 ISSN (Online): 3140-5312

A Factor Decomposition Method for Separating Store Substitution and Embedding Placement Effects in Vector Database Migration

Junha Yoon[1], Ohhyeon Gwon[2], Junhaeng Lee[3], Hyeji Roh[3], Soowan Cho[4], Suhee Kim[4], and Sangsoon Lim[5]

[1]BS Course, Department of Media Software, Sungkyul University, South Korea
[2]BS Course, Department of Computer Engineering, Sungkyul University, South Korea
[3]MS Course, College of Art & Technology, Chung-Ang University, South Korea
[4]Deep Insight Lab, South Korea
[5]Professor, College of Art & Technology, Chung-Ang University, South Korea
Corresponding author: Sangsoon Lim (slim@cau.ac.kr)

Abstract

When the vector layer of a retrieval-augmented generation (RAG) system is migrated to a different database, the effect is usually assessed by an end-to-end comparison before and after the migration. That comparison, however, blends two independent but co-occurring changes into a single measurement, namely substituting the store engine and relocating where the embedding is computed, so it cannot tell which factor caused the change. This paper introduces a bridge condition, in which the target store is loaded with the embeddings the source system has already produced, so that the entangled comparison splits into two comparisons that each differ in one factor only. It requires no re-embedding, so it is inexpensive and leaves the running service untouched. Applied to a Korean product-review RAG service in production (1,347 chunks, 43 queries), the store substitution showed no detectable change in retrieval quality (top-10 Jaccard 0.971, relevance difference −0.042, $p = 0.523$), whereas relocating the embedding into the database produced a significant, medium-sized effect ($p = 0.003$, Cohen's $d_z = 0.473$). Virtually all of the observed change therefore comes from the placement factor. The method does not judge any particular product and applies to migrations to converged databases in general.

Keywords: Retrieval-Augmented Generation, Vector Database, In-Database Embedding, Migration, Factor Decomposition

1. Introduction

A retrieval-augmented generation (RAG) system consists of an embedding model that turns queries and documents into vectors, a store that indexes and searches those vectors, and a language model that generates an answer grounded in the retrieved results [1], [2]. Among these, the component changed most often after deployment is the vector layer. A representative path leads from a general-purpose relational database with a vector extension to a converged database that has a vector type and an inference runtime inside it, that is, one in which the database itself performs the embedding computation.

The effect of such a migration is usually assessed by an end-to-end comparison that weighs the whole system before and after the change at once. In this comparison, however, two factors change at the same time. The first is the store, that is, the engine that holds the vectors and the indexing method it uses. The second is the placement, that is, whether the embedding is computed by an external service or by an inference runtime inside the database. When a migration is motivated by governance, moving the embedding inside as well is the preferred configuration, and a database that hosts a model imposes a model size limit. This is why the two factors change together in practice.

As a result, the end-to-end difference alone does not reveal which factor caused the change. When quality drops, it cannot be told apart whether the store or the model is at fault, and when nothing changes, it cannot be known whether the two factors cancelled each other out. In either case the conclusion cannot be carried over as is to an environment with a different instance class or model size limit.

This paper inserts an intermediate condition, the bridge condition, that loads the target store with the embeddings the source system has already produced, and thereby splits one comparison in which the two factors are entangled into two comparisons that each differ in one factor at a time. We apply the proposed method to a Korean review-analysis service migrated from a PostgreSQL vector extension to a converged database that supports in-database embedding, and measure it on 1,347 review chunks from the production corpus and 43 fixed queries. This paper does not judge which product is superior, and the latency figures reflect the measurement environment rather than the products.

2. Proposed Method

2.1 System Architecture

The target system retrieves Korean product reviews and generates answers that cite the retrieved evidence. Both architectures operate in four stages: query input, query embedding, vector search, and answer generation. Retrieval is a hybrid scheme that combines vector similarity with metadata filters and source-based prioritization, and this rule and its parameters were kept identical across the three conditions. Because both are

ISSN (Online): 3140-5312

expressed as a single SQL statement, the store and the embedding placement can be swapped while the application logic is left as it is.

