OmniOData: Unleashing Small Language Models for OData Query Generation with Synthetic Data and Reinforcement Learning

Tao Bai, Zhaochen Li, Hongxin Shao, Daniel Dahlmeier
SAP
{tao.bai, zhaochen.li, hongxin.shao, d.dahlmeier}@sap.com

Abstract

Despite the success of Large Language Models (LLMs) in structured query generation, OData—a critical RESTful protocol for enterprise APIs—remains under-researched due to a lack of high-fidelity, execution-validated datasets. To bridge this gap, we introduce OMNIODATA, a framework that generates SYNODATA, the first large-scale OData corpus featuring execution-grounded queries and reasoning traces. Using this corpus, we develop OMNIODATA-R1 (1.5B–3B parameters), a family of models that match or surpass frontier proprietary systems, such as GPT-4o and Gemini 3, on realistic industrial benchmarks. Our results demonstrate that the synergy of execution-verified synthetic data and Reinforcement Learning (RL) effectively unlocks the latent reasoning of Small Language Models (SLMs), providing a high-performance, low-latency solution for specialized enterprise query generation. The code and data will be released under an open-source license.

1 Introduction

Structured query languages serve as the foundation for programmatic data access across modern applications. In the relational database domain, Structured Query Language (SQL) has long been the gold standard, providing a robust interface for complex data manipulation. In recent years, the "Text-to-SQL" task has witnessed a paradigm shift driven by the emergence of Large Language Models (LLMs). Leveraging massive pre-training corpora and sophisticated in-context learning, state-of-the-art models have demonstrated a remarkable ability to bridge the gap between unstructured natural language and executable code. From early sequence-to-sequence architectures to modern reasoning-heavy models such as GPT-4o and DeepSeek-R1 (Cai et al., 2025; Pourreza and Rafiei, 2024; Hong et al., 2025), the research community has made significant strides in handling schema linking, nested joins, and ambiguous user intent across benchmarks like SPIDER (Yu et al., 2018; Lei et al., 2024) and BIRD (Li et al., 2024), where there are extensive question-query-reasoning triples with tens of thousands of examples.

However, while SQL dominates the relational landscape, a significant portion of enterprise data—particularly within the Microsoft and SAP ecosystems—is exposed via the *Open Data Protocol (OData)*. OData provides a standardized approach for building and consuming RESTful APIs, offering a URL-based syntax for rich query capabilities such as `$filter`, `$expand`, and `$select`. Despite its prevalence in enterprise-grade software, the **Text-to-OData** task remains significantly under-explored.

In contrast to the mature ecosystem surrounding Text-to-SQL, the development of Text-to-OData systems is currently hindered by a critical bottleneck: the total absence of large-scale, high-quality public datasets. This scarcity presents a dual challenge: a) **Syntactic Uniqueness:** OData's URI-centric syntax and specific operators (e.g., `eq`, `gt`, `contains`) differ fundamentally from traditional SQL clauses, requiring models to learn a distinct logical mapping that is rarely represented in general-purpose code datasets. b) **Resource Constraints:** While proprietary frontier models can generate OData through sheer scale, deploying these massive models for routine API calls is often cost-prohibitive and presents latency hurdles for real-time enterprise applications.

The industry is increasingly paying attention to Small Language Models (SLMs)—models typically under 10 billion parameters—due to their efficiency, privacy benefits, and lower operational costs (Belcak et al., 2025; Wang et al., 2025). However, without domain-specific alignment, these smaller models frequently suffer from "hallucinations" and syntax violations when generating OData queries.

To bridge this gap, we propose OmniOData, a framework designed to unlock the full potential of SLMs for the Text-to-OData task. Our approach addresses data scarcity through a novel five-stage synthetic data generation pipeline, which we employ to construct a comprehensive corpus of natural language and OData pairs, dubbed SynOData. Furthermore, we develop OmniOData-R1, a suite of high-performance Text-to-OData models available in two scales: 1.5B and 3B parameters. We evaluate OmniOData-R1 on the held-out test split of SynOData and an internal, realistic evaluation set, RealOData. Our experimental results indicate that OmniOData-R1 achieves state-of-the-art performance across these datasets—matching or even outperforming significantly larger open-source models (e.g., GPT-OSS-120B (OpenAI, 2025)) and frontier, closed-source models, (e.g., GPT-4o (OpenAI, 2024) and the Gemini 3 series models (Google DeepMind, 2026)), despite its compact parameter count. Our main contributions are summarized as follows:

- **The OmniOData Framework:** A novel five-stage pipeline that eliminates OData data scarcity through execution-verified synthesis and automated self-correction.
- **High-Fidelity Corpus & Benchmark:** The release of SynOData, a first-of-its-kind dataset with 110K+ validated queries, and RealOData, a specialized benchmark for production-level enterprise metadata.
- **State-of-the-Art (SOTA) SLMs:** The development of OmniOData-R1 (1.5B–3B), demonstrating that execution-grounded training allows compact models to outperform frontier systems like GPT-4o and Gemini 3.

2 Related Work

Text-to-Query Benchmarks. The task of mapping natural language to structured queries is dominated by Text-to-SQL research, supported by large-scale, human-annotated benchmarks such as Spider (Yu et al., 2018) and BIRD (Li et al., 2024). Similarly, the API-calling domain has seen significant progress with benchmarks like ToolBench (Guo et al., 2024), which focus on RESTful parameter filling and tool invocation. Beyond relational data, Text-to-Cypher (Ozsoy et al., 2025) and SPARQL synthesis (Ben Amor et al., 2025) have gained traction for querying graph databases. These tasks emphasize the model's ability to navigate non-linear entity relationships and path-based reasoning. Despite the maturity of these adjacent fields, OData synthesis remains significantly underexplored with no datasets or benchmarks publicly available. We argue that progress in this area is primarily hindered by a critical scarcity of high-fidelity, large-scale data capable of capturing the unique intersection of relational depth (e.g., multi-level joins) and protocol-specific URL syntax.

Reinforcement Learning for Reasoning. Reinforcement Learning (RL) has proven highly effective at enhancing the reasoning capabilities of LLMs for complex, multi-step tasks (Le et al., 2022; Ouyang et al., 2022). Recent breakthroughs, most notably DeepSeek-R1 (Guo et al., 2025), demonstrate that models can self-evolve sophisticated reasoning traces through pure RL, achieving state-of-the-art results in mathematics (Tu et al., 2025), programming (Pennino et al., 2025), and autonomous agents (Gao et al., 2025). Despite these advances, the application of RL to the Text-to-OData domain remains unexplored. Inspired by the success of execution-grounded reasoning, we extend these capabilities to OData synthesis, utilizing RL to bridge the gap between flexible natural language and the rigid requirements of OData v4 syntax.

3 Data Synthesis Framework

3.1 Overview

As illustrated in Figure 1, the OmniOData synthesis framework processes data through a five-stage pipeline: (1) **OData Service Creation**, (2) **OData Query Synthesis**, (3) **Service-Query Alignment**, (4) **Multi-style Question Synthesis**, and (5) **Chain-of-Thought Generation**. Each stage synergistically leverages LLMs (defaulting to GPT-4o) alongside automated pre- and post-processing strategies to ensure high-fidelity, diverse outputs while minimizing the need for human intervention. Distinct from prior data synthesis methodologies (Li et al., 2025), OmniOData introduces a novel *Self-Correcting Alignment Loop*. This mechanism dynamically synchronizes abstract metadata with procedural query logic, ensuring that generated samples are not merely syntactically plausible but fully executable in production-grade environments.

