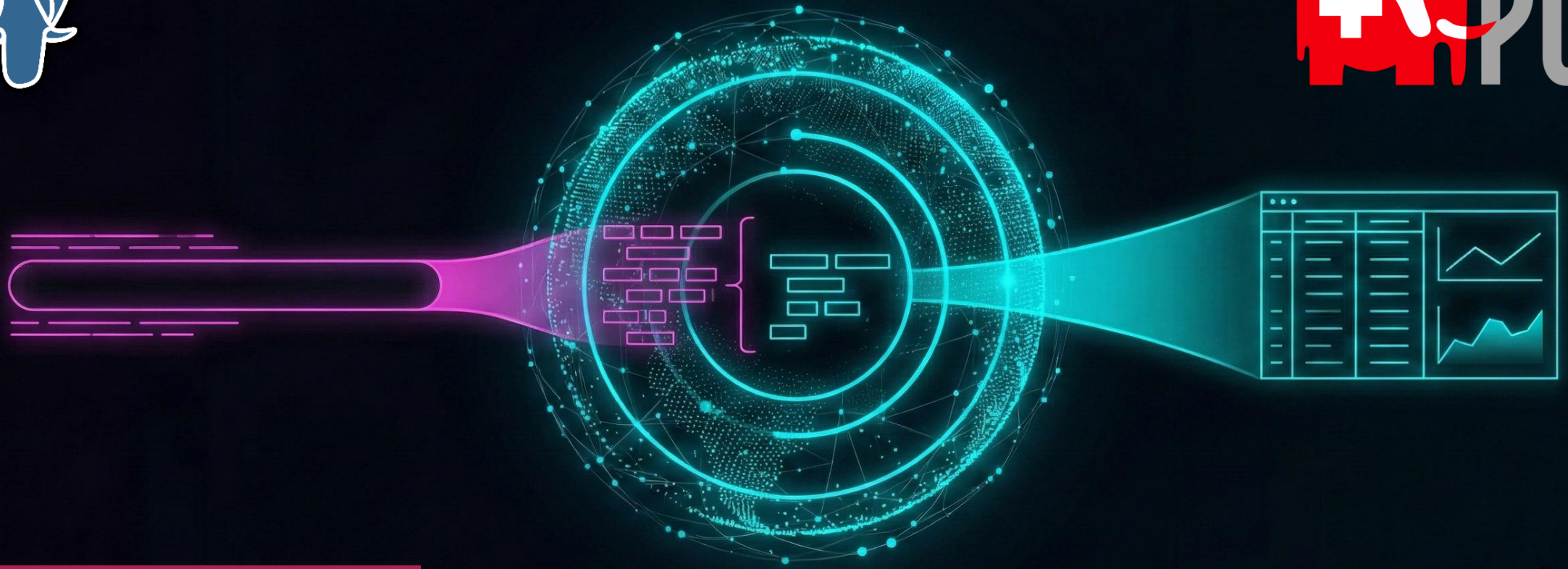


Stefan Keller & Lars Herrmann

Freitext zu SQL: psql-chat als natürlich-sprachliche SELECT-Erweiterung

2. Dezember 2025
Ein SwissPUG-Event
Red Hat Switzerland Zürich

Ausgangslage

- Freitext zu SQL: psql-chat als natürlich-sprachliche SELECT-Erweiterung
- Für viele Nutzer – insbesondere Business-Nutzer und Einsteiger – ist das Erlernen von Datenanalysen und Datenbankabfragen mit SQL eine Hürde.
- GenAI verspricht eine bessere Zugänglichkeit und eine Demokratisierung.
- GenAI verspricht in allen Informatikberufen – auch Datenbank-Experten und -Administratoren – eine Produktivitätssteigerung.

> «KI hat das Potenzial, bis 70 % der heutigen Arbeitszeit durch Automatisierung effizienter zu gestalten» (Quelle: McKinsey-Report 2023 "Die KI-Revolution…").

