Extensive Exploration of Databases and Preparation for Developing a VQL Debugger

Sau Lai YIP

Abstract

This progress report details my research under the UROP1100 project titled "Knowledge Discovery Over Database." My activities during this summer session are divided into two primary stages: (1) a comprehensive exploration of the field of databases, and (2) preparations for developing a VQL debugger. As a novice in database research, I dedicated the first half of the session to acquiring a broad understanding of the field by studying various sub-areas extensively. I reviewed suggested papers on two main streams: algorithm-based and model-based problems. Based on my interest and comprehension, I decided to work on developing a debugger for model-generated VQL queries. The latter half was spent preparing the pipeline design by reviewing related work on SQL and other programming languages.

1. Introduction

As inferred from [1], our society has experienced an exponential increase in data volume, facilitated by the reduced costs of data collection and processing and the democratization of data access. Consequently, database research has become increasingly vital, aiming to efficiently extract valuable information from large-scale data to address various challenges. Motivated by this trend, I began exploring the field of databases during this UROP session. Based on my interests and strengths, I decided to develop a debugger for model-generated VQL queries and initiated the preparatory stage. This report is organized as follows: Section 2 discusses my comprehensive exploration of databases, Section 3 details my preparation for developing the VQL debugger, and Section 4 presents the conclusion.

2. Extensive Exploration of Databases

As a beginner in the field, I spent the initial part of this session gaining an overall understanding of databases by studying various areas within this field extensively. The field of databases encompasses two major streams: algorithm-based problems and model-based problems. I read several papers focusing on different targeted problems within these streams.

2.1 Algorithm-Based Problems

Algorithm-based problems typically involve designing or optimizing algorithms to enhance efficiency and reduce costs, which often includes proving correctness and computing costs. The papers I reviewed on algorithm-based problems can be categorized into three areas based on their modelling settings: tabular DB, spatial DB, and graph DB. Table 1 is a summary of these papers.

2.1.1 Tabular DB

In a tabular DB, data is typically modelled as a d-dimensional dataset D consisting of tuples with d dimensions. As demonstrated in [2], this model can be applied to preference learning problems, which aim to identify the top-k tuples based on multi-criteria decisions and interactive user feedback. Their approach considers the probability of user errors to enhance accuracy.

2.1.2 Spatial DB

Spatial DB is useful for problems involving geospatial attributes, such as geographic information systems (GIS), computer graphics and vision, scientific data 3D modelling, online 3D virtual games, and spatial data mining [5]. I further divided the modelling settings in spatial DB into three types: static, constrained, and dynamic spatial DB. For static spatial DB, a modelling setting may consist of a 3D

surface (e.g., terrain surface and polyhedral surface) representing the spatial environment and a set of points on it representing interested objects. As in [3] and [4], this is typically used to solve shortest-path problems. When introducing additional attributes to each point, as the set of keywords of each object in [5], it becomes the constrained spatial DB. The practicability can be suggested by a trivial application that a tourist wants to find a hotel (object) with "sight-seeing" and "dining" (keywords). When considering a changeable setting of the spatial DB, it becomes a dynamic DB, which is useful for real-world problems like route recommendation with real-time traffic situations. The frequent updating can be costly, and [6] addresses it with an effective algorithm maintaining an Architecture-Intact Property.

Database modelling setting	Targeted problem	Paper title	Paper contribution summary
Tabular DB			
Tabular DB	Preference learning	Finding Best Tuple via Error-prone User Interaction [2]	Incorporated random errors in user responses into interactive multi-criteria preference learning and proposed algorithms for 2D and higher dimensions [2].
Spatial DB			
Static Spatial DB	Shortest path	Shortest Path on a Polyhedron [3]	Developed an algorithm derived from "continuous Dijkstra" for single source shortest path problems, achieving cost of O(n2) time and Θ(n) space by applying techniques such as sequence tree, planar layout, and subdivision [3].
	Shortest distance query	Distance Oracle on Terrain Surface [4]	Introduced a ϵ-approximate distance oracle with efficient construction time, oracle size, and query time, by exploiting compact ways to store information [4].
Constrained spatial DB (Spatial key word search)	Recommendation with both geospatial and attribute aspect constraints	Cost-Aware and Distance-Constrained Collective Spatial Keyword Query [5]	Aggregating aspect constraints as cost and geographical ones as distance, provided an exact version and an approximate version of algorithms for Cost-Aware and Distance-Constrained Collective Spatial Key Word Query (CD-CoSKQ) [5].
Dynamic special DB	Shortest path and distance in dynamic network	Architecture-Intact Oracle for Fastest Path and Time Queries on Dynamic Spatial Networks [6]	Introduced an algorithm with effective preprocessing, querying, and high-probability updating cost by maintaining an architecture-intact property [6].
Graph DB			
Graph keyword search	Real-time / dynamic recommendation	Index-Free Approach with Theoretical Guarantee for Efficient Random Walk with Restart Query [7]	Introduced an algorithm named Residue-Accumulated approach (ResAcc), satisfying index-free, output-bounded, and high-efficiency, to address Random Walk with Restart (RWR) problems [7].

