NL-to-SQL: Einsatzmöglichkeiten, Grenzen und Verbesserungspotential

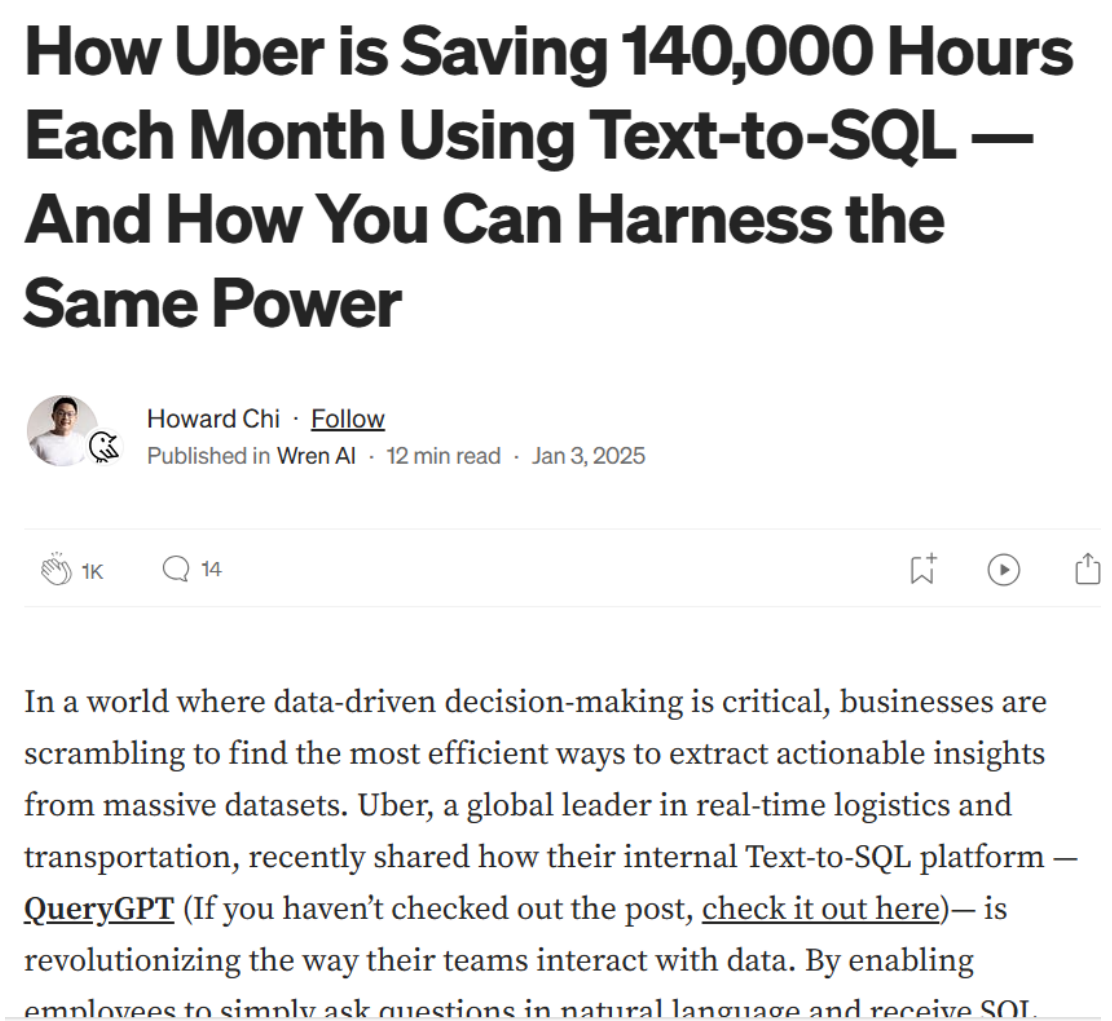
Database Synergy Day 2025 – Swiss Oracle & PostgreSQL User Groups United Mi. 26. März 2025, Die Mobiliar, Bern