- Achtung: KI != GenAI, GenAI ist ein Teilgebiet von KI

NLQ mit KI-Chatbots: Zwei Recherchen zur Güte

- Zwei Hilfen: Consensus.app (KI-Suchmaschine für wiss. Forschung) und ChatGPT
- Frage 1: Wie gut sind "Natural Language Queries" (NLQ) und "Natural Language to SQL" (NL-to-SQL)? D.h. Natürliche Sprache in SQL übersetzen (wobei es mehr Fragekategorien gibt)
- Antworten:
  - Gut bei Standard-Benchmarks,
  - wobei die Leistung je nach Komplexität der Abfrage und Domäne variiert
- Wichtige Faktoren, die die Leistung beeinflussen:
  - Einfache Abfragen: Hohe Genauigkeit für einfache SELECT-Abfragen mit einer Tabelle.
  - Komplexe Abfragen: Sinkende Genauigkeit bei verschachtelten, mehrere Tabellen umfassenden oder sehr domänenspezifischen Abfragen. Aktuelle Modelle erreichen je nach Datensatz etwa 60 bis 80 %.
  - Domänenanpassung: Schema-"bewusste" und Fine-Tuning-Techniken verbessern die Ergebnisse erheblich, insbesondere in spezialisierten Domänen oder (Query-)Sprachen mit geringen Ressourcen.

NLQ mit KI-Chatbots: Frage 2 – Thesen

Suche nach wissenschaftlichen Studien zu den folgenden Thesen (Behauptungen) über KI-Chatbots, die direkt in Datenbanksysteme integriert sind und Nutzern ermöglichen, Fragen in natürlicher Sprache an die verbundene Datenbank zu stellen (NLQ, NL-to-SQL):

- These 1: Nutzer ohne tiefgreifende SQL-Kenntnisse können mit KI-Chatbots Datenanalysen einfacher und schneller durchführen. Das fördert datengestützte Entscheidungen und führt zu Effizienzsteigerungen in Unternehmen.
- These 2: Experten können mit KI-Chatbots SQL-Abfragen schneller erstellen und verbessern die Datenbank-Administrierung. Das führt zu Effizienzsteigerungen in Unternehmen.
- These 3: Studierende kommen mit KI-Chatbots zu besseren Lernergebnissen beim Lösen von SQL-Aufgaben. Das führt zu besserer und zeitgemässer Lehre.

Gib eine Zusammenfassung der relevanten Studien, die diese Thesen entweder belegen oder widerlegen.

NLQ mit KI-Chatbots: Frage 2 – Antworten

- Laien & Nicht-SQL-Experten
  - Studien zeigen höhere Aufgabenlösung, schnellere Bearbeitung und geringere kognitive Belastung.
  - Betriebswirtschaftliche Effekte sind noch wenig quantifiziert.
- SQL-Expert:innen & Analyst:innen
  - Neuere Studien (< 3 Jahre) zeigen Verbesserungen in Bearbeitungszeit und Genauigkeit auch bei erfahrenen Nutzern (Quelle: SQL-LLM vs. Snowflake 2025).
  - Für klassische Datenbank-Administration (DBA-Aufgaben) liegen bislang nahezu keine Studien vor.
- Universitäre Lehre
  - Chatbot-gestützte SQL-Lernplattformen korrelieren mit besseren Prüfungsleistungen (Quellen: Balderas et al. 2024 "Chatbot-Based Learning Platform...”; Pérez-Mercado et al. 2023 "ChatbotSQL…").
  - Interaktive Erklär- und Feedbackmechanismen fördern selbstständiges Üben und reduzieren kognitive Belastung. (Quellen: plus Tian et al. 2024 "SQLucid…"; Sirbu et al. 2025 "Explanation Provision…")

Weitere Thesen und Evidenzen

- Schlechte Performance, d.h. hohe Latenz:
  - Für Codevervollständigung kaum geeignet, abgebrochene Antworten und limitierte Kontextlängen.
  - Ressourcen- und Energiebedarf pro Anfrage ist hoch.
- Halluzinationen/Konfabulationen:
  - LLM-Chatbots können komplexe Aufgaben lösen, erzeugen aber grundlegende sachliche und logische Fehler: falsche Fakten (falsche DB-Dokumentation!), unpassende Begriffe oder frei erfundene Quellen.
  - Bei NLQ sind Abfragen oft syntaktisch korrekt, aber semantisch falsch (falsche Joins, Filter, Aggregationen).
- Mangelnde Transparenz und Erklärbarkeit:
  - LLMs sind weitgehend Black Box und schwer interpretierbar (Nicht-Erklärbarkeit, Nicht-Nachvollziehbarkeit).
  - Von LLMs generierte "Erklärungen" können selbst wieder halluzinieren und sind daher nicht zuverlässig.