Table 1. Summary of Paper Reading on Algorithm-Based Problems

2.1.3 Graph DB

In the modelling setting of a graph DB, there is usually a set of nodes to represent objects and a set of edges representing relationships between nodes with weights measuring the intimacy, which is aggregated according to the problem definition. Such a graph neural network can be more powerful if equipped with suitable algorithms, taking [7] as an example, a computing efficient algorithm is introduced for Single-source Random Walk with Restart query by integrating a concept of Residue Accumulation with GNN, achieving elegant properties such as index-free.

2.2 Model-Based Problems

Model-based problems typically involve designing and improving models to achieve better performance in given settings, which often involves architecture selection and performance analysis. The papers I reviewed related to model-based problems can be categorized into three areas according to their system architecture settings: recommendation systems, NLP-based systems, and speech-based systems. Table 2 is a summary of these papers.

System architecture setting	Targeted problem	Paper title	Paper contribution summary
Recommendation system			
Session-based Recommendation System	Relation of importance and granularity of sequence information	Session-based Recommendation with Local Invariance [8]	Introduced the concept of local variance, which distinguishes the order of actions into detailed and high-level order, focusing more on the latter [8].
	Application of GNN for encoding	Handling Information Loss of Graph Neural Networks for Session-based Recommendation [9]	Developed a lossless encoding scheme and an edge-order preserving aggregation layer to solve the problem of lossy encoding and introduced a shortcut graph attention layer to address the ineffective long-range dependency capturing issue [9].
	Social-aware models	An Efficient and Effective Framework for Session-based Social Recommendation [10]	Introduced a heterogeneous graph neural network and relevance-based filter to improve the efficiency of social-aware models [10].
Speech-based System			
Hybrid of NLP and ASR (automatic speech recognition)	Speech-to-SQL	VoiceQuerySystem: A Voice-driven Database Querying System Using Natural Language Questions [11]	Developed a system consisting of a cascaded and an end-to-end speech-to-SQL component that takes natural language questions and outputs SQL queries [11].
NLP-based System			
NLP (natural language processing)	Text-to-visualization	RGVisNet: A Hybrid Retrieval-Generation Neural Framework Towards Automatic Data Visualization Generation [12]	Integrated a retrieval-generation framework on a DV query codebase to enhance efficiency [12].

Table 2. Summary of Paper Reading on Model-Based Problems

2.2.1 Recommendation System

One of the popular focuses in recommendation systems is session-based recommendation, which is a task to predict users' next actions based on a sequence of previous actions in the same session [8]. As illustrated by [8, 9, 10], systems developed for session-based recommendation can be improved by considering some specific nature of such problem settings. [8] enhances efficiency by considering importance differences embedded in level differences of granularity of sequent information, while [9] improves the encoding accuracy of systems using GNN by addressing information loss. When adding a social-aware setting to the session-based recommendation, computation cost increases due to a greater number of sessions involved, which is addressed by [10] with the integration of embedding precomputation and unrelated session filtering.

2.2.2 NLP-Based System and Speech-Based System

Database systems can be more universally beneficial if they cooperate with techniques that process direct human signals such as natural language and natural speech, without repelling non-technical users. [11] is a sample work using NLP and ASR to design a speech-to-visualization system, while [12] is another using NLP and a retrieved-argument framework to improve text-to-visualization systems.

