2026 European Conference on Computing in Construction
Corfu, Greece
July 12–15, 2026

CHAT WITH PRE-DEMOLITION AUDITS: KNOWLEDGE-GRAPH BASED INFORMATION RETRIEVAL

Ioanna Mitropoulou[1], Beril Önalan[1], and Catherine De Wolf[1]
[1]ETH Zurich, Switzerland
Circular Engineering for Architecture.

Abstract

Although pre-demolition audits (PDAs) provide early information on building materials, they are underutilized partly because their data require domain expertise to access and interpret. This raises the question: can natural-language querying enable non-experts to retrieve information from PDA datasets? We propose a knowledge-graph-based Retrieval-Augmented Generation (RAG) framework for natural-language querying of PDAs, applied to a case study of 40 French audits. Evaluated against a semantic RAG baseline across 21 queries, Graph-RAG achieves a success rate of 90% compared to 57% using vector-based retrieval, though at 2.34× higher cost, demonstrating its potential to making PDA content accessible to non-expert stakeholders.

Introduction

Pre-demolition audits (PDAs) are increasingly recognized as a key source of information for enabling reuse (Wu et al., 2026; Davis et al., 2025). Produced before the building intervention (refurbishment or demolition), they capture materials, quantities, conditions, and potential reuse pathways. However, despite their early availability and potential to inform design, PDAs remain largely disconnected from early-stage reuse decisions and design workflows. In practice, architects and clients rarely consult PDAs during conceptual design, when material selections and building configurations are most flexible.

The primary barrier to the effective utilization of PDAs is semantic accessibility. These audits often consist of vast quantities of fragmented information spread across extensive structured tabular inventories (material quantities, dimensions, conditions) with unstructured textual reports (building context, accessibility constraints, intervention plans). Extracting insights from these heterogeneous data currently requires users to understand schema conventions and to have the technical proficiency to formulate precise filters or database queries. For non-expert stakeholders, this technical overhead acts as a significant deterrent to data-driven reuse, limiting accessibility, user agency, and the practical use of PDA data in decision-making.

This paper aims to bridge this gap by developing a natural-language interface that enables easy and intuitive access to the contents of these large and complex documents through natural-language queries. We propose a Knowledge-Graph-based Retrieval-Augmented Generation (Graph-RAG) workflow structured to address specifically the typical formats of PDAs, so their information becomes more accessible and interpretable to non-expert users. We compare it against a baseline RAG workflow, as shown in Fig. 1), and conclude the strengths and weaknesses of each system. With the proposed Graph-RAG workflow, we enable users to extract specific material details and building-level context through natural language, effectively democratizing access to reuse data for the architecture, engineering, and construction industry.

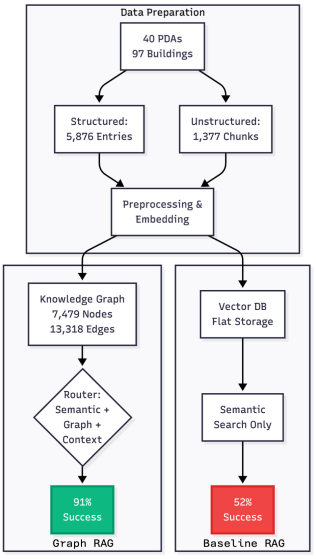

Figure 1: Overview of the proposed Graph-RAG framework for PDA data and its comparison with a baseline RAG approach.

Background

Pre-demolition audits in circular construction. PDAs have emerged as essential tools in circular construction for identifying, quantifying, and managing building materials at end-of-life stages to minimize end-of-life waste (Wu et al., 2026; Godina et al., 2025; Luciano et al., 2022). These audits systematically collect information about the type, quantity, quality, and accessibility of building materials, supporting selective demolition planning, material recovery pathways, and integration into digital building logbooks (Oluleye et al., 2023).

However, PDA datasets present significant challenges for

effective data retrieval. First, heterogeneity and inconsistency across different audits complicate data comparison, transferability, and generalization across different buildings (Akanbi et al., 2020). While recent research has addressed these issues through automated classification, structuring, and enrichment of heterogeneous data into unified schemas (Mitropoulou et al., 2026), challenges in querying remain. Second, PDAs consist of both structured data (tabular inventories with quantitative material information) and unstructured data (textual reports describing building conditions, accessibility, and contextual information), which are interlinked but difficult to query cohesively. This necessitates computational approaches that can jointly query both data modalities.