Fig. 1 shows the Oracle-based architecture. The application sends a single SQL statement carrying the query string (step 1), and the database converts the query into a 384-dimensional vector with its built-in Open Neural Network Exchange (ONNX) inference runtime (step 2) and then, within the same SQL statement, performs an exact search over the VECTOR column together with metadata filtering (step 3). The language model cites the returned top-*k* chunks to generate an answer (step 4).

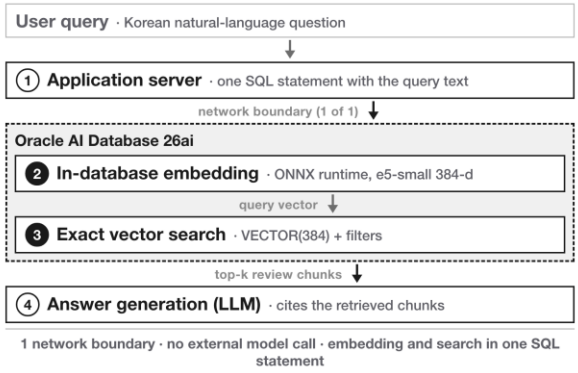

Fig. 1. Oracle-based system architecture

Fig. 2 shows the PostgreSQL-based architecture. The application calls an external embedding service to convert the query into a 1024-dimensional vector (step 2) and then requests the search with that vector as an argument (step 3). The store explores candidates with a hierarchical navigable small world (HNSW) approximate index and applies the same filters, and the remaining steps are as in Fig. 1.

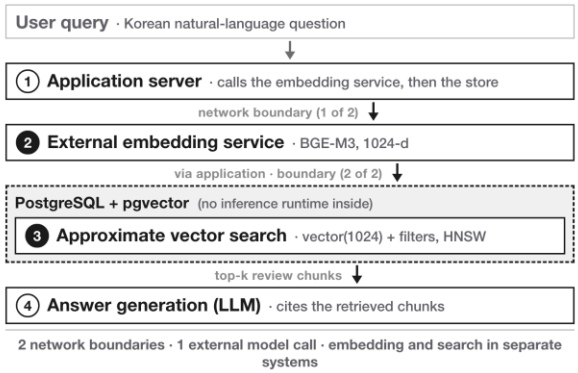

Fig. 2. PostgreSQL-based system architecture

The two architectures differ in two things at once, the store and the embedding placement. One is which engine holds the vectors and with what indexing method, and the other is whether the embedding is computed outside or inside the database. An end-to-end comparison alone cannot tell which of the two produced the difference. The next two subsections summarize the characteristics of each architecture from these two viewpoints, and Section 3 presents a way to separate them.

2.2 Characteristics of the Oracle-Based Architecture

First, it provides a native VECTOR type and a built-in ONNX inference runtime, so embedding and search are combined into a single SQL statement [3]. Second, it performs an exact search that compares every vector, without an approximate index that narrows candidates in advance, so no correct result is missed because of the index. Third, the network boundary is reduced to one, and because the query and the retrieved results stay inside the database, controls such as encryption and auditing apply to the entire retrieval path. Fourth, a built-in model must satisfy the model size limit the database allows, and in this environment the largest multilingual model within that limit was multilingual-e5-small (384 dimensions) [4].

2.3 Characteristics of the PostgreSQL-Based Architecture

First, it provides vector search through an extension module (pgvector) and contains no inference runtime [5]. The embedding model can be chosen separately from the store, so there is no size constraint, and this architecture uses the 1024-dimensional BGE-M3 [6]. Second, it uses an HNSW approximate index that narrows candidates as it searches, so search time grows gently as the data grows, at the cost of missing some correct results [7]. Third, because the embedding is computed externally, the query text leaves the store boundary and one more network round trip is required.