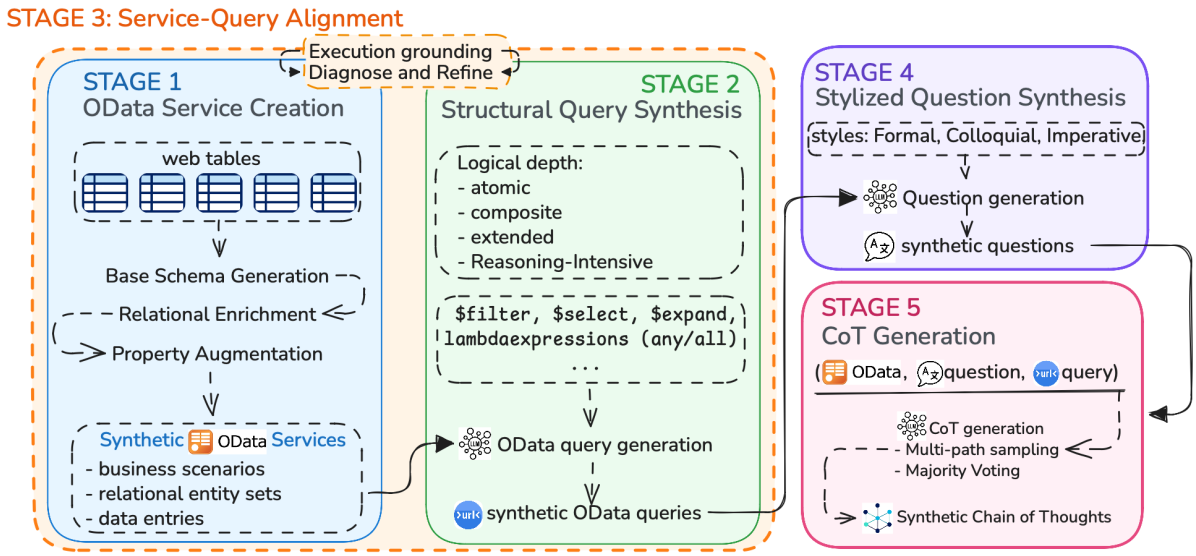


Figure 1: Overview of the `OmniOData` Framework. The architecture consists of five sequential stages designed to generate high-fidelity, execution-guaranteed OData datasets.

3.2 OData Service Creation

A primary challenge in the Text-to-OData task is the scarcity of publicly available, high-fidelity enterprise databases, which are typically shielded by strict privacy and security protocols (but see Klein et al. (2024) for an exception). To address this, we build upon the methodology proposed by Li et al. (2025) and develop a multi-phase pipeline to synthesize production-grade OData v4 services from open-domain tabular data.

Specifically, we transform flat web table schemas (Peeters et al., 2024) into complex relational structures via a three-phase enhancement process: **Base Schema Generation:** we extract EntityTypes from raw table columns, performing automated type inference and primary key designation to establish a foundational schema; **Relational Enrichment:** to mirror the complexity of industrial enterprise systems, we inject auxiliary entities—such as reference tables (e.g., `Categories`, `Statuses`), audit logs (e.g., `History`), and junction tables—to support many-to-many relationships; **Property Augmentation:** we enrich entity attributes by adding system metadata (e.g., `CreatedDate`, `ModifiedBy`), quantitative metrics (e.g., `Ratings`), and detailed semantic descriptions for each property. To ensure that the services are queryable across complex paths, we synthesize navigation properties bidirectionally, creating an explicit relationship graph between entities. Finally, we employ domain classification algorithms to contextualize each service within specific business verticals—such as healthcare, finance, or logistics—ensuring that the synthesized entities and properties are domain-appropriate. The synthesized services comprise three core components: **business context**, **service metadata**, and **synthetic records**. These services are containerized and deployed as OData v4 mock servers, establishing a deterministic backend environment for execution-based validation.

3.3 Structural Query Synthesis

While leveraging LLMs for synthetic data generation is an established practice, its application to the OData domain is particularly vital due to the scarcity of high-quality, publicly available enterprise datasets. Despite this scarcity, the extensive exposure of LLMs to diverse code and protocol specifications during pre-training renders them uniquely capable of synthesizing complex OData queries, as shown in experiments.

To ensure rigorous control over dataset quality and structural diversity, we perform a systematic syntactic analysis of the OData v4 specification. Based on this analysis, we define four distinct complexity levels: **Atomic**, **Composite**, **Extended**, and **Reasoning-Intensive**. This hierarchy, detailed in Table 5 (Appendix A.2), allows a granular evaluation of model performance across varying degrees of logical and relational depth.

To synthesize queries that are both syntactically valid and contextually relevant, we employ

a complexity-aware generation strategy. LLMs are provided with an informative prompt containing the service's structural constraints and the target logical depth. Specifically, the prompt for OData query synthesis incorporates the following structural components to ensure technical precision and logical variety: *Task Formalization, Schema Definition, Sample Grounding, Dynamic Function Library, Complexity-Specific Criteria, Validation Guardrails* (explanations and examples can be found in Appendix A.2.2).

3.4 Service-Query Alignment

While synthetic generation for relational databases often relies on standardized DDL and DML patterns (Li et al., 2025), OData service synthesis poses distinct challenges. These stem from the **semantic density** of the Common Schema Definition Language (CSDL) and the hierarchical nature of OData's Entity Data Model (EDM). Unlike SQL schemas, where constraints are often localized to table definitions, OData metadata requires LLMs to maintain long-range coherence across navigation properties, complex types, and service-side capabilities. Consequently, vanilla generation frequently results in **relational misalignment**—queries that are syntactically valid according to OData v4 grammar but executionally "orphaned" as they reference non-existent paths or mismatched property types.

To mitigate this, we propose an iterative **Self-Correcting Alignment Loop**. Rather than treating query generation as a single-pass transduction task, our framework adopts a "verify-and-refine" paradigm. The process leverages the mock server infrastructure as an execution-grounded oracle:

1. **Execution Grounding:** Every synthesized query is dispatched to a mock server instance. This provides an immediate "grounding signal" for the query's validity within its specific business context.
2. **Diagnostic Extraction:** Upon execution failure, the server's diagnostic logs are parsed to identify the root cause, such as invalid property resolution or unauthorized navigation traversal.
3. **Recursive Refinement:** A critic LLM ingests the failure trace alongside the service metadata to perform corrective actions. This includes *Intent Recalibration* (adjusting the query to honor schema limits) and *Metadata Augmentation* (dynamically evolving the mock service to support complex, reasoning-heavy logical jumps).

By establishing this rigorous alignment, we ensure the structural integrity and mutual consistency of both the OData services and their corresponding queries.

3.5 Stylized Question Synthesis

Given the diverse professional backgrounds and interaction patterns of enterprise users, a robust Text-to-OData system must exhibit high linguistic flexibility. Beyond structural validation, we prioritize the linguistic diversity of the natural language inputs by adopting the nine-style taxonomy proposed by Li et al. (2025) to simulate a comprehensive range of enterprise scenarios: *Formal* (standardized professional vocabulary), *Colloquial* (informal conversational phrasing), *Imperative* (direct functional commands), *Interrogative* (explicit information seeking), *Descriptive* (context-heavy narratives), *Concise* (minimalist shorthand), *Vague* (underspecified or ambiguous intent), *Metaphorical* (figurative expressions), and *Multi-turn* (context-dependent dialogue fragments). The prompt for synthesizing natural language questions has the following key components: *Style Conditioning, Enriched Entity Metadata, Structural Translation Guidelines, Dialogue State Management* (see Appendix A.2.3 for explanations and examples).

We employ a high-temperature sampling strategy during the generation phase to maximize linguistic diversity. While this promotes creative variation, it may occasionally introduce undesirable semantic shifts that deviate from the original OData intent. To mitigate this noise while preserving semantic consistency, we adopt a centroid-based selection strategy inspired by Rossiello et al. (2017) and Zhang et al. (2023). For each execution-validated query, we synthesize a candidate set of 3–5 variations across randomly sampled styles. We then encode these candidates into a high-dimensional vector space using a SentenceTransformer model (all-mpnet-base-v2) (Reimers and Gurevych, 2019) and identify the optimal question that minimizes the average cosine distance to all other variations in the stylistic cluster. This approach ensures that the selected question represents the semantic centroid of the cluster—effectively filtering out stylistic outliers and hallucinations while maintaining the linguistic variety necessary for robust model training.

3.6 Chain-of-Thought Generation with All-Property Context

The final step of our framework synthesizes reasoning chains that bridge natural language intent and OData syntax. We extract full property descriptions and navigation cardinalities from the service metadata. This allows the generator to justify property selection through business logic (e.g., return status vs. fine amounts) rather than simple keyword matching, ensuring the model learns the underlying relational reasoning.