Natural-language interfaces for built-environment data. Natural-language interfaces have emerged as a critical mechanism for making complex built-environment data accessible to non-expert users. Retrieval-augmented generation (RAG) based on semantic similarity has been increasingly explored for knowledge-base querying (Li and Wang, 2026). RAG has become widely adopted in pre-trained language models to enhance performance in domain-specific tasks by dynamically accessing external knowledge bases, improving factual grounding and relevance in generated responses (Shang et al., 2025). However, standard RAG approaches face significant limitations when handling structured knowledge: they treat the data as isolated chunks of plain text, thus losing information about their relations. As a result, they fail to capture the global semantics and relational dependencies present in built-environment datasets (Procko and Ochoa, 2024; Xu et al., 2024).

To address these challenges, for tabular data specifically, recent methods have explored how LLMs can search, interpret, and reason over tables stored in comma-separated values (CSV) format. However, such LLM-to-CSV search methods suffer from similar loss of semantic information and high failure rates on complex queries (Sequeda et al., 2024). Knowledge graphs offer a promising alternative by representing heterogeneous information through semantically meaningful nodes and relationships. Knowledge graphs preserve the hierarchies and connections present in source data (Shang et al., 2025), enabling more successful retrieval of information through Graph-RAG (Li and Wang, 2026; Sequeda et al., 2024; Xu et al., 2024). Recent advances in Graph-RAG have demonstrated substantial improvements over standard RAG: the Vector-Graph RAG framework for querying semantic web building data achieved a 20-40% improvement in accuracy over state-of-the-art LLMs using traditional RAG (Li and Wang, 2026). Similarly, graph-based approaches for Building Information Modeling (BIM) data showed 75% accuracy compared to just 16.7% for direct LLM querying (Hsiu Tung et al., 2025). Additionally, Graph-RAG has been implemented to enable natural-language interaction with digital twins, demonstrating how such technologies can democratize access to complex built-environment information for non-expert users (Pan et al., 2026). However, to date, Graph-RAG has not been applied to datasets in which structured and unstructured information are semantically interdependent, as in PDA datasets.

Summary. Despite advances in structuring heterogeneous PDA data and developing natural-language interfaces for built-environment information, a gap remains in retrieval frameworks capable of jointly handling semantically linked *structured* data (e.g., tables) and *unstructured* data (e.g., textual reports). This paper addresses these limitations by proposing a Graph-RAG workflow that integrates structured and unstructured PDA data within a graph-based retrieval framework, enabling non-expert users to query this information in natural language.

Methods

We first describe the initial preprocessing of PDA data and the subsequent construction of the knowledge graph that combines structured and unstructured data. Following this, we introduce our retrieval algorithm, which uses a router to decompose the query into smaller steps. Each step is then executed through either semantic search or graph search. Finally, we present a comparative evaluation of our retrieval framework against a baseline semantic RAG interface (lacking access to the knowledge graph) by testing both strategies on a set of 21 questions.

Data preprocessing

The goal of the preprocessing step is to transform the raw PDAs into an accessible format suitable for subsequent knowledge-graph creation. Our case study consisted of a set of 40 PDAs sourced from different buildings across France that collectively describe entries from 97 distinct buildings. The information in these audits consisted of both structured data (tables), which describe building materials and products, and unstructured data (textual reports), which describe contextual information (Fig. 2).

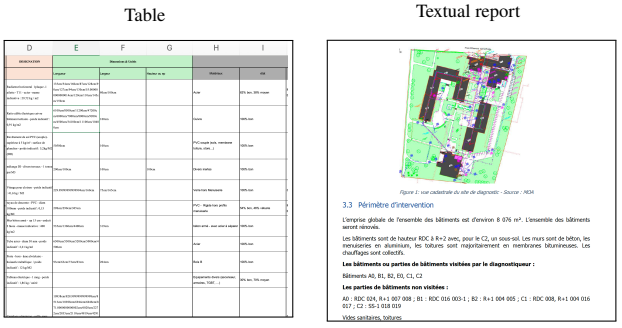

Figure 2: Example of the contents of a PDA. Left: structured input (table). Right: unstructured input (textual report).

Structured data preprocessing. To address the heterogeneity of the structured tabular data across different PDAs, we applied the methodology proposed by Mitropoulou et al. (2026). This process structured and classified the

entries according to a unified schema and classification system. This resulted in a CSV file containing over 7000 entries translated to English in a unified format. Following Mitropoulou et al. (2026), the entries were initially categorized into *Materials* and *Products*, each using a different classification system. For simplicity in the current study, we focused exclusively on the Product entries to maintain a single classification system, specifically the Uniformat II schema (Charette and Marshall, 1999), which provides a hierarchical structure consisting of three classification levels.

Unstructured data preprocessing. A subset of PDAs contained unstructured textual reports. These reports were first translated from French into English using OpenAI GPT-4o. Subsequently, the translated text was segmented into discrete text chunks, each limited to a maximum length of 1200 characters, with an overlap of 200 characters between neighboring chunks.

