

Knowledge-Graph Scope Expansion for Text-to-SQL RAG over Czech Public-Health Tables

Peter Vajdečka¹

¹Faculty of Informatics and Statistics, Prague University of Economics and Business, nám. W. Churchilla 1938/4, Prague, Czech Republic

Abstract

This paper presents a reproducible demo for Czech public-health question answering over official infectious-disease aggregates. The system treats retrieval-augmented generation as controlled scope construction before Text-to-SQL summarization: user phrases denoting disease families, age groups, regions, or years must be resolved to executable table values before aggregation. We compare two Text-to-SQL RAG architectures: a *Flat* RAG baseline and a *KG-RAG* variant over the same 272,858-row DuckDB answer table enriched with MKN-10-CZ, the Czech ICD-10 classification. Flat RAG retrieves table-present values from embedded text; KG-RAG adds deterministic MKN-10-CZ hierarchy expansion that maps disease-family phrases to explicit diagnosis-code sets before SQL. On ten saved Czech demo queries, KG-RAG improves diagnosis micro-F1 from 0.545 to 0.935, macro-F1 from 0.567 to 0.921, exact diagnosis-scope matches from 1/10 to 8/10, and complete-filter accuracy from 1/10 to 6/10, while producing no false-positive diagnosis codes. The interface exposes retrieved evidence, selected entities, graph expansion, generated SQL, and returned tables, making both gains and residual failures auditable.

Keywords

retrieval-augmented generation, knowledge graphs, KG-RAG, semantic retrieval, text-to-SQL, public-health open data, tabular summarization

1. Problem and Contribution

Public-health question answering can fail even when the generated SQL is syntactically correct, because a broad Czech phrase must first be mapped to the complete set of database values it denotes. For example, “střevní infekční nemoci” (intestinal infectious diseases) denotes the MKN-10-CZ range A00–A09; the current fact table contains A01–A09. Only some of these diagnoses have labels that closely resemble the wording of the question, so flat retrieval can omit valid sibling codes or select similar diagnoses outside the intended family. We call the conversion of disease, age, region, and time expressions to executable table values *scope construction*. Our research question is: *when the candidate-selection and SQL layers are shared, does MKN-10-CZ hierarchy expansion produce a more complete and precise executable diagnosis scope for disease-family questions?* The contribution is therefore not a new generic RAG+KG architecture, but a controlled Czech public-health demo comparing Flat RAG with a KG-assisted scope-closure step. Both systems use the same DuckDB answer table, aggregate measure, SQL constraints, and output format. The reported measure is always reported `case_count` and is not interpreted as unique patients, incidence, prevalence, causal evidence, or clinical guidance.

2. Related Work

Retrieval-augmented generation improves grounding and updateability for knowledge-intensive tasks [1], but standard RAG does not guarantee that all implied members of a controlled-vocabulary scope are selected. Graph-based RAG and KG-RAG use explicit relations among entities, documents, or concepts for retrieval and answer construction [2, 3, 4, 5]. Our demo uses the graph in a constrained role: it closes a diagnosis scope before aggregate SQL execution. The system also relates to table question answering and text-to-SQL [6, 7], but the SQL shape here is deliberately constrained and inspectable;

Posters, Demos, Blue Sky, and Tutorials at SEMANTiCS 2026, Sep 2026, Ghent, Belgium

✉ peter.vajdecka@vse.cz (P. Vajdečka)

ORCID 0009-0002-7167-0020 (P. Vajdečka)



© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

the main problem is mapping Czech disease, age, region, and time expressions to exact database values. The demo uses Czech public-health open data and DuckDB for local analytical SQL execution [8, 9, 10].

3. Dataset

The SQL answer layer focuses on a newly built infectious-disease fact table rather than claiming to answer over every profiled source. The final table contains 272,858 infectious-disease rows from 2018–2025, enriched with MKN-10-CZ terminology, derived MKN blocks, chapter labels, age intervals, region labels, sex labels, and the stable aggregate measure `reported_case_count`. MKN-10-CZ (39,136 rows) and the NZIS age-group code list (228 rows) are joined into this table. A 1,418,107-row hospitalization dataset is also loaded into DuckDB and represented in the KG through diagnosis, year, and measure context, but it is not yet joined into the final SQL answer table. This keeps the Flat RAG/KG-RAG comparison controlled on the infectious-disease build; cross-fact hospitalization analyses are left for future work.

