JKSCI

한국컴퓨터정보학회논문지
Journal of The Korea Society of Computer and Information
Vol. 31 No. 4, pp. 147-156, April 2026
https://doi.org/10.9708/jksci.2026.31.04.147

Improving Text-to-SQL Reliability by Leveraging Operational SQL

Myoungkuk Nam*, Namgyu Kim**

*PhD Student, Graduate School of Business IT, Kookmin University, Seoul, Korea
**Professor, Graduate School of Business IT, Kookmin University, Seoul, Korea

[Abstract]

Effective use of relational databases requires substantial SQL expertise, driving active research into Text-to-SQL, which aims to translate natural language queries into SQL statements. Recently, generation-based approaches leveraging large language models (LLMs) have gained considerable attention. However, purely LLM-based methods face inherent limitations in handling natural language ambiguity, schema linking, and the reliability of generated outputs. Against this backdrop, this study proposes OpSQL-Leverage (OSL), a methodology that exploits SQL queries accumulated in real-world operational environments as core knowledge assets. Specifically, OSL selects candidate SQLs from an operational database based on an LLM-generated SQL, then synthesizes these candidates with the original natural language query to produce the final SQL. Experiments on the SPIDER dataset show that OSL achieves higher accuracy than purely LLM-based Text-to-SQL models. This work is significant in that it extends retrieval-augmented generation (RAG) approaches to the Text-to-SQL domain by leveraging structured SQL assets within operational databases as external knowledge.

▸ **Key words**: Large Language Model, Text-to-SQL, DBMS, SPIDER dataset

[요 약]

관계형 데이터베이스의 효과적인 활용을 위해서는 SQL 전문 지식이 요구된다. 이에 자연어 질의를 SQL로 변환하는 Text-to-SQL 연구가 활발히 진행되어 왔으며, 최근 대규모 언어 모델(LLM) 기반 생성 접근법이 주목받고 있다. 그러나 순수 LLM 기반 방법은 자연어 모호성, 스키마 연결의 어려움, 생성 결과의 신뢰성 측면에서 한계를 가진다. 이러한 배경에서, 본 연구는 실제 운영 환경에 축적된 SQL을 핵심 지식 자산으로 활용하는 OpSQL-Leverage(OSL) 방법론을 제안한다. 구체적으로 OSL은 LLM이 생성한 SQL을 기준으로 운영 DB에서 후보 SQL을 선별하고, 이를 자연어 질의와 합성하여 최종 SQL을 생성한다. SPIDER 데이터셋 실험 결과, OSL은 순수 LLM 기반 모델 대비 향상된 정확도를 달성하였다. 본 연구는 정형 SQL 자산을 외부 지식으로 활용하는 RAG 접근법을 Text-to-SQL 영역으로 확장하였다는 데 의의가 있다.

▸ **주제어**: 대규모 언어 모델(LLM), Text-to-SQL, 데이터베이스 관리시스템, SPIDER 데이터셋

• First Author: Myoungkuk Nam, Corresponding Author: Namgyu Kim
*Myoungkuk Nam (nmk10068@kookmin.ac.kr), Graduate School of Business IT, Kookmin University
**Namgyu Kim (ngkim@kookmin.ac.kr), Graduate School of Business IT, Kookmin University
• Received: 2026. 02. 02, Revised: 2026. 03. 06, Accepted: 2026. 03. 23.

http://www.ksci.re.kr pISSN:1598-849X | eISSN:2383-9945

I. Introduction

관계형 데이터베이스는 오늘날 방대한 양의 정보를 저장하고 있으며, 의료기록, 금융시장, 고객관리와 같은 다양한 응용 시스템의 기반을 제공한다[1]. 관계형 데이터베이스에 접근하여 자료의 입력, 수정, 삭제, 그리고 갱신 등의 행위를 하기 위해서는 사용자가 SQL과 같은 질의 언어를 사용해야 한다. SQL은 범용성과 활용성이 높은 언어이지만, 데이터를 자유롭게 다루고 최적의 쿼리를 생성하기 위해서는 충분한 경험과 지식이 필요하다. 따라서 데이터베이스에 대한 이해가 없는 일반 사용자들이 SQL을 통해 관계형 데이터베이스를 사용하기에는 다소 어려움이 존재한다[1]. 이를 해소하기 위해서 많은 웹사이트에서 사용자가 자주 하는 질문의 집합을 구성하여 새로운 문제 해결에 도움을 주는 FAQ를 제공하고 있는데, 이와 유사한 맥락으로 기존에 생성되어 있는 SQL문을 다시 활용하거나 매개변수를 선택사항으로 매칭하여 그 결과를 제시하는 방식도 생각할 수 있다. 이처럼 인간이 자연어를 사용해 컴퓨터와 상호작용하기 위한 자연어 인터페이스에 대한 연구가 꾸준히 수행되어 왔다[2].

Text-to-SQL은 관계형 데이터베이스에 대한 자연어 질의를 SQL로 변환해주는 의미 분석 기법 중 하나이다. 이는 SQL에 대한 이해가 부족하더라도 데이터베이스에서 원하는 데이터에 쉽게 접근할 수 있도록 하여, SQL 비전문가의 학습 비용을 절약하고 사용자의 데이터 활용도를 높이는 것을 목적으로 한다[3]. 자연어를 이용해 데이터베이스를 효과적으로 질의할 수 있는 능력은 질의 언어에 익숙하지 않은 사용자가 대규모 데이터셋을 목적에 맞게 활용할 수 있는 가능성을 열어 준다. 이러한 배경에서, 많은 연구들은 자연어 질문을 데이터베이스 관리 시스템이 실행할 수 있는 SQL로 변환하는 문제에 주목해 왔다[3].

최근에는 대규모 언어 모델(Large Language Model, LLM)을 활용하여, 모델의 문맥 학습(In-context Learning) 방법을 통해 SQL을 생성하는 방법이 활발하게 연구되고 있다[4]. 문맥 학습 방법은 사전 학습된 LLM의 지식을 바탕으로, 사용자가 제공하는 퓨샷(Few-shot) 예시를 통해 SQL 생성에 필요한 패턴과 요구사항을 추론하는 방식이다. 이를 통해 추가적인 학습 없이도 복잡한 데이터베이스 구조와 사용자 질문을 효과적으로 이해할 수 있게 되었고, SQL 생성 성능도 크게 향상되었다[5]. 특히, 외부 지식 베이스에서 사용자의 질문과 유사한 예시를 검색하거나 데이터베이스 내에서 유사성이 가장 높은 자연어 질문을 선별하여 예시로 활용하는 등, 검색 증강 생성(Retrieval Augmented Generation, RAG) 기법이 효과적으로 적용되고 있다[6].