Stand der Technik: Oracle "SELECT AI"

- Ein "Feature" der Oracle Autonomous AI Database (Oracles Cloud-Datenbank)
- "Irgendwie" auch on-premise in Oracle Database seit Version 23ai (September 2023)
- Beispiele [1]:
  - Natural-Language-to-SQL:
    select ai how many customers in San Francisco are married
  - Chat-Interaktionen / SQL-Erklärungen:
    select ai explainsql how many customers in San Francisco are married
  - Generierung synthetischer Daten:
    BEGIN
    DBMS_CLOUD_AI.GENERATE_SYNTHETIC_DATA(profile_name=>'GENAI',object_name=>'MOVIE',owner_name=>'ADB_USER',record_count=>10);
    END;

[1] "Examples of Using Select AI"
https://docs.public.oneportal.content.oci.oraclecloud.com/iaas/autonomous-database-serverless/doc/select-ai-examples.html

Stand der Technik: PostgreSQL-bezogene Projekte

Name/Beschreibung	Abhängigkeiten	Lizenz	Bemerkungen	Beispiele (Abfragen)	Kurzbewertung
pg_ai_query – PostgreSQL-Erweiterung in C++.	PostgreSQL 17 empfohlen, CMake/Make, C++20-Compiler, OpenSSL; API-Keys von OpenAI- und Anthropic (GPT/Claude)..	Apache-2.0. Aktiv gepflegtes Projekt.	In-DB-Extension, reine SQL-API (CREATE EXTENSION pg_ai_query;), Konfiguration über Datei/Env-Variablen; „AI-Assistent fürs Schreiben und Tuning von SQL“ direkt in PG.	SELECT generate_query('list customers who have not placed an order in the last 90 days’); SELECT explain_query('SELECT * FROM orders WHERE ...’);	Attraktiver Ansatz für Nutzer mit PG-Extension-Know-how, aktive Entwicklung, klare Lizenz. Nachteile: Bindung an OpenAI/Anthropic; nur PG 17?
ai_toolkit – PostgreSQL-Erweiterung in C++ auf Basis ClickHouse-AI-Backend.	PostgreSQL 18, CMake, C++20-Compiler, ClickHouse SDK; API-Key von LLMs über ClickHouse AI SDK (OpenAI, Anthropic, OpenRouter etc..	Keine Lizenz angegeben. Sehr junges Projekt.	Verwaltet einen kleinen, eigenen, in SQL speicherbaren Prompt-Append mit Key-Value-Struktur.	SELECT ai_toolkit.query('list all customers in Berlin with orders > 1000 EUR’); SELECT ai_toolkit.explain_query('SELECT * FROM orders WHERE ...’); SELECT ai_toolkit.explain_error('ERROR: column customer_id does not exist’);	Technisch spannend, ClickHouse-AI-Abhängigkeit. Nachteile: keine Lizenzangabe; keine Releases.
psqlomni – Python-CLI ähnlich wie psql.	PostgreSQL 18, Python 3.9+, LangChain, OpenAI API, psycopg/DB-Treiber, API-Key via LangChain von OpenAI-Modellen (GPT).	Keine Lizenz angegeben. Seit 2024 eher ruhendes Projekt.	Reines Client-Tool (keine Server-Extension); generiert SQL, fragt vor Ausführung nach Bestätigung, führt aus und zeigt Ergebnisse. Kennt auch System Settings.	PSQLOMNI > show the columns of profile table > How many customers signed up last week? (im psqlomni-Prompt; das Tool erzeugt und führt die entsprechende SELECT-Query aus).	Niederschwelliger Einstieg in NL-to-SQL ohne Eingriff in den DB-Server. Nachteile: keine Lizenzangabe; OpenAI-Only; keine Releases und ausbleibende Weiterentwicklung.