2.3 Stage Summary

From the work reviewed, I realized that a breakthrough usually starts from noticing:

weaknesses of existing methods;
unconsidered properties of a specific problem setting;
potential for simplification or cost reduction by filtering or migrating operations to other stages.

It can be concluded that sensitive observation skills are crucial for research work.

3. Preparation for Designing a VQL Debugger

Based on my interest and comprehension, I decided to work on designing a debugger for model-generated VQL (Visualization Query Language) queries. The motivation for this project is that there are models that generate VQL to achieve goals such as text-to-VQL or text-to-visualization, but there might be syntax or logical errors in the generated VQL queries. To improve accuracy, an automatic debugger that can detect and correct errors is in demand.

I spent the second half of this UROP session preparing by conducting a literature review. This section first explains the intuition behind how I chose reference works and then discusses how the techniques might be transferred to designing a VQL debugger, in terms of the overall architecture and other potential techniques.

3.1 Intuition to Refer to Work of SQL and Other Programming Languages

VQL shares many similarities with SQL (Structural Query Language). As inferred from [13], both types of queries perform a series of operations such as selection, comparison, aggregation, and join on related tables to obtain data. Considering that there is more existing work on SQL, I mainly referred to them and focused on the possibility of transferring the techniques to VQL. Additionally, as VQL is also a kind of programming language, with syntax standards and a compiler, I also referred to existing work for general and other specific kinds of programming languages.

3.2 Overall Architecture of a Debugger

From the papers I have read on error detection and correction for SQL and other programming languages, I learned about several model frameworks for a debugger. Table 3 summarizes the architecture of each of the papers.

3.2.1 SQLFixAgent

[14] is a work for SQL debugging. As they stated, there are two types of LLM-based text-to-SQL tools: (1) prompting LLMs, which usually obtain better semantic accuracy; (2) fine-tuned LLMs, which usually obtain better syntactical accuracy. To leverage the advantages and disadvantages of the two types, their SQLFixAgent is a hybrid of three prompting-LLM agents and a fine-tuned SQLTool. The Reviewer detects errors using rubber duck debugging (CoT) to perform step-by-step analysis. The Crafter provides variants of a query to generate candidate SQL using SQLTool and performs multiple-choice (MCS) selection on candidates. Finally, the Refiner performs repairing by selecting the best from the filtered candidates repeatedly until a correct SQL is found while keeping a failure memory reflection (Reflxion) to improve efficiency. However, although they achieved a high detection rate, the repair rate is not satisfactory.

Related work	Architecture	Focused task	Remark
SQLFixAgent [14]	LLM-based agents: Reviewer (COT) Crafter (MCS) Refiner (Reflxion) + Fined-tuned SQLTool	semantic errors detection and repair	High detection rate Low repairing rate
RTLFixer [15]	LLM + ReAct + RAG + human guidance	syntax errors detection and repair	Transfer to RAG + VQL codebase
Error Detection for Text-to-SQL Semantic Parsing [16]	Base encoder: Pre-trained language model CodeBERT + GNN: Parse tree for NLQ Syntax tree for SQL	Error detection only	Parser-independent

Table 3. Comparison of Architecture of Debuggers

3.2.2 RTLFixer

[15] is a syntax error fixer for Verilog code. Their RTLFixer is LLM-based, applying interactive ReAct prompting, which is a mechanism to guide LLM to perform reasoning and action planning. The actions include interacting with a compiler to receive error messages and performing Retrieval-Augmented Generation (RAG) to make use of prepared human guidance. Considering that preparing manual guidance could be costly, reusing previously validated VQL requirements may be more efficient, similar to [12].

3.2.3 Error Detection for text-to-SQL

[16] is an error detection for SQL. Their model is parser-independent, meaning that it is applicable to SQL generated from text-to-SQL parsers with various architectures. Their model integrates a pre-trained language model for programming languages with a graph neural network (GNN) to capture structural features, which is built from the parse tree for natural language questions (NLQ) and syntax tree for SQL following specific algorithms. Since this model only performs error detection, a component for error repairing will be needed to build a debugger.

From the work reviewed, I noticed the following that might be applicable to the VQL debugger: (1) a language model might be useful for semantic accuracy, logical reasoning, and action planning thanks to the general knowledge gained from pre-training; (2) a text-to-VQL parser might be useful to generate high-quality candidates thanks to the knowledge of VQL gained from pre-training; (3) error messages

from a compiler can help the debugging of syntax errors; (4) reuse of validated VQL queries can improve efficiency.