3. Performance Evaluation

3.1 Experimental Setup

In this experiment we denote a retrieval configuration by the triple (S, E, P), where S is the store, E is the embedding model, and P is where the embedding is computed. Comparing the two architectures of Section 2 as they are amounts to comparing (S_0, E_0, external) with (S_1, E_1, in-database); all three coordinates differ, so the contribution of each factor cannot be separated. We therefore add (S_1, E_0, external), a bridge condition that uses the target store but is loaded with the vectors the source system produced, and the three conditions are summarized in Table 1. The bridge condition needs no recomputation of the vectors. Because the columns in production are left as they are and only one more vector column is created and filled, its preparation cost is a single bulk load and the running service is not touched.

Table 1. Experimental conditions

	A (baseline)	B (bridge)	C (migrated)
Store	PostgreSQL + pgvector	Oracle 26ai	Oracle 26ai

ISSN (Online): 3140-5312

Embedding model	BGE-M3 (1024)	BGE-M3 (1024)	e5-small (384)
Placement	external	external	in-database
Vector search	HNSW approx.	exact	exact

A→B changes only the store factor, and B→C only the placement factor. Oracle 26ai denotes Oracle AI Database 26ai [3], and e5-small denotes multilingual-e5-small [4].

In condition C, however, moving the placement also changes the model, because the model size limit of Section 2.2 makes it impossible to host the source model inside the database. Choosing in-database placement means accepting that limit as well, so this coupling was left in place on purpose. Separating the two would require one more condition, (S_1, E_1, external), which runs the smaller model outside the database, and we leave this to future work.

The corpus consists of 1,347 review chunks from a single product domain of the production service, loaded identically into both stores after verifying in advance that the chunk identifiers correspond one to one. The query set was fixed at 43 Korean queries, 13 drawn from production logs and 30 synthetic queries written so as to cover evenly the evaluation aspects and purchase-journey stages the service handles.

Three metrics are measured. The first is the agreement of the retrieved results: how much the top-10 results the two conditions return for the same query overlap (Jaccard), how similar their rankings are (Spearman rank correlation), and, taking the exact search of the bridge condition as ground truth, how many of those the baseline recovers (Recall@10). The second is the relevance of the retrieved evidence: a large language model scores the top-5 chunks on a 0–2 scale [8], and 215 judgments per condition are averaged per query and then compared in pairs across the same queries. The third is the search latency, the actual time taken to process a single query.

The target database ran on the lowest-tier free instance (Always Free, 2 GB RAM) and the external embedding service on a local laptop (Apple M5 Pro). Confidence intervals (CIs) were obtained by bootstrap resampling of the measurements 10,000 times.

3.2 Experimental Results

The measurements by factor are summarized in Table 2.

Table 2. Results by factor

Metric	A → B (store)	B → C (placement)
Top-10 Jaccard [95% CI]	0.971 [0.946, 0.992]	0.175 [0.133, 0.221]
Spearman rank correlation	0.958	—
Recall@10 (vs. exact search)	0.984	—
Relevance score (0–2)	1.228 → 1.270	1.270 → 1.014
Paired difference [95% CI]	−0.042 [−0.172, 0.084]	0.256 [0.098, 0.423]
Significance / effect size d_z	p = 0.523 / −0.098	p = 0.003 / 0.473
Mean search latency (ms)	150.2 → 110.0	110.0 → 947.0
External calls / boundaries	1→1 / 2→2	1→0 / 2→1

Relevance is the per-query mean over the 43 queries (215 judgments per condition), and p values are from a paired t-test.

Looking first at ranking agreement, the two stores loaded with identical vectors returned exactly the same top-10 set for 37 of the 43 queries, and even the minimum was 0.667. The results the baseline missed because of its approximate index amount to about 1.6%.

For the store factor, the paired difference in relevance scores across the same queries was not statistically significant and the effect size was negligible ($t(42)$ = −0.643). A Wilcoxon test on the 28 queries that remain after excluding those with identical scores leads to the same conclusion (p = 0.908).