그러나 사람이 인식하는 자연어를 컴퓨터가 인식하는 SQL로 변환하는 과정에는 다음과 같은 어려움이 존재한다[7]. 첫째, 자연어 질의의 모호성이다. 이는 하나의 단어가 여러 의미를 가지는 어휘적 모호성과 하나의 문장이 여러 가지로 해석이 될 수 있는 해석의 문제가 존재한다. 둘째, 스키마 연결의 문제이다. 자연어 질문의 어떤 부분이 데이터베이스 스키마의 어떤 요소를 가리키는지 이해하는 문제로, 단순한 텍스트 매칭으로 해결될 수 없는 경우 사용자의 문법적·구문적 오류로 인해 더욱 복잡한 문제를 야기하기도 한다. 셋째, 결과의 검증에 관한 문제이다. SQL에 익숙하지 않은 사용자들은 생성된 SQL이 실제로 자연어 질의의 의도와 일치하는지 확인하기가 어렵다. 마지막으로 다양한 데이터베이스에서 동일한 성능을 발휘할 수 있는지에 대한 보편성의 문제가 존재한다. 이상적인 Text-to-SQL 시스템은 관리자의 도움 없이 서로 다른 도메인과 스키마를 안정적으로 처리할 수 있어야 한다[7].

이에 본 연구에서는 관계형 데이터베이스 운영 환경에 축적된 SQL을 핵심 지식 자산으로 활용(Leverage)하여 Text-to-SQL의 신뢰성을 향상시키는 OpSQL-Leverage (OSL) 방법론을 제안한다. 본 방법론은 LLM이 생성한 SQL을 그대로 사용하지 않고, 이미 생성되어 있는 운영용 SQL을 활용하여 SQL의 구조적 일관성과 실행 안정성을 보완한다는 점에서 기존 접근법들과는 차별성을 갖는다. 실험에서는 SPIDER 데이터셋의 Student DB에 대해 실제 자연어 질의에 대한 응답 정확도를 분석하여, 제안하는 OSL 방법론의 유용성을 검증하고자 한다.

본 논문의 이후 구성은 다음과 같다. 2장에서는 LLM, DBMS, Text-to-SQL, 그리고 SPIDER 데이터셋 등에 대한 기존연구와 동향 등을 소개하고, 3장에서는 본 연구에서 제안하는 OSL 방법론을 소개한다. 4장에서는 제안 방법론의 성능 평가 결과를 제시하고, 5장에서는 본 연구의 기여와 한계를 요약한다.

II. Preliminaries

1. LLM

LLM은 통계 기반 초기 언어 모델에서 출발하여, 방대한 말뭉치와 사전 학습 기법을 활용하는 트랜스포머(Transformer)의 등장과 함께 인간 언어를 이해하고 새로운 텍스트를 생성하는 고도화된 모델로 진화해 왔다. 이

러한 발전을 가능하게 한 핵심 요인은 수천억 개 규모의 파라미터를 통해 대량의 텍스트 데이터를 학습함으로써, 모델의 언어 이해 및 생성 능력을 크게 향상시킨 점이다[8]. 그 결과, LLM은 자연어 처리 분야의 혁신을 이끌며 위험 평가, 소프트웨어 개발, 취약점 탐지, 의료 문서 분석, 그리고 검색 엔진 최적화 등 다양한 영역에서 활용되고 있다[9-13].

일반적으로 LLM이 갖추어야 할 특성은 다음 네 가지로 요약할 수 있다[14]. 첫째, 자연어 텍스트를 깊이 있게 이해하고 해석하는 능력으로, 정보 추출·요약·번역 등 다양한 언어 기반 작업을 수행할 수 있어야 한다. 둘째, 주어진 질문이나 조건에 따라 자연어와 유사한 품질의 텍스트를 생성할 수 있어야 한다. 셋째, 특정 도메인의 지식과 맥락을 반영하여 상황에 맞는 출력물을 생성할 수 있는 맥락 인식 능력이 필요하다. 넷째, 문제 해결과 의사결정 과정을 지원할 수 있어야 한다.

이러한 LLM은 학습에 사용된 소스의 공개 여부에 따라 개방형 모델 및 폐쇄형 모델로 나뉘며, Table 1에 나타난 바와 같이 다양한 모델들이 있다[15].

Table 1. Comparison of popular LLMs

Model	Provider	Params	Open -Source	Tunability
GPT-4	OpenAI	1.7T	×	×
GPT-3.5 turbo	OpenAI	175B	×	×
GPT-3	OpenAI	175B	×	×
BERT	Google	340M	○	○
PaLM	Google	540B	○	○
LLaMA	Meta AI	65B	○	○

개방형 모델은 개발자가 파라미터의 조정과 추가 학습을 통해 성능을 개선할 수 있으므로, 내부 자료를 보호하고 특정 목적을 달성하기 위해 모델이 필요한 기업이나 공공기관에서 사용하기에 좀 더 적합한 것으로 알려져 있다[15]. 한편 폐쇄형 모델 가운데 대규모 대화형 언어모델인 ChatGPT는 대화 능력과 코드 생성 능력에서 모두 뛰어난 성과를 보이고 있으며, 서로 다른 언어·설정·시나리오로 구성된 12개의 벤치마크 데이터셋에 대한 실험에서 강력한 Text-to-SQL 능력을 보유하고 있음과 실용적 응용 가능성이 있음을 입증하였다[16].

2. DBMS

데이터베이스 관리시스템(Database Management, DBMS)은 데이터베이스를 생성하고 관리하여, 데이터로부터 사용자의 물음에 대한 대답을 추출하는 프로그램의 집합을 말한다[17]. DBMS는 전체 데이터를 저장하고 관리하며, 이 데이터를 쉽고 효율적으로 사용할 수 있도록 도와주는 여러 가지 도구들을 제공하는 소프트웨어 패키지이다. DBMS를 통해 데이터를 중복없이 통합할 수 있으며, 효율적인 저장구조를 이용하여 저장할 수 있다. 또한 보안기능, 회복기능, 동시성 제어, 그리고 성능의 감시 등과 같이 데이터에 대한 고급 관리기능이 제공된다. 구체적으로 DBMS는 여러 이용자가 자료를 동시에 공동으로 이용할 수 있는 동시 공용성, 사용자의 요구에 대한 즉각적인 응답이 이루어지는 실시간 접근성, 수시로 반복되는 자료의 삽입, 삭제, 갱신의 용이성, 그리고 사용자가 원하는 결과를 반환하는 검색의 용이성 등을 통해 자료 이용의 효율을 높여준다[17].