4. Architecture

Given the infectious-disease answer table described in Section 3, the AI Scope Builder implements the paper’s controlled-scope view of RAG. A Czech question is not answered from retrieved prose. Instead, both compared systems first convert question phrases into executable table values, then generate an inspectable aggregate SQL query over the shared DuckDB answer table. The comparison is controlled: both systems use the same data, measure, SQL validator, and output format. They differ only in how the diagnosis scope is constructed before aggregation. The Flat RAG baseline selects table-present values from flat retrieved evidence, while KG-RAG adds deterministic MKN-10-CZ hierarchy expansion that converts selected disease seeds into explicit table-present `diagnosis_code` filters.

The shared runtime has four stages. First, the system extracts candidate spans from the user’s Czech question, including disease phrases, age expressions, regions, years, and grouping phrases. Second, each span is matched against a flat runtime catalog of public table values and grouping columns. The catalog contains diagnosis codes and Czech diagnosis labels, region labels, age-group labels, sex labels, years, months, and grouping aliases such as diagnosis or region. A database candidate is scored as

$$\text{score}_{DB} = 0.86 \cos(e_s, e_c) + 0.14 \text{lex}(s, c),$$

where $\cos(e_s, e_c)$ is embedding cosine similarity between the span and the catalog item, and $\text{lex}(s, c)$ is deterministic lexical similarity over normalized Czech text, token overlap, and explicit code or range matches. Candidates below cosine 0.20 are discarded. Third, a model selects filters, groupings, and context entities from candidate IDs. Deterministic safeguards may promote high-scoring diagnosis candidates for disease or code spans, with at most five promoted diagnoses per span, minimum score 0.28, and maximum drop 0.14 from the best diagnosis candidate. Fourth, SQL generation is constrained to a single read-only `SELECT` over `fact_infectious_disease_cases_enriched`, with the numeric answer computed as `SUM(reported_case_count)`. The AI methods are not allowed to filter directly on MKN block or chapter columns; hierarchy meaning must be compiled into concrete `diagnosis_code` `IN ( . . . )` predicates before SQL execution.

The Flat RAG baseline is flat at the diagnosis layer. The shared runtime catalog contains the 114 infectious-disease diagnosis values present in the answer table, and each diagnosis is matched independently through the same span-to-catalog scoring and entity-selection process. No MKN hierarchy is consulted when forming the final diagnosis filter. This makes the baseline strong for specific diagnoses and for disease families whose members are easy textual matches, but it does not guarantee complete scope construction: a broad family phrase may retrieve only some sibling codes or include plausible outsiders.

KG-RAG starts from the same extracted spans, candidates, and selected diagnosis seeds; it is not a separate end-to-end retrieval pipeline. Each seed code is mapped to its KG diagnosis node, its

derived MKN block is identified, candidate blocks are ranked, and the table-present diagnoses belonging to the selected scope are retrieved deterministically before SQL finalization. The KG contains 40,906 node documents and a graph with 40,906 nodes and 87,364 edges. The graph includes MKN chapters, derived MKN blocks, MKN diagnosis codes, infectious-disease diagnosis values, hospitalization diagnosis values, years, months, measures, datasets, and columns. The hierarchy edges used for scope construction are SAME_AS_MKN_CODE, CONTAINS_MKN_BLOCK, CONTAINS_MKN_CODE, and HAS_TABLE_DIAGNOSIS_VALUE. The graph also stores 2,418 weighted co-occurrence edges and 1,169 diagnosis profile-similarity edges for evidence and inspection, but these statistical edges do not choose the final SQL diagnosis scope.

KG-RAG maps selected concrete diagnosis-code seeds to KG diagnosis nodes, groups them by derived MKN block, and ranks candidate block scopes by

$$\text{score}_{\text{block}} = 0.55 \frac{|C_b|}{|C|} + 0.25 \frac{\sum_{d \in C_b} w_d}{\sum_{d \in C} w_d} + 0.20 \text{lex}(q, b).$$

Here C is the selected seed set, C_b is the subset of selected seeds inside block b , and w_d is the reported-case total for diagnosis d . The first term measures how many selected seeds support the block, the second term weights that support by reported cases, and the third term measures lexical support between the user question and the block label. After the highest-scored block is chosen, the system retrieves the table-present diagnosis values in that block and decides how far to expand. If block lexical similarity is at least 0.65, or if no seed code is already present in the table for that block, KG-RAG uses the whole table-present block. Otherwise, it uses the inclusive same-letter range from the minimum to maximum seed code when possible, or keeps the seed codes only. The result is a priority list of explicit `diagnosis_code` filters passed to the same SQL layer. Thus KG-RAG does not query by `mkn_block_code`; it converts hierarchy meaning into executable table-present diagnosis-code sets before aggregation. The numeric ratios are deterministic demo parameters chosen during interactive tuning with the live system. They are not learned model parameters and were not fit by optimizing the evaluation metrics.