We employ a three-stage architecture to synthesize reasoning chains that are both linguistically diverse and executionally sound:

1. **Structured Prompting:** The task is decomposed into a seven-step derivation: requirement analysis, entity set identification, predicate determination, property selection, expansion planning, query construction, and syntactic validation (examples in Appendix A.7).
2. **Multi-Path Sampling:** For each $\langle$service, question, query$\rangle$, we generate 5 reasoning variations triplet using high-temperature sampling to ensure logical diversity.
3. **Execution-based Majority Voting:** Rather than string-based consensus, we group reasoning chains by their *normalized execution results* from the mock server. The final CoT is sampled from the largest group whose execution output matches the ground-truth result.

Following generation, we validate all samples by comparing their execution outputs with ground truth and filter out mismatches and execution failures strictly. This rigorous filtering ensures that the resulting reasoning traces serve as high-fidelity supervision signals for instruction-tuning SLMs on complex, reasoning-intensive relational navigation tasks.

4 `OmniOData-R1`: State-of-the-art Text-to-OData SLM

4.1 Problem Formulation

Given a dataset with n samples, $\mathcal{D} = \{(q_i, s_i, r_i)\}$ where q_i is a natural language question, s_i is an OData service schema, and r_i is a reference reasoning chain with query, our goal is to train a policy $\pi_\theta(r|q, s)$ that generates valid OData queries through step-by-step reasoning. The policy receives as input: (1) natural language question q, (2) complete service metadata s, including all entity types, properties, navigation properties, and descriptions, and (3) optional few-shot examples. The output is a structured response containing:

```
<think>[Step-by-step reasoning explaining query
construction]</think>
<answer>[OData query URI]</answer>
```

4.2 Data Preparation

Utilizing the `OmniOData` framework, we curate SynOData, a high-fidelity corpus comprising more than 110K execution-guaranteed samples across 682 unique services. To ensure methodological rigor during training and evaluation, we implement *Service-Based Splitting* strategy, which prevents data leakage. Details of SynOData can be found in Appendix A.3.

From the primary SynOData corpus, we derive three specialized subsets through stratified sampling based on complexity: SynOData-10K, SynOData-5K, and SynOData-Eval, corresponding to our Supervised Fine-Tuning (SFT), RL, and evaluation phases, respectively. It is important to note that the RL subset SynOData-5K is specifically curated to include only extended and reasoning-intensive samples to facilitate complex logic discovery. In contrast, the SFT subset SynOData-10K and the evaluation subsets SynOData-Eval maintain a distribution consistent with the overall SynOData population.

4.3 Training

We adopt a two-stage training regime: SFT on the SynOData-10K subset, followed by RL using GRPO (Guo et al., 2025) on the SynOData-5K reasoning-intensive set. GRPO is particularly effective for the execution-grounded OData synthesis task, as it leverages the discrete execution signals from the mock server without the instability of a separate critic model.

For a given natural language question q and service schema s, the policy π_θ generates a group of G outputs $\{o_i\}_{i=1}^{G}$ from the old policy $\pi_{\theta_{\text{old}}}$. The GRPO surrogate objective $J(\theta)$ is maximized according to the following formulation:

$$J(\theta) = \mathbb{E}\Bigg[\frac{1}{G}\sum_{i=1}^{G}\min\Bigg(\frac{\pi_\theta(o_i \mid q,s)}{\pi_{\theta_{\text{old}}}(o_i \mid q,s)}\hat{A}_i, \\ \text{clip}\left(\frac{\pi_\theta(o_i \mid q,s)}{\pi_{\theta_{\text{old}}}(o_i \mid q,s)}, 1-\epsilon, 1+\epsilon\right)\hat{A}_i\Bigg) \\ - \beta D_{\text{KL}}(\pi_\theta \,\|\, \pi_{\text{ref}})\Bigg]. \tag{1}$$

where the advantage $\hat{A}_i = (r_i - \text{mean}\{r_k\})/(\text{std}\{r_k\} + \epsilon)$. By utilizing group-relative baselines, GRPO eliminates the need for a separate critic network, reducing VRAM overhead while grounding the model's stochastic reasoning traces in the deterministic execution signals of the OData mock server.

Reward Design Our composite reward function $\mathcal{R}(r, q, s) = R_{\text{fmt}} + R_{\text{len}} + R_{\text{exe}} + R_{\text{res}}$ balances structural integrity with execution-based grounding.

- **Format Compliance (R_{fmt}):** Enforces XML encapsulation. $R_{\text{fmt}} = 0.5$ if reasoning and query are correctly enclosed in `<think>` and `<answer>` tags; otherwise -1.0.
- **Reasoning Density (R_{len}):** Encourages concise yet sufficient logic via a Gaussian penalty centered around the average length of the thinking trace ($\mu = 2320$):

$$R_{\text{len}} = A \cdot \exp\left(-\frac{z^2}{2}\right), \quad z = \frac{|r| - \mu}{\sigma}. \quad (2)$$

  The amplitude coefficient $A = 2.0$ is chosen to provide a meaningful gradient signal while remaining subordinate to the correctness reward.
- **Execution Grounding (R_{exe}):** Provides direct feedback from the mock server. $R_{\text{exe}} = +1.0$ for successful responses; -0.5 for syntax or schema-level failures.
- **Semantic Correctness (R_{res}):** The primary signal ($+4.0$) for matching ground-truth results (data entries), otherwise 0. All the returned data entries are normalized (canonicalized and sorted) before comparison to ensure semantic equivalence.

5 Experiments and Results

5.1 Implementation Details

Model Configuration. Our primary models, `OmniOData-R1`, are initialized from the Qwen2.5-Coder series (Hui et al., 2024), specifically the 1.5B and 3B parameter variants. All experiments were conducted on an AWS EC2 p4de.24xlarge instance with 8× NVIDIA A100 GPUs (80GB VRAM each). The complete training recipe are documented in Appendix A.4.

Evaluation Benchmarks. The evaluations are conducted on two benchmarks: SynOData-Eval and RealOData. **SynOData-Eval** A held-out test set of 5,000 samples from SynOData, as described in Section 4. RealOData is a realistic industrial benchmark of 2,000 samples across diverse business scenarios, provided in two variants to test sensitivity to *context density*: (i) **Hard**, retaining full, production-level metadata, and (ii) **Lite**, a context-optimized version with redundant metadata pruned randomly.

Baselines. We compare against a diverse set of widely used proprietary and open-source language models under zero-shot evaluation, including Gemini-3-Flash, Gemini-3-Pro, GPT-4o-mini, GPT-4o, DeepSeek-R1-32B, DeepSeek-R1-70B, GPT-OSS-120B, Qwen2.5-1.5B, and Qwen2.5-3B. We do not include specialized Text-to-SQL systems as baselines, since preliminary exploration suggests that they transfer poorly to OData-specific, metadata-dependent operators such as $expand$, $apply$, and lambda predicates. Moreover, establishing a fair comparison would require substantial adaptation beyond the scope of this work.

Metrics. The primary metric is **Execution Accuracy (ExAcc)**, defined as the percentage of generated queries that return semantically equivalent results to the ground truth after normalization. To ensure semantic equivalence, we implement a strict normalization protocol:

1. **Canonicalization:** Removing metadata fields and transient properties.
2. **Order Invariance:** Sorting all JSON arrays and object keys to ensure consistent comparison.
3. **Comparison:** Exact match verification on the resulting canonical JSON structures.

5.2 Main Results

The empirical results summarized in Table 1 demonstrate a clear performance hierarchy across specialized OData synthesis tasks, illustrating the profound impact of domain-specific alignment. The primary breakthrough is the performance of the OmniOData-R1 series; notably, the 3B variant achieves the best performance across all benchmarks, reaching 33.84% accuracy on the RealOData-Hard set. This result is particularly significant, as it not only eclipses massive open-source baselines like `GPT-OSS-120B` (24.53%), but also outperforms frontier proprietary systems, such as `GPT-4o` (33.33%) and `Gemini-3-Pro` (22.50%). Even the 1.5B variant demonstrates remarkable performance, achieving 27.75% accuracy on the same hard benchmark and effectively outperforming general-purpose models nearly fifty times its

Model	SYN-EVAL	REALODATA	
		Lite	Hard
Proprietary Models (Zero-shot)			
Gemini-3-Flash	43.25%	27.07%	26.90%
Gemini-3-Pro	37.85%	22.50%	22.50%
GPT-4o-mini	42.85%	33.33%	28.43%
GPT-4o	39.50%	32.66%	33.33%
Open-Source Models (Zero-shot)			
DeepSeek-R1-32B	29.60%	27.58%	28.60%
DeepSeek-R1-70B	32.30%	22.67%	27.92%
GPT-OSS-120B	38.10%	25.21%	24.53%
Qwen2.5-1.5B	0.00%	0.00%	0.00%
Qwen2.5-3B	3.10%	0.00%	0.00%
SFT Only			
Qwen2.5-1.5B (SFT)	35.30%	32.83%	22.50%
Qwen2.5-3B (SFT)	45.45%	35.36%	15.91%
SFT + GRPO (Our Approach)			
OmniOData-R1 (1.5B)	42.90%	34.35%	27.75%
OmniOData-R1 (3B)	**50.55%**	**36.04%**	**33.84%**

Table 1: Execution accuracy (ExAcc) on synthetic and realistic industrial OData benchmarks. SYN-EVAL is short for SYNODATA-EVAL. REALODATA variants represent production-level complexity with varying context density.

size, such as `DeepSeek-R1-70B`. More results and discussions can be found in Appendix A.5.