DBMS의 종류로는 계층형(Hierarchical), 망형(Network), 관계형(Relational), 객체지향형(Object Oriented), 객체관계형(Object Relational) 등이 있으며, 이 중에서 관계형 데이터베이스 시스템(Relational Database Management System, RDBMS)이 학계와 실무에서 가장 많이 사용되고 있다[18]. DBMS의 사용 경향은 Fig. 1과 같이 웹사이트에서 검색되는 횟수, 기술적 토론의 빈도, 관련 구인 공고 및 구인 검색 엔진, 전문가 네트워크 프로필 수(LinkedIn, 링크드인), 그리고 소셜 네트워크에서의 연관성 등을 고려하여 매월 분석되고 있다[18]. DBMS 모델 순위 TOP 10에는 관계형 데이터베이스 관리시스템(RDBMS)이 대부분을 차지하고 있으며, 많은 기업의 업무 시스템에서 활용되고 있다[19].

Rank Nov 2025	Rank Oct 2025	Rank Nov 2024	DBMS	Database Model
1.	1.	1.	Oracle	Relational, Multi-model
2.	2.	2.	MySQL	Relational, Multi-model
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model
4.	4.	4.	PostgreSQL	Relational, Multi-model
5.	5.	5.	MongoDB	Multi-model
6.	6.	↑ 7.	Snowflake	Relational
7.	7.	↓ 6.	Redis	Key-value, Multi-model
8.	8.	↑ 14.	Databricks	Multi-model
9.	9.	9.	IBM Db2	Relational, Multi-model
10.	10.	↓ 8.	Elasticsearch	Multi-model

Fig. 1. DB-Engines Ranking[18]

또한 많은 데이터베이스 사용자들은 Fig. 2에서 보는 바와 같이, 질의의 효율성을 위해 이미 작성되어 있는 트랜잭션/Query를 활용하여 정보를 획득하는 것이 일반적이다[20].

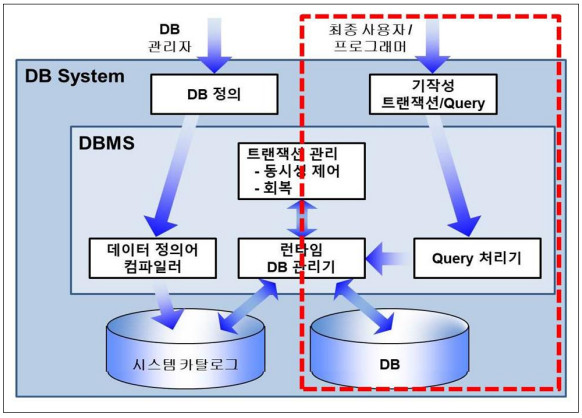


Fig. 2. Database System Architecture[20]

다양한 RDBMS 중 SQLite는 2000년부터 개발되어 온 퍼블릭 도메인의 경량 DBMS로, 별도의 서버 구축이나 설치·설정 과정 없이 응용 프로그램 내부에 직접 포함되어 실행될 수 있다는 장점으로 인해 많은 연구와 응용 분야에서 활용되고 있다. SQLite의 주요 특징은 테이블, 인덱스, 뷰 등 데이터베이스를 구성하는 모든 객체가 하나의 파일에 저장된다는 점으로, 운영체제나 하드웨어 환경이 달라도 해당 파일을 복사하는 것만으로 동일한 데이터베이스를 사용할 수 있다. 또한 SQLite는 소스 코드가 완전히 공개되어 있어 누구나 무료로 이용할 수 있으며, 전 세계 개발자 커뮤니티에 의해 지속적인 성능 개선과 안정화가 이루어지고 있다. 반면, 동시 접속 처리 성능과 대규모 데이터 처리 능력, 네트워크 기반 분산 환경에 대한 지원에는 한계가 있어, 대형 시스템보다는 소규모 또는 경량 애플리케이션에 주로 사용된다[21].

3. Text-to-SQL

Text-to-SQL은 자연어 질의를 관계형 데이터베이스에서 실행 가능한 SQL 쿼리로 변환하는 작업을 의미한다. Text-to-SQL의 목표는 자연어 질의가 주어졌을 때 사용자의 의도를 정확하게 반영하는 SQL을 생성하여, 이를 데이터베이스에서 실행했을 때 적절한 결과를 반환하도록 하는 것이다[22].

Text-to-SQL 솔루션의 발전은 규칙 기반, 신경망 기반, 사전학습 언어모델(Pre-trained Language Model, PLM) 기반, 그리고 LLM 기반의 네 가지 단계를 거쳐서 이루어졌다[22]. 초기에는 통계적 언어 모델을 활용하여 자연어 질의를 해석하고, 미리 정의된 규칙을 통해 자연어를 SQL 쿼리로 변환하였다. 이 시기의 변환은 토큰 수준의 자연어 이해와 단일 테이블 SQL 질의 생성에 중점을 두고 이루어져서, 적응성, 확장성, 그리고 일반어 능력 측면에서 많은 한계를 가지고 있었다[23-25]. 신경망 기반 단계에서는 변환 방법론이 다중 테이블을 활용하는 더 복잡한 형태로 확장되어 동의어 처리와 사용자의 의도 이해 능력이 향상되었지만, 모델 크기와 충분한 학습 데이터 확보라는 한계로 인해 일반화 능력은 여전히 제한적이었다. PLM 기반 단계에서는 대규모 코퍼스를 기반으로 학습된 PLM을 활용하여 자연어 이해 능력을 크게 향상시켰지만, 복잡한 스키마를 처리하는 데에는 여전히 도전 과제가 존재하였다[26-27]. 마지막으로 LLM 기반 단계에는 자연어 이해 능력이 더욱 향상되어, Text-to-SQL의 이슈가 프롬프트 설계 최적화[28]와 파인튜닝[29]에 초점을 맞추고 점점 도메인의 데이터베이스 특화 문제로 고도화되고 있다[28-29].

4. SPIDER Dataset

SPIDER는 Text-to-SQL 연구 분야에서 모델의 일반화 능력을 평가하기 위한 대표적인 벤치마크 데이터셋으로, Yale NLP 그룹에 의해 2018년 공개되었다[30]. 기존의 단일 도메인 혹은 단순 질의 중심의 데이터셋(SPARC, WikiSQL 등)이 갖는 한계를 극복하고자, 여러 도메인의 복잡한 자연어 질의와 해당 SQL 쿼리로 구성되어 있다. 해당 데이터셋은 일반적인 상식이나 수학적 연산을 요구하는 쿼리는 배제하고, 개발자만 알 수 있는 축약어 형태의 용어는 LLM이 이해할 수 있는 형태('stu_id' → 'student id' 등)로 정제하여 제공한다는 장점을 갖는다[30].

