Ontology-Based Access to World Cup Data through Virtual Knowledge Graphs

Lívia Cristina Silva do Nascimento[1,2], Rafaela Moos[2], João Couto[2], Renata I. S. Pereira[3]

[1] Institute of Informatics – Federal University of Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brazil

[2]Buildaz Group – São Paulo – SP – Brazil

[3]Arapiraca Campus – Federal Institute of Alagoas (IFAL)
Rodovia AL 110 – Deputado Nezinho – Arapiraca – AL – Brazil

{livia,rafaela,jcouto}@buildaz.io, renata.pereira@ifal.edu.br

Abstract. *This paper presents a semantic virtualization architecture for FIFA World Cup data (1930–2026), combining Ontocup, a World Cup ontology aligned with the Basic Formal Ontology, with mappings in Ontop's native syntax (equivalent to R2RML) to expose a PostgreSQL database as a Virtual Knowledge Graph queryable via SPARQL, without full RDF materialization. We present representative SPARQL queries with their SQL equivalents, verifying their equivalence and reporting execution times. Results indicate that the ontological layer reduces the dependency between queries and the physical schema and favors future Text-to-SPARQL extensions powered by large language models.*

Resumo. *Este artigo apresenta uma arquitetura de virtualização semântica de dados da Copa do Mundo FIFA (1930–2026), combinando a Ontocup, ontologia alinhada à Basic Formal Ontology, com mapeamentos na sintaxe nativa do Ontop (equivalente ao R2RML) para expor uma base PostgreSQL como um Grafo de Conhecimento Virtual consultável via SPARQL, sem materialização completa em RDF. São apresentadas consultas SPARQL com seus equivalentes SQL, verificando sua equivalência e reportando os tempos de execução. Os resultados indicam que a camada ontológica reduz a dependência entre as consultas e o esquema físico e favorece extensões futuras Text-to-SPARQL baseadas em modelos de linguagem de grande porte.*

1. Introduction

The growing availability of structured and semi-structured data has expanded analysis possibilities across many domains, including sports. Yet the mere existence of such data does not guarantee its integration or reuse: information often remains distributed across sources with their own schemas, dependent on implicit knowledge about the meaning of tables, attributes, and relationships, which hinders queries based on the meaning of data rather than its physical structure.

Knowledge graphs have emerged as a relevant approach for representing domain entities and their relationships in an explicit, connected, and semantically interpretable manner [Hogan et al. 2021], supported by Semantic Web technologies such as

RDF [W3C 2014] and SPARQL [W3C 2013]. Since materializing such graphs is costly and raises synchronization and maintenance issues when sources change frequently, an alternative is the use of Virtual Knowledge Graphs, related to the *Ontology-Based Data Access* (OBDA) paradigm, in which an ontology provides a conceptual view of the data while mappings link domain concepts to the original sources [Poggi et al. 2008]. Systems such as Ontop rewrite SPARQL queries into SQL and create the graph view at query time, keeping data in its original sources [Xiao et al. 2020].

Sports data constitute an interesting domain for this approach, bringing together competitions, editions, national teams, players, matches, stadiums, goals, and cards whose analysis frequently depends on specific joins and prior schema knowledge. Questions such as "which players scored goals in a given edition?" or "which national teams faced a particular opponent in the knockout stages?" require combining different entities and relationships from the domain.

This paper presents a semantic virtualization architecture for sports data based on virtual knowledge graphs, using World Cup data as a case study. The proposal combines the Ontocup ontology, aligned with the *Basic Formal Ontology*, mappings between relational sources and ontology concepts, and SPARQL queries at the conceptual level. Its main contributions are the architecture for semantic exposure of sports data, the modeling of the Ontocup ontology, the specification of the mappings, and the demonstration of the approach through representative SPARQL queries, without requiring complete RDF materialization.

2. Theoretical Background

This section introduces the fundamental concepts and essential terminology for understanding this study.