These findings confirm that for Text-to-OData, high-fidelity synthetic data is more impactful than raw parameter scaling. By providing execution-grounded reasoning traces, SYNODATA bridges the critical data-scarcity gap in enterprise protocols. This specialized alignment enables our SLMs to master complex relational mappings that are absent from general-purpose corpora, achieving frontier-level accuracy with minimal computational overhead.

6 Conclusion

This paper introduced OMNIODATA, a framework for advanced OData query generation with small language models, enabled by execution-guaranteed synthetic data and reinforcement learning. To address the lack of training data for relational API reasoning, we release SYNODATA, the first large-scale, verified corpus for this task. Building on SYNODATA, we develop OMNIODATA-R1, a family of 1.5B and 3B models trained with SFT and GRPO. Experimental results on both SYNODATA-EVAL and the industrial REALODATA benchmark show that OMNIODATA-R1 outperforms strong proprietary and open-source baselines, with the 3B variant achieving the best overall performance across all evaluation settings. Overall, our findings demonstrate that high-fidelity synthetic supervision together with execution-based reinforcement learning can make compact models highly effective for structurally grounded OData reasoning.

At the same time, this study has several limitations. Our evaluation focuses on OData query generation and does not yet cover broader API interaction settings such as multi-turn clarification, cross-service composition, or downstream tool use. In addition, although REALODATA reflects realistic industrial scenarios, its scale and domain coverage remain inherently limited.

References

Peter Belcak, Greg Heinrich, Shizhe Diao, Yonggan Fu, Xin Dong, Saurav Muralidharan, Yingyan Celine Lin, and Pavlo Molchanov. 2025. Small language models are the future of agentic ai. *arXiv preprint arXiv:2506.02153*.

Mehdi Ben Amor, Alexis Strappazzon, Michael Granitzer, Elöd Egyed-Zsigmond, and Jelena Mitrović. 2025. Instruct-to-sparql: A text-to-sparql dataset for training sparql agents. In *Proceedings of the 2025 ACM SIGIR Conference on Human Information Interaction and Retrieval*, CHIIR '25, page 390–395. Association for Computing Machinery.

Qifeng Cai, Hao Liang, Chang Xu, Tao Xie, Wentao Zhang, and Bin Cui. 2025. Text2sql-flow: A robust sql-aware data augmentation framework for text-to-sql. *arXiv preprint arXiv:2511.10192*.

Jiaxuan Gao, Wei Fu, Minyang Xie, Shusheng Xu, Chuyi He, Zhiyu Mei, Banghua Zhu, and Yi Wu. 2025. Beyond ten turns: Unlocking long-horizon agentic search with large-scale asynchronous rl. *arXiv preprint arXiv:2508.07976*.

Google DeepMind. 2026. Gemini 3: Next-generation multimodal intelligence. https://blog.google/technology/ai/gemini-3-announcement/. Accessed: 2026-02-08.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-R1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Zhicheng Guo, Sijie Cheng, Hao Wang, Shihao Liang, Yujia Qin, Peng Li, Zhiyuan Liu, Maosong Sun, and Yang Liu. 2024. Stabletoolbench: Towards stable large-scale benchmarking on tool learning of large language models. *Preprint*, arXiv:2403.07714.

Zijin Hong, Zheng Yuan, Qinggang Zhang, Hao Chen, Junnan Dong, Feiran Huang, and Xiao Huang. 2025. Next-generation database interfaces: A survey of llm-based text-to-sql. *IEEE Transactions on Knowledge and Data Engineering*.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, and 1 others. 2024. Qwen2. 5-coder technical report. *arXiv preprint arXiv:2409.12186*.

Tassilo Klein, Clemens Biehl, Margarida Costa, Andre Sres, Jonas Kolk, and Johannes Hoffart. 2024. SALT: Sales autocompletion linked business tables dataset. In *NeurIPS 2024 Third Table Representation Learning Workshop*.

Hung Le, Yue Wang, Akhilesh Deepak Gotmare, Silvio Savarese, and Steven Chu Hong Hoi. 2022. Coderl: Mastering code generation through pretrained models and deep reinforcement learning. *Advances in Neural Information Processing Systems*, 35:21314–21328.

Fangyu Lei, Jixuan Chen, Yuxiao Ye, Ruisheng Cao, Dongchan Shin, Hongjin Su, Zhaoqing Suo, Hongcheng Gao, Wenjing Hu, Pengcheng Yin, and 1 others. 2024. Spider 2.0: Evaluating language models on real-world enterprise text-to-sql workflows. *arXiv preprint arXiv:2411.07763*.

Haoyang Li, Shang Wu, Xiaokang Zhang, Xinmei Huang, Jing Zhang, Fuxin Jiang, Shuai Wang, Tieying Zhang, Jianjun Chen, Rui Shi, and 1 others. 2025. Omnisql: Synthesizing high-quality text-to-sql data at scale. *arXiv preprint arXiv:2503.02240*.

Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, and 1 others. 2024. Can LLM already serve as a database interface? a big bench for large-scale database grounded text-to-sqls. *Advances in Neural Information Processing Systems*, 36.

OpenAI. 2024. Hello GPT-4o. https://openai.com/index/hello-gpt-4o/. Accessed: 2026-02-08.

OpenAI. 2025. gpt-oss-120b & gpt-oss-20b model card. *Preprint*, arXiv:2508.10925.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Makbule Gulcin Ozsoy, Leila Messallem, Jon Besga, and Gianandrea Minneci. 2025. Text2cypher: Bridging natural language and graph databases. In *Proceedings of the Workshop on Generative AI and Knowledge Graphs (GenAIK)*, pages 100–108.

Ralph Peeters, Alexander Brinkmann, and Christian Bizer. 2024. The web data commons schema.org table corpora. In *Proceedings of the ACM Web Conference 2024*, page 4795–4804. Association for Computing Machinery.

Federico Pennino, Bianca Raimondi, Massimo Rondelli, Andrea Gurioli, and Maurizio Gabbrielli. 2025. From reasoning to code: Grpo optimization for underrepresented languages. *arXiv preprint arXiv:2506.11027*.

Mohammadreza Pourreza and Davood Rafiei. 2024. DTS-SQL: Decomposed text-to-sql with small large language models. *arXiv preprint arXiv:2402.01117*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Gaetano Rossiello, Pierpaolo Basile, and Giovanni Semeraro. 2017. Centroid-based text summarization through compositionality of word embeddings. In *Proceedings of the MultiLing 2017 Workshop on Summarization and Summary Evaluation Across Source Types and Genres*, pages 12–21. Association for Computational Linguistics.

Songjun Tu, Jiahao Lin, Qichao Zhang, Xiangyu Tian, Linjing Li, Xiangyuan Lan, and Dongbin Zhao. 2025. Learning when to think: Shaping adaptive reasoning in r1-style models via multi-stage RL. *arXiv preprint arXiv:2505.10832*.

Fali Wang, Zhiwei Zhang, Xianren Zhang, Zongyu Wu, TzuHao Mo, Qiuhao Lu, Wanjing Wang, Rui Li, Junjie Xu, Xianfeng Tang, Qi He, Yao Ma, Ming Huang, and Suhang Wang. 2025. A comprehensive survey of small language models in the era of large language models: Techniques, enhancements, applications, collaboration with llms, and trustworthiness. *ACM Trans. Intell. Syst. Technol.*, 16(6).