5. FAISS

FAISS(Facebook AI Similarity Search)[31]는 Facebook AI Research에서 개발한 오픈소스 라이브러리로, 고차원 벡터 데이터에 대한 효율적인 유사도 검색과 클러스터링을 지원한다. FAISS는 대규모 벡터 집합에 대한 검색을 위해 설계되었으며, RAM 용량을 초과하는 데이터셋에서도 효율적인 검색을 수행할 수 있는 다양한 알고리즘을 제공한다. 또한 L2 거리(Euclidean Distance), 내적(Dot Product), 코사인 유사도(Cosine Similarity) 등 다양한 거리 및 유사도 측정 방식을 지원하여, 다양한 검색 요구사항에 대응할 수 있다. 이러한 특성으로 인해 FAISS는 최근 벡터 기반 정보 검색 시스템 및 RAG 구조에서 벡터 인덱싱 및 검색을 위한 기술로 활용되고 있다.

III. The Proposed Scheme

1. Research Process

본 장에서는 본 연구의 제안 방법론인 OSL (OpSQL-Leverage)의 단계별 절차와 구성 요소를 체계적으로 설명한다. OSL은 기존의 Text-to-SQL 접근법과 달리, 이미 제작되어 관리하고 있는 운영 SQL 쿼리들을 적극적으로 활용한다는 특징을 갖는다. 구체적으로 OSL은 운영 데이터베이스에 존재하는 SQL 쿼리들 가운데 자연어 질의와 유사한 쿼리를 식별하고, 이 쿼리와 자연어 질의를 합성하여 새로운 쿼리를 생성한다. 제안 방법론의 전체 구조는 Fig. 3과 같다.

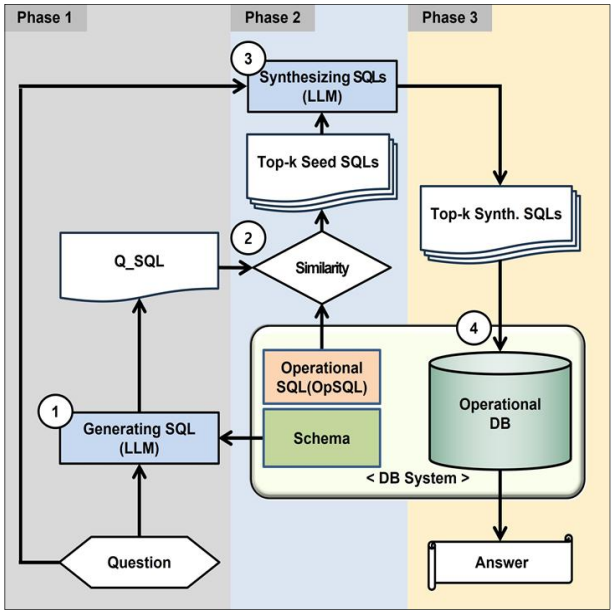


Fig. 3. Overall Architecture of OSL

먼저 Phase 1은 자연어 질의어를 입력으로 받아 LLM을 활용하여 Q_SQL을 생성하는 단계이다(Generation SQL). Phase 2에서는 운영용 데이터베이스 시스템에 구현되어 있는 SQL문, 즉 OpSQL 중 Q_SQL과 유사도가 높은 Seed SQL을 k개 선택한다(②). 다음으로 LLM을 활용하여, k개의 Seed SQL문 각각과 Phase 1에서 사용된 자연어 질의를 융합하여, 질의자의 의도가 반영된 SQL을 합성한다(Synth. SQLs)(③). 이후 Phase 3에서는 k개의 Synth. SQLs을 운영용 데이터베이스에 각각 적용하여 k개의 결과를 도출한다(④). 각 단계에 대한 세부 프로세스는 다음의 각 절에서 설명하며, 실제 데이터를 적용한 제안 방법론의 성능 평가 결과는 4장에서 제시한다.

2. Q_SQL Generation from NL Question

본 단계에서는 Fig. 3의 Phase 1, 즉 LLM을 활용하여 자연어 질의어로부터 초기 SQL을 생성한다(①). 제안 기법은 데이터베이스의 스키마 정보를 프롬프트 형태로 제공하고, 이를 활용하여 LLM이 SQL을 생성한다. Fig. 4에서와 같이 OpenAI사의 Text-to-SQL Template을 모델별로 준용하여 적용하였으며[32], 이렇게 생성된 Q_SQL의 결과 예시는 Table 2와 같다.

Fig. 4. Text-to-SQL Template[31]

Table 2. Example of Q_SQL

NL Question : Report the names of teachers who have between seven and eight students in their classes	
(GPT4o) SELECT t.Name_teacher FROM Teachers t JOIN csbc c ON t.Classroom = c.Classroom WHERE c.student_count BETWEEN 7 AND 8;	(Mistral 7B) SELECT DISTINCT Teachers.Name_teacher FROM Teachers INNER JOIN (SELECT Classroom, COUNT(*) as student_count FROM Students GROUP BY Classroom) as count_students_by_classroom ON Teachers.Classroom = count_ students_by_classroom.Classroom WHERE count_students_by_classroom .student_count BETWEEN 7 AND 8;

3. Seed SQL Selection and Synthesization

본 단계에서는 운영용 데이터베이스에 보관되어 있는 OpSQL 중 앞에서 생성된 Q_SQL과 유사도가 높은 상위 k개의 SQL을 Seed SQL로 선별하고(②), 이들 Seed SQL문 각각에 대해 최초 자연어 질의어를 융합하여 k개의 합성 SQL(TOP-k Synth. SQL)을 생성한다(③). 제안 방법론은

Seed SQL 선별 과정에서 자연어 질의어를 SQL 형태로 변환한 후 이를 이미 작성되어 있는 OpSQL과 비교하였는데, 이는 SQL 문장간 유사도가 SQL과 자연어 간의 유사도보다 더 높은 값을 나타낸다는 사전 실험 결과에 따른 것이다(Fig. 5). 유사도 비교를 위해 본 연구에서는 FAISS를 활용하여 각 SQL 문장을 벡터화하여 벡터 저장소에 저장하고, 그 값을 비교하여 유사도를 평가하였다. Fig. 5에서 Q는 자연어 상태의 질의어를 나타내며, Q_SQL과 OpSQL은 Fig. 3에서 소개된 SQL문을 나타낸다.

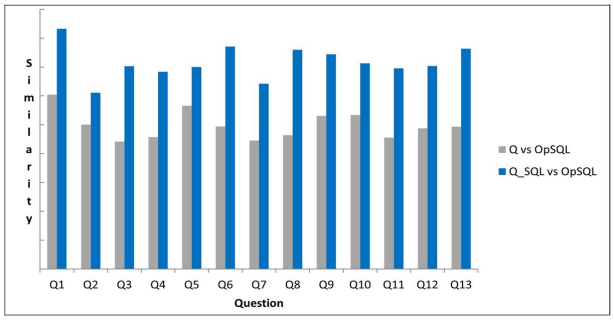


Fig. 5. Similarity between 'Q vs OpSQL' and 'Q_SQL vs OpSQL'

이러한 절차에 따라, OpSQL에 자연어 질의어를 융합하여 생성한 Synth. SQL의 예는 Table 3과 같다.