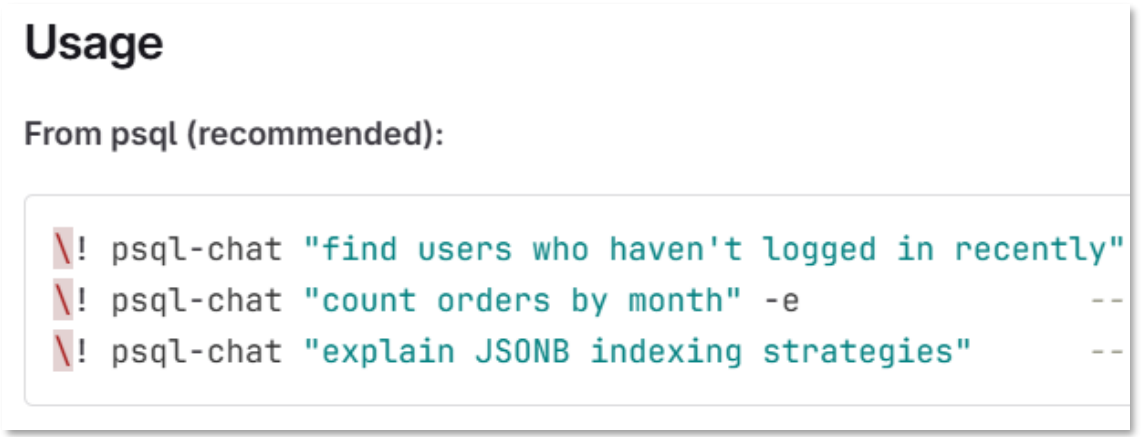
psql-chat – Ein Proof of Concept (PoC)

- Idee: Nutzer können Datenbank-Fragen in natürlicher Sprache (Text) stellen.
  - Erstelle eine SQL-Abfrage (NL-to-SQL) aus Text (v.a. Deutsch und englisch).
  - Erkläre oder korrigiere eine SQL-Abfrage (SQL-to-NL).
  - Erstelle Datenbank-Administrations-Befehle (NLQ).
  - Fokus nicht auf Datenbank-Optimierung.

- Implementiere einen KI-Chatbot …
  - direkt in psql, d.h. in PostgreSQL integriert (damit sind ein Teil des Kontexts und Autorisierung klar).
  - mit Berücksichtigung der korrekten, aktuellen PostgreSQL-Dokumentation.
  - in Python (viele GenAI-Bibliotheken sind in Python geschrieben).
  - mit Lizenz PostgreSQL License.