2.1. Ontologies

Ontologies are computational artifacts that represent, in an explicit, formal, and shareable manner, the concepts of a domain and the relationships among them. Studer, Benjamins, and Fensel define an ontology as a formal and explicit specification of a shared conceptualization [Studer et al. 1998], refining Gruber's classical definition [Gruber 1993]: the relevant concepts are made explicit, the specification is formal enough for computational interpretation, and the conceptualization is shareable across people, systems, and applications. Unlike a relational schema, focused on the physical organization of data, an ontology represents the meaning of the modeled elements: concepts such as *Player*, *Match*, and *Goal*, and relationships such as *playsFor* and *scoredIn*, form a shared conceptualization of the sports domain. In this work, the ontology is the semantic layer that organizes the football domain and grounds the virtual knowledge graph, enabling originally tabular data to be queried at the conceptual level through SPARQL.

2.2. Knowledge Graphs

Knowledge graphs represent domain entities and their relationships in an explicit, structured, and semantically interpretable way, modeling objects, events, places, people, and concepts as nodes connected by named relationships, which facilitates the integration of heterogeneous data through a common conceptual model [Hogan et al. 2021]. In a sports

domain, entities such as player, national team, match, and edition become interconnected elements, with relationships such as "played for" or "scored a goal in" making these connections explicit. When associated with an ontology describing the relevant concepts, their properties, and constraints, the graph gains a conceptual layer that organizes the interpreting vocabulary, which is particularly important when data is distributed across sources with distinct schemas.

2.3. RDF and SPARQL

The *Resource Description Framework* (RDF) represents information as triples formed by a subject, predicate, and object [W3C 2014]; for example, "Messi scored a goal in the 2022 final" relates an entity of type player, one of type goal, and one of type match. SPARQL queries RDF data through graph patterns [W3C 2013]. Unlike relational queries tied to table structures, SPARQL queries are expressed in terms of the entities and relationships of the semantic model, allowing questions at a level closer to the domain, such as listing players who scored goals for a national team.

2.4. Mapping Relational Data to RDF

Since much structured data remains in relational databases, querying it semantically without reformulating existing schemas requires mappings between the relational and RDF models. R2RML is a standardized language for such mappings [W3C 2012], associating tables, columns, and keys with classes, properties, and relationships of an ontology. While R2RML is natively expressed in RDF/Turtle, OBDA systems such as Ontop provide a more compact, human-readable syntax that is functionally equivalent to R2RML and exportable to it, facilitating the authoring and maintenance of mappings.

2.5. Ontology-Based Data Access and Virtual Knowledge Graphs

Ontology-Based Data Access (OBDA) uses an ontology as a conceptual interface between users and data sources [Poggi et al. 2008]: the ontology defines the domain vocabulary and mappings specify how the stored data corresponds to its concepts, so queries can be formulated over the conceptual model even when data remains relational and its internal organization changes without affecting user queries. Virtual Knowledge Graphs extend this idea by exposing relational or structured sources as knowledge graphs without full materialization in an RDF repository [Xiao et al. 2019]. The graph is produced virtually at query time: systems such as Ontop rewrite SPARQL queries formulated over the ontology into SQL executed on the underlying sources [Xiao et al. 2020], presenting the result as if the data were an RDF graph while it remains in its original databases, which avoids data replication, reduces synchronization issues, and supports independently updated sources.

3. Solution Architecture

The proposed architecture enables the semantic querying of sports data through a Virtual Knowledge Graph. Data remains stored in a PostgreSQL relational database, while a semantic layer of an ontology and mappings allows it to be accessed at the conceptual level, so users can query domain entities such as matches, national teams, players, stadiums, cities, World Cup editions, and game events without knowing the physical structure of the tables.

