MAPFORGE: A BENCHMARK FOR VIRTUAL KNOWLEDGE GRAPH CONSTRUCTION WITH FEDERATION, TRANSFORMATION AND DOCUMENTATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Virtual knowledge graphs (VKGs) unify access to relational data through an ontology, while keeping the data in the underlying sources. In practice, generating mappings for VKGs may require combining data from multiple databases, transforming source values, or referring to domain documentation. However, current benchmarks do not adequately evaluate these requirements. To fill this gap, we present MAPFORGE, a benchmark for VKG construction with three features: *(1)* Federation combining data from multiple databases; *(2)* Transformation converting source values to match target property requirements; *(3)* Documentation providing domain knowledge for mapping construction. To evaluate these capabilities, MAPFORGE provides 51 mapping generation tasks across 84 databases and 9,780 queries to assess the quality of generated mappings. Our evaluation of six baselines across four open-weight large language models (LLMs) shows that even the best-performing baseline achieves an F1 score of only 61.1%, with further performance drops on the transformation and federation subsets. Interestingly, we find that providing documentation does not always improve performance. These findings highlight the challenges of VKG mapping generation in more realistic settings and provide directions for future research.

1 INTRODUCTION

Ontology-based data access (OBDA), also known as virtual knowledge graph (VKG), provides unified access to relational data through an ontology while keeping the data in the underlying sources (Poggi et al., 2008; Lenzerini, 2018; Xiao et al., 2019; Schneider & Šimkus, 2020). The ontology is connected to the data sources through a set of mappings, which populate the classes and properties of the ontology from the data, written in a language such as R2RML (Das et al., 2012; Calvanese et al., 2023) or the native format of Ontop (Calvanese et al., 2017). When a SPARQL query is posed over the ontology of a VKG, it will be translated on the fly into SQL over the sources, and the graph that the mapping populates is never materialised (Calvanese et al., 2017; Xiao et al., 2019). VKGs support applications in manufacturing, healthcare and geospatial data (Kalaycı et al., 2020; Xiao et al., 2022; Ding et al., 2025). More broadly, mapping source data to domain concepts supports ontology-driven data integration in Palantir (Palantir Technologies, 2026) and Semantica (Semantica, 2026). However, generating high-quality mappings for a VKG is not a simple matter. It requires knowledge of the relational schema and the ontology, together with domain knowledge that is often provided by domain documentation, and thus still relies mainly on human effort (Pinkel et al., 2018; Calvanese et al., 2023; Xiao et al., 2025a).

Several systems have been developed to generate high-quality mappings automatically. Traditional systems such as BootOX (Jiménez-Ruiz et al., 2015) and the Ontop bootstrapper (Calvanese et al., 2017) mainly follow the W3C Direct Mapping specification (Arenas et al., 2012; Sequeda et al., 2011; Spanos et al., 2012), which defines a default way of generating RDF triples from a relational schema. Besides these rule-based systems, the recent LLM4VKG (Xiao et al., 2025a) uses large language models (LLMs) for semantic matching, guided by mapping patterns (Calvanese et al., 2023). Additionally, for the reliable and comparable evaluation of the practical utility of systems like these, Pinkel et al. (2018) propose RODI, a benchmark that scores a computed mapping by how well it works for query answering.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

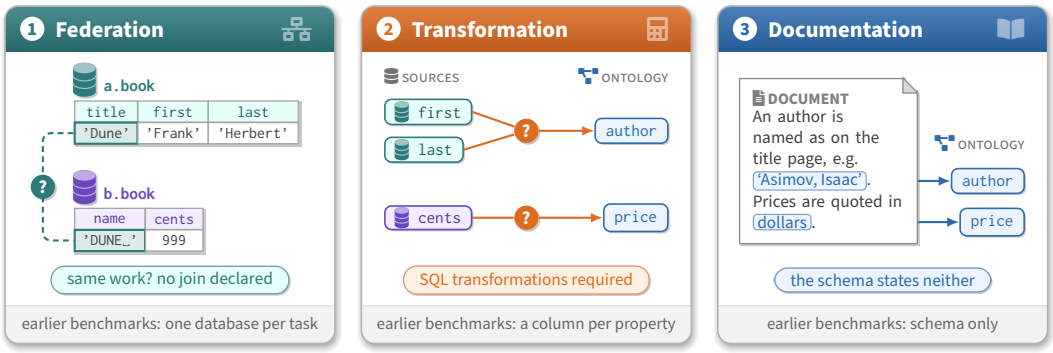


Figure 1: Federation, value transformation, and domain documentation in MAPFORGE.

However, existing methods and benchmarks mainly focus on matching the schema elements of a single data source to ontology classes and properties. In practice, generating mappings for VKGs may also require combining data from multiple databases, transforming source values, and referring to domain documentation. For example, Figure 1 and Listing 2 illustrate mappings that combine records for the same book across two sources, concatenate `first_name` and `last_name` into `author`, and convert `cents` to `price` in dollars, using domain documentation to interpret naming conventions and monetary units. The example shows how these requirements interact, as combining data across sources can involve value transformations informed by domain knowledge. Indeed, Pinkel et al. (2018) already noted that real-world cases demand data transformations such as unit conversion and string composition, but excluded them from their benchmark scenarios. In recent years, Auto-Pipeline and BAT have studied the synthesis of data preparation pipelines that transform source tables into a target table (Yang et al., 2021; Ge et al., 2026), and BLINKG has included transformation tasks in its benchmark for knowledge graph generation from structured files (Castedo et al., 2026). Yet evaluating mapping generation over multiple relational sources with value transformations remains a gap in these benchmarks.

To fill this gap, we introduce MAPFORGE, a benchmark for VKG construction covering federation, transformation, and documentation. It provides 51 mapping generation tasks across 84 databases. Each task asks a system to generate mappings from a target ontology, source databases, and domain documentation. We also propose an algorithm that automatically generates evaluation queries from the gold VKG of each task, guaranteeing coverage of every populated ontology class and property. Using this algorithm, we generate 9,780 queries and, following RODI (Pinkel et al., 2018), evaluate mapping quality through query answering.

Our evaluation of six baselines across four open-weight LLMs on MAPFORGE shows that even the best-performing baseline achieves an F1 score of only 61.1%. Beyond overall scores, we further examine performance on the federation and transformation subsets and compare results with and without domain documentation. Every combination of a baseline and an LLM scores lower on the transformation subset than on the remaining queries, and most show the same pattern for the federation subset. However, providing domain documentation does not consistently improve performance, helping the largest LLM but hurting the smallest. These findings point to directions for improving VKG mapping generation, including combining data across sources, transforming source values, and using domain knowledge effectively.

Our contributions are as follows.

- We introduce MAPFORGE, a benchmark for VKG mapping generation covering federation, transformation, and documentation.
- We develop an algorithm that generates evaluation queries with guaranteed coverage of every populated ontology class and property.
- We evaluate six baselines across four open-weight LLMs and analyse their performance on federation, transformation, and the use of domain documentation.

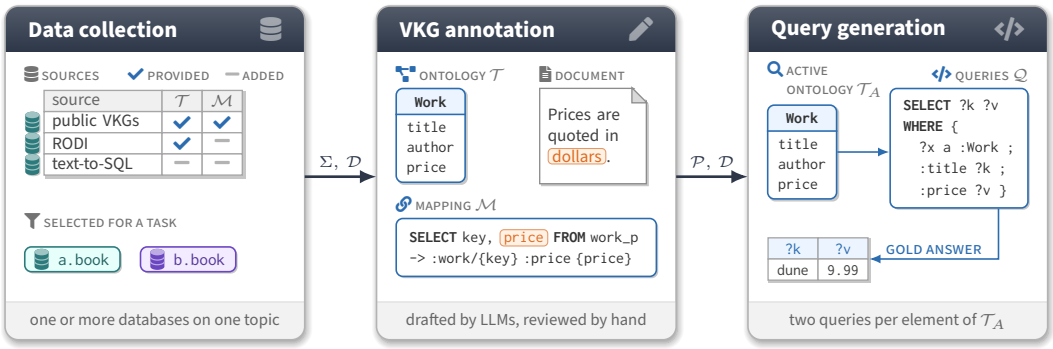


Figure 2: Constructing a MAPFORGE task through data collection, VKG annotation, and query generation.

2 PRELIMINARIES AND TASK FORMULATION

Virtual knowledge graphs. A VKG specification is a triple $\mathcal{P} = (\mathcal{T}, \mathcal{M}, \Sigma)$, where $\mathcal{T}$ is an ontology, Σ is a relational database schema, and $\mathcal{M}$ is a set of mappings linking them (Calvanese et al., 2023). Each assertion consists of a source query in SQL over Σ and a target made of triple templates that construct IRIs and literals from the query answers. In Ontop, source queries can reference *lenses*, named relational views defined over the data sources without modifying the underlying databases[1]. Given an instance $\mathcal{D}$ of Σ, the mappings expose a virtual RDF graph $\mathcal{M}(\mathcal{D})$. SPARQL queries are answered under $\mathcal{T}$ by translation into SQL over $\mathcal{D}$, without materialising the graph (Xiao et al., 2019). For multiple sources, a federation layer exposes their relational schemas as the disjoint union $\Sigma = \bigcup_{i=1}^{n} \Sigma_i$, yielding an ontology-based data federation (OBDF) specification of the same form (Gu et al., 2022). A VKG is single-source when $n = 1$ and federated otherwise.