Table 3. Example of Synthesization

(NL Question)
Report the names of teachers who have between seven and eight students in their classes

(Q_SQL)
```
SELECT DISTINCT Teachers.Name_teacher
FROM Teachers INNER JOIN (SELECT Classroom, COUNT(*)
  as student_count FROM Students GROUP BY Classroom)
  as count_students_by_classroom
ON Teachers.Classroom = count_ students_by_classroom.Classroom
WHERE count_students_by_classroom.student_count
BETWEEN 7 AND 8;
```

(Top-1 OpSQL)
```
SELECT DISTINCT
  Teachers.Name_teacher
FROM Students
INNER JOIN Teachers
  ON Students.Classroom
      = Teachers.classroom
INNER JOIN csbc
  ON Teachers.Classroom
      = csbc.classroom
GROUP BY csbc.student_count,
  Teachers. Name_teachers,
  Teachers.classroom
HAVING csbc.student_count >=?
  AND csbc.student_count <=?;
```

(Top-1 Synth. SQL)
```
SELECT DISTINCT
  Teachers.Name_teacher
FROM Students
INNER JOIN Teachers
  ON Students.Classroom
      = Teachers.classroom
INNER JOIN csbc
  ON Teachers.Classroom
      = csbc.classroom
GROUP BY csbc.student_count,
  Teachers. Name_teachers,
  Teachers.classroom
HAVING csbc.student_count >=7
  AND csbc.student_count <=8;
```

4. Querying Operational Databases with Top-k Synthesized SQLs

본 단계에서는 Fig. 3의 Phase 2에서 생성된 Top-k Synth. SQLs를 운영용 데이터베이스에 질의하여 결과를 출력한다(④). 본 연구에서는 여러 LLM 중 외부환경과 연결된 폐쇄용 범용모델로 GPT4o를, 로컬환경에서 활용할 수 있는 개방형 모델로 Mistral 7B를 선택하였다. Mistral 7B 모델은 매개변수가 73억 개임에도 불구하고 벤치마크 전 분야에서 메타의 'LLaMA-2 13B'를 능가했으며, 많은 분야에서 매개변수 340억 개인 'LLaMA-1 34B'를 뛰어넘는 성능을 갖춘 것으로 알려져 있다. 또한, Mistral 7B 모델은 파인튜닝 등을 통해 특정 도메인에 부합된 모델을 만드는데 용이한 장점을 가지고 있다. Table 4는 입력된 자연어 질문을 Q_SQL로 변환하여 이와 가장 유사한 OpSQL을 선별한 뒤, 자연어 질문을 합성하여 Synth. SQL을 생성하고, 이를 운영용 데이터베이스에 질의하여 도출한 결과값을 요약하여 보여 준다.

Table 4. Example of NL Question, Top-1 Synth. SQL, and Results

NL Question	Report the names of teachers who have between seven and eight students in their classes

Top-1 Synth. SQL	SELECT DISTINCT Teachers.Name_teacher
	FROM Students
	INNER JOIN Teachers
	  ON Students.Classroom = Teachers.classroom
	INNER JOIN csbc
	  ON Teachers.Classroom = csbc.classroom
	GROUP BY csbc.student_count, Teachers.Name_teachers,
	  Teachers.classroom
	HAVING csbc.student_count >=7 AND csbc.student_count <=8;

Results	JEROME COVIN LEIA TARRING GORDON KAWA

IV. Experiment

1. Experiment Overview

본 장에서는 앞서 소개한 제안 방법론을 실제 데이터에 적용한 실험 결과를 소개한다. 실험을 위해 SPIDER 데이터셋의 Student DB를 활용하여 운영용 데이터베이스를 구현하였으며, 해당 데이터를 기반으로 총 128개의 자연어 질의를 구성하여 실험 데이터로 사용하였다. 실험 환경은 Python을 통해 구축하였으며, 구체적인 H/W 및 S/W 환경은 Table 5와 같다. 또한 성능 비교 실험의 전체 프로세스는 Fig. 6과 같다.

Table 5. System Environments

H/W	CPU	AMD Ryzen 7
	GPU	Radeon 780M
	RAM	32GB
S/W	OS	Windows 11
	Python	3.12.3
	LLM	GPT4o / Mistral 7B
	Ollama	0.3.13
	LangChain	0.2.1
	FAISS	1.8.0
	SQLite	3. 4. 17

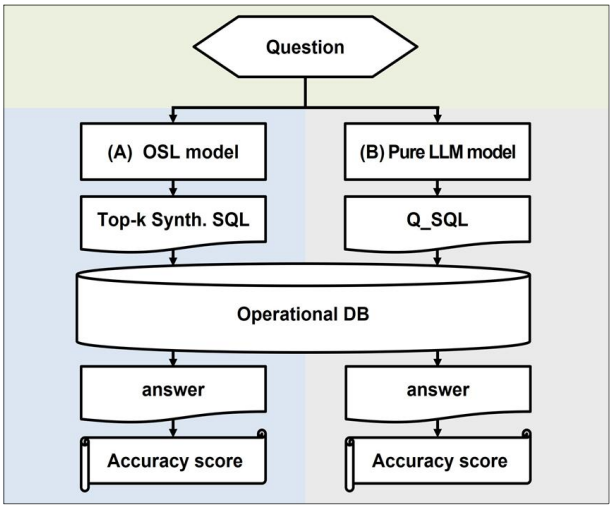


Fig. 6. Overall Process of Performance Evaluation

Fig. 6의 (A)는 본 연구에서 제안한 방법론의 성능을 평가하기 위한 과정으로, 자연어 질의에 대해 Fig. 3의 과정에 따라 Top-k Synth. SQL을 생성하고 이를 운영용 데이터베이스에 적용하여 결과를 도출한다. 이렇게 도출된 결과를 실제 정답과 비교하여 모델의 정확도를 평가한다. 한편, Fig. 6의 (B)는 제안 방법론과의 성능 비교를 위한 실험으로, LLM을 통해 자연어 질의로부터 SQL(Q_SQL)을 생성한 뒤, 이를 운영용 데이터베이스에 적용한 결과의 정확도를 평가한다.

2. Operational DB and Queries

본 연구에서는 Fig. 7 및 Fig. 8과 같이, SPIDER 데이터셋에서 제공하는 Student DB의 13개 질의어를 사용자 요구사항으로 간주하고 두 개 테이블(Teachers, List)을 활용하여 Fig. 9의 운영용 DBMS와 22개의 OpSQL을 구축한다. 실험에 사용된 128개의 질의어 데이터는 13개 질의어에서 매개변수로 생성이 가능한 각 정보를 테이블에 존재하는 60명의 학생, 12명의 선생님, 7개 학년, 12개 교실 데이터를 조합하여 구성한다. 답변 내용의 정확성을 평가하기 위한 정답 데이터는 이미 구축된 OpSQL에 상기 데이터를 직접 활용하여 구성한다.