5. Evaluation

The saved evaluation is a fixed ten-query Czech regression suite chosen to exercise the main demo mechanism—disease-family scope construction—together with year, age, region, and grouping constraints. Six queries are variations of the A00–A09 intestinal-infection scope; the remaining queries test B00–B09, B15–B19, and J12–J18 scopes. We use *reference scope* rather than an independently adjudicated clinical gold standard. Diagnosis reference sets are predefined MKN-10-CZ ranges stored with the evaluation cases and intersected with diagnosis codes actually present in DuckDB; for example, A00–A09 resolves to A01–A09 because A00 has no rows in the fact table. Year, age-group, and region expectations come from the saved evaluation mappings and DuckDB values. Complete-filter accuracy requires every expected filter to match exactly, with no unexpected filter values. The suite is preliminary: the ten queries were available during development, the deterministic expansion thresholds were tuned interactively on the live system, and no held-out calibration split was used. The metrics therefore demonstrate this demo configuration rather than general Czech Text-to-SQL performance.

Table 1

Saved ten-query evaluation. Diagnosis values report precision/recall/F1.

Metric	Flat RAG	KG-RAG
Diagnosis micro P/R/F1	.72/.44/.55	1.00/.88/.94
Diagnosis macro P/R/F1	.72/.48/.57	1.00/.89/.92
Exact diagnosis scope	1/10	8/10
Complete-filter accuracy	1/10	6/10
False-positive diagnosis codes	14	0

Table 1 shows the central result. KG-RAG improves diagnosis recall while preserving precision, raising exact diagnosis scopes from 1/10 to 8/10 and complete-filter accuracy from 1/10 to 6/10. The remaining complete-filter failures are informative: two involve age or regional aliases handled by the shared flat scope builder, and two involve partial diagnosis expansion under conservative block-expansion settings. Thus the gain is not a more fluent answer sentence, but a more complete executable scope over which DuckDB computes the same aggregate measure.

Qualitative diagnosis-scope example. The demo query “*Porovnej střevní infekční nemoci u dětí podle věkové skupiny.*” illustrates the failure mode. The intended diagnosis family is the MKN block A00–A09; in the current answer table, the table-present members are A01–A09. Flat RAG selected

{A08, A07, A04, A48, B99}.

The three correct seeds, A08, A07, and A04, all contain the Czech token *střevní* in their labels. The two outsiders, A48 and B99, are broadly infectious or bacterial but outside the intended intestinal-infection block. At the same time, flat RAG misses six table-present siblings: A01, A02, A03, A05, A06, and A09.

KG-RAG starts from the same flat seeds but closes the scope through the MKN-10-CZ block relation. The final SQL diagnosis scope is

{A01, A02, A03, A04, A05, A06, A07, A08, A09}.

Several recovered labels, such as typhoid/paratyphoid, salmonella infections, shigellosis, food-borne intoxication, amoebiasis, and infectious gastroenteritis, are not direct textual matches to the query. They are recovered because they are table-present members of the same official controlled vocabulary block. This example shows the main mechanism: flat RAG approximates family meaning by textual similarity, while KG-RAG uses textual evidence to find seeds and then expands them through typed MKN-10-CZ relations before SQL aggregation.

6. Demo Scenario

The live demo lets a user enter a Czech public-health question and inspect both architectures side by side. For each run, the interface displays extracted spans, retrieved flat evidence, selected database entities, KG seeds, graph expansion, generated SQL, and the returned aggregate table. This makes it clear whether an answer changed because of different generated prose or because a different executable diagnosis-code set was passed to DuckDB.

The deployed demo is available at <https://kgrag.13.63.202.150.sslip.io/>. For review purposes, the demo can be accessed using the following temporary credentials: username: kgrag and password: bHj+K0NQrKc1+AtWSdizGo4lG13vkrYl. The public repository is available at <https://github.com/pvajdecka/semantics-kg-rag-public-health>, and the demo video is available at <https://www.youtube.com/watch?v=LYnnABiFcZk>.

7. Discussion and Conclusion

The results support the paper’s core claim: KG-RAG is valuable for query-conditioned table summarization when a user phrase denotes a coded family rather than a literal table value. Flat RAG often retrieves plausible high-precision seeds, but it does not guarantee the complete controlled-vocabulary set. KG-RAG separates seed retrieval from scope closure: retrieval proposes candidate diagnoses, the graph expands them through typed MKN-10-CZ relations, and DuckDB computes the aggregate over explicit `diagnosis_code` values.