In Listing 1, the two assertions link works to publishers by constructing the same publisher IRI from matching identifier values.

Listing 1 Two mapping assertions

```
SELECT id, label FROM press      ->  :publisher/{id} a :Publisher ; :name {label} .
SELECT name, pid FROM book       ->  :work/{name} :publisher :publisher/{pid} .
^-- source query                     ^-- target
```

Task formulation. Given an ontology $\mathcal{T}$, a schema Σ with an instance $\mathcal{D}$, and an optional domain document Doc, the task is to produce a set of mappings $\hat{\mathcal{M}}$ and hence the specification $\hat{\mathcal{P}} = (\mathcal{T}, \hat{\mathcal{M}}, \Sigma)$. Following RODI, the generated mappings are evaluated by comparing query answers over the resulting VKG with those over the gold VKG (Pinkel et al., 2018).

3 BENCHMARK CONSTRUCTION

We build MAPFORGE from three groups of data sources (Section 3.1), annotate the VKGs (Section 3.2), and construct queries to evaluate mapping quality (Section 3.3). The overall process is shown in Figure 2.

3.1 DATA COLLECTION

To construct MAPFORGE, we draw on three groups of data sources, retaining their ontologies and mappings where available: *(1)* Public VKGs include NPD FactPages (Skjæveland et al., 2013; Lanti et al., 2015) and an Ontop VKG over EasyBgee (Bastian et al., 2021). Their schemas, ontologies, and mappings are retained as distributed, with the mappings serving as gold annotations; *(2)* The 25

[1] https://ontop-vkg.org/guide/advanced/lenses.html

conference-domain scenarios selected from RODI (Pinkel et al., 2018) provide schemas, ontologies, and paired SQL and SPARQL queries with equal answers, but no mappings. The mappings are therefore written and verified against these query pairs; *(3)* The Text-to-SQL benchmarks Spider (Yu et al., 2018) and BIRD (Li et al., 2023) provide databases from which the ontologies and mappings are constructed.

3.2 VKG Annotation

Constructing a VKG typically begins with designing a domain ontology and then defining mappings that connect the data sources to it. The ontology provides a conceptual view of the domain, while the mappings specify how source data populate its classes and properties. However, the same data sources can support different ontology and mapping designs, depending on how the domain is modeled and the data are interpreted. To address this ambiguity, VKG designers often refer to domain documentation to guide ontology design and mapping construction (Xiao et al., 2025b). Listing 2 illustrates how mappings combine and transform source data, using domain documentation to determine author-name formatting and price units. Rows with matching normalized titles yield one instance, with author names and prices following the document.

Listing 2 Mapping assertions for federation and transformation guided by domain documentation

a.book

title	first_name	last_name
'Dune'	'Frank'	'Herbert'
'Emma'	'Jane'	'Austen'

b.book

name	cents
'DUNE␣'	999
'EMMA'	799

Document

The catalogue lists books by title, with author names written surname first, followed by the given name, e.g. 'Asimov, Isaac'. All prices in the catalogue are listed in dollars.

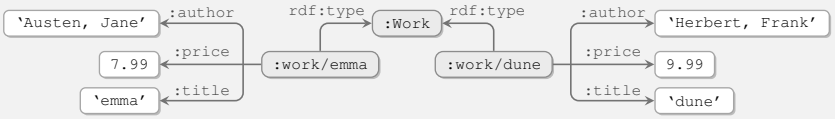
Lenses

```
work_a := SELECT lower(trim(title)) AS key,
                 last_name || ', ' || first_name AS author FROM a.book
work_p := SELECT lower(trim(name)) AS key, cents / 100.0 AS price FROM b.book
work   := SELECT key FROM work_a UNION SELECT key FROM work_p
```

Mapping

```
SELECT key FROM work          ->  :work/{key} a :Work ; :title {key} .
SELECT key, author FROM work_a ->  :work/{key} :author {author} .
SELECT key, price FROM work_p  ->  :work/{key} :price {price} .
```

Result



We construct each task with a four-stage pipeline, reusing available ontologies and mappings:

$$\text{sources} \xrightarrow{\text{❶ select}} (\Sigma, \mathcal{D}) \xrightarrow{\text{❷ design}} \overset{\circlearrowleft}{\mathcal{T}} \xrightarrow{\text{❸ map}} \overset{\circlearrowleft}{\mathcal{M}} \xrightarrow{\text{❹ document}} \overset{\circlearrowleft}{\text{Doc}}.$$

❶ Select one or more databases covering the same topic from the sources described in Section 3.1. ❷ Design the ontology $\mathcal{T}$ to represent the domain concepts and relations. ❸ Construct the mapping $\mathcal{M}$ in the native format of Ontop (Calvanese et al., 2017), including basic assertions and assertions for federation and value transformation. Define SQL expressions for key normalization and value transformations in lenses. ❹ Write the domain document Doc to describe the domain and its conventions, such as naming practices and measurement units.

In steps ❷, ❸, and ❹, LLM-based coding assistants[2] draft the missing artifacts under expert supervision. Experts review the ontology, mappings, and domain document, and drafts that require changes return to the corresponding stage for revision, as indicated by the loop arrows ($\circlearrowleft$).

3.3 Query Generation

Following RODI (Pinkel et al., 2018), mapping quality is evaluated by comparing query answers over the generated and gold VKGs. Therefore, a comprehensive evaluation requires queries that cover the populated classes and properties, both individually and in combination. To this end, we design an algorithm that generates a SPARQL query suite $\mathcal{Q}$ from the gold VKG for each task. The query generation algorithm consists of the following two steps.

$$(\mathcal{P}, \mathcal{D}) \xrightarrow{\text{❶ probe}} \mathcal{T}_A \xrightarrow{\text{❷ instantiate}} \mathcal{Q}.$$

[2]Claude Code, Codex and OpenCode.

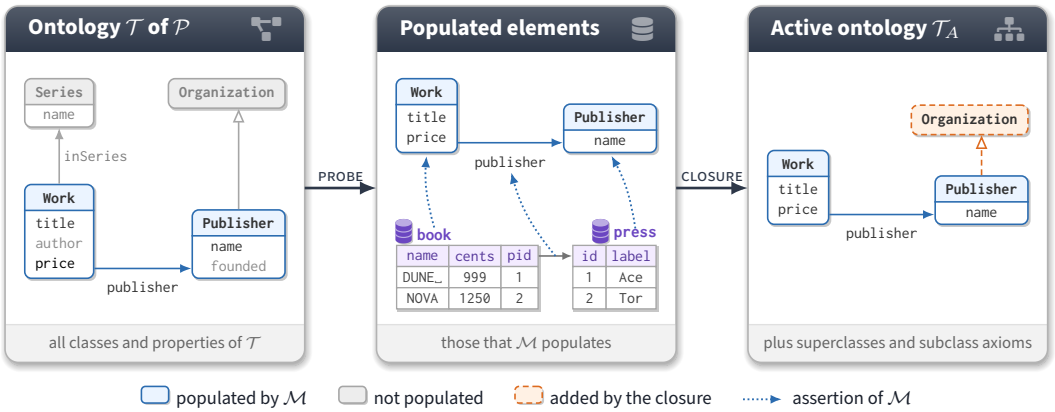


Figure 3: An example of the active ontology $\mathcal{T}_A$.

❶ Probe the gold VKG to construct an active ontology $\mathcal{T}_A$ that records the populated classes and properties. Specifically, the gold VKG is queried for instances of the classes named in the gold mapping and their superclasses. For each class with instances, $\mathcal{T}_A$ records the populated properties and the target classes of object properties, providing the information needed to instantiate query templates. Figure 3 illustrates this process with `Work` and `Publisher`, together with `Organization` added through superclass closure.

❷ Instantiate query templates using the active ontology $\mathcal{T}_A$ to generate SPARQL queries $\mathcal{Q}$ for evaluating mapping quality. There are twelve query shapes, including instance enumeration, property-value retrieval, counting, object-property chains, filters, aggregations, and superclass queries. During instantiation, queries are generated for every populated class and property in $\mathcal{T}_A$ to evaluate whether the generated mappings correctly recover their instances, values, and links. The generated queries are then checked against the gold VKG, and those with empty answers are discarded. Appendix A details the query templates; Appendix D provides a task example and prompt excerpts.

4 Dataset Statistics

MapForge contains 51 mapping tasks and 9,780 gold SPARQL queries across 84 relational data sources. The tasks comprise 24 derived from text-to-SQL benchmarks, 25 from RODI, and two from public VKGs. These tasks contain 1,386 tables, 4,932 columns and over 47 million rows. Task sizes vary substantially, with 2–70 tables and 12–962 columns per task, and 3–342 classes per ontology. Appendix A provides additional dataset statistics and query category definitions.

Query coverage. The query suite covers populated classes and properties individually, as well as multi-hop relationships, filtering, aggregation, and superclass instances. Queries for class instances, property values, and direct relationships account for 5,048 queries (51.6%). The remaining 4,732 queries (48.4%) cover the other five categories. Table 1 reports the counts for all eight categories.

Federation and transformation. We identify two query subsets from the gold mappings. A query belongs to the federation subset when the classes and properties it covers are together populated from multiple data sources. It belongs to the transformation subset when it covers at least one datatype property whose population requires a SQL value transformation. The federation subset contains 2,102 queries across 14 tasks, accounting for 21.5% of the suite. The transformation subset contains 260 queries across 12 tasks, accounting for 2.7%. The subsets may overlap, and seven tasks contain queries from both. Transformation coverage is concentrated in property-value retrieval and equality filtering, which account for 215 of its 260 queries (82.7%).