Prof. Stefan Keller

Institut für Software, FH OST Campus Rapperswil, ost.ch/ifs |

Medienmitteilung

- Uber QueryGPT:
- Produktiver Einsatz zur Datenabfrage bei Uber.
- Ermöglicht Mitarbeitern ohne tiefgehende SQL-Kenntnisse komplexe Abfragen.
- Kommerziell
- "Uber has cut query authoring time by 70%. Considering they run about 1.2 million queries per month, this translates into an astonishing 140,000 hours saved monthly."
- (Quellen: Uber Blog und Medium)



How Uber is Saving 140,000 Hours Each Month Using Text-to-SQL — And How You Can Harness the Same Power

Howard Chi · Follow
Published in Wren AI · 12 min read · Jan 3, 2025

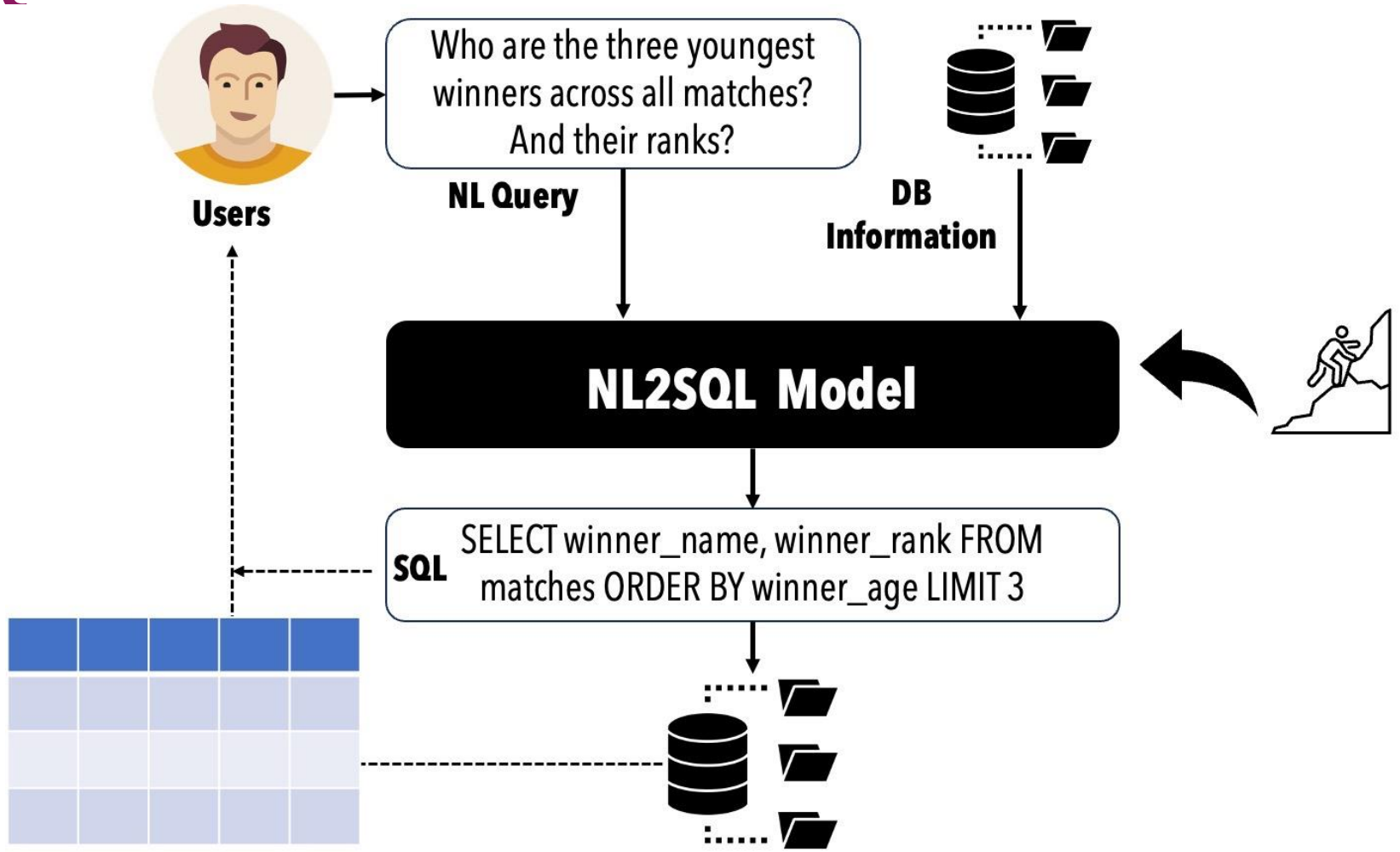
1K 14

In a world where data-driven decision-making is critical, businesses are scrambling to find the most efficient ways to extract actionable insights from massive datasets. Uber, a global leader in real-time logistics and transportation, recently shared how their internal Text-to-SQL platform — **QueryGPT** (If you haven't checked out the post, check it out here)— is revolutionizing the way their teams interact with data. By enabling employees to simply ask questions in natural language and receive SQL

1. Was ist NL-to-SQL? Definition und Relevanz.

- NL-to-SQL revolutioniert den Datenbankzugriff, indem es natürliche Sprache in SQL-Abfragen übersetzt.
  - Dies führt zu einer breiteren Nutzung der Daten.
  - NL-to-SQL ist spezifischer als Text-to-SQL
- Vorteile der Technologie:
  - Einfacher Zugang: Sie erweitert für Anwender ohne SQL-Kenntnisse die Möglichkeiten, komplexe Datenbankabfragen zu erstellen.
  - Effizienzsteigerung: Für versierte Anwender bietet sie eine Zeitersparnis, da sie die manuelle Erstellung komplexer SQL-Abfragen unterstützt und das Nachschlagen von Tabellen- und Attributnamen erleichtert.
  - Unterstützung der Lehre und Weiterbildung: Sie erleichtert das Erlernen von SQL in der Aus- und Weiterbildung.

