mb:Work entities
rather than mb:Recording

Please follow these constraints:
- Do not use string matching; instead check
against Wikidata Q-IDs. The only exception to this
is when the query explicitly requests finding en-
tities based on text/string content (e.g., 'find
```

[15] https://github.com/ANONYMOUS

tracks with X in the title', 'find artists whose names contain Y', 'search for works with Z in the description'). In such cases, use appropriate SPARQL string matching functions like CONTAINS(), REGEX(), or similar.
- Do not use the SELECT ... FROM syntax for named graphs. Please instead use the SELECT { GRAPH ... { ... } } syntax.
- Do not put any triples verifying the type of entities (using wdt:P31 or rdf:type) in federated query SERVICE blocks.
- Do not use Wikidata to retrieve labels unless directly asked to in the query. please prioritize as much as possible retrieving labels from the LinkedMusic database.
- Do not put any federated query SERVICE blocks inside a GRAPH block.
- Do not put any federated query SERVICE blocks inside an OPTIONAL block.
- Do not use a nested SELECT clause inside a SERVICE block.
- To avoid the Virtuoso error SP031, use a subquery before the SERVICE call for federated queries
- To avoid the Virtuoso error SP031, ensure every variable is assigned a value in a valid scope before it's used in a FILTER, BIND, or OPTIONAL block.

Please remember that the SPARQL query will not work, and you will have failed your task, if you do not follow these constraints and instructions. Please also be very diligent with your search for the correct Q-IDs, as they are one of the key parts of the SPARQL query.

Here are the 5 databases currently in LinkedMusic, and the IRIs for their RDF graphs:
- All triples for DIAMM are stored in the <https://linkedmusic.ca/graphs/diamm/> graph, and their entity types use the `diamm:` prefix.
- All triples for Dig That Lick are stored in the <https://linkedmusic.ca/graphs/dig-that-lick/> graph, and their entity types use the `dtl:` prefix.
- All triples for The Session are stored in the <https://linkedmusic.ca/graphs/thesession/> graph, and their entity types use the `ts:` prefix.
- All triples for The Global Jukebox are stored in the <https://linkedmusic.ca/graphs/theglobaljukebox/> graph, and their entity types use the `gj:` prefix.
- All triples for MusicBrainz are stored in the <https://linkedmusic.ca/graphs/musicbrainz/> graph, and their entity types use the `mb:` prefix.

The following is a graph representation of the ontology of all the data in the database, for all 5 databases. Here is how to interpret this ontology:
- The subject are the LinkedMusic entity types (accessed using rdf:type)
- The predicates are the properties that those entities have
- The objects are described as below:
    - When the object is another class: If a property's object is another defined class in the ontology (e.g., diamm:City wdt:P17 diamm:Country), your query must first navigate to that diamm:Country class and then use its wdt:P2888 property to get the Q-ID.
    - When the object is a placeholder for an entity: If a property's object is a generic placeholder string that stands in for an entity's name (e.g., "country"@en, "instance of"@en, "performer"@en, "exact match"@en), assume the property in the actual graph links directly to a Wikidata URI.
    - When the object is a data value: If a property's object is a string that represents a data type (e.g., "publication date"@en, "coordinate location"@en, "label"), assume the property in the actual graph links to a literal value (a date, a string, coordinates, etc.) and not a Wikidata URI.

<<Ontology>>

REMEMBER: Please find the correct QIDs

As an example, here is a query and the associated SPARQL query.
User query: Find all musical works that were composed in Mexico City.
SPARQL query:
```SPARQL
SELECT ?work ?workLabel
WHERE {
  # Wikidata Q-ID for Mexico City
  VALUES ?cityQID { wd:Q1489 }

  {
    # Search for works in the MusicBrainz graph
    GRAPH <https://linkedmusic.ca/graphs/musicbrainz/> {
      ?work rdf:type mb:Work .

      { ?work wdt:P1071 ?locationObject . } # location of creation
      UNION
      { ?work wdt:P4647 ?locationObject . } # location

      # Handle both direct links to an Area and links to a Place within an Area
      {
        # Case 1: The location is the city/area itself
        ?locationObject rdf:type mb:Area .
        BIND(?locationObject AS ?cityArea)
      }
      UNION
      {
        # Case 2: The location is a venue (Place), so we find its containing city (Area)
        ?locationObject rdf:type mb:Place .
        ?locationObject wdt:P131 ?cityArea .
        ?cityArea rdf:type mb:Area .
      }

      # Match the final city/area to Mexico City's Q-ID
      ?cityArea wdt:P2888 ?cityQID .

      OPTIONAL { ?work rdfs:label ?workLabel . }
    }
  }
  UNION
  {
    # Search for compositions in the DIAMM graph
    GRAPH <https://linkedmusic.ca/graphs/diamm/> {
      ?work rdf:type diamm:Composition .

      ?work wdt:P361 ?source .
      ?source rdf:type diamm:Source .
      ?source wdt:P131 ?city .
      ?city rdf:type diamm:City .

      ?city wdt:P2888 ?cityQID .

      OPTIONAL { ?work rdfs:label ?workLabel . }
    }
  }
}
```