Using LLM Artificial Intelligence Systems as Complex SQL Programming Assistants

Putsadee Pornphol
Department of Digital Technology
Phuket Rajabhat University
Phuket, Thailand
putsadee.p@pkru.ac.th

Suphamit Chittayasothorn
School of Engineering
King Mongkut's Institute of Technology Ladkrabang
Bangkok, Thailand
suphamit.ch@kmitl.ac.th

2024 12th International Conference on Information and Education Technology (ICIET) | 979-8-3503-7177-2/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICIET60671.2024.10542806

***Abstract*—Learning database programming such as SQL programming is a challenging task when the queries become more complex. SQL is a declarative language based on relational calculus which describes the definition of the query results instead of describing the procedure or steps used to obtain the query result. Tutorial sessions using tutorial assistances are generally required to support the learning of advanced part of the language. Recently generative AI systems demonstrated question answering capabilities including programming codes generation. This paper verifies the SQL code generating capabilities of four generative AI systems: Bing, Bard, ChatGPT, and Copilot and their suitability as SQL programming assistants.**

Keywords—SQL, LLM, database, complex queries

I. INTRODUCTION

The relational database language SQL was developed by a team led by Donald D. Chamberlin at IBM in 1974 as a commercial, English-like version of the relational language Tuple Relational Calculus. The language was originally called SEQUEL (Structured English Query Language) [1] and pronounced "see-quail". Later, IBM changed it to SQL (Structured Query Language). SQL is a database language. It is not a general-purpose language. In the early years of database technology, database languages were referred to as "data sub languages" since their functions are mainly database manipulations and retrieval. SQL was adopted as the standard relational database language by the ANSI (American national Standard Institute) in 1986 and the ISO (International Organization for Standardization) in 1987. To date, it has gone through several revisions with many specialized features such as temporal database definitions and manipulations in SQL 2011[2]. The full detail of recent SQL standard documents can be purchased from ISO or ANSI.

SQL is a high productivity language. An SQL query that retrieves a desired result requires minutes or seconds to write. It is a more declarative language. SQL programmers define what they want instead of describing how to do it. A procedural language program which gives the same query result requires much longer time to develop. The basic part of SQL is very straight forward and does not require many efforts to learn. The more advanced complex query parts, which are very useful in practice, are relatively hard for students and require formal approaches and training.

The introduction of large language model (LLM) systems [3,4] to the public starts a new chapter of human-computer communication and defines new approaches of machine assistance including programming assistance. These models, such as Bard [5], Bing [6], Copilot [7], and ChatGPT [8], are the results of many years of research and innovation. They are trained on a large number of textual sources and have the ability to generate relevant responses such as question answering based on learned knowledge and data. There are premium ones which need subscription fees, provide more up to date information. These LLM systems find various applications and assist human users more directly to the needs than the use of conventional search engines. One of the useful applications includes code generations, including the Structured Query Language (SQL) codes [9,10] .

Several LLM systems can generate SQL code. They are trained on massive datasets of texts and codes, including SQL codes. SQL code can be generated by the following steps: First, the LLM system first needs to understand the query in a natural language such as English. This also includes the understanding of the tables which are to be queried, the columns, and the search conditions. Then, it generates the SQL code. Data manipulation as well as retrieval statements can also be generated.

LLM systems know SQL syntax through their training data and the patterns they have learned from various sources that include SQL queries and related text. During the pre-training phase of its training process, they are exposed to a wide range of text data from the internet, books, articles, technical documentation, and more. This data includes examples of SQL queries, discussions about databases, and related content. The exposure to a diverse range of SQL-related text allows them to learn how SQL queries are constructed, even though they may not possess an in-depth comprehension of the semantic meanings of individual column names or the specifics of database schemas.

Programmers and students use LLM systems as SQL programming assistants. Since the knowledge of generating SQL statements is based on the training sets which are supposed to be big enough to cover semantically correct cases, verification of the ability to generate correct SQL statements needs to be done. This paper presents the verification of complex SQL database retrieval statements which are generated by LLM systems and discusses the suitability of these systems as SQL database programming assistants.

```
Presidential Database Structure
President (
        pres_name char (15) not null,
        birth_yr number not null,
        yrs_serv number not null,
        death_age number,
        party char (12) not null,
        state_born varchar (14) not null );
pres_marriage(
        pres_name char (15) not null,
        spouse_name char (15) not null,
        pr_age number not null,
        sp_age number not null,
        nr_children number not null,
        mar_year number not null );
pres_hobby(
        pres_name char (15) not null,
        hobby char (18) not null );
election (
        election_year number not null,
        candidate varchar(30),
        votes number,
        winner_loser_indic char(1));
```

Fig. 1. Presidential Database Table Structures.