Knowledge-graph creation

Schema. To create the knowledge graph, we first defined the schema, which specifies the nodes, how they are connected, and which attributes each node stores. The schema includes the nodes *Project*, *Building*, *Entry*, *Classification*, *Report*, and *Text Chunk*, connected as shown in Fig. 3. We developed this model by first clarifying the purpose of the graph, its intended users, and the types of queries it should be able to answer. In our case, the dataset was designed for non-expert stakeholders seeking reliable information on documented materials to inform early-stage reuse decisions. Based on this, we manually identified the core entities and relationships present in the dataset and iteratively refined the structure by testing it on a small data sample. The attributes associated with each node type are listed in Table 1. The nodes Project, Building, Entry, and Text Chunk each store an embedding vector, enabling their contents to be retrieved through semantic search.

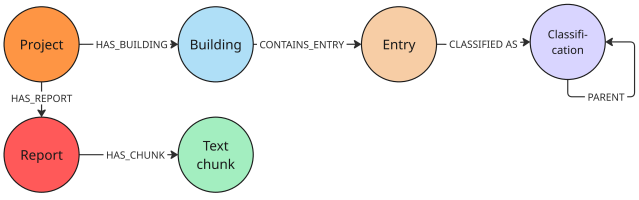

Figure 3: Schema of knowledge graph containing the following nodes: Project, Building, Entry, Classification, Report and Text chunk, connected with semantic edges.

Graph creation. The knowledge graph was constructed using the Graph Data Science Python Client (Neo4j, Inc., 2025a) to connect to Neo4j (Neo4j, Inc., 2025b), a web-based platform for building and querying knowledge graphs. We imported both the structured data (a CSV file containing the processed entries) and the unstructured data (text chunks), and created nodes for each entity defined in the schema, including their embedding vectors. We then added the semantic edges specified in the schema. An example of the resulting graph for a subset of entries belonging to a single building is shown in Fig. 4.

Table 1: Node types and their attributes in the knowledge graph. The nodes Project, Building, Entry, and Text Chunk each contain an embedding vector, so that their contents can be retrieved through semantic search.

py

Node	Attributes
Project	name, zipcode, **embedding vector**
Building	name, total area, usage, year of construction, **embedding vector**
Entry	entry id, assembly type, condition, description, notes, hazardous, quantitative data (mass, volume, dimensions, etc.), **embedding vector**
Classification	code, level
Report	report id, project name
Text Chunk	chunk id, chunk index, **embedding vector**

The full knowledge graph consisted of a total of 7,479 nodes and 13,318 relationships. Querying was performed using the OpenAI GPT-4o, while semantic representations were generated with the OpenAI `text-embedding-3-small` model, which produces 1,536 dimensional embeddings.

Knowledge-graph-based information retrieval

Our workflow for retrieving information using the knowledge graph is shown in Fig. 5 (proposed Graph-RAG) and compared with the baseline RAG workflow. The process begins with the user submitting a natural-language query. In the Graph-RAG, the query is routed through a sequence of steps, described below, aiming to assemble a comprehensive context that will produce an accurate answer.

LLM Router. The query is first handled by the router, an LLM-based function that decomposes the user query into a sequence of smaller retrieval steps. Each step is assigned to either a *graph search* or a *semantic search*, depending on which method is better suited to retrieve relevant context. The purpose of this decomposition is not to answer the question directly, but to make a plan for how to construct a maximally informative context that will enable answering the question at a later step. A few-shot router prompt instructs the LLM on the available tools, the criteria for selecting each tool, and examples of how complex queries should be broken down. For every generated sub-step, the router specifies:

- *Tool*: semantic search | graph search
- *Prompt*: a concise description of what to retrieve
- *Report context*: true | false (whether to also retrieve the most relevant linked report text chunks)