psql-chat: Ablauf

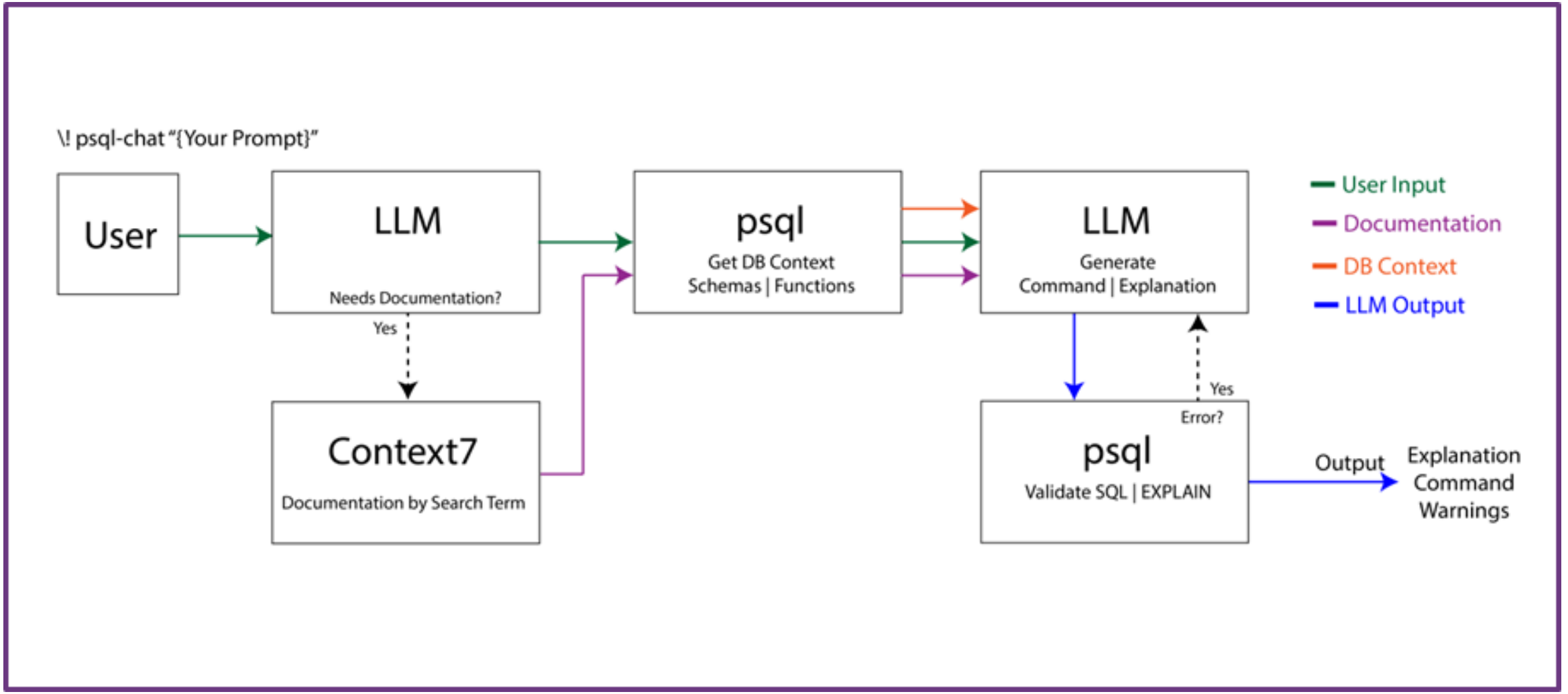
- Möglicher Nutzer-orientierter Ablauf mit psql und installiertem psql-chat:
  - Nutzer startet psql in der Command Shell und verbindet sich mit «mydb»
  - Nutzer gibt eine Datenbank-bezogene Frage ein in der Form:
    mydb=# \! psql-chat "{Your Prompt}"
  - Der Chatbot schlägt eine SQL-Abfrage oder einen psql-Befehl vor und wartet.
  - Nutzer kontrolliert die Abfrage und führt den Befehl aus – oder passt ihn an.