Domain documentation. All 51 tasks include a domain document. We evaluate mapping generation with and without domain documentation while keeping the tasks and query suites fixed.

Table 1: Gold SPARQL query counts by category. Federation and Transformation denote subsets of All and may overlap.

Query category	All	Federation	Transformation
Class instances	1,267	209	12
Property values	2,664	751	142
Direct relationships	1,117	193	0
Multi-hop relationships	1,410	281	0
Equality filters	1,000	249	73
Range filters	434	89	10
Counts and sums	1,608	296	22
Superclass instances	280	34	1
Total	9,780	2,102	260

5 EXPERIMENTS

We evaluate mapping generation systems on MAPFORGE to assess their overall performance and their ability to handle federation and value transformation. The analysis also examines whether domain documentation improves mapping quality and whether the benefit of documentation depends on how much of the target ontology is provided.

5.1 SETUP

Baselines. We compare six mapping generation systems grouped into three types, which cover the three approaches to mapping generation in Section 6. *(1)* DM+LOGMAP and DM+LLM first construct a mapping using the W3C Direct Mapping specification (Arenas et al., 2012), then align the resulting ontology with the target ontology using LogMap (Jiménez-Ruiz & Cuenca Grau, 2011) and an LLM, respectively. *(2)* MAGNETO-LLM (Liu et al., 2025) and LAMP (Ashiddiiqi et al., 2026) predict correspondences between database schema elements and ontology terms, then assemble mappings from these correspondences. *(3)* LLM4VKG (Xiao et al., 2025a) and VANILLA LLM generate complete mappings. LLM4VKG uses a catalogue of mapping patterns, whereas VANILLA LLM receives one prompt per task and can generate arbitrary SQL expressions in the mapping source queries. Following Xiao et al. (2025a), we repair the mappings generated by VANILLA LLM that Ontop rejects where possible.

LLMs and evaluation settings. Each of the five LLM-based systems is evaluated with Qwen-3.5-9B (Qwen Team, 2026a), Qwen-3.8-27B (Qwen Team, 2026b), Gemma-4-12B (Gemma Team, 2026), and Ministral-3-14B (Liu et al., 2026)[3]. For each system, the prompts and evaluation protocol are held fixed across LLMs. DM+LOGMAP uses neither an LLM nor domain documentation, so its results are reused across these experimental conditions.

Evaluation metric. Following RODI (Pinkel et al., 2018), mapping quality is measured by comparing query answers over the generated and gold VKGs. Let $\hat{A}_q$ and A_q denote the respective answer multisets for query q. Query-level F1 is computed as

$$\mathrm{F1}_q = \frac{2|\hat{A}_q \cap A_q|}{|\hat{A}_q| + |A_q|}.$$

We average query-level F1 within each task and then across tasks, reporting scores on a scale of 0–100. Additional experimental settings and scoring details are provided in Appendix B.

5.2 OVERALL PERFORMANCE

Table 2 reports overall F1 under both documentation conditions. The highest score is 61.1, achieved by VANILLA LLM with Qwen-3.8-27B and domain documentation. Without documentation, VANILLA LLM with Qwen-3.8-27B achieves the highest score of 48.7.

[3]The main comparison alone requires 2,040 generation runs, so we only use locally deployed open-weight models. Nevertheless, Qwen-3.8-27B performs close to Claude Opus 4.6 Max on some public benchmarks (Qwen Team, 2026b).

Table 2: F1 on MAPFORGE. For federation and transformation, w/o. and w. denote queries without and with each feature under the gold mappings; both use no document. Documentation compares overall F1 without and with the document. Δ is w. minus w/o.

Method	Federation			Transformation			Documentation		
	w/o.	w.	Δ	w/o.	w.	Δ	w/o.	w.	Δ
DM+LogMap	23.4	8.5	−14.9	22.3	2.8	−19.5	22.1	22.1	0.0
Qwen-3.8-27B									
DM+LLM	31.5	27.0	− 4.5	31.9	4.4	−27.5	30.9	32.4	+ 1.5
Magneto-LLM	27.5	25.5	− 2.0	26.4	7.0	−19.4	25.6	24.2	− 1.4
LAMP	35.4	37.5	+ 2.1	35.2	9.6	−25.6	34.2	35.4	+ 1.2
LLM4VKG	38.3	18.0	−20.3	38.3	8.2	−30.1	38.1	43.0	+ 4.9
Vanilla LLM	**49.6**	**47.6**	− 2.0	**49.3**	**38.0**	−11.3	**48.7**	**61.1**	+12.4
Qwen-3.5-9B									
DM+LLM	25.2	22.7	− 2.5	25.2	6.8	−18.4	24.6	25.4	+ 0.8
Magneto-LLM	24.3	17.1	− 7.2	22.2	4.3	−17.9	21.5	19.5	− 2.0
LAMP	23.3	**28.0**	+ 4.7	23.1	**8.2**	−14.9	22.5	19.3	− 3.2
LLM4VKG	**35.4**	15.5	−19.9	**34.2**	7.3	−26.9	**35.1**	**30.5**	− 4.6
Vanilla LLM	11.4	18.3	+ 6.9	15.2	0.0	−15.2	15.2	11.4	− 3.8
Gemma-4-12B									
DM+LLM	30.1	25.5	− 4.6	30.0	9.1	−20.9	29.2	30.0	+ 0.8
Magneto-LLM	26.9	22.7	− 4.2	25.7	4.4	−21.3	24.8	23.5	− 1.3
LAMP	30.4	**32.7**	+ 2.3	29.7	**9.6**	−20.1	28.8	28.7	− 0.1
LLM4VKG	**33.4**	24.0	− 9.4	**34.6**	8.3	−26.3	**34.4**	**36.4**	+ 2.0
Vanilla LLM	12.3	11.3	− 1.0	10.8	3.6	− 7.2	10.7	12.0	+ 1.3
Ministral-3-14B									
DM+LLM	29.5	**26.7**	− 2.8	29.5	**6.1**	−23.4	28.8	29.1	+ 0.3
Magneto-LLM	22.9	18.3	− 4.6	23.1	3.7	−19.4	22.3	22.2	− 0.1
LAMP	17.0	16.3	− 0.7	17.3	3.0	−14.3	16.9	15.5	− 1.4
LLM4VKG	**36.0**	18.8	−17.2	**35.8**	**6.1**	−29.7	**35.7**	**29.6**	− 6.1
Vanilla LLM	17.4	23.8	+ 6.4	20.6	4.5	−16.1	20.1	25.8	+ 5.7

On Qwen-3.8-27B without documentation, VANILLA LLM outperforms LLM4VKG by 10.6 points and LAMP, the strongest schema-matching system, by 14.5 points. With the other three LLMs, however, LLM4VKG ranks first under both documentation conditions. Without documentation, LLM4VKG scores between 34.4 and 38.1 across the four LLMs, whereas VANILLA LLM scores between 10.7 and 48.7.

For the direct-mapping systems, replacing LogMap with an LLM improves F1 from 22.1 to 24.6–30.9 without documentation. MAGNETO-LLM and LAMP do not consistently outperform DM+LLM under the same condition.

5.3 ANALYSIS

The federation and transformation analyses below use runs without domain documentation. We then examine the effect of adding the document.

Federation. Most systems score lower on queries inside the federation subset than outside it (Table 2). For example, with Qwen-3.8-27B, LLM4VKG scores 18.0 inside the subset and 38.3 outside it, whereas VANILLA LLM scores 47.6 and 49.6, respectively. On the ten tasks that contain queries in both subsets, every system scores lower inside the subset on all four LLMs, by 0.4 to 21.2 points. Figure 4 reports federation F1 by source count. On Qwen-3.8-27B, the highest score is 42.9 for queries involving three sources and 17.0 for those involving four sources.

Transformation. All systems score lower on queries inside the transformation subset than outside it (Table 2). Even VANILLA LLM with Qwen-3.8-27B, which achieves the highest transformation F1, scores 38.0 inside the subset compared with 49.3 outside it. To examine performance by transformation type, we group queries by the operations used in their gold mappings. Table 3 shows that non-zero scores for string parsing or formatting and arithmetic or unit conversion are limited to

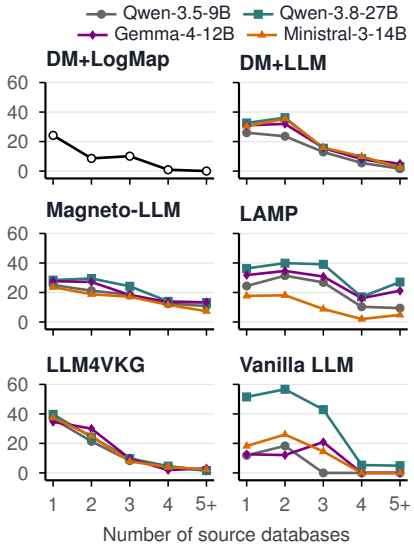


Figure 4: F1 by query source count without documentation.