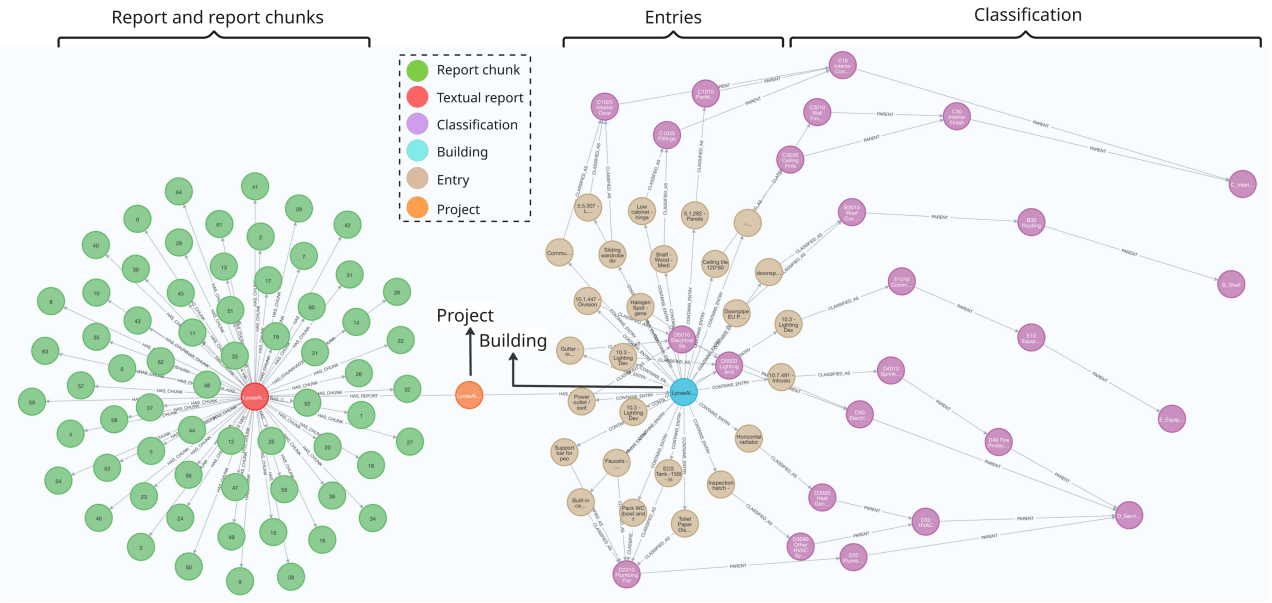

Figure 4: Example of a graph created for a subset of entries of a single building. Left: The Textual report chunk nodes (green) contain unstructured text parts of a Report node (red). Middle: A Building node (blue) is connected to a Project node (orange) and contains Entries (brown). Right: The Entry nodes are connected to their Classification nodes (purple).

Semantic search. The semantic-search component follows a standard RAG workflow. All nodes that contain embeddings (Projects, Buildings, Entries, and Textual Report Chunks) are searchable. For each step, the LLM semantic router determines which node type(s) should be queried. A similarity search is then performed, ranking nodes according to embedding similarity and selecting the top-k most relevant items, with $k = 20$.

Graph search. Graph search uses an LLM that generates Cypher code to traverse the knowledge graph, filter nodes, count elements, or retrieve specific relationships. The prompt for generating the Cypher code includes a description of the graph schema, relationships, and attributes, so that the generated queries are syntactically correct. Once generated, the Cypher query is executed to retrieve the required data.

Linked report context. For each retrieval step, the router may indicate that linked report information should also be included. In such cases, the algorithm starts from nodes obtained via semantic or graph search and traverses the graph to collect all linked textual report chunks. For example, when starting from a Building node, the traversal proceeds via the corresponding Project and its linked Report to retrieve all associated text chunks. If such chunks exist (not all projects have reports), they are semantically ranked, and the top-k most relevant fragments are added to the context.

Answering. Once the final context has been assembled by combining results from graph search, semantic search, and optionally linked report content, it is passed to the LLM along with the original user query and an answer prompt that instructs the model to base its output strictly on the provided context. The LLM then produces the final answer.

Evaluation

To evaluate the performance of the proposed approach, we designed a set of representative questions organized into five categories testing different competencies: (A) simple filtering, (B) counting, (C) more complex queries, (D) queries requiring report context, and (E) unanswerable questions (Table 2). Ground-truth answers were established through manual inspection of the dataset. The evaluation focused on end-to-end question-answering performance, that is, how well the final generated answer agreed with the established ground truth, without separately assessing the success of the intermediate retrieval steps. We compared the performance of our model against a baseline RAG system, described below.

Baseline RAG. The baseline consists of a standard RAG pipeline, illustrated in Fig. 5 left. Given a user query, the LLM semantic router determines which type(s) of searchable nodes containing embeddings to search (Projects, Buildings, Entries, or Textual Report Chunks). These nodes are then ranked according to semantic textual similarity. The top-k most relevant nodes are retrieved and incorporated into the context. For all semantic searches that retrieved the top-k most relevant nodes, we used $k = 20$.

Results

Evaluation questions

The evaluation questions, shown in Table 2, are organized into five categories that test different competencies.