Since “no difference” is itself our claim, we did not rely only on a non-significant p value but tested equivalence separately [9]. Two one-sided tests examining whether the difference falls within ±0.15 points gave p = 0.052: the lower bound of the confidence interval, −0.172, falls outside this range, so the significance level was not met. We therefore report not that equivalence has been established but that no effect was detected and that its magnitude is bounded to a small range. Since the direction in which the interval extends is the one where the target store scores higher, there is no basis for concluding that substituting the store degraded quality.

By contrast, between B and C, which share the same store and the same search method, the results differ greatly. The top-10 Jaccard falls to 0.175, so mostly different evidence is retrieved for the same query, and the relevance score decreases from 1.270 to 1.014. This difference is significant with a medium effect size ($t(42)$ = 3.104), and a Wilcoxon test on the 38 queries that remain after excluding those with identical scores leads to the same conclusion (p = 0.006). Because no store effect was detected in the change from A to B, virtually all of the end-to-end quality change is due to the placement factor and the model size limit that comes with it. Had only an end-to-end comparison been performed, the roughly 20% drop in relevance would have been blamed on the store, and reverting the store, the remedy that follows from that diagnosis, would not bring the quality back.

For search latency, the ratio of C to B was 8.27 times at the median (Wilcoxon p < 0.001). The increase in C is not a property of in-database embedding itself but a consequence of running inference on a free-tier instance, so it cannot be generalized to higher-tier hardware. Since A and B produce the same vectors with the same external service, the query embedding time was measured once and applied to both conditions, and the latency difference between them therefore reflects only the store search time.

Finally, because 30 of the 43 queries are synthetic, we checked whether they behave differently from the 13 real queries. Relevance scores were systematically lower on the synthetic subset (A: 1.147 vs. 1.415, p = 0.045; C:

 ISSN (Online): 3140-5312

0.880 vs. 1.323, p = 0.007; Mann–Whitney), so the absolute level of relevance must not be read as service quality. In contrast, ranking agreement, the basis on which the cause is attributed, showed no significant difference between the two subsets (p = 0.441, p = 0.570). Absolute scores depend on where the queries come from, but the conclusion about the cause is the same on both.

4. Conclusion

Adding a single bridge condition makes it possible to split an end-to-end comparison into two single-factor comparisons. Applied to a Korean review RAG service in production, no effect of the store factor was detected, and the entire observed change originated from the placement factor. Changing the placement is a trade in governance rather than in performance. In exchange for keeping the query and the review text inside the database boundary, one has to accept the model size limit and the inference performance of the database tier. Therefore, if governance of the stored data is the only goal, replacing the store alone is enough, and if the query path must also stay inside the boundary, one should first confirm that the largest model that can be hosted is sufficient for that domain. The bridge condition makes that confirmation possible before the migration is carried out.

The limitations are as follows. The study targets a single Korean corpus of 1,347 chunks, human assessors were not involved in the relevance scoring, and the measurements were taken on an entry-level instance. In addition, as stated in Section 3.1, placement and model size are coupled by construction, and the result for the store factor does not establish equivalence.

Conflict of Interest

The authors declare that there are no potential conflicts of interest related to this paper.

References

[1] P. Lewis et al., “Retrieval-augmented generation for knowledge-intensive NLP tasks,” Advances in Neural Information Processing Systems, vol. 33, pp. 9459-9474, 2020.

[2] Y. Gao et al., “Retrieval-augmented generation for large language models: A survey,” arXiv preprint arXiv:2312.10997, 2023.

[3] Oracle, “Oracle AI Vector Search User's Guide,” Oracle AI Database 26ai documentation, Available: https://docs.oracle.com/en/database/oracle/oracle-database/26/vecse/, 2026, [Accessed: Aug. 8, 2026].

[4] L. Wang, N. Yang, X. Huang, L. Yang, R. Majumder, and F. Wei, “Multilingual E5 text embeddings: A technical report,” arXiv preprint arXiv:2402.05672, 2024.