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Yi Zhang, Jan Deriu, George Katsogiannis-Meimarakis, Catherine Kosten, Georgia Koutrika, and Kurt Stockinger. 2023. Sciencebenchmark: A complex real-world benchmark for evaluating natural language to sql systems. *arXiv preprint arXiv:2306.04743*.

A Appendix

A.1 OData Preliminaries and Syntax Taxonomy

This section provides a formal definition of the OData v4 syntax scope utilized in the development of OMNIODATA and the curation of the SYNODATA corpus. OData (Open Data Protocol) is an OASIS standard that abstracts relational database operations into RESTful URI strings. The challenge for Text-to-OData synthesis lies in mapping natural language to a rigid, path-based hierarchical structure that requires both syntactic precision and schema-aware relational reasoning.

A.1.1 OData Query Options

OData query options (prefixed with $) define the primary transformation applied to the requested resource. These options act as the functional equivalents of high-level SQL clauses. As summarized in Table 2, our framework supports the full suite of OData shaping operations, including complex server-side aggregations.

Option	Relational Function and Description
`$filter`	Restricts the result set based on boolean expressions (`WHERE`).
`$select`	Projects specific properties to optimize the payload (`SELECT`).
`$expand`	Performs relational joins across navigation properties (`JOIN`).
`$apply`	Enables data transformations and aggregations (`GROUP BY`).
`$orderby`	Controls result sorting in ascending or descending order.
`$top/$skip`	Implements pagination logic and result set offsets.
`$count`	Returns the total number of matched items in the metadata.
`$search`	Executes full-text search across multiple entity properties.

Table 2: Taxonomy of OData System Query Options.

A.1.2 Predicate Logic and Canonical Functions

OData supports a rich set of built-in functions within the `$filter` clause to handle data-type-specific logic. These functions allow for nuanced queries that go beyond simple equality, such as regex matching or temporal extraction. Table 3 details the categories and operators available for predicate synthesis.

Category	Supported Syntax and Operators
Comparison	`eq, ne, gt, ge, lt, le`
Logical	`and, or, not`
String	`contains, startswith, endswith, length, tolower, toupper, trim, concat, substring, indexof`
Advanced	`matches` (Regex), `replace, padleft, reverse, isempty, normalize`
Date/Time	`year, month, day, hour, minute, second`
Math	`round, floor, ceiling`

Table 3: Canonical Functions and Filter Operators.

A.1.3 Relational Navigation and Collection Logic

The most reasoning-intensive component of the OData protocol involves traversing schema relationships. Our dataset evaluates the model's ability to navigate nested paths and apply collection-level logic through Lambda expressions. Table 4 outlines the operations used to test these capabilities.

Operation	Functional Definition
`any`	Returns true if at least one collection member matches the predicate.
`all`	Returns true only if every member of the collection matches.
`in`	Checks if a property value exists within a literal set or array.
One-to-One	Navigation to a related single entity (e.g., `Profile/Photo`).
Deep Nav	Path traversal up to five relational levels (e.g., `A/B/C/D/E`).

Table 4: Relational Navigation and Collection Operations.

A.2 Extended Explanations of `OmniOData`

A.2.1 OData Service Creation

From flat web tables, we obtained the 682 initial OData services into a high-fidelity relational corpus, specifically engineered to bridge the data-scarcity gap in structured API reasoning. By expanding the total entity count by 2.18× and properties by 3.39×, the framework transitioned from sparse, two-entity services to dense, enterprise-grade environments. Most critically, navigation properties—the fundamental building blocks for relational joins—increased 3.15×, shifting the median from 2 to 7 per service. This structural scaling effectively eliminated all "Atomic" complexity classifications, resulting in a pool where 60% of services are now classified as "Extended" (score ≥ 50). To support execution-grounded training, record volume was simultaneously scaled by 10–25× per entity, providing the necessary depth for validating advanced operations such as Lambda expressions and data aggregations.

A.2.2 Structural Query Sythesis

Query Taxonomy. The synthesis process is governed by a hierarchical taxonomy designed to evaluate the limits of relational reasoning in Small Language Models (SLMs). As detailed in Table 5, we categorize the SynOData corpus into four complexity tiers, ranging from basic property projection to multi-level relational aggregation.

Query Generation Prompt. The synthesis of the SynOData corpus relies on a multi-

Logical Depth	Structural Operations	Illustrative Example (OData URI)	Natural Language Intent
Atomic	Elementary $filter, $select, pagination.	/Products?\protect\T1\textdollarfilter=Pricegt100&\protect\T1\textdollarselect=Name,Price&\protect\T1\textdollartop=10	List the name and price of the first 10 products costing over 100.
Composite	Boolean logic, string functions, single $expand.	/Orders?\protect\T1\textdollarfilter=Statuseq'Pending'andOrderDategt2024-01-01&\protect\T1\textdollarexpand=Customer	Get pending orders from 2024 onwards, including the customer's profile.
Extended	Multi-level $expand, lambda expressions (any/all).	/Customers?\protect\T1\textdollarfilter=Orders/any(o:o/Statuseq'Shipped')&\protect\T1\textdollarexpand=Orders(\protect\T1\textdollarexpand=Items)	Find customers with at least one shipped order, including their orders and line items.
Reasoning-Intensive	$\geq$ 2 level $expand, $apply (aggregation).	/Orders?\protect\T1\textdollarexpand=Customer,Items&\protect\T1\textdollarapply=groupby((Status),aggregate(\protect\T1\textdollarcountasTotal))	Group all orders by status to show the total count, while including customer and item details.

Table 5: Taxonomy of OData query complexity based on logical depth. We categorize queries by their structural requirements and natural language intent to evaluate relational reasoning capabilities.

component prompt architecture designed to simulate the constraints of real-world enterprise environments. By decoupling schema metadata from logical complexity constraints, we ensure that the generated queries are both execution-guaranteed and representative of production-level depth. The following elements constitute the core structural blocks of our generation prompts:

- **Task Formalization:** Defines the objective of generating an executable OData v4 query adhering to strict syntactic and metadata constraints.
- **Schema Definition:** A flattened representation of the service metadata, comprising `EntityType` specifications (keys and typed properties) and `NavigationProperty` mappings with explicit cardinality (e.g., *MainEntity.Name* $\rightarrow$ *NavigationProperty.Name*).
- **Sample Grounding:** A subset of representative records (up to 5 per entity) providing realistic literal values for filter predicates and analytical targets.
- **Dynamic Function Library:** A complexity-weighted selection of OData functions—ranging from basic comparison operators for *Atomic* queries to advanced collection and math functions for *Reasoning-Intensive* levels.
- **Complexity-Specific Criteria:** Explicit logical constraints that dictate the required relational depth, allowed operators (e.g., `$expand`, `$apply`), and business context to ensure cross-domain alignment.
- **Validation Guardrails:** A set of five core OData v4 compliance rules focusing on metadata synchronization, meaningful result set generation, and URL naming conventions.

A.2.3 Stylized Question Generation

To bridge the gap between rigid URI syntax and the diversity of human language, we employ a style-conditioned generation pipeline. This stage transforms the ground-truth OData queries into natural language questions across multiple linguistic dimensions. The prompt architecture for this phase is structured as follows:

- **Style Conditioning:** A target linguistic descriptor selected from our nine-style taxonomy (e.g., *Metaphorical*, *Vague*, or *Multi-turn*). This ensures the generated text captures the specific communicative nuances required for robust model training.
- **Enriched Entity Metadata:** A detailed extraction of the `entity_info` derived from resolved property anchors. This includes property descriptions, primitive data types, and navigation context, providing the LLM with the necessary semantic grounding to describe technical OData entities in natural language.
- **Structural Translation Guidelines:** Style-dependent instructions that dictate the complexity of the translation. For standard styles, we utilize guidelines without external knowledge dependencies, whereas *Vague* and *Metaphorical* styles utilize extended reasoning steps to inject required external semantic context.
- **Dialogue State Management:** For *Multi-turn* variations, the prompt includes specific formatting directives to simulate iterative user-system

exchanges, effectively decomposing complex queries into sequential dialogue acts.

A.3 Dataset Details

The global corpus consists of more than 110K execution-validated OData queries distributed across 682 unique service schemas. To ensure the model's capacity for out-of-distribution generalization, we employed a service-based partitioning strategy with a 90/10 split ratio. This approach ensures that the evaluation environment remains entirely isolated at the schema level, preventing metadata leakage during the learning process.