NL-to-SQL

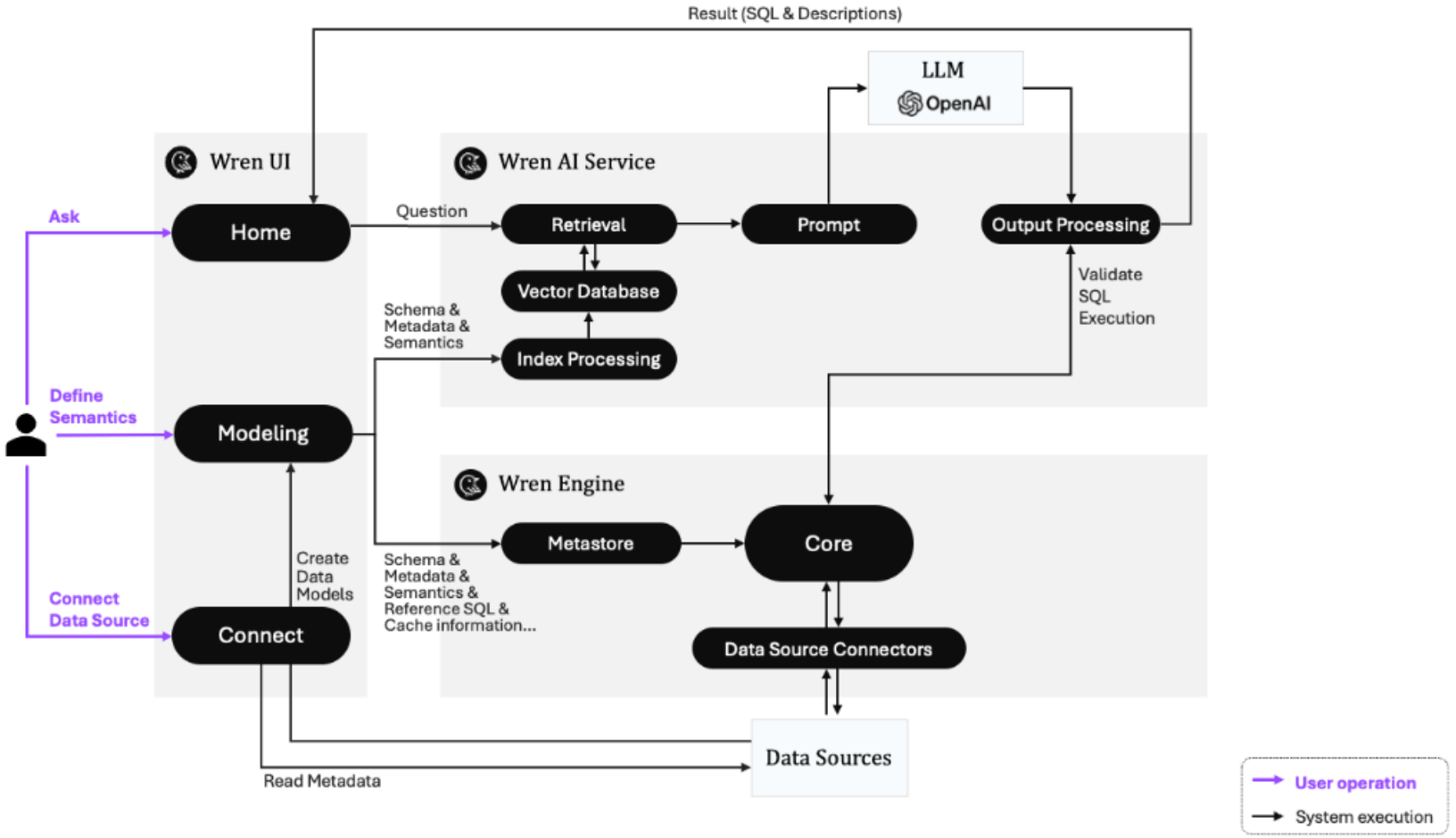


Source: NL2SQL Handbook https://github.com/HKUSTDial/NL2SQL_Handbook

2. Stand der Technik (Auswahl)

- "A Survey of NL2SQL with Large Language Models: Where are we, and where are we going?"
  - Umfassende Studie (2024) mit Überblick über NL2SQL-Techniken. Deckt Aspekte wie Modellarchitekturen, Datensätze, Evaluierungsmethoden und Fehleranalysen ab. URL https://arxiv.org/abs/2408.05109 und das "NL2SQL Handbook" https://github.com/HKUSTDial/NL2SQL_Handbook
- Weitere Quellen:
  - "Benchmarking Large Language Models for NL-to-SQL: A Comprehensive Evaluation of Accuracy, Cost, and Throughput" (2024). Studie, die die Leistung grosser Sprachmodelle in NL-to-SQL-Aufgaben untersucht und wertvolle Einblicke bietet in Genauigkeit, Kosten und Durchsatz in realen Anwendungen. URL https://www.techrxiv.org/users/848214/articles/1237086-benchmarking-large-language-models-for-nl-to-sql-a-comprehensive-evaluation-of-accuracy-cost-and-throughput
  - "Deep-Learning-basierte Modelle zur Steigerung der Genauigkeit und Robustheit". Hong et al. (2024). Modellgenerierung, generalisierte Performanz. URL https://arxiv.org/pdf/2406.08426
  - "BASE-SQL: A powerful open source Text-To-SQL baseline approach" (2025). Arbeit zu einem leistungsstarken Open-Source-Ansatz für Text-to-SQL, der eine Pipeline-basierte Methode zur Verbesserung der Genauigkeit bei der SQL-Generierung nutzt. URL https://arxiv.org/abs/2502.10739

RAG mit WREN.ai



Source: https://docs.getwren.ai/oss/overview/how_wrenai_works

3. Endnutzer-orientierte Tools

… vs. 4. Entwickler-orientierte Frameworks (siehe später)