3.3 Other Potential Techniques

Apart from the overall framework, this section discusses some additional techniques I have learned from the literature review that might improve performance. Table 4 summarizes these potential techniques.

3.3.1 Text-to-SQL Parsing

[17] introduces a model to synthesize high-quality text-SQL pairs for text-to-SQL parsing. They found that many errors in generated SQL are related to the mismatching of data types and table joining through non-foreign-primary key pairs, for which they proposed strong typing to preserve data types and key identity in column names and restrict to foreign-primary key relationships for some operations. Since VQL also targets relational databases and performs similar operations, these two techniques can be integrated to assist related syntax error detection. Additionally, they use an SQL-IR-NLQ process to synthesize SQL-text pairs, which means generating SQL first, then simplifying to intermediate representation (IR), and finally generating natural language questions (NLQ). A potential application is to build a similar framework for VQL to generate corresponding IR and NLQ given a VQL, which might be helpful for semantic accuracy checks.

3.3.2 Classification of Logical Errors

[18] defined a set of logical error types and relationships to improve logical error classification for general programming languages. As suggested by them, such general definitions should be tailored to a specific language, VQL in my targeted problem, to better assist classification and improve the debugging process.

Related work	Techniques	Potential application
Importance of Synthesizing High-quality Data for Text-to-SQL Parsing [17]	Strong typing Key relation	Assist syntax error detection
	Schema-weighted distance	Assist error correction using original VQL as basis
	SQL-to-IR process SQL-to-NLQ process	Transfer to VQL-to-IR and VQL-to-NLQ processes to assist semantic error detection
Improving Classification of Logical Errors by Integrating Error Relationship into Prompts [18]	Logical error types and relationships	Tailor for VQL language Assist error detection

Table 4. Summary of Other Potential Techniques

4. Conclusion

In this summer UROP session, I gained an overview of database research through extensive exploration and performed preparation for developing a debugger for model-generated VQL queries by conducting a literature review. I not only obtained knowledge in the field of databases and techniques related to my project topic but also enriched my general skills and experience about how to conduct a research topic and transform techniques from related work.

Acknowledgments

I would like to express my sincere gratitude to my supervisor, Prof. WONG, who has provided inspiring guidance and constructive feedback throughout the process.

References

[1] Claudio Gutierrez. A Copernican Revolution in Data. arXiv:2306.08766v1, 2023. doi:10.48550/arXiv.2306.08766.

[2] Qixu Chen, Raymond Chi-Wing Wong. Finding Best Tuple via Error-prone User Interaction. In *Proceedings of the 39th IEEE International Conference on Data Engineering (ICDE 2023)*, Anaheim, CA, USA, 3-7 April 2023, article number 10184674, pp. 1803-1816, Apr. 2023. DOI: 10.1109/ICDE55515.2023.00141. Identifier: https://hdl.handle.net/1783.1/125674.

[3] Jindong Chen and Yijie Han. Shortest paths on a polyhedron. In Proceedings of the sixth annual symposium on Computational geometry (SCG '90). Association for Computing Machinery, New York, NY, USA, 360–369. 1990. https://doi.org/10.1145/98524.98601

[4] Victor Junqiu Wei, Raymond Chi-Wing Wong, Cheng Long, and David M. Mount. Distance Oracle on Terrain Surface. In Proceedings of the 2017 ACM International Conference on Management of Data (SIGMOD '17). Association for Computing Machinery, New York, NY, USA, 1211–1226. 2017. https://doi.org/10.1145/3035918.3064038

[5] Harry Kai-Ho Chan, Shengxin Liu, Cheng Long, and Raymond Chi-Wing Wong. Cost-Aware and Distance-Constrained Collective Spatial Keyword Query. *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 2, pp. 1324-1336, Feb. 2023. DOI: 10.1109/TKDE.2021.3095388. Identifier: https://hdl.handle.net/1783.1/114478.