II. The Presidential Database and Test Queries

The sample database tables which are used in this chapter is the Presidential Database [11], [12] which was widely used during the early eighties as sample database tables. The tables show presidential details of US presidents. Figure 1 shows the table structure of some of the tables of the presidential database that are referred to. There are seven tables in the presidential database. In this paper, four of them are used. The table names and the column names are meaningful enough for the MML systems to understand without any further explanations.

The first one is the president table which is the main table that represents each president as a row. The pres_name is the president's name and also the primary key of the table, birth_yr is the president's birth year, yrs_serv is the total number of years that the person served as a president, death_age is the age at death, party is the party that the president belonged to, and state_born is the birth state of the president. The second table is the pres_marriage table which represents each presidential marriage as a row. A president may have more than one marriage. The primary key of this table is the combination of pres_name and spouse_name. Pr_age, sp_age are the president's and the spouse's ages at marriage time, nr_children is the number of children from the marriage, and mar_year is the marriage year. The third table is the pres_hobby table. This is a two-column table which represents a many-to-many relationship between president and hobby. The primary key is the combined key of pres_name and hobby. The pres_marriage table only have rows of married presidents. Likewise, the pres_hobby table only have rows of presidents who had their hobbies recorded.

The following four test queries are prepared. 1) From the Presidential database, give an SQL query of the following question: List hobbies that at least 3 Democratic presidents have in common. 2) List details of presidents who belonged to the party which has the highest number of presidents in Ohio. 3) List the rows of the top two candidates by votes for each election year. 4) List the name of the youngest second wife. These queries are hard to very hard to formulate, even by experienced SQL programmers. We ask four LLM systems to generate SQL queries with explanations to assist learners understanding of the more complex parts of SQL programming. All the codes are tested on Oracle Database Express Edition [13].

The Test Query 1: From the Presidential database, give an SQL query of the following question: List hobbies that at least 3 Democratic presidents have in common.

This query tests the use of a join or a subquery to check that the presidents belong to the Democratic party, then applies a GROUP BY Hobby and the built-in function count(*) to count the number of presidents. The HAVING clause checks if there are at least 3 presidents. This query is considered a query with complex search conditions. Students find it hard to do. The corresponding SQL queries as coded manually by a human tutorial assistant are as shown in Fig 2:

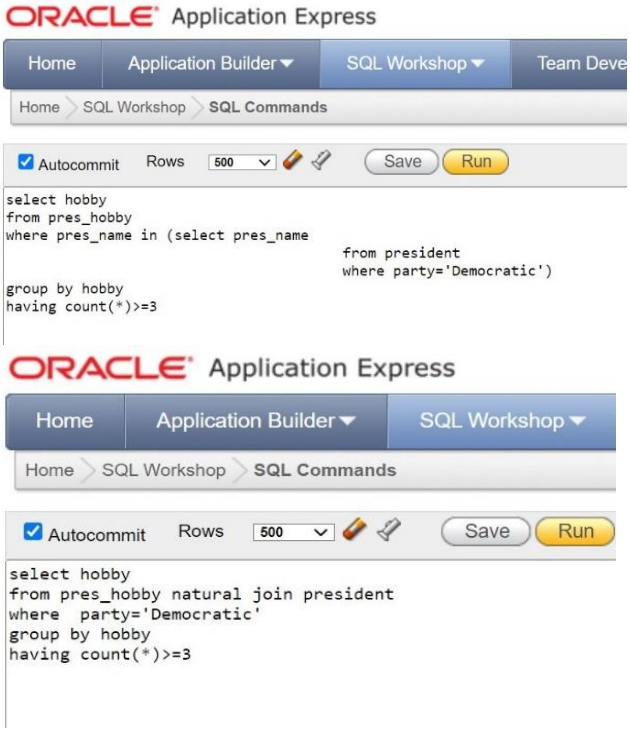

Fig. 2. SQL Queries for test question 1.

The Test Query 2 List details of presidents who belonged to the party which has the highest number of presidents in Ohio.