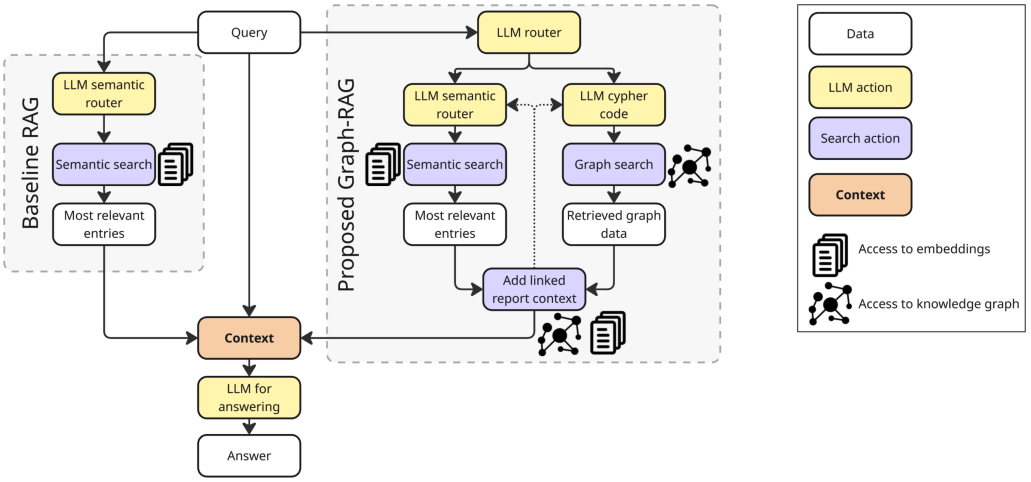

Figure 5: Diagram comparing two information-retrieval workflows for building the context that enables answering natural-language queries; baseline RAG (left), and the proposed Graph-RAG (right). The baseline RAG uses a single semantic search to retrieve relevant nodes by checking their embeddings for semantic similarity. The proposed graph-RAG first sends the query to an LLM router, which decomposes it into sub-steps executed through either semantic or graph search. The retrieved results, together with any linked report context selected by the router, are aggregated into the context.

Category A: Simple filtering. These queries involve retrieving specific entities based on explicit attribute matches, such as material condition or safety status. The baseline RAG performed well on basic listing tasks. The Graph-RAG successfully resolved these logical constraints but occasionally struggled with precise attribute filtering, such as in question A3: ‘‘Find windows in good condition”. There, it produced an invalid filter which returned zero entries, leading to failure in answering this question.

Category B: Counting. This category requires the system to aggregate and count information across the entire dataset to provide quantitative totals. Here, the baseline RAG failed in every query because it relies on local chunk similarity, which cannot capture a global count. In contrast, Graph-RAG demonstrated high competency by using graph operations to provide accurate totals.

Category C: More complex queries. These questions require both filtering and semantic search, and may also include multi-step retrieval. Graph-RAG answered all questions, whereas the baseline failed in half of the questions. This underscores a fundamental limitation of standard RAG: if the information needed is not contained within the same text chunk, the vector search cannot reliably link data from different chunks. Further, question C5, which required a global comparison of window counts across all buildings, was not successfully answered by baseline RAG. This is because this method only sees the top-k chunks it retrieves and has no way of knowing which building has the most windows. Graph-RAG successfully answered the questions by combining semantic search and graph traversal. It also handled semantic variations, such as identifying that ‘‘residential buildings” and ‘‘houses” in C1 and C2 refer to the same category. However, this accuracy came at significantly higher token consumption and cost as the retrieval required more steps and generated more extensive context.

Category D: Queries requiring report context. These queries evaluate the system’s ability to retrieve information from the reports and link it to the corresponding entries. Both methods succeeded in questions D1 and D3, where building-level attributes like construction material or future use could be found in the report text chunks. However, in D2 the baseline RAG failed to link a specific component (glazing panels) to building properties (the heating system) found elsewhere in the report, illustrating the RAG’s inability to link information stored in different chunks. In contrast, Graph-RAG successfully resolved the query by following the graph edges from the identified nodes (glazing panels) to their parent building nodes and finally to their report text chunks where the information about the heating system is stored.

Category E: Answer cannot be found in the dataset. This final category serves as a robustness check to evaluate the systems’ tendency to hallucinate when faced with queries regarding information absent from the dataset, such as market values or carbon-footprint metrics. The baseline RAG demonstrated consistent reliability; it correctly identified the absence of data. In contrast, Graph-RAG failed in E1 by confusing a different numerical property (mass) with the requested market value. Although Graph-RAG successfully provided correct refusals for E2 and E3, it did so at a significantly higher token cost since the system performed a more extensive search before concluding that the data was missing.

Table 2: Comparison of performance across 21 query types for baseline RAG and the proposed Graph-RAG approach. The symbol ✓denotes success and ✗denotes failure. The total amount of tokens used and the total cost for the creation of the full answer are reported for each approach.