<Teachers>

LastName	FirstName	Classroom
'MACROSTIE'	'MIN'	101
'COVIN'	'JEROME'	102
'MOYER'	'OTHA'	103
'NIBLER'	'JERLENE'	104
'MARROTTE'	'KIRK'	105
'TARRING'	'LEIA'	106
'URSERY'	'CHARMAINE'	107
'ONDERSMA'	'LORIA'	108
'KAWA'	'GORDON'	109
'SUMPTION'	'GEORGETTA'	110
'KRIENER'	'BILLIE'	111
'SUGAI'	'ALFREDA'	112

<List>

LastName	FirstName	Grade	Classroom
'CAR'	'MAUDE'	2	101
'KRISTENSEN'	'STORMY'	6	112
'VANDERWOUDE'	'SHERWOOD'	3	107
'NOGODA'	'ISMAEL'	0	105
'DANESE'	'JANEE'	4	111
'AMY'	'PATRINA'	1	102
'PREHM'	'SHANEL'	0	104
'GRUNIN'	'EMILE'	5	109
'GELL'	'TAMI'	0	104
'MADLOCK'	'RAY'	4	110
'SUDA'	'DARLEEN'	4	110
'DROP'	'SHERMAN'	0	104

Fig. 7. Tables of Student DB

Question
Find all students who study in classroom 111. For each student list first and last name.
For each classroom report the grade that is taught in it. Report just the classroom number and the grade number.
Find all teachers who teach fifth grade. Report first and last name of the teachers and the room number.
Find all students taught by OTHA MOYER. Output first and last names of students
For each teacher teaching grades K through 3, report the grade (s)he teaches. Each name has to be reported exactly once.
Find all first-grade students who are NOT taught by OTHA MOYER. Report their first and last names
Find and report the number of students taught by LORIA ONDERSMA.
For each grade, report the number of classrooms in which it is taught and the toal number of students in the grade.
Report the names of teachers who have between seven and eight students in their classes.
For each kindergarden classroom, report the total number of students.
Find the teacher name who teach(es) the largest number of students.
Find all classrooms which have fewer students in them than the average number of students in a classroom in the school. Report the classroom numbers
For each grade with more than one classroom, report the last name of the teacher who teachers the classroom with the largest number of students in the grade.

Fig. 8. Questions in Natural Language

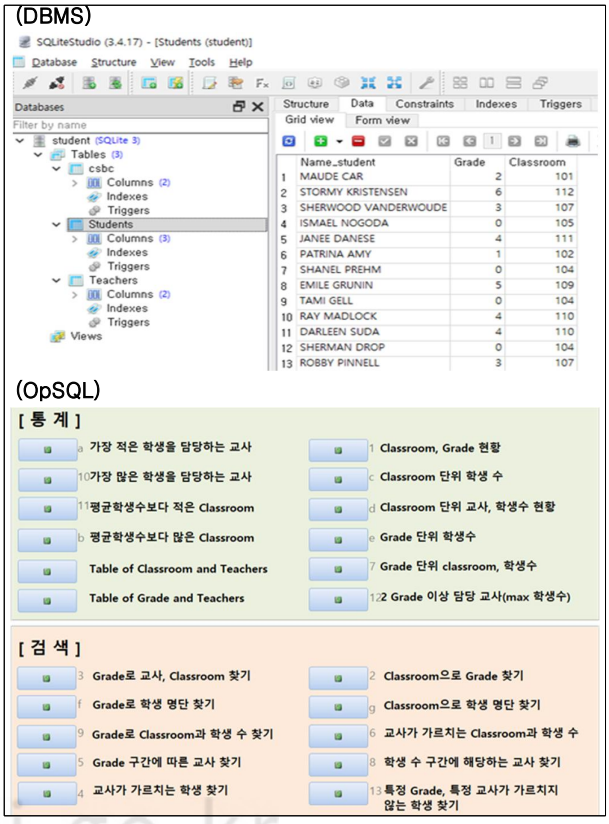


Fig. 9. Student DBMS and OpSQL

3. Performance Evaluation

본 절에서는 관계형 데이터베이스 운영 환경에 축적된 SQL을 핵심 지식 자산으로 활용하여 Text-to-SQL의 신뢰성을 향상시키는 OSL 모델과, 자연어 질의어를 단순 생성 방식으로 처리하는 Pure LLM 모델 간의 정확도를 비교한 실험 결과를 제시한다.

우선 Table 6은 질의에 대한 답변을 나타낸 예시로, 실행 결과는 정확한 답변인 'Correct Answer', 정확하지 않은 답변인 'Incorrect Answer', 그리고 쿼리 실행이 불가한 'Error'와 응답이 없는 'no result'로 구분된다. 본 실험에서 성능 지표로 사용한 정확도는 정답과 완전히 일치하는 답이 k개의 OpSQL 중 하나라도 존재하면 정답으로 간주하였다.

Table 6. Example of Answers

Question	Report the names of teachers who have between seven and eight students in their classes
Correct Answer	JEROME COVIN LEIA TARRING GORDON KAWA
Incorrect Answer	KIRK MARROTTE OTHA MOYER CHARMAINE URSERY
Execution Error	ERROR: Execution failed on sql 'SELECT Students.name'
Empty Result	(no result)

또한 Table 7은 384개의 질의 응답 과정에서 발생한 오류의 유형을 분석한 결과를 제시한다. 해당 표에서 'Execution Error'는 컬럼, 테이블, 문법, 함수 등 SQL 구성요소와 관련된 실행 오류를 의미하고, 'Empty Result'는 SQL과 자연어를 결합하는 과정에서 매개변수 적용이 적절하게 이루어지지 않아 발생한 오류를 의미한다. 이러한 결과는 모델의 성능에 따라 SQL 생성 능력이 오류 발생에 큰 영향을 미친다는 점을 보여준다.