Method	Concat.	Parse	Cond.	Date	Num.	Mix	Avg.
Qwen-3.8-27B							
DM+LLM	3.2	0.0	18.9	0.0	0.0	0.0	3.7
Magneto-LLM	8.0	0.0	15.0	14.3	0.0	0.0	6.2
LAMP	9.8	0.0	18.9	14.3	0.0	0.0	7.2
LLM4VKG	8.6	0.0	12.6	0.0	0.0	0.0	3.5
Vanilla LLM	35.8	74.1	47.7	0.0	84.4	0.0	40.3
Qwen-3.5-9B							
DM+LLM	5.5	0.0	8.1	0.0	0.0	0.0	2.3
Magneto-LLM	5.3	0.0	1.2	0.0	0.0	0.0	1.1
LAMP	7.8	0.0	19.0	14.3	0.0	0.0	6.8
LLM4VKG	7.5	0.0	10.6	0.0	0.0	0.0	3.0
Vanilla LLM	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Gemma-4-12B							
DM+LLM	9.4	0.0	19.0	0.0	0.0	0.0	4.7
Magneto-LLM	5.2	0.0	10.0	0.0	0.0	0.0	2.5
LAMP	9.8	0.0	18.9	14.3	0.0	0.0	7.2
LLM4VKG	8.8	0.0	12.6	0.0	0.0	0.0	3.6
Vanilla LLM	4.8	0.0	0.0	0.0	0.0	0.0	0.8
Ministral-3-14B							
DM+LLM	5.7	0.0	14.0	0.0	0.0	0.0	3.3
Magneto-LLM	3.9	0.0	15.0	0.0	0.0	0.0	3.2
LAMP	4.1	0.0	0.0	0.0	0.0	0.0	0.7
LLM4VKG	7.6	0.0	8.7	0.0	0.0	0.0	2.7
Vanilla LLM	3.7	6.8	0.0	0.0	33.3	0.0	7.3

Table 3: F1 by transformation type (no document). Avg. is the mean of six type scores.

VANILLA LLM with Qwen-3.8-27B and Ministral-3-14B. All evaluated LLM-based systems score zero on queries covering properties from multiple transformation categories.

Domain documentation. Adding domain documentation does not consistently improve mapping quality (Table 2). With Qwen-3.8-27B, overall F1 increases by 12.4 points for VANILLA LLM and 4.9 points for LLM4VKG, while the other three LLM-based systems change by less than two points. With Qwen-3.5-9B, however, four of the five LLM-based systems score lower with documentation. The effect also varies across systems on Gemma-4-12B and Ministral-3-14B, with changes ranging from -6.1 to $+5.7$ points. These results show that providing domain knowledge alone does not ensure better mappings, and the benefit of documentation depends on both the system and the LLM used. Transformation category definitions, additional results by query subset and analyses of mapping failures are provided in Appendix B.

5.4 ONTOLOGY VISIBILITY

As noted by LLM4VKG (Xiao et al., 2025a), the initial ontology in VKG construction may be incomplete and require extension based on the database schema and data. Motivated by this setting, we examine how mapping quality depends on the amount of target-ontology information available during generation and whether domain documentation helps when this information is limited. We use VANILLA LLM with Qwen-3.8-27B for generation, followed by LLM-based alignment to the full target ontology.

From zero to full visibility, F1 rises from 33.0 to 47.8 without documentation and from 44.1 to 60.9 with it (Figure 5a). Documentation adds 13.1 points even at full visibility, suggesting complementary benefits from the two inputs. With documentation, range filters show the largest gain across visibility settings (Figure 5c), but deployment success does not improve alongside F1 (Figure 5b). Appendix C details the procedure.

6 RELATED WORK

VKGs and mapping generation. The VKG setting was formalised as OBDA by Poggi et al. (2008) and is implemented by query rewriting in systems such as Ontop (Calvanese et al., 2017; Xiao et al., 2019). OBDF extends the setting to several sources, which a federation layer exposes as one relational schema, and takes the mapping over that schema as given (Gu et al., 2022; 2025).

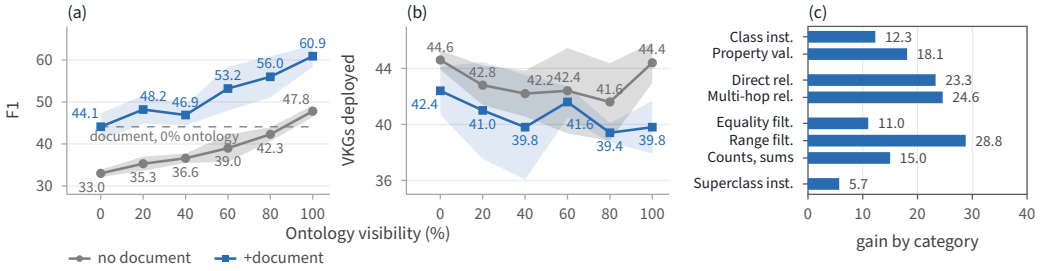


Figure 5: Target-ontology visibility on Qwen-3.8-27B. Panels show (a) F1, (b) successfully deployed VKGs, and (c) per-category F1 gains from zero to full visibility with documentation. Shaded bands indicate 95% confidence intervals.

Existing systems take three broad approaches to mapping generation. *(1)* Schema-only systems, including the Ontop bootstrapper and MIRROR, derive the mapping from tables, columns and foreign keys (Calvanese et al., 2017; de Medeiros et al., 2015). *(2)* Systems that also use the target ontology, including COMA++, IncMap, BootOX and Karma, match schema elements to ontology terms by their names and structure (Aumueller et al., 2005; Pinkel et al., 2013; Jiménez-Ruiz et al., 2015; Knoblock et al., 2012), a role that schema matchers built on language models such as Magneto now play, so we adapt Magneto as a baseline (Liu et al., 2025). *(3)* LLM-based systems generate mappings, including LLM4VKG guided by mapping patterns (Xiao et al., 2025a; Calvanese et al., 2023) and the RML and YARRRML generators of Hofer et al. (2024) and Schmidt et al. (2025).

Benchmarks for mapping generation. RODI, the benchmark for relational-to-ontology mapping generation, scores a generated mapping by the answers it returns to a query workload, treats one source at a time and leaves value transformations out of its default benchmark (Pinkel et al., 2018). BLINKG evaluates LLMs on mapping structured files to an ontology and compares task-specific mapping decision tables with gold annotations (Castedo et al., 2026). Spider defines execution accuracy and BIRD uses execution-based evaluation, but their task specification is a natural-language question rather than an ontology (Yu et al., 2018; Li et al., 2023). MAPFORGE keeps the execution-based scoring of RODI over complete data sources and evaluates mapping generation involving federation, value transformation, and domain documentation.

Data preparation and transformation synthesis. The transformation feature draws on data preparation, where Auto-Pipeline and BAT synthesise a pipeline of relational operators that turns input tables into a target table (Yang et al., 2021; Ge et al., 2026), and Auto-Join, TDE and Auto-Transform synthesise single value transformations from examples or from a corpus of tables (Zhu et al., 2017; He et al., 2018; Jin et al., 2020). There the target is a flat table, whereas in MAPFORGE it is an OWL ontology, the value-transformation rules are stated in the domain document, and the mapping is scored through the VKG it defines.

7 CONCLUSION

This paper presented MAPFORGE, a benchmark for VKG mapping generation covering federation, transformation, and documentation. It evaluates generated mappings by deploying them as VKGs and comparing their query answers with those of the gold mappings. Our evaluation shows that all systems score lower inside the transformation subset than outside it, and most show the same pattern for the federation subset. Furthermore, providing domain documentation does not consistently improve these outcomes, as its benefits vary across methods and backbones, and F1 gains can accompany lower deployment success. These findings motivate further work on generating deployable mappings that combine sources, transform values, and use domain knowledge effectively.

AI USE STATEMENT

We used LLM-based coding assistants to draft missing ontologies, mappings and domain documents as described in Section 3.2, and to write evaluation code and figure-generation scripts. We also used an LLM assistant to edit the paper text. The authors reviewed the AI-assisted work and take responsibility for the final content.

ETHICS STATEMENT

MAPFORGE is built from publicly available VKG resources, RODI scenarios and databases from public text-to-SQL benchmarks. The study does not involve human participants. We do not anticipate adverse societal impacts from using the benchmark to evaluate mapping-generation systems.

REPRODUCIBILITY STATEMENT

To support reproducibility, we describe task construction in Section 3, with source statistics, validation details and query templates in Appendix A. The experimental setup and scoring procedure are described in Section 5.1 and Appendix B, with details of the ontology visibility experiment in Appendix C. Alongside these descriptions, we provide dataset examples, code and results in the supplementary material. The dataset and code will be made publicly available upon acceptance.

REFERENCES

Marcelo Arenas, Alexandre Bertails, Eric Prud'hommeaux, and Juan Sequeda. A direct mapping of relational data to RDF. W3C Recommendation, `https://www.w3.org/TR/rdb-direct-mapping/`, 2012. 27 September 2012.

Sulthoni Ashiddiiqi, Fajar J. Ekaputra, Muhammad Zuhri Catur Candra, and Gusti Ayu Putri Saptawati Soekidjo. Reducing human effort in ontology-based data integration: An LLM-assisted mapping pipeline. In *Proc. of KGCW'2026*, volume 4245 of *CEUR Workshop Proceedings*, pp. 54–68, 2026.

David Aumueller, Hong-Hai Do, Sabine Massmann, and Erhard Rahm. Schema and ontology matching with COMA++. In *Proc. of SIGMOD'2005*, pp. 906–908, 2005. doi: 10.1145/1066157.1066283.

Frederic B. Bastian, Julien Roux, Anne Niknejad, Aurélie Comte, Sara S. Fonseca Costa, et al. The Bgee suite: Integrated curated expression atlas and comparative transcriptomics in animals. *Nucleic Acids Research*, 49(D1):D831–D847, 2021. doi: 10.1093/nar/gkaa793.