This query is a complex query which requires nested subqueries. To find the party which has the highest number of

presidents in Ohio, one must first find the number of presidents in Ohio for each party. This is done at the lowest level subquery. The next level subquery checks the party which has the maximum number of presidents from the set. The outer most level query lists the president details of all presidents which belong to the party. The corresponding SQL queries as coded manually by a human tutorial assistant are as shown in Fig3:

Fig. 3. A SQL Query for test question 2.

The Test Query 3 List the rows of the top two candidates by votes for each election year.

Using the conventional SQL statement without any vendor-specific extension, this is a very hard query to comprehend. To find the top two candidates by votes for each election year, the idea is to find the second maximum votes for each election year first. This can be achieved by using a correlated subquery which returns the number of row(s) whose candidate has votes greater than the one which is being checked. If the returned number from the subquery is 1, it means the row which is being checked is the 2nd one. Since both the maximum and the second maximum numbers of votes are required, the condition is to check that number of rows returned is <=1. The corresponding SQL queries as coded manually by a human tutorial assistant are as shown in Fig 4:

Fig. 4. A SQL Query for test question 3.

The Test Query 4 List the name of the youngest second wife.

This is a very hard question. In conventional SQL, either a view or a temp table is required to obtain the second wife rows first. The youngest second wife is then obtained by using a subquery that returns the minimum age from the view or temp table. The corresponding SQL queries as coded manually by a human tutorial assistant are as shown in Fig 5:

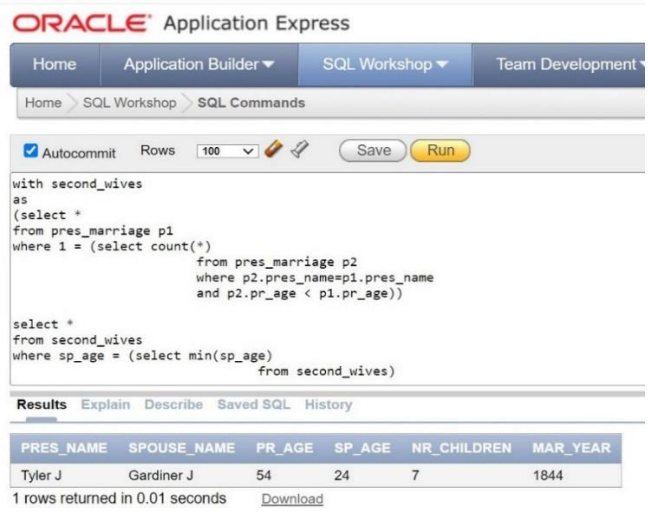

Fig. 5. A SQL Query for test question 4.

III. Experiments Using LLM Systems for Query Generations

A. Experiment with ChatGPT

ChatGPT gives the same correct SQL query as the one obtained from the human tutorial assistant (TAs) for test query 1. For the test query 2, however, it gives an incorrect SQL query given the following prompt: From the Presidential database, give an SQL query for each of the following question: List details of presidents who belonged to the party which has the highest number of presidents in Ohio. It gives an SQL statement which retrieves only the row of a Republican president who were born in Ohio instead of retrieving all presidents who belong to the Republican party which is the party that has the highest number of presidents in Ohio.

The generated query for test query 3: List the rows of the top two candidates by votes for each election year; is impressive. It does not use the correlated subquery technique that the human TA used. Instead, it uses the RANK() OVER (PARTITION BY ELECTION_YEAR ORDER BY VOTES DESC) which is understood by the Oracle Express[ ] system that we use for testing the queries, together with the temp table WITH as shown in Fig 6. The query gives correct results.

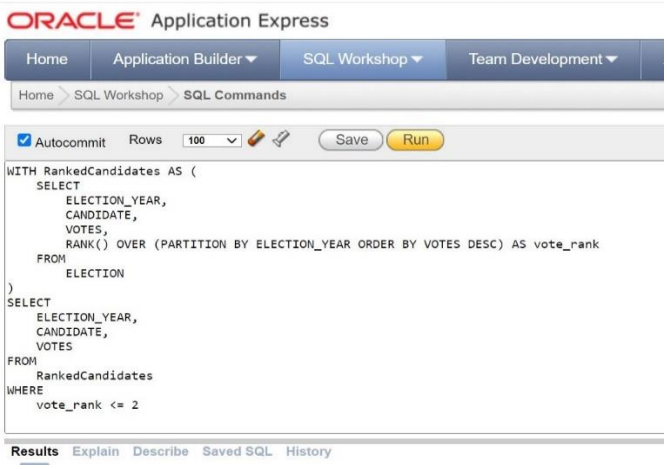

Fig. 6. A SQL Query for test question 3 as generated by ChatGPT.