Table 7. Comparing SQL Errors of Each Model

Model	Sum	Empty Result	Execution Error
GPT4o	67	62	5
Mistral 7B	129	60	69

실험 결과, Table 8 및 Fig. 10에서와 같이, 전반적인 정확도는 모델 자체의 성능에 큰 영향을 받음을 알 수 있었다. 구체적으로 후보자의 수를 1개, 2개, 3개로 증가시킴에 따라, GPT4o는 정확도가 50%, 77%, 85%로 향상되었다. 또한 후보자의 수가 1개, 2개인 경우에는 Pure LLM에 비해 28%p 및 1%p 하락하였지만, 3개인 경우에는 Pure LLM의 방법보다 7%p 높은 성능을 보였다. 또한 Mistral 7B의 경우에는 정확도가 45%, 51%, 55%로 향상되었고, 후보자 수의 증가에 따라 Pure LLM보다 1%p, 5%p, 11%p 우수한 성능을 나타냈다. 이를 통해 모델의 성능에 따라 차이는 있지만, 제안 방법론에서 제공하는 후보자 수의 유연성을 통해 성능을 향상시킬 수 있다는 가능성을 확인하였다. 특히 Mistral 7B의 경우 Pure LLM 기반 모델의 성능이 매우 낮아서, 제안 모델을 적용한 모든 실험에서 성능 향상을 확인할 수 있었다.

Table 8. Performance Comparison

Accuracy	(A) OpSQL-Leverage model			(B) Pure LLM
	k=1	k=2	k=3	
GPT 4o	50% (28%p ↓)	77% (1%p ↓)	**85%** (7%p ↑)	78%
Mistral 7B	**45%** (1%p ↑)	**51%** (5%p ↑)	**55%** (11%p ↑)	44%

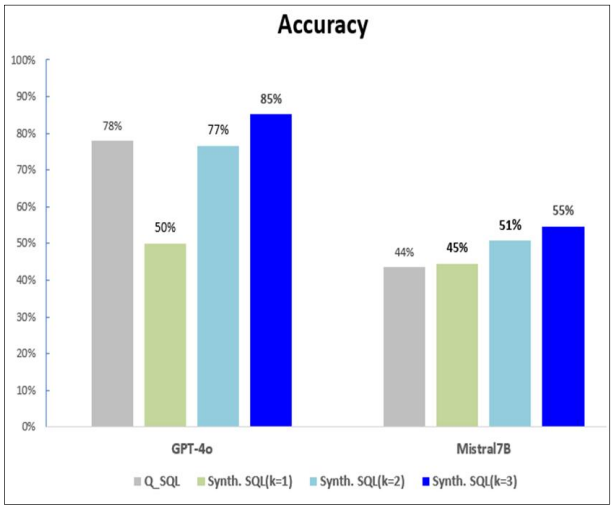


Fig. 10. Performance Comparison

한편, GPT 4o에서 k=1인 경우 정확도가 매우 낮게 나타나는데, 이는 SQL간 유사도를 비교하는 과정에서 OpSQL의 매개변수가 영향을 미친 것에서 원인을 찾을 수 있다. 즉, 기존에 생성되어 있는 OpSQL은 매개변수를 포함하고 있기 때문에, 자연어 질의어의 의미를 충분히 정확하게 이해하기 어렵다. 이러한 이유로 Table 9에서와 같이 GPT-4o에서 k=1의 정확도가 Pure LLM보다 낮게 나타난 것으로 해석할 수 있다.

Table 9. Similarity Comparison

	Q_SQL(k=1)	Q_SQL(k=2)
Similarity	0.8198	0.8505

V. Conclusions

기존의 Text-to-SQL 연구는 데이터베이스의 스키마 정보를 기반으로, LLM이 자연어 질의어를 직접 SQL 쿼리로 생성하는 방식에 초점을 맞추어 진행되어 왔다. 실제 운영 환경의 데이터베이스에는 사용자 요구사항을 반영하여 사전에 정의되고 검증된 SQL 쿼리가 다수 축적되어 있음에도 불구하고, 이러한 SQL 자산은 기존 Text-to-SQL 연구에서 충분히 활용되지 못한 것이 현실이다. 이에 본 연구에서는 운영용 데이터베이스에 이미 구축된 SQL을 적극적으로 활용하는 OSL(OpSQL-Leverage) 방법론을 제안하였다. 제안 방법론은 LLM의 생성 능력을 활용할 뿐 아니라 운영용 데이터베이스에 저장된 기존 OpSQL도 적극적으로 활용하여, 순수 생성 기반 Text-to-SQL 방식이 갖는 구조적 한계를 보완하였다.

본 연구는 운영용 데이터베이스에 이미 존재하는 SQL과 자연어 질의어를 결합하여 SQL을 생성하는 새로운 접근 방식을 제안했다는 점에서 학술적 의의를 갖는다. 또한, 데이터베이스 내부의 정형 데이터를 외부 지식으로 활용한다는 측면에서, 최신 정보 활용을 목표로 하는 RAG 기반 접근법을 Text-to-SQL 영역으로 확장할 수 있는 가능성을 제시했다는 의미가 있다. 특히 실제 운영 환경에서 지속적으로 갱신되는 데이터를 안정적으로 질의할 수 있는 기반을 제공함으로써, Text-to-SQL 시스템의 실용성과 확장성을 동시에 확보할 수 있을 것으로 기대된다.

한편, 본 연구에서는 하나의 도메인을 선택하여 제안 방법론의 우수성을 평가하였다. 향후에는 여러 도메인에 대한 자연어 질의어를 선택하는 방법을 포함하여, 본 방법론을 Cross-domain 환경에서 적용 가능하도록 확장하는 방안이 모색되어야 한다. 또한 본 연구에서는 후보군의 크기 k가 증가할수록 정확도가 향상되는 경향을 확인하였지만, 여러 후보 중 어떤 SQL을 최종적으로 선택해야 하는지에 대한 이슈는 충분히 다루어지지 않았다. 따라서 향후 연구에서는 다수의 후보 SQL에 대한 정교한 선택 및 순위화 전략을 고도화하여, 최종 SQL 선택 과정의 신뢰성과 효율성을 향상시키는 다양한 방안이 다루어져야 한다.

REFERENCES

[1] V. Zhong, C. Xiong, and R. Socher, “SEQ2SQL: Generating Structured Queries from Natural Language Using Reinforcement Learning,” arXiv preprint arXiv: 1709.00103, 2017.

[2] Androutsopoulos, G. D. Ritchie, P. Thanisch, “Natural Language Interfaces to Databases–An Introduction”, Natural Language Engineering, Volume 1, Issue 1, March 1995, pp. 22-81, DOI: https://doi.org/10.1017/S135132490000005X

[3] B. Wang, R. Shin, X. Liu, O. Polozov, and M. Richardson, “RAT-SQL: Relation-Aware Schema Encoding and Linking for Text-to-SQL Parsers,” Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, Jul. 2020. DOI: 10.18653/v1/2020.acl-main.677

[4] Z. Tan, X. Liu, Q. Shu, X. Li, C. Wan, D. Liu, Q. Wan, and G. Liao. “Enhancing Text-to-SQL Capabilities of Lnaguage Models through Tailored Promptings,” LREC-COLING 2024, pages 6091-6109, 20-25 May, 2024.