Diego Calvanese, Benjamin Cogrel, Sarah Komla-Ebri, Roman Kontchakov, Davide Lanti, et al. Ontop: Answering SPARQL queries over relational databases. *Semantic Web*, 8(3):471–487, 2017. doi: 10.3233/SW-160217.

Diego Calvanese, Avigdor Gal, Davide Lanti, Marco Montali, Alessandro Mosca, et al. Conceptually-grounded mapping patterns for virtual knowledge graphs. *Data & Knowledge Engineering*, 145:102157, 2023. doi: 10.1016/j.datak.2023.102157.

Carla Castedo, Enrique Iglesias, Manuel Lama, Alberto Bugarín-Diz, Maria-Esther Vidal, et al. BLINKG: A benchmark for LLM-integrated knowledge graph generation. *Transactions on Graph Data and Knowledge*, 4(2):9:1–9:35, 2026. doi: 10.4230/TGDK.4.2.9.

Souripriya Das, Seema Sundara, and Richard Cyganiak. R2RML: RDB to RDF mapping language. W3C recommendation, W3C, 2012. URL `https://www.w3.org/TR/r2rml/`.

Luciano Frontino de Medeiros, Freddy Priyatna, and Oscar Corcho. MIRROR: Automatic R2RML mapping generation from relational databases. In *Proc. of ICWE'2015*, pp. 326–343, 2015. doi: 10.1007/978-3-319-19890-3_21.

Linfang Ding, Guohui Xiao, Albulen Pano, Mattia Fumagalli, Dongsheng Chen, et al. Integrating 3D city data through knowledge graphs. *Geo-spatial Information Science*, 28(2):780–799, 2025. doi: 10.1080/10095020.2024.2337360.

Congcong Ge, Yachuan Liu, Yixuan Tang, Yifan Zhu, Yaofeng Tu, et al. BAT: Target-instance-free data preparation synthesis via LLM-driven tree search. *Proceedings of the ACM on Management of Data*, 4(3):143:1–143:25, 2026. doi: 10.1145/3802020.

Gemma Team. Gemma 4 technical report. arXiv:2607.02770, 2026. URL https://arxiv.org/abs/2607.02770.

Zhenzhen Gu, Davide Lanti, Alessandro Mosca, Guohui Xiao, Jing Xiong, et al. Ontology-based data federation. In *Proc. of IJCKG'2022*, pp. 10–19, 2022. doi: 10.1145/3579051.3579070.

Zhenzhen Gu, Davide Lanti, Francesco Corcoglioniti, Marco Di Panfilo, Alessandro Mosca, et al. Ontology-based data federation and query optimization. *Knowledge-Based Systems*, 329:114216, 2025. doi: 10.1016/j.knosys.2025.114216.

Yeye He, Xu Chu, Kris Ganjam, Yudian Zheng, Vivek Narasayya, et al. Transform-data-by-example (TDE): An extensible search engine for data transformations. *Proceedings of the VLDB Endowment*, 11(10):1165–1177, 2018. doi: 10.14778/3231751.3231766.

Marvin Hofer, Johannes Frey, and Erhard Rahm. Towards self-configuring knowledge graph construction pipelines using LLMs: A case study with RML. In *Proc. of KGCW'2024*, volume 3718 of *CEUR Workshop Proceedings*, 2024.

Ernesto Jiménez-Ruiz and Bernardo Cuenca Grau. LogMap: Logic-based and scalable ontology matching. In *Proc. of ISWC'2011*, pp. 273–288, 2011. doi: 10.1007/978-3-642-25073-6_18.

Ernesto Jiménez-Ruiz, Evgeny Kharlamov, Dmitriy Zheleznyakov, Ian Horrocks, Christoph Pinkel, et al. BootOX: Practical mapping of RDBs to OWL 2. In *Proc. of ISWC'2015*, pp. 113–132. Springer, 2015. doi: 10.1007/978-3-319-25010-6_7.

Zhongjun Jin, Yeye He, and Surajit Chaudhuri. Auto-Transform: Learning-to-transform by patterns. *Proceedings of the VLDB Endowment*, 13(11):2368–2381, 2020. doi: 10.14778/3407790.3407831.

Elem Güzel Kalaycı, Irlan Grangel González, Felix Lösch, Guohui Xiao, Anees ul Mehdi, et al. Semantic integration of Bosch manufacturing data using virtual knowledge graphs. In *Proc. of ISWC'2020*, pp. 464–481. Springer, 2020. doi: 10.1007/978-3-030-62466-8_29.

Craig A. Knoblock, Pedro Szekely, José Luis Ambite, Aman Goel, Shubham Gupta, et al. Semi-automatically mapping structured sources into the semantic web. In *Proc. of ESWC'2012*, pp. 375–390, 2012. doi: 10.1007/978-3-642-30284-8_32.

Davide Lanti, Martín Rezk, Guohui Xiao, and Diego Calvanese. The NPD benchmark: Reality check for OBDA systems. In *Proc. of EDBT'2015*, pp. 617–628, 2015. doi: 10.5441/002/edbt.2015.62.

Maurizio Lenzerini. Managing data through the lens of an ontology. *AI Magazine*, 39(2):65–74, 2018. doi: 10.1609/aimag.v39i2.2802.

Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, et al. Can LLM already serve as a database interface? A big bench for large-scale database grounded text-to-SQLs. In *Proc. of NeurIPS'2023*, 2023. Datasets and Benchmarks Track.

Alexander H Liu, Kartik Khandelwal, Sandeep Subramanian, Victor Jouault, Abhinav Rastogi, et al. Ministral 3. *arXiv preprint arXiv:2601.08584*, 2026.

Yurong Liu, Eduardo H. M. Pena, Aécio Santos, Eden Wu, and Juliana Freire. Magneto: Combining small and large language models for schema matching. *Proceedings of the VLDB Endowment*, 18 (8):2681–2694, 2025. doi: 10.14778/3742728.3742757.

Palantir Technologies. Ontology overview. Palantir Foundry documentation, https://www.palantir.com/docs/foundry/ontology/overview, 2026. Accessed 2 September 2026.

Christoph Pinkel, Carsten Binnig, Evgeny Kharlamov, and Peter Haase. IncMap: pay as you go matching of relational schemata to OWL ontologies. In *Proc. of OM'2013*, pp. 37–48, 2013.

Christoph Pinkel, Carsten Binnig, Ernesto Jiménez-Ruiz, Evgeny Kharlamov, Wolfgang May, et al. RODI: Benchmarking relational-to-ontology mapping generation quality. *Semantic Web*, 9(1): 25–52, 2018. doi: 10.3233/SW-170268.

Antonella Poggi, Domenico Lembo, Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, et al. Linking data to ontologies. In *Journal on Data Semantics X*, volume 4900 of *Lecture Notes in Computer Science*, pp. 133–173. Springer, 2008. doi: 10.1007/978-3-540-77688-8_5.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026a. URL `https://qwen.ai/blog?id=qwen3.5`.

Qwen Team. Qwen3.8-27B model card, August 2026b. URL `https://huggingface.co/Qwen/Qwen3.8-27B`.

Wilma Johanna Schmidt, Irlan Grangel-González, Tobias Huschle, Lena Wagner, Evgeny Kharlamov, et al. LLM-supported mapping generation for semantic manufacturing treasure hunting. In *Proc. of ESWC'2025*, pp. 84–101, 2025. doi: 10.1007/978-3-031-94578-6_5.

Thomas Schneider and Mantas Šimkus. Ontologies and data management: A brief survey. *KI – Künstliche Intelligenz*, 34(3):329–353, 2020. doi: 10.1007/s13218-020-00686-3.

Semantica. Semantica: Graph-native infrastructure for context and accountable AI systems. `https://github.com/semantica-agi/semantica`, 2026. Version 0.6.7, released 28 August 2026.

Juan F. Sequeda, Syed Hamid Tirmizi, Oscar Corcho, and Daniel P. Miranker. Survey of directly mapping SQL databases to the semantic web. *The Knowledge Engineering Review*, 26(4):445–486, 2011. doi: 10.1017/S0269888911000208.

Martin G. Skjæveland, Espen H. Lian, and Ian Horrocks. Publishing the Norwegian Petroleum Directorate's FactPages as semantic web data. In *Proc. of ISWC'2013*, volume 8219 of *Lecture Notes in Computer Science*, pp. 162–177. Springer, 2013. doi: 10.1007/978-3-642-41338-4_11. Part II.

Dimitrios-Emmanuel Spanos, Periklis Stavrou, and Nikolas Mitrou. Bringing relational databases into the semantic web: A survey. *Semantic Web*, 3(2):169–209, 2012. doi: 10.3233/SW-2011-0055.

Guohui Xiao, Linfang Ding, Benjamin Cogrel, and Diego Calvanese. Virtual knowledge graphs: An overview of systems and use cases. *Data Intelligence*, 1(3):201–223, 2019. doi: 10.1162/dint_a_00011.

Guohui Xiao, Emily Pfaff, Eric Prud'hommeaux, David Booth, Deepak K. Sharma, et al. FHIR-Ontop-OMOP: Building clinical knowledge graphs in FHIR RDF with the OMOP common data model. *Journal of Biomedical Informatics*, 134:104201, 2022. doi: 10.1016/j.jbi.2022.104201.

Guohui Xiao, Lin Ren, Guilin Qi, Haohan Xue, Marco Di Panfilo, et al. LLM4VKG: Leveraging large language models for virtual knowledge graph construction. In *Proc. of IJCAI'2025*, pp. 4715–4723, 2025a. doi: 10.24963/ijcai.2025/525.