ChatGPT shows a promising attempt for the very hard test question 4. The query is: List the name of the youngest second wife. The intended query result is expected to be the name of the youngest among the wives from the second presidential marriages. Bard misinterpreted the request and generated a query which retrieves the wife of the youngest president who had a second marriage instead.

B. *Experiment with Bing*

Bing generates SQL query for test query 1 correctly. It generates a query which gives correct results for test question 2 using an extension of SQL which does not run on our Oracle Express system but runs on MySQL. It generates an impressive SQL query for test question 3: List the rows of the top two candidates by votes for each election year; is very impressive. Bing does not use the WITH temp table that ChatGPT uses as shown in Figure 6. Instead, Bing uses the advanced technique of a subquery in the FROM clause as shown in Figure 7 with correct results. However, for the very hard test question 4, Bing admitted that it cannot provide an answer.

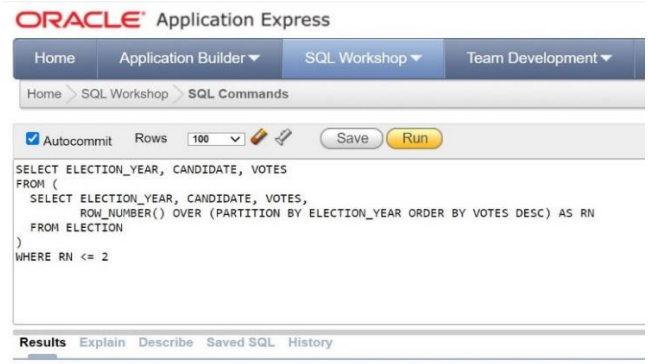

Fig. 7. A SQL Query for test question 3 as generated by Bing.

C. *Experiment with Bard*

Google's Bard is an LLM system released by Google in May 2023. It is capable of generating SQL statements, given the table structures and queries definition in English. It does not need further explanations on the meaning of the column names since the names already have understandable meanings.

Bard gives a correct SQL query for test question 1 and test question 2. The generated SQL statements for question 2 and 3 do not run on Oracle Express but run on MySQL. For the very hard question 4, Bard creates a view which correctly shows row of second marriages only but fails to find the youngest second wife from the view.

D. *Experiment with Copilot*

Copilot is an AI companion provided by Microsoft. For the test question 1: From the Presidential database, give an SQL query of the following question: List hobbies that at least 3 Democratic presidents have in common. Copilot correctly generates an SQL statement, together with comments. Figure 8 shows the Copilot-generated SQL query with results from Oracle Express. Note that copilot not only shows the hobbies as asked, but also the party name together with the number of presidents which had the hobbies, the extra information that we do not ask.

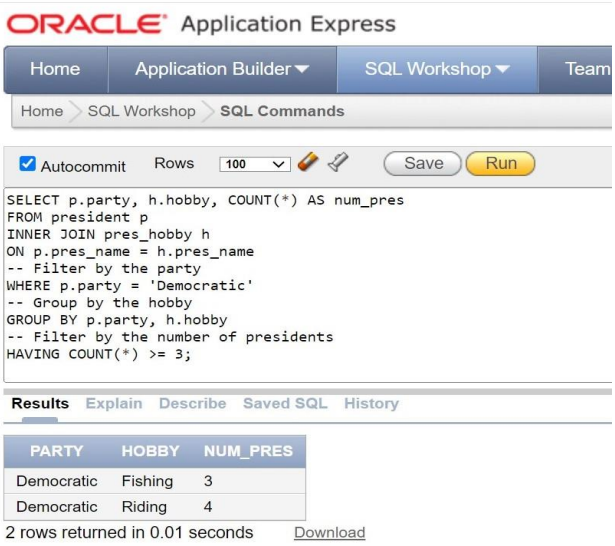

Fig. 8. A SQL Query for test question 1 as generated by Copilot.

For the test question 2: List details of presidents who belonged to the party which has the highest number of presidents in Ohio. Copilot generates SQL codes that do not run on Oracle Express but run correctly on MySQL.

For the hard test question 3: List the rows of the top two candidates by votes for each election year. Copilot generates identical SQL codes that ChatGPT generates plus comments, as shown in Fig 9.