[5] Z. Li, X. Wang, J. Zhao, S. Yang, G. Du, X. Hu, B. Zhang, Y. Ye, Z. Li, R. Zhao, and H. Mao, “PET-SQL: A Prompt-Enhanced Two-Round Refinement of Text-to-SQL with Cross-consistency,” arXiv:2403.09732, DOI: https://doi.org/10.48550/arXiv.2403.09732

[6] J. Lee, J. Lee, and Y. Choi, “USD-RAG: Unique Schema-Driven Retrieval Augmented Generation for Text-to-SQL via In-Context Learning,” Journal of Korean Institute of Intelligent Systems Vol. 34, No. 5, October 2024, pp. 448-454

[7] G. Katsogiannis-Meimarakis and G. Koutrika, “A Deep Dive into Deep Learning Approaches for Text-to-SQL Systems,” in SIGMOD, 2021.

[8] R. Jain, N. Gervasoni, M. Ndhlovu, and S. Rawat, “A Code Centric Evaluation of C/C++ Vulnerability Datasets for Deep Learning based Vulnerability Detection Techniques,” Proceedings of The 16th Innovations in Software Engineering Conference, pp. 1–10, 2023.

[9] S. Y. Feng, V. Gangal, J. Wei, S. Chandar, S. Vosoughi, T. Mitamura, and E. Hovy, “A Survey of Data Augmentation Approaches for NLP,” 2021. arXiv preprint arXiv:2105.03075

[10] C. Novelli, F. Casolari, A. Rotolo, M. Taddeo, and L. Floridi, “Taking AI Risks Seriously: A New Assessment Model for The AI Act,” AI & Society, pp. 1–5, 2023.

[11] Y. Cai, S. Mao, W. Wu, Z. Wang, Y. Liang, T. Ge, C. Wu, W. You, T. Song, and Y. Xia, “Low-code LLM: Visual Programming over LLMs,” 2023. arXiv preprint arXiv:2304.08103.

[12] A. J. Thirunavukarasu, D. S. J. Ting, K. Elangovan, L. Gutierrez, T. F. Tan, and D. S. W. Ting, “Large Language Models in Medicine,” Nature Medicine Vol. 29, No. 8, pp. 1930–1940, 2023.

[13] B. B. Arcila, “Is It a Platform? Is It a Search Engine? It’s ChatGPT! The European Liability Regime for Large Language Models,” J. Free Speech L. 3, 455, 2023.

[14] J. Yang, H. Jin, R. Tang, X. Han, Q. Feng, H. Jiang, B. Yin, and X. Hu, “Harnessing The Power of LLMs in Practice: A Survey on ChatGPT and Beyond,” 2023, arXiv preprint arXiv:2304.13712.

[15] M. K. Nam and N. K. Kim, “Sentence-Based Extraction

Methodology from External References to Enhance Performance in RAG," Journal of The Korea Society of Computer and Information Vol. 29, No. 12, pp. 29-39, December 2024.

[16] A. Li, X. Hu, L. Wen, and P. Yu, "A Comprehensive Evaluation of ChatGPT's Zero-Shot Text-to-SQL Capability," arXiv preprint arXiv:2303.13547, 2023.

[17] J. H. Kim, "Guidelines for the Adoption and Evaluation of Database Management Systems (DBMS)," Database world, no. 9 - no. 16, pp. 10-15, 1994.

[18] K. S. Kim, "Microsoft Visual Basic Programming," Samyang, 2001.

[19] DB-Engines, DB-Engines Ranking, https://db-engines.com/

[20] E. K. Hong, Database Academy based MS SQL Server, Saengneung, 2017. 2. 17.

[21] About SQLite, SQLite, https://www.sqlite.org/

[22] X. Liu, S. Shen, B. Li, P. Ma, R. Jiang, Y. Zhang, J. Fan, and G. Li, "A Survey of Text-to-SQL in the Era of LLMs: Where Are We, and Where Are We Going?," Journal of LATEX CLASS FILES, vol. 18, no.9, 2020.

[23] N. Rajkumar, R. Li, and D. Bahdanau, "Evaluating the Text-to-SQL Capabilities of Large Language Models," CoRR, 2022.

[24] F. Li and H. V. Jagadish, "Nalir: an Interactive Natural Language Interface for Querying Relational Databases," in ACM SIGMOD, 2014.

[25] T. Yu, C. S. Wu, X. V. Lin, B. Wang, Y. C. Tan, X. Yang, D. Radev, R. Socher, and C. Xiong, "Grappa: Grammar-Augmented Pre-Training for Table Semantic Parsing," in ICLR, 2021.

[26] B. Li, Y. Luo, C. Chai, G. Li, and N. Tang, "The Dawn of Natural Language to SQL: Are We Fully Ready?" Proc. VLDB Endow, 2024.

[27] Z. Gu, J. Fan, N. Tang, S. Zhang, Y. Zhang, Z. Chen, L. Cao, G. Li, S. Madden, and X. Du, "Interleaving Pre-Trained Language Models and Large Language Models for Zero-Shot NL2SQL Generation." arXiv:2306.08891, 2023.

[28] D. Gao, H. Wang, Y. Li, X. Sun, Y. Qian, B. Ding, and J. Zhou, "Text-to-SQL Empowered by Large Language Models: A Benchmark Evaluation," Proc. VLDB Endow, 2024.

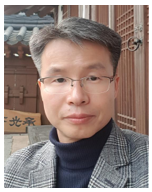
[29] H. Li, J. Zhang, H. Liu, J. Fan, X. Zhang, J. Zhu, R. Wei, H. Pan, C. Li, and H. Chen, "Codes: Towards Building Opensource Language Models for Text-to-SQL," SIGMOD, 2024.

[30] T. Yu, R. Zhang, K. Yang, M. Yasunaga, D. Wang, Z. Li, J. Ma, and I. Li, "Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task," Department of Computer Science, Yale University, arXiv:1809.08887, 2018.

[31] ProjectPro, FAISS Vector Database: A High-Performance AI Similarity Search, https://www.projectpro.io/article/faiss-vector-database/1009.

[32] https://platform.openai.com/docs/examples/default-sql-translate.

Authors



Myoungkuk Nam received the B.S. degree in Information Engineering from Korea Military Academy in 1997, M.S. degree in Computer Engineering from Korea National Defense University in 2009, and currently enrolled in Graduate School of Business IT, Kookmin University. He is interested in natural language processing, text mining, and RAG.

Namgyu Kim received the B.S. in Computer Engineering from Seoul National University in 1998, M.S. and Ph.D. degrees in Management Engineering from KAIST, Korea, in 2000 and 2007, respectively. Dr. Kim joined the faculty of the School of Management Information Systems at Kookmin University, Seoul, Korea, in 2007. He served as the Dean of the Graduate School of Business IT at Kookmin University and is currently a professor at the Business IT. He is interested in deep learning, text mining, and data modeling.