The training pool comprises 100K samples derived from 614 services, while the test pool contains 10K samples from 68 services. For standardized benchmarking, a primary evaluation subset (SYNODATA-EVAL) of 2K samples was extracted from the test pool, with the remaining 9K samples held in reserve for additional validation.

A.3.1 Training Subset Allocations

From the primary training pool, specific subsets were isolated to support a multi-stage optimization pipeline. A rigorous overlap verification protocol confirmed zero commonality between SFT, Validation, and RL-specific datasets, ensuring that reasoning traces were not contaminated by samples seen during the supervised phase. The SFT dataset SYNODATA-10K follows a natural complexity distribution of approximately 30% *Atomic*, 28% *Composite*, 27% *Extended*, and 15% *Reasoning-Intensive*. In contrast, the *RL* dataset (SYNODATA-5K) priortizes on *Extended* and *Reasoning-Intensive* queries.

A.3.2 Data Integrity and Leakage Verification

To maintain the statistical significance of the results, a deduplication and isolation check was performed across all data partitions. Service-level isolation was verified at 100%, ensuring no overlap between training and testing schemas. Furthermore, sample-level isolation between SFT and RL phases was confirmed at 0% overlap to prevent the model from memorizing training targets.

A.4 Training Recipe

The optimization of our models was conducted in two sequential phases: initial alignment via Supervised Fine-Tuning (SFT) and subsequent reasoning enhancement through Reinforcement Learning (RL). Both the 1.5B and 3B parameter variants were trained using a unified hyperparameter framework to ensure architectural consistency.

A.4.1 Stage 1: Supervised Fine-Tuning (SFT)

The SFT phase utilized full-parameter tuning on the SYNODATA-10K dataset. Models were initialized from the Qwen2.5-Coder-Instruct series and trained for 3 epochs. We employed a learning rate of 5×10^{-6} with a cosine decay scheduler and a weight decay of 0.1. To accommodate the long-context requirements of OData metadata, the maximum sequence length was set to 4096 tokens. The training was distributed across an 8-GPU cluster using DeepSpeed ZeRO-3 with BF16 precision, utilizing a per-device batch size of 8 and a gradient accumulation step of 2.

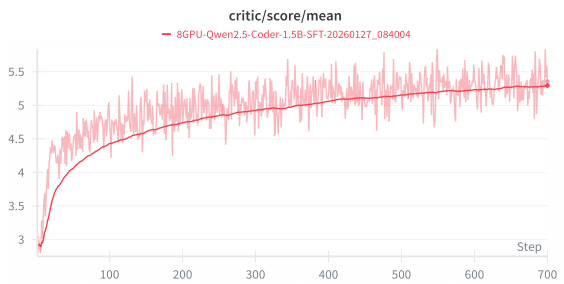
A.4.2 Stage 2: Reinforcement Learning (GRPO)

Following SFT, we employed Group Relative Policy Optimization (GRPO) to refine the models' structural reasoning and syntactic precision on SYNODATA-5K. This stage spanned 10 epochs with a reduced learning rate of 1×10^{-6}. The RL configuration utilized a rollout sample size of $n = 16$ per prompt, with the vLLM engine serving as the generation backbone at a temperature of 1.1.

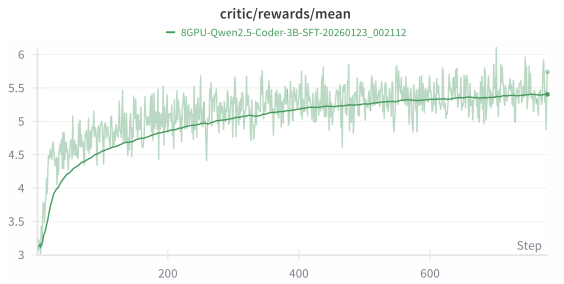
To support the internal reasoning traces, the maximum response length was capped at 2048 tokens. Policy stability was maintained via a `low_var_kl` loss type, with the KL coefficient set at 0.01 for the 1.5B model and 0.001 for the 3B model. For memory efficiency, the 3B model utilized a Tensor Parallel (TP) size of 2, while the 1.5B variant operated with a TP size of 1. Both models were optimized on the same 8-GPU hardware configuration with a targeted GPU memory utilization of 0.5. Visualization of training process is shown in Figure 2.

A.5 Experimental Results

The experimental results across the three benchmarks (see Figure 3) identify a persistent performance hierarchy, where the transition from *Atomic* to *Reasoning-Intensive* tiers marks a significant "complexity ceiling" for general-purpose models. In the *Atomic* tier, which requires basic projections and single-entity filtering, we observe high syntactic proficiency across most modern LLMs. For instance, even the baseline GEMINI-3-FLASH achieves a score of 64.06% on SYNODATA-EVAL, suggesting that mapping simple natural language at-



(a) OmniOData-R1(1.5B)



(b) OmniOData-R1(3B)

Figure 2: Visualization of GRPO Training Process.

tributes to metadata properties is largely within the emergent capabilities of large-scale pre-training.

However, as structural depth increases in the *Composite* and *Extended* tiers, a sharp relational gap emerges. While closed-source models like Gemini-3 show substantial performance decay when navigating multi-level expansions, our fine-tuned QWEN2.5-3B-GRPO maintains a remarkably stable performance curve, scoring 46.02% and 47.12% respectively. This stability indicates that the group-relative reinforcement learning phase successfully instilled a stateful understanding of path-based navigation, allowing the model to treat relational joins as consistent logical operations rather than sporadic syntactic hurdles.

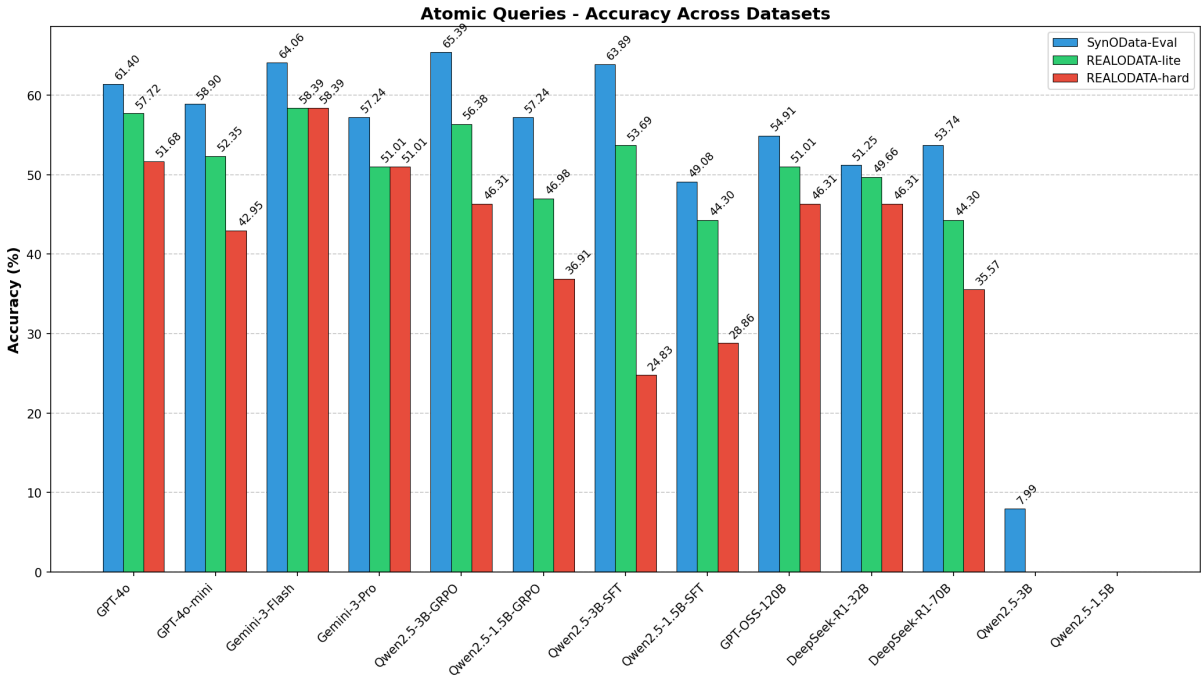
The *Reasoning-Intensive* tier, which necessitates complex `$apply` aggregations and ≥ 2 levels of nesting, serves as the ultimate stress test. Here, the specialized QWEN2.5-3B-GRPO demonstrates a distinct "reasoning premium", outperforming its SFT counterpart by 7.11%. This superiority is further validated in the REALODATA-HARD benchmark, where the 3B-GRPO model achieves its highest categorical score of 43.06% in the most complex tier. Such results prove that the use of reasoning traces allows the model to "plan" intricate structural transformations, bridging the gap between isolated data points and the high-fidelity relational networks required for authentic enterprise environments.