[6] Victor Junqiu Wei, Raymond Chi-Wing Wong, Cheng Long. Architecture-Intact Oracle for Fastest Path and Time Queries on Dynamic Spatial Networks. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, New York, NY, USA, 2020, pp. 1841-1856. DOI: 10.1145/3318464.3389718. Identifier: https://hdl.handle.net/1783.1/104115.

[7] Dandan Lin, Raymond Chi-Wing Wong, Min Xie, Victor Junqiu Wei. Index-free approach with theoretical guarantee for efficient random walk with restart query. In *Proceedings - International Conference on Data Engineering*, vol. 2020-April, April 2020, article no. 9101811, pp. 913-924. DOI: 10.1109/ICDE48307.2020.00084. Identifier: https://hdl.handle.net/1783.1/104195.

[8] Tianwen Chen and Raymond Chi-Wing Wong. Session-based Recommendation with Local Invariance. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, pp. 994-999, 2019.

[9] Tianwen Chen and Raymond Chi-Wing Wong. Handling Information Loss of Graph Neural Networks for Session-based Recommendation. In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '20). Association for Computing Machinery, New York, NY, USA, 1172–1180. 2020. https://doi.org/10.1145/3394486.3403170.

[10] Tianwen Chen and Raymond Chi-Wing Wong. An Efficient and Effective Framework for Session-based Social Recommendation," in *WSDM 2021 - Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, New York, USA, Mar. 2021, pp. 400-408, doi: 10.1145/3437963.3441792.

[11] Yuanfeng Song, Raymond Chi-Wing Wong, Xuefang Zhao, and Di Jiang. VoiceQuerySystem: A Voice-driven Database Querying System Using Natural Language Questions. In Proceedings of the 2022 International Conference on Management of Data (SIGMOD '22). Association for Computing Machinery, New York, NY, USA, 2385–2388. 2022. https://doi.org/10.1145/3514221.3520158.

[12] Yuanfeng Song, Xuefang Zhao, Raymond Chi-Wing Wong, and Di Jiang. RGVisNet: A Hybrid Retrieval-Generation Neural Framework Towards Automatic Data Visualization Generation. In Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD

'22). Association for Computing Machinery, New York, NY, USA, 1646–1655. 2022. https://doi.org/10.1145/3534678.3539330.

[13] Yang Wu, Yao Wan, Hongyu Zhang, Yulei Sui, Wucai Wei, Wei Zhao, Guandong Xu, and Hai Jin. Automated Data Visualization from Natural Language via Large Language Models: An Exploratory Study. Proc. ACM Manag. Data 2, 3 (SIGMOD), Article 115 (June 2024), 28 pages. 2024. https://doi.org/10.1145/3654992.

[14] Jipeng Cen, Jiaxin Liu, Zhixu Li, Jingjing Wang. SQLFixAgent: Towards Semantic-Accurate Text-to-SQL Parsing via Consistency-Enhanced Multi-Agent Collaboration. https://arxiv.org/abs/2406.13408v2, 2024. doi:10.48550/arXiv.2406.13408.

[15] Yun-Da Tsai, Mingjie Liu, Haoxing Ren. RTLFixer: Automatically Fixing RTL Syntax Errors with Large Language Models. https://arxiv.org/abs/2311.16543v3, 2023. doi:10.48550/arXiv.2311.16543.

[16] Shijie Chen, Ziru Chen, Huan Sun, Yu Su. Error Detection for Text-to-SQL Semantic Parsing. https://arxiv.org/abs/2305.13683v2, 2023. doi:10.48550/arXiv.2305.13683.

[17] Yiyun Zhao, Jiarong Jiang, Yiqun Hu, Wuwei Lan, Henry Zhu, Anuj Chauhan, Alexander Li, Lin Pan, Jun Wang, Chung-Wei Hang, Sheng Zhang, Marvin Dong, Joe Lilien, Patrick Ng, Zhiguo Wang, Vittorio Castelli, Bing Xiang. Importance of Synthesizing High-quality Data for Text-to-SQL Parsing. https://arxiv.org/abs/2212.08785v1, 2022. doi:10.48550/arXiv.2212.08785.

[18] Yanggyu Lee, Suchae Jeong, Jihie Kim. Improving LLM Classification of Logical Errors by Integrating Error Relationship into Prompts. https://arxiv.org/abs/2404.19336v2, 2024. doi:10.48550/arXiv.2404.19336.