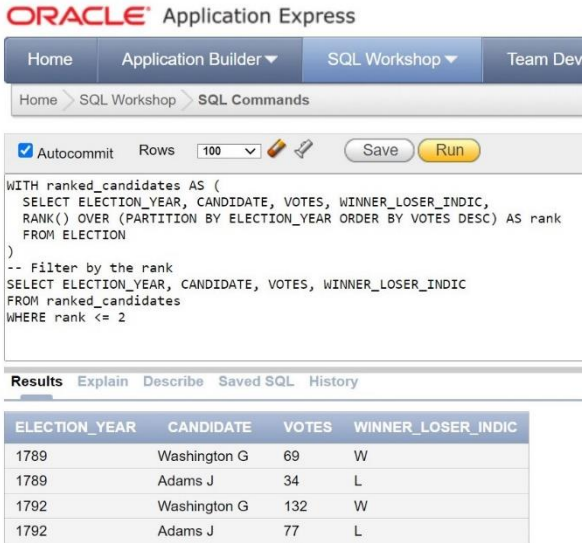

Fig. 9. A SQL Query for test question 3 as generated by Copilot.

Copilot shows a promising attempt for the very hard test question 4. The query is: List the name of the youngest second wife. The intended query result is expected to be the name of the youngest wife among the wives from the second presidential marriages. ChatGPT misinterpreted the request and generated a query which retrieves the youngest wife of the president who had a second marriage and shows the one from his first marriage instead.

IV. CONCLUSION

The four LLM AI systems used in our project all demonstrate very high capability for SQL queries generation. All of them generate SQL codes correctly for simple queries that have only the simple relational algebra select, project, and join operations. All of them provide very clear explanations which explain how the codes work step by step. When it comes to complex to very complex questions, they also perform well. All the four systems give correct SQL queries for the first three complex queries. The fourth one, which is very hard, is still a challenge for all of them. The LLM systems used in this project are free of charge at the time of this project. They provide very high productivity application development assistance and reasonable trustworthiness. All of them are recommended as SQL database programming assistants. Nonetheless, the generated SQL codes should always be quality controlled and tested before production.

REFERENCES

[1] D. D. Chamberlin et al., "SEQUEL: A Structured English Query Language," in Proc. 1974 ACM SIGFIDET (now SIGMOD) Workshop on Data description, access and control, 1974, pp. 249-264.

[2] Information technology, Database languages, Part 2 Foundations, SQL, ISO/IEC 9075-1:2011, accessed 19 January, 2024, https://www.iso.org/standard/53682.html

[3] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I. (2019). Language Models are Unsupervised Multitask Learners. arXiv:1901.02860, accessed 19 January 2024, https://d4mucfpksywv.cloudfront.net/better-language-models/language_models_are_unsupervised_multitask_learners.pdf

[4] Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., ... & Amodei, D. (2020). Language Models are Few-Shot Learners. arXiv:2005.14165, accessed 19 January 2024, https://arxiv.org/pdf/2005.14165.pdf

[5] Bard, accessed 19 January 2024, https://bard.google.com/chat

[6] Bing, accessed 19 January 2024, https://www.microsoft.com/en-us/bing?form=MG0AUO

[7] Copilot, accessed 19 January 2024, https://copilot.microsoft.com/

[8] OpenAI Blog. ChatGPT: Engaging and Contextual AI Conversations, accessed 19 January 2024, https://openai.com/blog/chatgpt

[9] Immanuel Trummer. 2022. CodexDB: Synthesizing code for query processing from natural language instructions using GPT-3 Codex. Proceedings of the VLDB Endowment 15, 11 (2022), 2921–2928.

[10] Raul Castro Fernandez, Aaron J. Elmore, Michael J. Franklin, Sanjay Krishnan, and Chenhao Tan. 2023. How Large Language Models Will Disrupt Data Management. Proceedings of the VLDB Endowment Volume 16 Issue 11 pp 3302–3309 https://doi.org/10.14778/3611479.3611527

[11] G.M. Nyssen and E. D. Falkenberg, IBM SQL Covering SQL/DS, Brisbane, Australia: The University of Queensland, 1984.

[12] Ter, Arthur & Proper, Henderik & Weide, Theo. (1996). Query Formulation as an Information Retrieval Problem. Comput. J.. 39. pp. 255-274. 10.1093/comjnl/39.4.255.

[13] Oracle Database Express Edition, accessed 19 January, 2024, https://www.oracle.com/database/technologies/appdev/xe.html