A.6 Inverse Accuracy Trends and Model Calibration

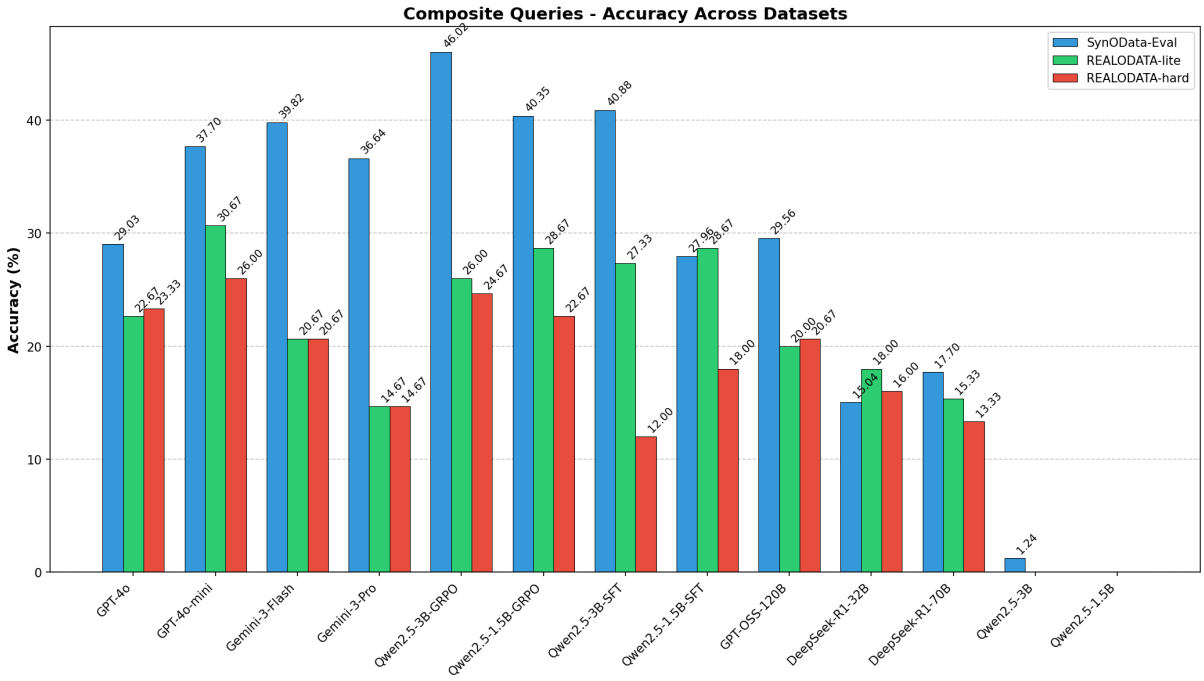
We notice a counter-intuitive accuracy discrepancy across the REALODATA benchmark tiers, as shown in Figure 3d. Specifically, reasoning-intensive models, such as the DEEPSEEK-R1 series (32B and 70B), exhibit higher precision on REALODATA-Hard compared to the REALODATA-Lite subset. In contrast, general-purpose frontier models, exemplified by the GEMINI-3 series, follow a traditional performance decay curve where accuracy diminishes as structural complexity increases.

An investigation into the models' reasoning process and final URI outputs suggests two distinct failure modes. The GEMINI-3 series frequently demonstrates a "conservative fallback" behavior; when the model fails to explicitly recognize a complex aggregation intent within the natural language prompt, it defaults to a syntactically safe but semantically incomplete `$filter` or `$select` clause. This results in a high success rate for simple queries but catastrophic failure on complex ones.

Conversely, the DEEPSEEK-R1 series suffers from what we term *over-reasoning* on low-complexity tasks. Because these models are optimized for multi-step chain-of-thought, they tend to inject unnecessary structural transformations or nested expansions into simple queries where a flat URI would suffice. However, this same tendency becomes a competitive advantage on REALODATA-Hard, where the high intrinsic complexity of the task finally aligns with the model's exhaustive reasoning depth. This suggests that while specialized reasoners are more robust against "complexity ceilings", they currently lack the calibration necessary to distinguish between trivial and reasoning-intensive API requests.

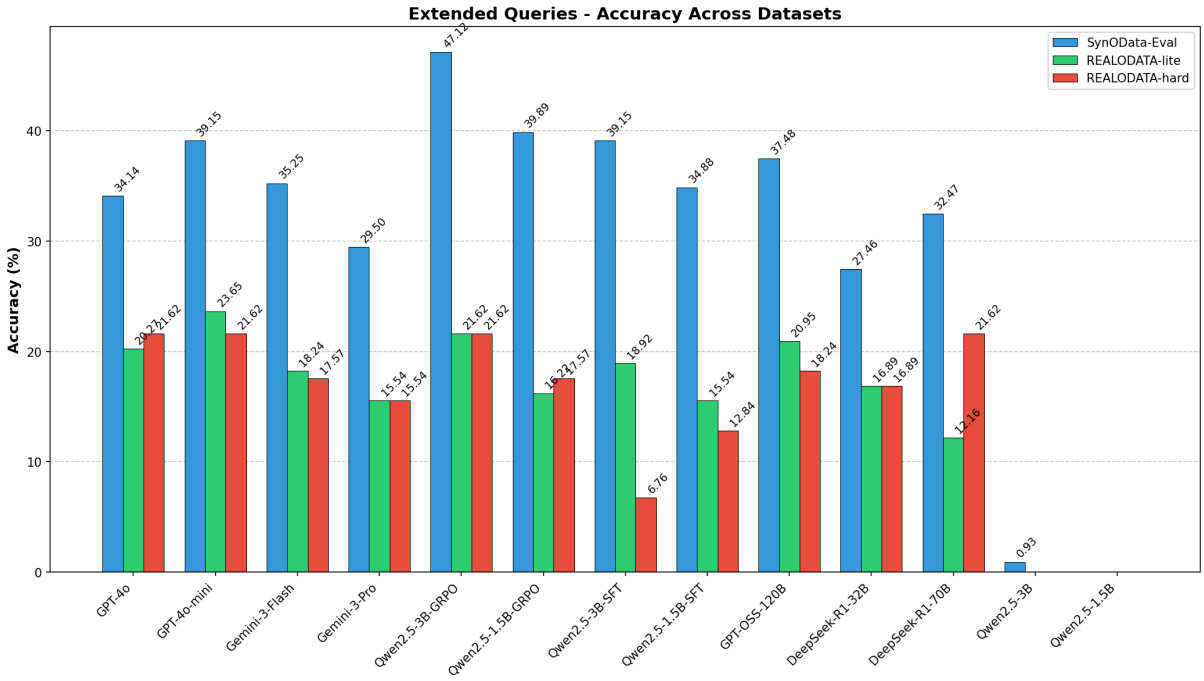


(a) Evaluation results on atomic queries.