Guohui Xiao, Diluxan Sathalingam, Albulen Pano, Yu Feng, and Linfang Ding. ATKIS-DLM-KG: a unified knowledge graph framework for integrating and querying fragmented topographic-cartographic data in the German digital landscape model. *International Journal of Digital Earth*, 18(1):2528703, 2025b. doi: 10.1080/17538947.2025.2528703.

Junwen Yang, Yeye He, and Surajit Chaudhuri. Auto-Pipeline: Synthesizing complex data pipelines By-Target using reinforcement learning and search. *Proceedings of the VLDB Endowment*, 14 (11):2563–2575, 2021. doi: 10.14778/3476249.3476303.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, et al. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. In *Proc. of EMNLP'2018*, pp. 3911–3921, 2018. doi: 10.18653/v1/D18-1425.

Erkang Zhu, Yeye He, and Surajit Chaudhuri. Auto-Join: Joining tables by leveraging transformations. *Proceedings of the VLDB Endowment*, 10(10):1034–1045, 2017. doi: 10.14778/3115404.3115409.

A Dataset and Query Construction Details

A.1 Source Statistics and Gold-Mapping Validation

Table 4 summarizes tasks by data origin, and Figure 6 shows their size distributions. Shared ontologies contribute their declaration counts once per task.

Table 4: Dataset statistics by data origin. Each row reports totals across the corresponding tasks.

		Relational sources			Target ontology			
Data origin	**Tasks**	**Sources**	**Tables**	**Columns**	**Classes**	**Data prop.**	**Object prop.**	**Queries**
Text-to-SQL	24	57	344	1,588	364	528	200	4,772
RODI	25	25	966	2,353	1,163	430	974	4,065
Public VKGs	2	2	76	991	374	245	165	943
All	**51**	**84**	**1,386**	**4,932**	**1,901**	**1,203**	**1,339**	**9,780**

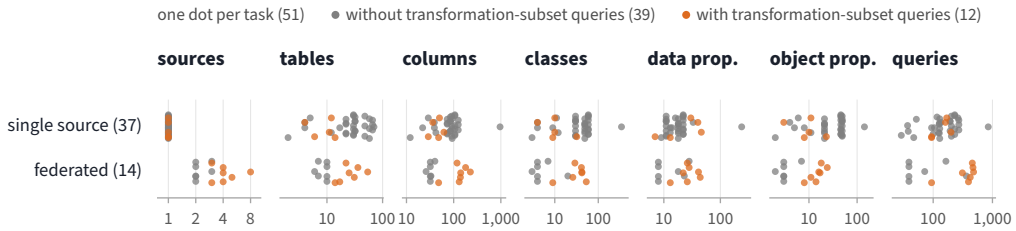


Figure 6: Distribution of the 51 tasks of MapForge by the size of the data sources, the target ontology and the query suite.

Gold-mapping validation. Gold-mapping validation combines expert review with automated checks. Public VKGs retain their maintainers' mappings, while RODI mappings are checked against paired SQL and SPARQL queries. Experts review every assertion, and automated checks confirm that all 4,092 declared key, foreign-key, non-null and unique constraints hold on the source data. As a sanity check of the evaluation pipeline, every gold mapping scores 100.0 F1 against the reference answers for its task. Automated checks also confirm that the domain documents contain no schema table or column names.

A.2 Active Ontology and Query Templates

The probing step constructs the active ontology used to instantiate queries (Algorithm 1). For each populated class, it records populated properties and value counts, a key when available, and the target classes of object properties. It also includes populated superclasses and the subclass axioms connecting the retained classes. Some populated classes have no key, such as Car in the trains task. For these classes, we omit member enumeration but retain counts and query property values without keys.

Table 5 gives the query templates. The first six shapes provide two queries per populated class or property, while the remaining shapes are sampled under a per-shape budget. The multi-hop template shows a two-edge path, with three-edge paths included in the same category.

A.3 Query Categories and Shared Suites

The eight categories in Table 6 group queries by the ontology construct they cover. Listing and counting instances both belong to class instances, or to superclass instances when the class has at least two populated subclasses.

Query suites are shared by the four bookshelf tasks and by three citation-network tasks. We evaluate each query once per task, giving 9,780 task–query pairs from 9,475 distinct queries.

Algorithm 1 Probing the gold VKG for the active ontology $\mathcal{T}_A$.

Require: gold VKG $\mathcal{P}$ over $\mathcal{D}$ as a SPARQL endpoint, $\mathcal{T}$, $\mathcal{M}$
Ensure: active ontology $\mathcal{T}_A$

$\mathcal{T}_A \leftarrow \emptyset$
for all classes C named in $\mathcal{M}$ that have an instance **do** ▷ ASK(`?x a C`)
  PROBECLASS(C)
for all superclasses S in $\mathcal{T}$ of a class in $\mathcal{T}_A$ **do** ▷ closure over $\mathcal{T}$, instances entailed
  PROBECLASS(S)
add the subclass axioms of $\mathcal{T}$ between the classes of $\mathcal{T}_A$
return $\mathcal{T}_A$

procedure PROBECLASS(C)
  add C to $\mathcal{T}_A$ with its number of instances
  for all data properties p with a value on C **do** ▷ one query per class
    add p under C with its number of values
  $\text{key}(C) \leftarrow$ a p whose values identify the instances of C, if any
  for all object properties p of C **do** ▷ one query per class
    $T \leftarrow$ the class of the objects of p ▷ range in $\mathcal{T}$ confirmed by ASK, else queried
    add $C \xrightarrow{p} T$ to $\mathcal{T}_A$ with its number of pairs

Table 5: Query shapes. `C1`, `C2` are classes with keys `k1`, `k2`, `d` is a data property, `o` an object property, `S` a superclass, `c` a constant and `q1`, `q3` the quartiles of `d`.

Shape	Query
Class and property coverage	
Class instances (list)	`SELECT ?k WHERE { ?x a C1 ; k1 ?k }`
Class instances (count)	`SELECT (COUNT(DISTINCT ?x) AS ?n) WHERE { ?x a C1 }`
Property values (list)	`SELECT ?k ?v WHERE { ?x a C1 ; k1 ?k ; d ?v }`
Property values (count)	`SELECT (COUNT(?v) AS ?n) (COUNT(DISTINCT ?v) AS ?m) WHERE { ?x a C1 ; d ?v }`
Direct relationships (list)	`SELECT ?k1 ?k2 WHERE { ?x a C1 ; k1 ?k1 ; o ?y . ?y k2 ?k2 }`
Direct relationships (count)	`SELECT (COUNT(?y) AS ?n) (COUNT(DISTINCT ?x) AS ?s) (COUNT(DISTINCT ?y) AS ?t) WHERE { ?x a C1 ; o ?y }`
Filtering, aggregation, paths and hierarchy	
Equality filter	`SELECT ?k WHERE { ?x a C1 ; k1 ?k ; d ?v . FILTER(?v = c) }`
Range filter	`SELECT ?k WHERE { ?x a C1 ; k1 ?k ; d ?v . FILTER(q1 <= ?v && ?v <= q3) }`
Count aggregation	`SELECT (COUNT(?y) AS ?n) (COUNT(DISTINCT ?x) AS ?m) WHERE { ?x a C1 ; o ?y }`
Sum aggregation	`SELECT (SUM(?v) AS ?s) WHERE { ?x a C1 ; d ?v }`
Multi-hop relationships	`SELECT ?k1 ?k2 WHERE { ?x a C1 ; k1 ?k1 ; o1 ?y . ?y o2 ?z . ?z k2 ?k2 }`
Superclass instances (count)	`SELECT (COUNT(DISTINCT ?x) AS ?n) WHERE { ?x a S }`

Table 6: The eight query categories, grouped by what a query checks.

Query category	What it checks
Entities and values	
Class instances	the size and members of one class
Property values	values of one datatype property
Relationships	
Direct relationships	keys across one object property
Multi-hop relationships	keys across a chain of two or three object properties
Filtering and aggregation	
Equality filters	a property equal to a literal
Range filters	a property inside a quantile band
Counts and sums	counts and sums over a property
Class hierarchy	
Superclass instances	a class over all its subclasses

B Evaluation Details and Supplementary Results

B.1 Experimental Settings

We set the temperature to zero for all five LLM-based baselines. The ontology visibility experiment uses the same temperature setting for both generation and alignment. The main tables use one mapping output per task and experimental condition. The five-run aggregation used only for the ontology visibility experiment is described in Appendix C.

B.2 Scoring and Run Coverage

We compare predicted and gold answers as multisets using the F1 definition in Section 5.1. Because gold answers are non-empty, an empty prediction receives a score of zero. Query errors and timeouts also score zero. A mapping rejected by Ontop scores zero on every query of its task and remains in the mean.

F1 is averaged within each task and then equally across applicable tasks. We report deployment and query execution separately. The deployment count is the number of tasks whose mappings Ontop loads, while the executed-query rate is the fraction of all queries that execute successfully, pooled across tasks.

All documentation comparisons cover the 51 tasks. DM+LogMap uses neither an LLM nor documentation and serves as the same reference across conditions.