- Zielgruppe: Benutzer ohne SQL-Kenntnisse oder mit Basiswissen.
- Merkmale: Einfachheit, Benutzerfreundlichkeit, schnelle Ergebnisse.
- Beispiele:
  - DataLine: Intuitive Oberfläche, minimaler Lernaufwand.
  - SQL Chat: Konversationsbasierte Datenbankabfragen, unterstützt PostgreSQL.
  - Vanna.AI: KI-assistierte Abfragen mit geringer Einstiegshürde, per default auf Basis ChatGPT, Open Source und PostgreSQL-kompatibel.
  - DBeaver: Open Source Datenbankverwaltung mit PostgreSQL-Unterstützung und erweiterter NL-Unterstützung (Community Version verlangt nachinstallation einer NL-Erweiterung).
  - Database.built (→ nächste Folie)

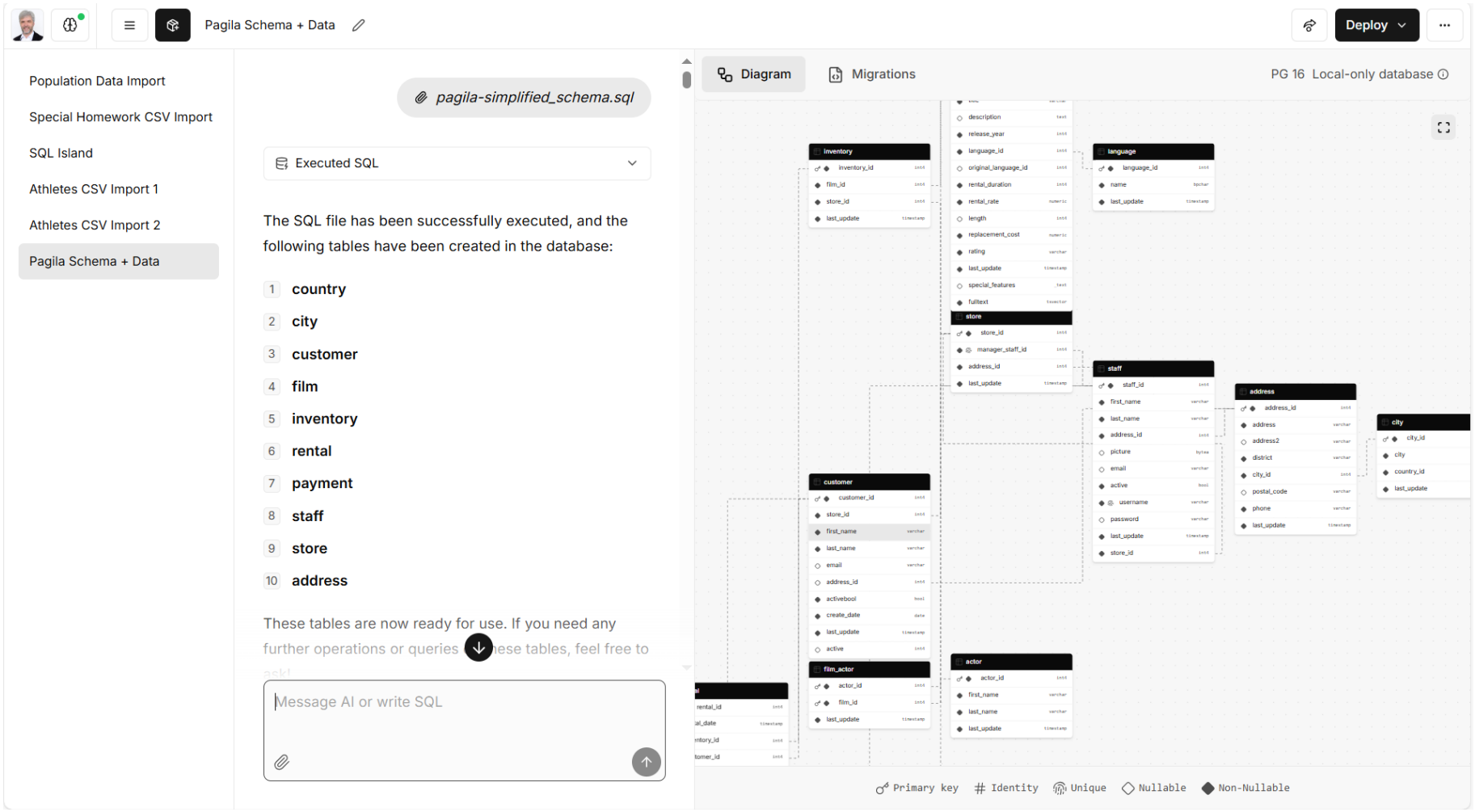
Database.built – Überblick

- Experimentelle Plattform von Supabase, https://database.built
  - PostgreSQL läuft lokal direkt im Browser mittels PGlite (WebAssembly, kein Server nötig)
  - Einloggen mittels GitHub, um Missbrauch zu reduzieren
- Erstellung und Verwaltung von Datenbanken im Browser
  - Lokale Speicherung aller Einstellungen und API-Schlüssel; keine Daten verlassen den Browser
- KI-Unterstützung
  - beim Aufbau der Datenbanken, bei der Abfrage und bei der Erstellung von Businessgrafiken
- Einfache Datenintegration
  - Drag-and-Drop von CSV-Dateien; automatische Erkennung der Tabellenstruktur
- Integration eigener LLMs
  - "Bring your own LLM": OpenAI-kompatible Services ("OpenAI Tools API") mit eigenem Superprompt

Database.built Superprompt (original gekürzt)

```
You are a helpful database assistant. Under the hood you have access to an in-browser Postgres database called Pglite.

The following extensions are available: plpgsql , vector.

When generating tables, do the following:
- For primary keys, always use "id bigint primary key generated always as identity" (not serial)
- Prefer 'text' over 'varchar'
- Keep explanations brief but helpful
- Don't repeat yourself after creating the table

When creating sample data:
- Make the data realistic, including joined data
- Check for existing records/conflicts in the table

When querying data, limit to 5 by default. The maximum number of rows you're allowed to fetch is 100 (to protect AI from token abuse). If the
user needs to fetch more than 100 rows at once, they can export the query as a CSV or SQL.

Output as table.

When importing CSVs try to solve the problem yourself (eg. use a generic text column, then refine) vs. asking the user to change the CSV. No need
to select rows after importing.

When performing FTS, always use 'simple' (languages aren't available).

No images are allowed. Do not try to generate or link images, including base64 data URLs.

Feel free to suggest corrections for suspected typos.
```

Database.built Screenshot mit Pagila-Daten



4. Entwickler-orientierte Frameworks

- Zielgruppe: Entwickler und Data Engineers
- Merkmale: Hohe Genauigkeit, Anpassbarkeit und Integration in Entwicklungsprozesse.
- Beispiele von Open Source-Projekten:
  - Vanna: Hybridmodell mit hoher Anpassbarkeit, Open Source und PostgreSQL-Unterstützung.
  - Wren AI: Umfangreichere Lösung mit Fokus auf Integration in Produktivsysteme, Open Source und PostgreSQL-Unterstützung.
  - SQLCoder: Open-Source-Framework mit Fokus auf Präzision, Erweiterbarkeit und PostgreSQL-Unterstützung.
  - DSPy: Framework für die systematische Entwicklung und Optimierung von NL-to-SQL-Modellen, Open Source.

4. Entwickler-orientierte Frameworks (Forts.)

Weitere Beispiele akademischer (Open-Source-) Projekte:

- XiYan-SQL – Ein im Februar 2025 veröffentlichtes Open-Source-Projekt, das ein Multi-Generator-Ensemble-Framework für Text-to-SQL bietet. Es umfasst verschiedene Modellgrößen und zielt darauf ab, die Genauigkeit und Effizienz der SQL-Generierung zu verbessern. URL GitHub https://github.com/XGenerationLab/XiYan-SQL
- CodeS – Ein im Februar 2024 vorgestelltes Open-Source-Projekt, das Sprachmodelle mit 1 bis 15 Milliarden Parametern speziell für Text-to-SQL-Aufgaben entwickelt hat. CodeS erreicht eine hohe Genauigkeit bei der SQL-Generierung und bietet robuste Leistungen auf verschiedenen Benchmarks. URL arXiv https://arxiv.org/abs/2402.16347
- Awesome-Text2SQL: Ein kontinuierlich aktualisiertes Repository, das Tutorials und Ressourcen für Text-to-SQL, einschließlich Feinabstimmungstechniken und Datensätzen, bereitstellt. URL GitHub https://github.com/topics/nl2sql

Bachelor Thesis Fiona Pichler + Benjamin Kern (1)

- "Text-to-SQL für DataGovernance Technologies (DGT) und die Aus- + Weiterbildung", 2024
- Es wurden vier Ansätze für die Integration von Schemainformationen in LLMs ermittelt:
  1. reines LLM als Basis,
  2. kontextbezogenes Lernen,
  3. Feinabstimmung des LLM und
  4. RAG (Retrieval Augmented Generation).
- Feinabstimmung (3) wurde aufgrund einer unzureichenden Menge an Trainingsdaten ausgeschlossen.
- Die drei Ansätze wurden in Python mit LLM-APIs implementiert.
- Die ausgewählten LLMs waren Mistral, phi3:3.8b, phi3:14b, lama3.1:8b und GPT-4o-mini.
- Zwei Datensätze: 1. DGT (MS SQL Server), und 2. der Pagila-Datensatz (PostgreSQL).
- Die SQL-Testabfragen und die entsprechenden Prompts wurden aus einer Datenbankvorlesung extrahiert und andere wurden mit ChatGPT generiert.

Bachelor Thesis by Fiona Pichler + Benjamin Kern (2)

- Evaluationskriterien:
  - 1. similarity (how similar is an output compared to the example solution?),
  - 2. validity (is the output valid SQL?),
  - 3. executability (can the output be executed and are the generated column names correct?),
  - 4. reliability (how similar is the output to the same user prompt?).
- Bemerkung:
  - executability (Ausführbarkeit) schliesst zwar die Gültigkeit einer Abfrage mit ein. Gültigkeit ist aber wichtig v.a. wegen reinem reinen LLM-Ansatz, der Tabellennamen "erfindet".
- Challenges:
  - Es war eine Herausforderung, die LLMs dazu zu bringen, ohne zusätzliche Erklärung gültiges SQL auszugeben. Sie erraten Spalten- und Tabellennamen.
  - Es erwies sich als schwierig, die Ähnlichkeit mit der üblichen Kosinusfunktion zu bewerten, da dasselbe SQL-Resultset mit unterschiedlichen SQL-Abfragen erzielt werden kann.

Bachelor Thesis by Fiona Pichler + Benjamin Kern (3)

- Die Abbildung zeigt das Ergebnis für RAG bei der Ausführung der SQL-Abfragen.
  - Von den komplexeren Testfällen konnten durchschnittlich 45 % ausgeführt werden, wobei nur 70% der angeforderten Spalten extrahiert wurden.
  - Halluzinationen konnten nicht vollständig beseitigt werden, was zu der geringen Anzahl ausführbarer SQL-Abfragen führte.
- Erkenntnisse
  - Modell phi3:3.8b hatte hohe Anzahl von "Halluzinationen". Modell llama3.2 hatte das grösste Potenzial.
- Mögliche Weiterentwicklungen
  - Beseitigung der Halluzinationen
  - Bereitstellung verbesserter Testmetriken
  - Strukturierte Ausgabe und agenten-ähnliche Funktionsaufrufe.
  - Neue Open Source-Modelle z.B. QwQ 32B