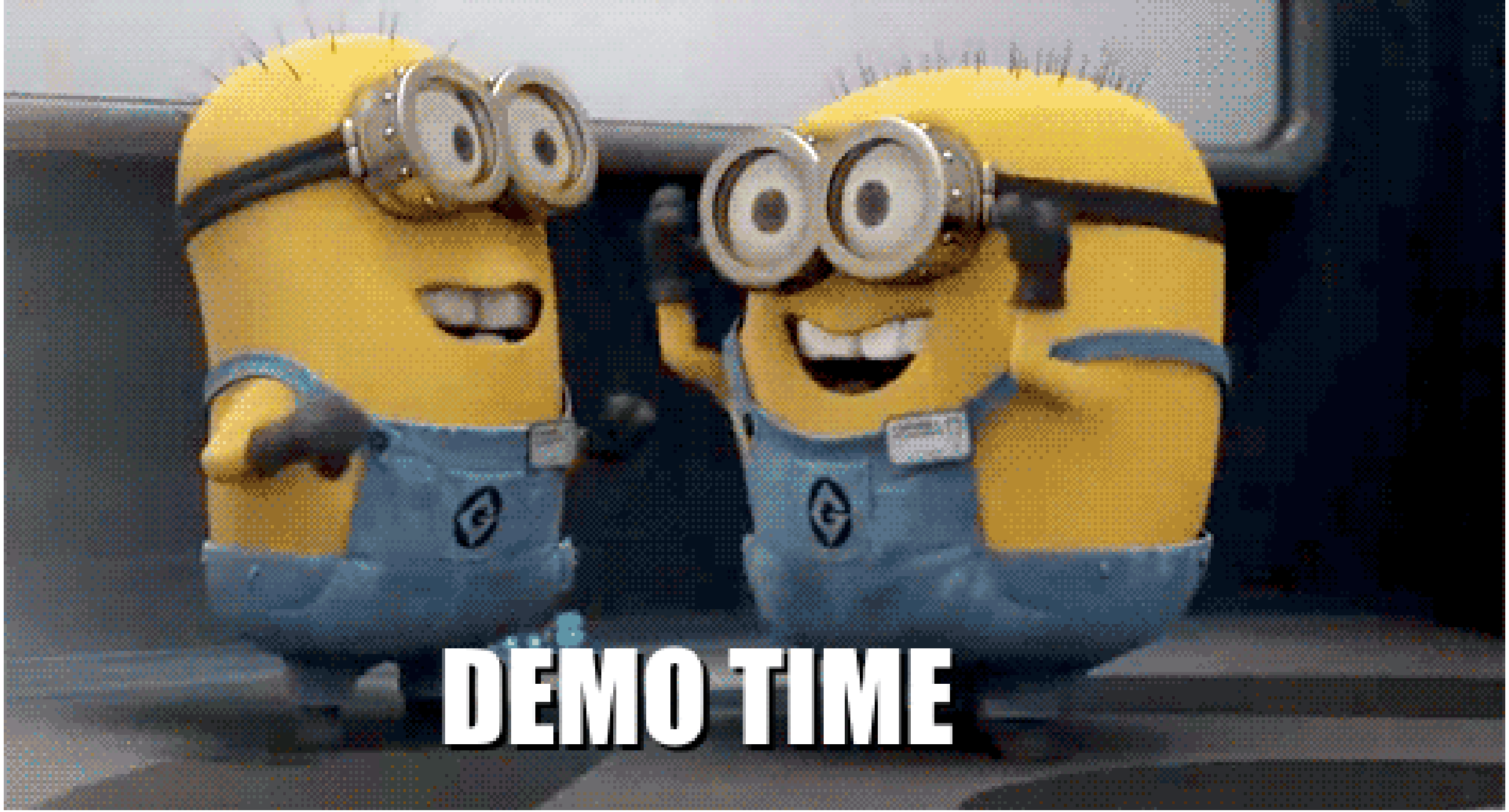


psql-chat: Merkmale

- Kontext:
  - psql-chat verwendet die Verbindungsinformationen der aktuell verbundenen Datenbank.
  - Jeder Abfrage wird dem LLM das Schema der Datenbank als Kontext (Metadaten) mitgegeben.
  - Zusätzlich wird vor dem LLM-API-Aufruf ermittelt, ob die PostgreSQL-Dokumentation benötigt wird. Per Suchbegriff über das Context7 API werden Textausschnitte geholt und ebenfalls als Kontext mitgegeben (Retrieval Augmented Retrieval RAG).
- SQL-Validierung:
  - Bevor dem Nutzer ein SQL-Statement zurückgegeben wird, wird dieses mittels «EXPLAIN <SQL command>» mit dem SQL-Parser validiert und bei Fehlern iterativ korrigiert
- Implementation:
  - psql-chat ist ein Python-Package; Abhängkeiten sind: openai, typer (cli), rich (Ausgabe Formatierung).
- Installation:
  - psql-chat lässt sich via pipx run oder uvx oder direkt aus PyPI ausführen.

psql-chat: Architektur




DEMO TIME

psql-chat: Wie weiter?