Evaluation questions	Baseline RAG			proposed Graph-RAG		
	Tokens	Cost $	Result	Tokens	Cost $	Result
A. Simple filtering						
A1. List five items that are in good condition.	6,475	0.012	✓	9,262	0.021	✓
A2. List five hazardous entries.	8,433	0.024	✓	4,657	0.010	✓
A3. Find windows in good condition.	7,615	0.022	✓	14,379	0.029	✗
B. Counting						
B1. How many buildings are in the dataset?	3,332	0.009	✗	3,872	0.006	✓
B2. Count all the residential buildings in the dataset.	3,795	0.014	✗	3,999	0.007	✓
B3. Count how many entries are in each building in the dataset.	3,733	0.013	✗	8,351	0.026	✓
B4. What percentage of entries are marked as hazardous?	6,327	0.016	✗	5,531	0.013	✓
C. More complex queries						
C1. Find five windows from residential buildings	6,499	0.020	✗	20,720	0.061	✓
C2. Find five windows from houses	6,302	0.019	✗	21,072	0.072	✓
C3. Show exterior cladding options from schools in good condition.	5,627	0.017	✓	23,716	0.062	✓
C4. Find acoustic ceiling panels from office buildings.	6,550	0.018	✗	17,473	0.049	✓
C5. Which building has the most available windows and what are its details?	3,452	0.010	✗	28,802	0.072	✓
C6. We need 30 sq. meters of exterior facade cladding in good condition, which options exist in the dataset?	6,676	0.021	✓	19,263	0.038	✓
C7. Find reusable double glazing windows with more than 2 units available.	6,458	0.019	✓	20,262	0.056	✓
C8. I'm looking for 10 identical toilet separator walls that can be reused.	7,645	0.022	✓	20,217	0.052	✓
D. Queries requiring report context.						
D1. Find a residential building in the dataset whose load-bearing system is made of reinforced concrete.	8,261	0.021	✓	16,322	0.058	✓
D2. I'm looking for available glazing panels. What kind of heating system does the building have that these panels originate from?	8,557	0.022	✗	21,371	0.056	✓
D3. Which building is undergoing transformation into offices?	7,705	0.020	✓	12,761	0.035	✓
E. Answer cannot be found in the dataset.						
E1. What is the market value of the double-glazing windows recorded in the dataset?	8,227	0.025	✓	21,228	0.064	✗
E2. Which reusable material in the dataset has the highest market value?	7,634	0.022	✓	25,592	0.069	✓
E3. What is the carbon footprint of the structural steel parts in the dataset?	7,707	0.023	✓	20,678	0.053	✓

Reasoning plans

Table 3 shows examples of the reasoning plan of sub-steps created by Graph-RAG to gather the necessary context to answer the given questions.

Table 3: Step-by-step reasoning plans followed by Graph-RAG for building the context that can help the LLM answer selected evaluation questions.

ID	Question and reasoning plan
A4:	**Find three entries from buildings constructed before 1970.** 1. *Graph search.* Find buildings constructed before 1970. 2. *Graph search.* Find entries from the buildings found in the previous step.
B4:	**What percentage of entries are marked as hazardous?** 1. *Graph search.* Count all entries. 2. *Graph search.* Count all entries marked as hazardous.
C4:	**Find acoustic ceiling panels from office buildings.** 1. *Semantic search.* Find office buildings. 2. *Graph search.* Find all entries from the buildings found in the previous step. 3. *Semantic search.* Find acoustic ceiling panels from the entries found in the previous step.
C6:	**We need 30 sq. meters of exterior facade cladding in good condition, which options exist in the dataset?** 1. *Semantic search.* Exterior facade cladding entries. 2. *Graph search.* Filter entries found in the previous step to those in good condition with at least 30 square meters available.

Discussion

Overall performance. The evaluation results demonstrate a clear performance-accuracy trade-off between the baseline RAG and the Graph-RAG methods. Across 21 queries, Graph-RAG achieved 90% success rate (19/21 successful queries) compared to 57% (12/21 successful queries) for baseline RAG. Graph-RAG significantly outperformed the baseline RAG on tasks requiring global knowledge of the dataset, multi-step retrieval, and counting, while the baseline RAG proved effective for simple, localized retrieval (Category A) and exhibited higher robustness against hallucination in the absence of data (Category E). However, this improved accuracy came at a 2.34× higher cost, with Graph-RAG averaging $0.0433 per query compared to $0.0185 for baseline RAG. The main strengths and weaknesses of each method are summarized in the following:

- **Relations.** Graph-RAG successfully used graph traversal to find relations between separate nodes and address the inability of baseline RAG to link information that is stored in different text chunks.