[5] pgvector, “pgvector: Open-source vector similarity search for Postgres,” GitHub repository, Available: https://github.com/pgvector/pgvector, 2026, [Accessed: Aug. 8, 2026].

[6] J. Chen, S. Xiao, P. Zhang, K. Luo, D. Lian, and Z. Liu, “M3-Embedding: Multi-linguality, multi-functionality, multi-granularity text embeddings through self-knowledge distillation,” Findings of the Association for Computational Linguistics: ACL 2024, Bangkok, Thailand, pp. 2318-2335, https://doi.org/10.18653/v1/2024.findings-acl.137, Aug., 2024.

[7] Y. A. Malkov and D. A. Yashunin, “Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 42, no. 4, pp. 824-836, https://doi.org/10.1109/TPAMI.2018.2889473, 2020.

[8] L. Zheng et al., “Judging LLM-as-a-judge with MT-Bench and Chatbot Arena,” Advances in Neural Information Processing Systems, vol. 36, Datasets and Benchmarks Track, 2023.

[9] D. Lakens, “Equivalence tests: A practical primer for t tests, correlations, and meta-analyses,” Social Psychological and Personality Science, vol. 8, no. 4, pp. 355-362, https://doi.org/10.1177/1948550617697177, 2017.

Author Information

Junha Yoon
2026 ~ now: B.E. Candidate, Media Software, Sungkyul University
Email: wnsgk314@gmail.com
Research Interest: Human-Computer Interaction, Applied AI, RAG

Ohhyeon Gwon
2026 ~ now: B.E. Candidate, Computer Engineering, Sungkyul University
Email: ruud5521@gmail.com
Research Interest: Large Language Models (LLM), Data-driven Analysis, Applied AI

Junhaeng Lee
2026: B.S., Computer Engineering, Sungkyul University
2026 ~ now: M.S., Dept. of Applied Art and Technology, Chung-Ang University
Email: hkhk9495@cau.ac.kr
Research Interest: Natural Language Processing, Faithfulness in Natural Language Generation, Retrieval-Augmented Generation, Large Language Model-based Agent Systems

ISSN (Online): 3140-5312

Hyeji Roh
2021: BE, Computer Engineering, Sungkyul University
2021 ~ now: Integrated MS-PhD, Dept. of Applied Art and Technology, Chung-Ang University
Email: shgpwl509@cau.ac.kr
Research Interest: AIoT, AI, Physical AI, Deep Learning-based Remote Sensing Analysis

Soowan Cho
2004: BS, Mathematics, Sogang University
2007: MA, Consumer Psychology, Ewha Womans University
2008 ~ 2019: Senior Researcher, Global Research Company (HANKOOK, IPSOS)
2019 ~ 2026: Director of PR & Marketing, Education Industry
2026 ~ now: COO, Deep Insight Lab
Email: soowanc@naver.com

Suhee Kim
2004: BA, Business Administration, Sookmyung Women's University
2009: MA, International Business with Marketing, University of Exeter, UK
2006 ~ 2019: Senior Researcher, Global Research Company (KANTAR, IPSOS)
2019 ~ 2021: Manager of Consumer Research, Global Manufacturer
2025 ~ now: CEO, Deep Insight Lab
Email: shkim8161@gmail.com

Sangsoon Lim (IEEE Senior Member)
2013: Ph.D, Electrical Engineering and Computer Science, Seoul National University
2013 ~ 2017: Senior Engineer, Samsung Electronics
2017 ~ 2025: Assistant Professor, Dept. of Computer Engineering, Sungkyul University
2023 ~ 2024: Visiting Professor, Dept. of Electrical and Computer Engineering, the University of British Columbia
2025 ~ now: Associate Professor, College of Art & Tech., Chung-Ang University
Email: slim@cau.ac.kr
Research Interest: AIoT, AI, ML, Image Analysis, Intelligent Computing