- Datenschutz:
  - Aktuell wird OpenAI mit GPT-4o verwendet; als Nächstes stellen wir auf LLMHub by IFS OST mit offenem LLM um.
  - Prüfenswert wäre ein komplett lokales, auf SQL fine-tuned Modell.
- Kontext:
  - Berücksichtigen der PostgreSQL-Version und vorhandenen Extensions für die versionsspezifische Abfrage der Dokumentationen.
  - Auch current_user, search_path und alle sichtbaren Schemas berücksichtigen.
- Transparenz:
  - Referenzen auf gefundenen Dokumentations-Stellen (PostgreSQL, PostGIS) angeben.

Problem generischer Code-Generierung

- Problem: Die KI schreibt Code inkl. SQL der zwar läuft, aber von schlechter Qualität und voller Anti-Patterns ist (TEXT statt VARCHAR, kein money, timestampz, Indizes auf FKs, BIGINT GENERATED ALWAYS AS IDENTITY).
- Grund: LLMs sind auf gemischten SQL-Daten trainiert, wodurch idiomatisches PG-Wissen untergeht.
- Warum Codequalität wichtig ist:
  - Schema-Entscheidungen sind teuer zu ändern. „fast richtig“ führt zu Performance- und Migrationsproblemen in der Produktion.
  - Studien zeigen: Entwickler:innen brauchen mit generischer KI teils länger wegen subtiler Fehler (METRA).
- Ziel: KI soll von Haus aus korrekter, performanter Code schreiben, ohne Prompt-Hacks.
- pg-aiguide - Open-Source-PostgreSQL Guide, der KI-Agenten Beurteilungen gibt (tigerdata.com):
  - SKILL.md (Claude.ai-spezifisch): Kuratierte Best Practices als Markdown-Text.
  - Semantische Suche über offizielle Dokumentation (version-aware).
  - Alles wird als MCP-Server exponiert, plus spezielles Claude-Code-Plugin.

Die Modularisierung und das Teilen von KI-Wissen

- Was ist SKILL.md?
  - Eine Anthropic-spezifische Konfigurationsdatei für «Claude Skills» (insbesondere Claude Code und Agenten).
  - Dient als modulares Wissenspaket oder «Bedienungsanleitung» für die KI.
- Steht für «Context-as-Code» (Anweisungen als Code verwalten) im Gegensatz zu den Cloud-basierten Konfigurationen (z.B. bei OpenAI GPTs), da das Wissen lokal im Projektordner gespeichert und versionierbar ist.
- Funktionsweise: Claude scannt Metadaten; passt die «description» zur Nutzeranfrage, wird der gesamte Skill in den Kontext geladen.
- Technisches Format: Einfache Markdown-Datei, bestehend aus YAML-Frontmatter (Metadaten «name», «description») und Markdown-Body (detaillierte Anweisungen und Beispiele).
- Fazit:
  - Das Konzept («Context-as-Code» – also das Ablegen von KI-Regeln in simplen Dateien direkt im Projektordner) ist gerade im Aufwind im Gegensatz zu «Regeln in der Cloud verstecken» (OpenAI).
  - Tipp: SKILL.md-Inhalt kann in .cursorrules Datei kopiert oder als System-Prompt bei OpenAI eingefügt werden.

Inspired by Ovomaltine 2015-2017:

«Mit KI kannst du's nicht besser – nur schneller»
Besser wirst du durch Verständnis von Daten, Prozessen und Domäne!

psql-chat: Lasst uns diskutieren und dran bleiben

- psql-chat Repository: https://gitlab.com/geometalab/psql-chat (feedback welcome)
- Kontakt: Stefan Keller, IFS, FH OST, ost.ch/ifs, stefan.keller@ost.ch
- Weitere Events:
  - 6. February 2026: CERN PGDay 2026. CERN, Meyrin (Switzerland). www.swisspug.org/cern-pgday-2026.html
  - 22./23. Januar 2026: Kurs PostGIS-Einführung (inkl. PostgreSQL) (2 Tage), FH OST Campus Rapperswil. giswiki.ch/Agenda
  - 25./26. Juni 2026: Swiss PGDay 2026 - die Schweizer PostgreSQL-Datenbank-Konferenz (en+de), FH OST Campus Rapperswil. pgday.ch
  - Siehe swisspug.ch
- Diese Folien: CC-BY-4.0