- **Global vs. local context.** The baseline RAG is inherently limited by its top-k retrieval mechanism, making it incapable of performing global aggregations

such as counting (Category B). Graph-RAG, conversely, was successful at answering questions that require global context.

- **Efficiency and cost.** The increased accuracy of Graph-RAG comes at a substantial cost. On average, Graph-RAG came at a 2.34× higher cost than the baseline, making standard RAG the more cost-effective solution for simple keyword-based retrieval.

- **Hallucination.** As seen in Category E, the baseline RAG was more conservative and better at identifying knowledge gaps, whereas Graph-RAG occasionally became confused by the retrieved information and failed to identify the lack of the requested data.

- **Flexibility.** Graph-RAG's modular approach, which builds a series of sub-steps, enables adaptive query decomposition, providing flexibility that baseline RAG lacks.

Retrieval methods overlap. Semantic search and graph search have overlapping capabilities. As a result, most queries can be answered through multiple retrieval paths, though with varying degrees of reliability. For instance, identifying a specific material category can be achieved either through vector-based semantic similarity or by querying specific nodes within the graph.

Reliability. When Graph-RAG fails, errors typically originate from the stochastic nature of the query-decomposition phase, inaccuracies in automated Cypher generation, or system implementation issues. For instance, Graph-RAG was unable to answer question A3 because the graph-search query for windows in good condition produced the invalid filter `entry.condition = "Good condition"`, whereas the correct encoding is `entry.condition = "good"`. This illustrates that, while the approach provides a robust foundation for PDA information retrieval, its practical performance remains dependent on the accuracy with which the LLM translates natural-language inputs into structured, executable operations.

Evaluation limitations. The question-based evaluation cannot capture nuances such as incomplete answers, varying response quality, or incorrect confidence levels in generated outputs. Additionally, the manual ground-truth verification introduces potential subjectivity for queries with multiple valid interpretations. Furthermore, the evaluation does not assess user-experience factors such as query formulation difficulty or result interpretability, which are critical considerations given that the intended users are non-experts. Finally, the study focuses on a single dataset of French PDAs with specific structural characteristics. Applicability to PDAs from other regulatory contexts or formats remains to be validated.

Outlook

Several directions could advance this research beyond the current prototype. First, enriching the knowledge graph with additional semantic relationships, such as explicit MENTIONS edges linking Report nodes to referenced Entry or Building nodes and implicit edges for relations such as spatial proximity, material compatibility, and dimensional similarity, would facilitate multi-step queries. Second, extending the system to support continuous conversations with memory would enable iterative query refinement (for example, ‘‘show me more details about that building,’’ ”now filter those to only non-hazardous items”) without re-specifying context. Third, incorporating multimodal data such as photos from PDA site visits would provide richer context; photographic documentation that many PDAs already contain could be leveraged but remains unexploited in text-only frameworks. Fourth, integrating external knowledge sources such as material databases, reuse marketplaces, or embodied carbon calculators would enable queries combining PDA inventory with market availability, pricing, or environmental impact assessments. Fifth, implementing explainability mechanisms that visualize which graph paths, semantic similarities, or retrieval steps contributed to answers would build user trust and enable systematic debugging of retrieval errors. Finally, human-centered deployment studies in real design workflows are needed to evaluate the system as a decision-support tool, including its effects on user trust, understanding, perceived agency, decision quality, decision-making time, and stakeholder engagement.

Conclusion

This paper presented a Graph-RAG framework to enable natural-language querying of PDA data. The approach integrates structured tabular inventories with unstructured textual reports in a knowledge graph to enable non-expert users to retrieve information using natural-language queries without requiring schema knowledge or technical skills. In this way, it improves the accessibility of PDA data and provides a foundation for more human-centered interaction with reuse information.

The evaluation across 21 queries demonstrated that Graph-RAG achieved a higher success rate than baseline RAG, with the performance advantage most pronounced in queries requiring counting, global aggregation, and multi-step reasoning. Both methods performed equally well on purely semantic queries, confirming that semantic similarity-based retrieval remains effective for concept matching when graph structure is not required. However, Graph RAG's higher accuracy comes at a higher cost, as it requires significantly more tokens to answer queries. The study demonstrates that knowledge-graph representations of PDA data enable more flexible and context-aware information retrieval compared to semantic search over unstructured text chunks. The framework's design provides a foundation for applying natural-language interfaces to heterogeneous built-environment datasets where structured

and unstructured information coexist.

Acknowledgments

Research reported in this publication was supported by Bouygues construction R&D and Innovation Department. The authors also wish to thank Dr. Roxanne Goldberg for her valuable feedback and insightful advice. The authors further acknowledge Thibaut Menny and Agyre for facilitating access to PDA data.