B.3 Federation and Transformation Subsets

We derive subset membership from the gold mappings. For each query, we trace its classes and properties through the mapping assertions and lenses to the catalogues containing the underlying relations. The query belongs to the federation subset when these catalogues span at least two data sources. Multiple sources may populate an element through separate assertions or through a union or join lens. For example, the `work` lens in Listing 2 draws from both `a.book` and `b.book`, so queries covering `:Work` belong to this subset. This label records source coverage and does not establish whether a generated mapping must reconcile entities across sources.

The federation subset contains 2,102 queries in 14 tasks, and its complement contains 7,678 queries in 47 tasks. All queries in the four bookshelf tasks belong to the federation subset, leaving ten tasks with queries in both subsets. On these ten tasks, every system scores lower inside the federation subset than outside it on all four LLMs, by 0.4 to 21.2 points without documentation.

A query belongs to the transformation subset when it covers at least one datatype property populated by a SQL value transformation in the gold mapping. The subset contains 260 queries in 12 tasks, and its complement contains 9,520 queries in all 51 tasks. Tasks without an applicable query are excluded from the corresponding subset mean. Federation and transformation membership may overlap.

Transformation types. For Table 3, we assign each query in the transformation subset to one of six groups based on the operations used to populate its datatype properties. Concatenation accounts for 145 queries across nine tasks, and parsing or formatting accounts for 57 across three tasks. The remaining groups contain 21 conditional-selection or fallback queries in four tasks, 14 date-extraction queries in one task, 15 arithmetic or unit-conversion queries in three tasks, and eight mixed-type queries in two tasks. For nested SQL expressions, classification prioritizes numeric operations (`FLOOR`/`ROUND`), then date extraction (`YEAR`), conditional selection (`CASE`/`COALESCE`), parsing or formatting, and concatenation. The mixed group contains queries covering properties from multiple operation groups, rather than a single property with nested functions. Scores are averaged within each task and type, then equally across tasks containing that type. Avg. averages the six type scores equally, whereas transformation F1 in Table 2 averages over tasks. The evaluator's exported scores are retained, including scores for truncated answers. Performance within a group depends on the complete mapping, including whether the required facts and relationships are populated. The groups also cover different task sets, with numeric operations appearing in three tasks and

date extraction in only one. Their scores therefore characterize these workloads rather than establish a general ordering of operator difficulty.

B.4 Detailed Results on Qwen-3.8-27B

Tables 7 and 8 supplement Table 2 with results by data origin and query category, together with deployment counts. Scores are on a 0–100 scale, with the best score in each row in bold. Both conditions cover all 51 tasks.

Table 7: Results on Qwen-3.8-27B without the domain document.

	Direct mapping		Schema matching		End-to-end	
	DM +LogMap	DM +LLM	Magneto-LLM	LAMP	LLM4VKG	Vanilla LLM
By data origin						
Text-to-SQL (24)	18.0	43.4	38.7	50.4	31.8	**66.0**
RODI (25)	27.3	20.8	14.4	20.2	**46.5**	35.3
Public VKGs (2)	5.7	7.3	7.8	**14.3**	7.5	7.6
By query category						
Class instances (1267)	40.5	43.6	31.0	43.8	**54.5**	54.4
Property values (2664)	21.4	40.0	32.6	41.8	45.4	**52.2**
Direct relationships (1117)	9.4	19.4	18.3	25.0	19.2	**47.9**
Multi-hop relationships (1410)	7.2	7.7	7.2	12.6	15.1	**36.7**
Equality filters (1000)	22.6	48.3	44.3	47.1	40.3	**51.0**
Range filters (434)	33.7	37.8	56.9	60.4	60.6	**62.1**
Counts and sums (1608)	20.4	25.1	22.1	33.3	34.3	**52.0**
Superclass instances (280)	47.6	26.7	15.9	29.0	**61.5**	37.1
By transformation-subset membership						
Outside transformation subset (9520)	22.3	31.9	26.4	35.2	38.3	**49.3**
Inside transformation subset (260)	2.8	4.4	7.0	9.6	8.2	**38.0**
Overall F1 (51)	22.1	30.9	25.6	34.2	38.1	**48.7**
VKG deployed (/51)	**51**	**51**	**51**	**51**	**51**	40

Table 8: Results on Qwen-3.8-27B with the domain document. Δ is relative to the condition without it.

	DM+LLM		Magneto-LLM		LAMP		LLM4VKG		Vanilla LLM	
	+doc	Δ	+doc	Δ	+doc	Δ	+doc	Δ	+doc	Δ
By data origin										
Text-to-SQL (24)	44.4	+1.0	36.3	−2.4	50.3	−0.1	39.6	+7.8	**63.9**	−2.1
RODI (25)	23.1	+2.3	13.9	−0.5	22.6	+2.4	49.2	+2.7	**63.2**	+27.9
Public VKGs (2)	4.8	−2.5	6.8	−1.0	**15.6**	+1.3	5.9	−1.6	0.0	−7.6
By query category										
Class instances (1267)	42.3	−1.3	30.5	−0.5	44.4	+0.6	55.8	+1.3	**62.8**	+8.4
Property values (2664)	42.7	+2.7	31.5	−1.1	43.7	+1.9	48.9	+3.5	**62.1**	+9.9
Direct relationships (1117)	21.3	+1.9	15.0	−3.3	25.3	+0.3	30.0	+10.8	**57.4**	+9.5
Multi-hop relationships (1410)	8.2	+0.5	5.5	−1.7	12.4	−0.2	23.3	+8.2	**57.7**	+21.0
Equality filters (1000)	50.3	+2.0	43.1	−1.2	50.0	+2.9	47.3	+7.0	**60.2**	+9.2
Range filters (434)	38.4	+0.6	54.6	−2.3	60.4	±0.0	58.7	−1.9	**62.5**	+0.4
Counts and sums (1608)	25.4	+0.3	20.7	−1.4	34.4	+1.1	42.4	+8.1	**62.9**	+10.9
Superclass instances (280)	27.4	+0.7	15.2	−0.7	30.2	+1.2	**54.6**	−6.9	51.8	+14.7
By transformation-subset membership										
Outside transformation subset (9520)	33.3	+1.4	25.0	−1.4	36.4	+1.2	43.3	+5.0	**61.5**	+12.2
Inside transformation subset (260)	6.2	+1.8	6.6	−0.4	9.6	±0.0	6.5	−1.7	**16.8**	−21.2
Overall F1 (51)	32.4	+1.5	24.2	−1.4	35.4	+1.2	43.0	+4.9	**61.1**	+12.4
VKG deployed (/51)	**51**	±0	**51**	±0	**51**	±0	**51**	±0	35	−5

B.5 Empty Answers and Deployment Failures

Empty answers in the federation subset. We compare empty-answer rates inside and outside the federation subset on the same ten tasks, using Qwen-3.8-27B and Qwen-3.5-9B without documentation (Figure 7). An answer is counted as empty only when the query executes successfully and returns no rows. For each task and subset, we divide the number of empty answers by the total number of queries, retaining failures in the denominator, then average these rates across the ten tasks. Empty answers are more frequent inside the federation subset for every system on these two LLMs, but this observation does not identify the cause. A low empty-answer rate can also accompany frequent failures. For example, Vanilla LLM on Qwen-3.5-9B has an empty-answer rate of 4.8% and a failure rate of 90.0% within the federation subset of these tasks.

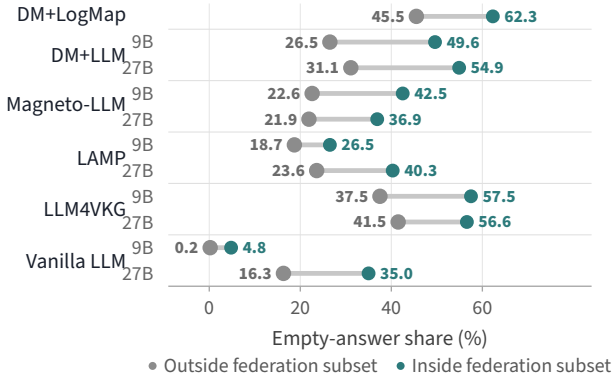


Figure 7: Empty-answer rates on queries outside and inside the federation subset of the same ten tasks, without documentation. Each point is a task-averaged percentage. Execution failures remain in the denominator but are not counted as empty answers.

Documentation and deployment. Documentation can change both mapping accuracy and deployment success. Figure 8 reports F1 with and without documentation on the federation and transformation subsets, covering 14 and 12 tasks respectively, for Qwen-3.8-27B and Qwen-3.5-9B. On Qwen-3.8-27B, Vanilla LLM deploys on 40 tasks without documentation and 35 with it, while its executed-query rate falls from 75.1% to 53.6%. Within the 12 tasks containing queries in the transformation subset, deployment falls from eleven tasks to four. On the four tasks that deploy in both conditions, transformation F1 changes from 53.6 to 50.4. The decline from 38.0 to 16.8 over all 12 tasks therefore reflects both deployment failures and lower F1 on tasks that deploy in both conditions.

C Ontology Visibility Experiment Details

C.1 Generation and Alignment

Generation. A single Qwen-3.8-27B call receives a textual database description, an optional domain document, and the selected ontology information, and generates an OWL ontology and an Ontop mapping to it. At intermediate visibility levels, we randomly select the specified percentage of ontology classes, rounding to the nearest whole number and retaining at least one class when any are available. We also include properties whose declared domain contains a selected class and sample the same proportion of properties whose domain does not specify any named class. The retained ontology statements are serialized as Turtle, omitting direct references to unselected named entities. At 0% visibility, no target ontology is supplied, whereas 100% supplies the full ontology. Because properties are included according to their domains, the visibility percentage does not denote the exact proportion of all ontology entities supplied.