The main limitation is evaluation scale. The ten-query suite was available during development, thresholds were tuned interactively, and no held-out calibration set was used; the results should therefore be read as preliminary demo evidence. MKN blocks are derived from configured ranges

because the source CSV provides chapters but no explicit block rows. Age and regional aliases remain in the shared flat scope builder. Hospitalization is loaded as separate database/KG context but excluded from the answer table because it represents a different fact table and aggregate measure for which a cross-fact query contract has not yet been validated. Future work should evaluate larger held-out query sets, typed age and region concepts, and multi-fact analyses. The system does not provide clinical advice or patient-level inference.

Ethics and Environmental Footprint

The demo uses aggregate public-health data and does not process patient-level records, personally identifiable information, or clinical notes. It is intended for exploratory analysis only, not for medical advice, causal claims, diagnosis, treatment recommendations, or patient-level inference. The interface exposes retrieved evidence, graph expansion, generated SQL, and returned tables to reduce misinterpretation risk. The environmental footprint is limited to offline embedding and graph preparation plus small-scale live inference over a local DuckDB layer; no large-scale model training is performed.

Declaration on Generative AI

The author used OpenAI tools for language polishing, editorial revision support, and TeX/LaTeX correction of the manuscript. The author used ElevenLabs for correction of speech in the demo video. OpenAI language and embedding models are also components of the evaluated demo system, as described in Section 4; their use is part of the research method rather than manuscript preparation. The author reviewed and edited all AI-assisted materials and remains fully responsible for the final content, claims, references, submission metadata, and audiovisual materials.

References

- [1] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Kuttler, M. Lewis, W.-t. Yih, T. Rocktaschel, S. Riedel, D. Kiela, Retrieval-augmented generation for knowledge-intensive nlp tasks, in: *Advances in Neural Information Processing Systems*, volume 33, 2020, pp. 9459–9474. URL: <https://arxiv.org/abs/2005.11401>.
- [2] D. Edge, H. Trinh, N. Cheng, J. Bradley, A. Chao, A. Mody, S. Truitt, J. Larson, From local to global: A graph rag approach to query-focused summarization, *arXiv preprint arXiv:2404.16130* (2024). URL: <https://arxiv.org/abs/2404.16130>.
- [3] B. Peng, Y. Zhu, Y. Liu, X. Bo, H. Shi, C. Hong, Y. Zhang, S. Tang, Graph retrieval-augmented generation: A survey, *arXiv preprint arXiv:2408.08921* (2024). URL: <https://arxiv.org/abs/2408.08921>.
- [4] D. Sanmartin, Kg-rag: Bridging the gap between knowledge and creativity, *arXiv preprint arXiv:2405.12035* (2024). URL: <https://arxiv.org/abs/2405.12035>.
- [5] C. Mavromatis, G. Karypis, Gnn-rag: Graph neural retrieval for large language model reasoning, *arXiv preprint arXiv:2405.20139* (2024). URL: <https://arxiv.org/abs/2405.20139>.
- [6] T. Yu, R. Zhang, K. Yang, M. Yasunaga, D. Wang, Z. Li, J. Ma, I. Li, Q. Yao, S. Roman, Z. Zhang, D. R. Radev, Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task, in: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018, pp. 3911–3921. URL: <https://aclanthology.org/D18-1425/>.
- [7] J. Herzig, P. K. Nowak, T. Mueller, F. Piccinno, J. M. Eisenschlos, Tapas: Weakly supervised table parsing via pre-training, in: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020, pp. 4320–4333. URL: <https://aclanthology.org/2020.acl-main.398/>.
- [8] Ústav zdravotnických informací a statistiky ČR, Infekční nemoci v ČR (otevřená data), Národní zdravotnický informační portál, 2025. URL: <https://www.nzip.cz/data/2621-infekcni-nemoci-otevrena-data>, open dataset. Accessed 2026-06-14.

- [9] Ústav zdravotnických informací a statistiky ČR, Číselník mkn-10-cz (česká verze 10. revize mezinárodní klasifikace nemocí) (otevřená data), Národní zdravotnický informační portál, 2025. URL: <https://www.nzip.cz/data/2479-ciselnik-mkn-10-cz-otevrena-data>, open dataset. Accessed 2026-06-14.
- [10] M. Raasveldt, H. Muehleisen, Duckdb: An embeddable analytical database, in: Proceedings of the 2019 International Conference on Management of Data, ACM, 2019, pp. 1981–1984. URL: <https://duckdb.org/pdf/SIGMOD2019-demo-duckdb.pdf>. doi:10.1145/3299869.3320212.