References

Akanbi, L. A., Oyedele, A. O., Oyedele, L. O., and Salami, R. O. (2020). Deep learning model for Demolition Waste Prediction in a circular economy. Journal of Cleaner Production, 274:122843.

Charette, R. and Marshall, H. (1999). UNIFORMAT II Elemental Classification for Building Specifications, Cost Estimating and Cost Analysis. NIST Interagency/Internal Report (NISTIR) 6389, National Institute of Standards and Technology, Gaithersburg, MD. Accessed June 6, 2025.

Davis, A., Audí, N. M., and Hall, D. M. (2025). A review of circular industrialised construction for sustainable and affordable housing: Towards a process-driven framework. Sustainable Cities and Society, 133:106837.

Godina, M., Gowler, P., Rose, C. M., Wiegand, E., Mills, H. F., Koronaki, A., Ramage, M. H., and Shah, D. U. (2025). Strategies for salvaging and repurposing timber elements from existing buildings in the UK. Journal of Cleaner Production, 489:144629.

Hsiu Tung, Y., Rezvani, S., Asgharzadeh, F., Neumann, M., and Hartmann, T. (2025). IFC Whisperer: Querying Building Information Models with Large Language Models and IFC-based Knowledge Graphs. Journal of Physics: Conference Series, 3140(16):162007. Publisher: IOP Publishing.

Li, M. and Wang, Z. (2026). BuildingGPT: Query building semantic data using large language models and vector-graph retrieval-augmented generation. Building and Environment, 287:113855.

Luciano, A., Cutaia, L., Altamura, P., and Penalvo, E. (2022). Critical issues hindering a widespread construction and demolition waste (CDW) recycling practice in EU countries and actions to undertake: The stakeholder's perspective. Sustainable Chemistry and Pharmacy, 29:100745.

Mitropoulou, I., Forth, K., Vangelova, S., Menny, T., and De Wolf, C. (2026). Automated structuring, classification, and thermal properties enrichment of pre-demolition audits using llms. Artificial Intelligence for a Sustainable Built Environment. Under review.

Neo4j, Inc. (2025a). Neo4j graph data science python client. `https://neo4j.com/docs/graph-data-science-client/current/`. Accessed: 2025-12-11; Official documentation of the `graphdatascience` Python package for Neo4j Graph Data Science.

Neo4j, Inc. (2025b). Neo4j graph database platform. `https://neo4j.com/`. Accessed: 2025-12-11; official website of the Neo4j graph database platform, a native graph database for managing and querying interconnected data.

Oluleye, B. I., Chan, D. W. M., Olawumi, T. O., and Saka, A. B. (2023). Assessment of symmetries and asymmetries on barriers to circular economy adoption in the construction industry towards zero waste: A survey of international experts. Building and Environment, 228:109885.

Pan, Y., Wang, M., Lu, L., Lamsal, R., Pärn, E., Zlatanova, S., and Brilakis, I. (2026). Llm-enabled multi-agent framework for natural language interaction with graph-based digital twins. Automation in Construction, 183:106791.

Procko, T. T. and Ochoa, O. (2024). Graph Retrieval-Augmented Generation for Large Language Models: A Survey. In 2024 Conference on AI, Science, Engineering, and Technology (AIxSET), pages 166–169, Laguna Hills, CA, USA. IEEE.

Sequeda, J., Allemang, D., and Jacob, B. (2024). A Benchmark to Understand the Role of Knowledge Graphs on Large Language Model's Accuracy for Question Answering on Enterprise SQL Databases. In Proceedings of the 7th Joint Workshop on Graph Data Management Experiences & Systems (GRADES) and Network Data Analytics (NDA), pages 1–12, Santiago AA Chile. ACM.

Shang, Y., Ke, Z., Lin, P., Ren, Q., Zhang, W., Wang, X., Li, X., Gong, F., Wang, S., Wang, B., Xu, Z., Sun, M., and Tan, S. (2025). Empowering knowledge graphs with hybrid retrieval-augmented generation for the intelligent mix scheme of mass concrete. Case Studies in Construction Materials, 23:e04979.

Wu, P.-Y., El-Assady, M., and De Wolf, C. (2026). A unified intelligence-augmented framework for building audits via physical and virtual inspection. Expert Systems with Applications, 302:130514.

Xu, Z., Cruz, M. J., Guevara, M., Wang, T., Deshpande, M., Wang, X., and Li, Z. (2024). Retrieval-Augmented Generation with Knowledge Graphs for Customer Service Question Answering. In Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 2905–2909, Washington DC USA. ACM.