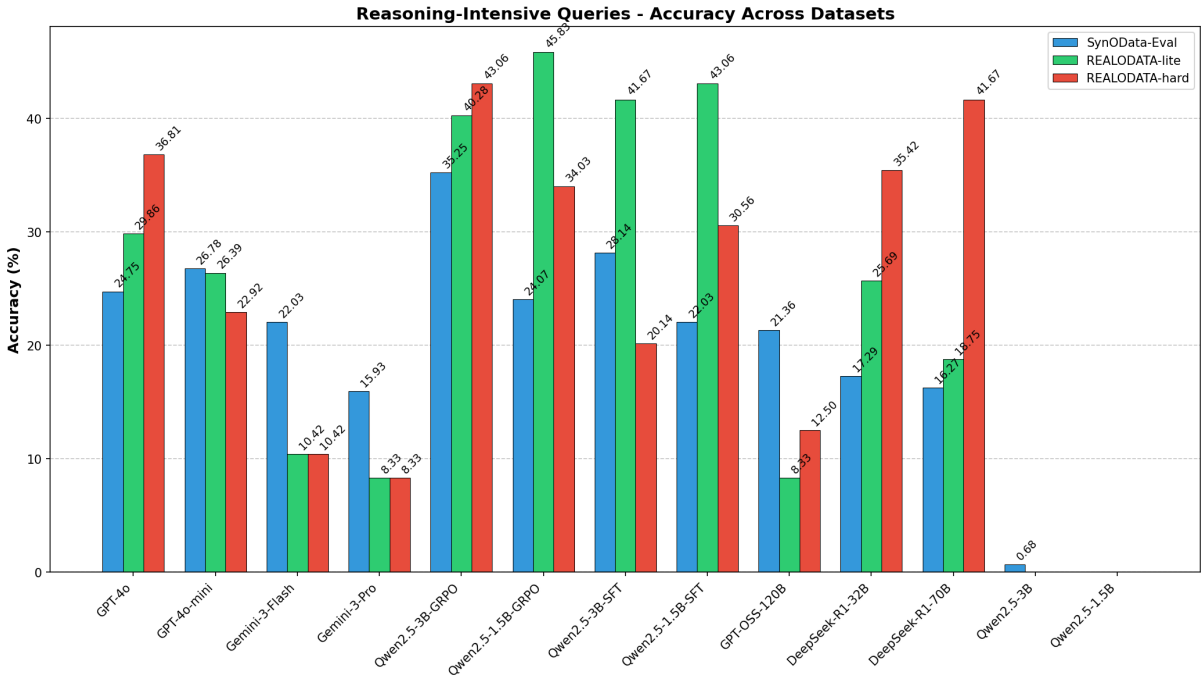


(b) Evaluation results on composite queries.

Figure 3: Evaluation results by complexity across three datasets (part 1/2)



(c) Evaluation results on extended queries.



(d) Evaluation results on reasoning-intensive queries.

Figure 3: Evaluation results by complexity across three datasets (part 2/2)

A.7 Prompt Templates

Task Overview
Create an executable OData v4 query based on the provided service metadata and sample data.

OData Service Metadata
Entity Types: `{entity_types}`
Navigation Properties: `{navigation_properties}`

Sample Data
`{sample_data_context}`

Available OData Functions
`{selected_odata_functions}`

Query Complexity Level: `{complexity}`
`{complexity_criteria}`

Business Context
Generate a query that addresses a realistic business scenario relevant to the data domain.

- Data filtering and searching
- Performance reporting and analytics
- Data relationships and navigation
- Aggregation and grouping operations

Output Format Requirements
Provide the complete OData URL query:
`/EntitySet?$filter=...&$select=...&$expand=...`

OData v4 Requirements

1. Use proper OData v4 syntax and operators.
2. Ensure all referenced entities and properties exist in the metadata.
3. Use appropriate functions for the specified complexity level.
4. Create queries that return meaningful business results.
5. Follow OData URL conventions and encoding.

Answer
Based on the provided metadata and sample data, I'll create a `{complexity}` OData query:

Figure 4: The structured prompt template utilized for the synthesis of executable OData queries, featuring modular injection points for metadata, sample grounding, and complexity constraints.

Task Overview
Your task is to create a high-quality natural language question based on a given OData v4 query and other contextual information.

Question Style
The natural language question should follow this style:
`{style_desc}`

OData Service Version
`{odata_version}`

Entity Property Information
Below are entity properties and their corresponding descriptions from the OData service metadata:
`{entity_info}`

OData Query
Given OData v4 query:

`{odata_query}`

Reasoning Steps
`{steps}`

Guidelines
`{guidelines}`

Output Format
`{output_format}`

Instructions
`{instruction}`

Figure 5: The prompt template for stylized question generation. This stage utilizes the ground-truth OData query, property-level metadata, and specific linguistic guidelines to synthesize diverse natural language inquiries.

System Instruction
You are a senior API analyst specializing in OData v4. Your task is to demonstrate step-by-step reasoning for translating a natural language question into an executable OData query. You are provided with a CORRECT, VALIDATED OData query as the target solution. Your reasoning should show the logical thought process...

Context Injections
[OData Service Metadata]: `{service_metadata}`
[Available Properties and Their Descriptions]: `{property_descriptions}`
[Natural Language Question]: `{question}`
[Target OData Query]:

`{odata_query}`

Reasoning Framework (Step-by-Step)

1. **Understand the Business Requirements:** Identify entities, relationships, and data needs.
2. **Identify the Primary Entity Set:** Select the starting point based on metadata.
3. **Determine Filtering Requirements:** Construct the `$filter` clause with appropriate operators.
4. **Select Required Properties:** Use `$select` to optimize the data payload.
5. **Plan Entity Expansions:** Identify navigation properties for `$expand` operations.
6. **Construct the Complete Query:** Assemble components and verify syntactic correctness.
7. **Validate Against Requirements:** Ensure the query is performant and answers the prompt.

Key Learning Points
Highlight important OData concepts demonstrated: Query option usage, navigation traversal, and performance considerations.

Instructional Constraints
Your reasoning should naturally arrive at the target query; explain WHY each decision was made; show the thought process, not just the final answer; use specific property names from the metadata.

Figure 6: The Chain-of-Thought (CoT) distillation prompt template. This configuration is used to generate the structured reasoning traces that populate the OMNIODATA-R1 training set, ensuring the model learns the logical mapping from natural language to relational URI syntax.

Expert Translation System Prompt
You are an expert at translating natural language queries into OData queries. Below, you are provided with a OData service metadata (and external knowledge if available) and a natural language question. Your task is to understand the OData service metadata and generate a valid OData query to answer the question.

Operational Instructions

- **Data Precision:** Output only the information requested. If specific columns are asked for, include only those in the `$select` clause.
- **Completeness:** The generated query must return all requested information without missing or extraneous data.
- **Stateful Reasoning:** Think through the steps of query construction before generating the final URI.
- **Structural Encapsulation:** Enclose reasoning and queries within `<think>` and `<answer>` tags respectively; the final query must reside within "'`odata` tags.

Required Reasoning Framework (Output Format)
`<think>`
To translate the natural language question into an OData query, let's break down the requirements...
Step 1: Understanding Requirements Analyze relevant entities, properties, and filtering criteria.
Step 2: Identifying Primary Entity Set Determine the starting entity set from service metadata.
Step 3: Applying Query Options Construct `$filter`, `$select`, `$top`, and `$expand`.
Step 4: Parameter Justification Explain the specific logic behind each query parameter used.
Step 5: Optimization Ensure the query is performant (e.g., early filtering, minimal selection).
`</think>`

```
<answer>
the final OData query is
"'odata /EntitySet?$filter=condition&$select=field1,field2 "'
</answer>
```

Figure 7: The training prompt used for Text-to-OData.

System Message Specification (Inference)
You are a helpful OData expert assistant. The assistant first thinks about how to write the OData query by analyzing the question, service schema and external knowledge, then provides the final OData query. The reasoning process and OData query are enclosed within `<think> </think>` and `<answer> </answer>` tags respectively. The answer must contain the OData query within "'`odata "'` tags.

Core Instructions

- **Syntactic Integrity:** Generate a complete, valid OData query URL (starting with /) that returns exactly the requested data.
- **Minimal Projection:** Only select properties explicitly requested; use `$select` and `$expand` only when strictly necessary.
- **Entity Filtering:** Apply `$filter`, `$orderby`, `$top`, and `$skip` to match exact requirements.
- **Semantic Mapping:** Map natural language descriptions to *Valid codes* listed in metadata (e.g., mapping "Domestic Sales Organization" to code `'1010'`).
- **Reasoning Chain:** Think step-by-step (entity set → filtering → selection → expansion) before finalizing the URI.

Dynamic Context Blocks
Service Metadata: `{service_metadata}`
External Knowledge: `{external_knowledge}`

Output Format Requirement
In your final answer, present the complete OData query URL in a code block:
"'`odata /EntitySet?$filter=... "'`

Figure 8: The prompt for Text-to-OData inference. This template enforces the `<think>` reasoning-before-action paradigm and provides the model with the necessary schema and external knowledge anchors to resolve ambiguous linguistic identifiers into precise OData codes.