Alignment and mapping rewriting. We use Qwen-3.8-27B with temperature zero to match classes first, then properties using the class matches as context. The entities to be matched come

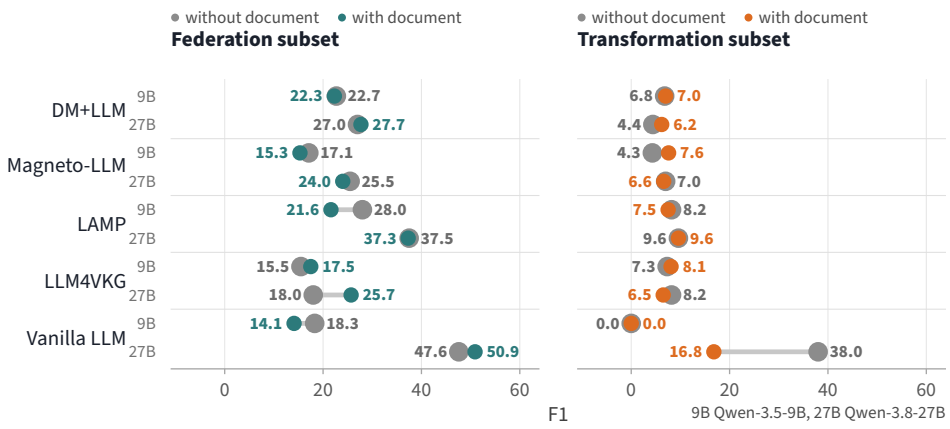


Figure 8: F1 with and without documentation on the federation subset (left) and transformation subset (right), for Qwen-3.8-27B and Qwen-3.5-9B. Each score averages over tasks containing queries in the corresponding subset.

from the generated mapping and are described using schema information and annotations from the generated ontology. If the generated ontology cannot be parsed, we use only the schema information to describe these entities. Target descriptions come from the full target ontology at every visibility setting. The domain document is supplied to generation in the document condition but is not included in the alignment prompts. Both matching calls return source-to-target correspondences in JSON format. We discard correspondences with unknown entities, reversed source and target roles, incompatible entity types or incompatible datatypes detected by the aligner. We remove duplicate matches but do not filter matches by confidence score. Before alignment, generated IRIs are placed in a common namespace. We rewrite each mapping assertion using the matched target class or property, preserving its SQL query, individual IRI templates and literal values. An assertion with no accepted correspondence is dropped, while one with several correspondences is expanded once per target entity. Standard-vocabulary assertions, such as those using `rdfs:label`, pass through unchanged, and rules with no remaining assertions are removed. The resulting mapping is evaluated with the same queries and scoring procedure as the main experiments. Appendix D.2 provides the prompt examples.

C.2 Visibility Settings, Repeated Runs and Aggregation

We evaluate Qwen-3.8-27B on all 51 tasks at visibility levels of 0%, 20%, 40%, 60%, 80% and 100%, with and without documentation. We repeat each condition five times, using the same ontology sample for each task across repetitions. Ontology sampling uses a fixed seed of zero. Runs that complete without a usable mapping retain a zero score.

To compute the plotted F1 means, we first average the five repeated-run scores for each task, then average equally across the 51 tasks. The shaded bands are 95% confidence intervals over the five repetitions, computed in the same way for deployment counts. The intervals describe variation across repeated executions on this fixed benchmark and fixed ontology samples, rather than variation across ontology samples.

C.3 Generation Failures

The NPD task scores zero at every visibility level under both documentation conditions, because generation either exceeds the context window or returns a mapping that Ontop rejects.

D TASK AND PROMPT EXAMPLES

D.1 A TASK FROM THE BOOKSHELF COLLECTION

The task `merge_01_bookshelf` combines the catalogues `bookshelf_a` and `bookshelf_b`. We illustrate how its domain document, gold mapping and queries distinguish work identity from title values, extending the simplified example in Listing 2.

Domain convention. Entries describe printings, whereas `Work` represents the written text shared by those printings. The document specifies how librarians identify works and record titles (Listing 3). These passages contain domain conventions, with no source-schema references, SQL expressions, IRI templates or mapping construction steps. The task assumes that distinct works have distinct normalized titles, so this rule does not address general entity resolution.

Listing 3 Domain-document excerpts

```
A work is known by its title, read without regard to capitalisation and without regard to
    space
before or after it. Two entries whose titles agree on that reading are entries for the
    same work, and
two whose titles do not are entries for different works.
[...]
A title is published exactly as it was written down. Capitalisation and surrounding space
    are part of
what was written and are kept.
```

Gold mapping. The gold mapping implements these conventions over the two source tables. Each source has a `book` table with `book_id` and `title` columns. The gold lenses compute a normalized `work_key` and union the sources with duplicate elimination, retaining original title strings in a separate lens. Listing 4 shows this implementation, not text supplied by the domain document. The lens operations are rendered as SQL and the gold assertion IRIs are abbreviated. Matching titles therefore yield one work IRI while distinct original strings remain title values.

Listing 4 Gold lens operations and mapping assertions

```
-- SQL rendering of the gold title lenses
SELECT CAST(lower(trim("title")) AS VARCHAR) AS work_key, title
FROM "bookshelf_a"."main"."book"
UNION
SELECT CAST(lower(trim("title")) AS VARCHAR) AS work_key, title
FROM "bookshelf_b"."main"."book"
-- The work lens unions only work_key, without the title column.

-- ex: = http://example.org/bookshelf#
-- work: = http://example.org/bookshelf/work/
mappingId  merge01-u1-work
target     work:{work_key} a ex:Work .
source     SELECT work_key FROM "lenses"."work"

mappingId  merge01-u1-title
target     work:{work_key} ex:title {title}^^xsd:string .
source     SELECT work_key, title FROM "lenses"."work_title"
```

Evaluation queries. The suite checks both the number of works and their title values (Listing 5). Title retrieval uses ISBN values rather than generated IRI strings to compare answers. The queries are unchanged apart from namespace abbreviation and line wrapping.

Listing 5 Two queries from the task's evaluation suite

```
PREFIX ex: <http://example.org/bookshelf#>
SELECT (COUNT(DISTINCT ?x) AS ?n) WHERE { ?x a ex:Work }

PREFIX ex: <http://example.org/bookshelf#>
SELECT ?key ?v WHERE {
  ?x a ex:Work ; ex:isbn13 ?key ; ex:title ?v FILTER(isLiteral(?v))
}
```

D.2 Prompts Used for Mapping Generation and Matching

The following prompt excerpts illustrate mapping generation with Vanilla LLM and ontology matching with DM+LLM, using Qwen-3.8-27B on the same bookshelf task with documentation. Prompt text is reproduced verbatim apart from line wrapping, and `[...]` marks omitted material.

Input format. For Vanilla LLM, the database is represented as a textual listing of tables with fully qualified SQL names, row counts, primary keys, column types, foreign-key references, and example values. The target ontology is included in Turtle syntax, and the document condition adds the domain document as a separate text block. For DM+LLM, the class and property prompts instead list source and target entities under local identifiers such as `S1` and `T1`. Source entries describe their associated schema elements, while target entries include labels and descriptions, with domains and ranges for properties. The property prompt also includes the class matches established in the preceding stage.

End-to-end generation. Vanilla LLM receives the database description, target ontology and domain document, and returns an Ontop mapping file. Its full prompt includes OBDA formatting rules before the task inputs. Listing 6 retains the task instruction, output constraint and the rule governing shared entity IRIs. The gold mapping shown above is a reference annotation, not an input to this prompt.

Listing 6 Vanilla LLM prompt excerpt

```
You are given a relational database, an OWL ontology, and a document describing how the
     data was
catalogued. Write an Ontop OBDA mapping file that exposes the database as a virtual
     knowledge graph
conforming to that ontology.

Answer with the contents of the .obda file and nothing else: no explanation before it, no
     commentary
after it, no markdown fence. The first line of your answer must be `[PrefixDeclaration]`.
[...]
* IRI templates are free: any base URI and any key column may be used. Two rules that must
      describe
  the same entity have to build the same IRI from the same columns, or the triples will
     not join.
[...]
Write the .obda file now. Start with [PrefixDeclaration] and nothing before it.
```

Ontology matching. DM+LLM instead asks the LLM to align a mechanically generated ontology with the target ontology. The class-matching prompt supplies source identifiers `S1`, `S2`, and so on, target identifiers `T1`, `T2`, and so on, and the domain document. Its output is a JSON array of correspondences, not SQL or a complete mapping. Listing 7 shows the matching instruction and output rules. The JSON object illustrates the required output format.

Listing 7 DM+LLM class-matching prompt excerpt

```
You are aligning two OWL ontologies that describe the same relational database.
[...]
Match each class of A to the class or classes of B that denote the same things.
A class of A is a database table; a class of B is a kind of thing in the domain. The two
often differ in granularity: a join table may be one class in B or none at all, and two
tables may be two subclasses of one class in B.
[...]
Answer with a JSON array and nothing else -- no prose, no code fence:

[{"source": "S1", "target": "T4", "confidence": 0.95}, ...]

Rules:
- Use only the identifiers listed above. Do not invent identifiers and do not write out
     IRIs.
- Omit a source entry entirely when nothing in B means the same thing. A wrong match is
     worse than
  no match: it makes every query over that entity return the wrong rows.
- One source entry may match several target entries; emit one object per pair.
- confidence is your own estimate in [0, 1].
```