LLM	columns correctly identified (%)				queries executable on database (%)			
	basic testcases		advanced testcases		basic testcases		advanced testcases	
	avg	max	avg	max	avg	max	avg	max
Mistral-7b-v.01	0.66	0.73	0.51	0.88	0.25	0.30	0.06	0.09
phi3:3.8	0.56	0.78	0.43	0.7	0.40	0.45	0.08	0.09
llama3.2	0.68	0.68	0.45	0.61	0.62	0.64	0.30	0.38
phi3:14	0.62	0.69	0.42	0.5	0.17	0.21	0.12	0.19
gpt-4o-mini:	0.70	0.73	0.50	0.55	0.62	0.66	0.45	0.55

Experiment "nl2sql" am Institut IFS FH OST

- Hauptuntermerkmal: Echtzeit-Abfragegenerierung von PostgreSQL oder DuckDB
  1. Der Benutzer stellt eine Frage in natürlicher Sprache
  2. Das System verarbeitet die Frage mithilfe von LLM, um SQL zu generieren
  3. SQL wird validiert und anhand der Datenbank ausgeführt
  4. Alle Artefakte werden zur späteren Verwendung gespeichert
- Verwendete Python-Pakete
  - Kern: SQLAlchemy, Pandas, NumPy
  - LLM: OpenAI, LLMHub (qwq:32b)
  - Infrastruktur: Pydantic, Loguru (Logging)
- Was die Lösung ausmacht
  - Kontextbewusst: Verwendet Schemaverständnis
  - Robust: Threadsichere Verbindungen mit automatischer Wiederholung
  - LLM-Services: OpenAI API (mit Plus-Konto), LLMHub mit QwQ (langsam!)

5. Grenzen aktueller NL-to-SQL Technologien

- Schwierigkeiten bei komplexen und mehrdeutigen Anfragen
- Probleme bei kontextabhängigen Abfragen
- Probleme bei der Erkennung des Datenbankschemas insbesondere der Constraints
- Datenschutz und Sicherheitsaspekte

5. Grenzen aktueller NL-to-SQL Technologien

- Video von Tatiana Krupenya, CEO DBeaver ("Where is the Place of AI in SQL Querying?")

- Offensichtliche Limitationen von LLMs
  - "Context" (→ Schema- und Daten-Kontext)

- Weniger offensichtliche Limitationen von LLMs
  - Grosse Schemata mit Hunderten und Tausenden von Tabellen
  - Lange Namen für Tabellen und Spalten
  - Analytische Datenbanken
  - Neue Datenbanken mit fehlender Dokumentation
  - NoSQL-Datenbanken

6. Verbesserungspotential und zukünftige Trends

- Schnellere und bessere Large Language Models (LLMs).
- Weiterentwicklung der Kontextsensitivität und Robustheit.
- Bessere Nutzung des OpenAI "Tools API"
- Nutzung des neuen Model Context Protocols (MCP).
  - Mit dem MCP können Large Language Models (LLMs) externe Datenquellen und Werkzeuge über ein standardisiertes Client-Server-Modell nutzen, ähnlich wie bei Remote Procedure Calls (RPCs).
  - Die LLM-Anwendung (z.B. Claude Desktop) fungiert als Client, der über das MCP entfernte Funktionen oder Prozeduren auf einem Server aufruft, um von dort Daten abzurufen oder dort Funktionen auszuführen. Dies ermöglicht eine bessere Integration zwischen der Anwendung und externen Ressourcen. Siehe z.B. pg-mcp.

7. Fazit und Ausblick

- Ich experimentiere noch mit einem Vorlesungs-Chatbot Ethel der ETHZ

- Fazit
  - Kontinuierliche Weiterentwicklung und Forschung ist nötig
  - Noch Potenzial für erweiterte Anwendungsszenarien und bessere Benutzererfahrungen
  - Bin noch unsicher, wie NL-to-SQL genau in den Unterricht einbauen für Einsteiger und Fortgeschrittene

- NL-to-SQL ist ein vielversprechender Schritt zur Demokratisierung des Datenzugriffs

- NL-to-SQL wird den "Menschen" nicht ersetzen aber (einzelne) produktiver machen

Kontakt: Stefan Keller, stefan.keller@ost.ch, ost.ch/ifs

Fragen und Diskussion

nl2sql-Experiment: Datenfluss-Diagramm