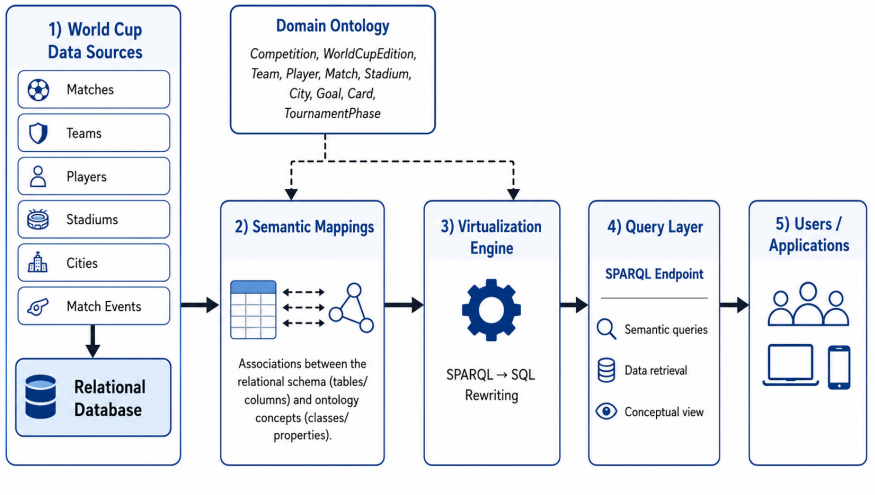


Figure 1. Proposed architecture for semantic virtualization of World Cup data through a Virtual Knowledge Graph.

The solution is organized into five layers, as illustrated in Figure 1: a data source with structured World Cup records; a domain ontology defining the relevant concepts and relationships; the mappings linking relational elements to ontology terms; a virtualization engine that rewrites SPARQL over the ontology into SQL on the original source; and a query layer through which users and applications access the data as a semantic view.

The data were obtained from an openly available football data API and from open-source repositories, covering every FIFA World Cup edition from 1930 to 2026. The consolidated source is the `world_cup` relational database, hosted in PostgreSQL 17.10, occupying approximately 14 MB across 21 tables and about 44,900 records, the largest being player call-ups and per-match player statistics (`match_lineup_player`, 14,905 rows, and `match_player_statistic`, 7,545). Over this base, an OWL ontology grounded in the Basic Formal Ontology (BFO 2020) was built, comprising 35 classes, 22 object properties, 16 datatype properties, and 4 annotation properties, connected to the database through 43 OBDA mappings defined in Ontop. Table 1 summarizes the dataset, the ontology, and the tool versions.

Table 1. Dataset, ontology, and tool versions used in the evaluation.

Item	Value	Item	Value
Database	`world_cup`	Ontology classes	35
Coverage	1930–2026	Object properties	22
Size / tables	∼14 MB / 21	Datatype properties	16
Records	∼44,900	Annotation properties	4
PostgreSQL	17.10	OBDA mappings (Ontop)	43
Ontop	5.5.0	PostgreSQL JDBC driver	42.7.4
Java (runtime)	OpenJDK 11.0.30 Temurin	Protégé Desktop	5.6.9

The second layer consists of the domain ontology. This ontology provides a conceptual representation of the main elements related to the World Cup, aligned with the *Basic Formal Ontology* as the top-level ontology. Table 2 presents the Ontocup class hierarchy organized according to the BFO and IAO categories.

Table 2. Ontocup class hierarchy according to BFO and IAO

BFO/IAO Category	Parent class	Ontocup ontology
Continuant	Object	person
	Object Aggregate	team, state, organization, club, national_team
	Site	country, city, continent, stadium
	Role	player, coach, referee, host_country, host_city, goalkeeper, defender, midfielder, forward
	Quality	nationality
	Information Content Entity	goal, card, foul, scoreboard, result, substitution, season
Occurrent	Process	match, score_goal, commit_foul, issue_card, make_substitution

The third layer holds the mappings between the relational schema and the ontology, expressed in Ontop's native mapping syntax (functionally equivalent to R2RML and exportable to it). Each mapping pairs a SQL query with a target template that projects the selected rows into RDF triples, translating tables, columns, and keys into classes, object properties, and data properties of Ontocup. This layer also operationalizes the BFO design decisions: roles such as *player* or *coach* are instantiated as BFO Roles borne by *Person* individuals, rather than as subclasses of *Person*, and occurrents such as *goal* are linked to the underlying process that generated them. Table 3 presents two representative mappings of these patterns.

The fourth layer is the virtualization engine, which receives SPARQL queries formulated over the ontology and translates them into SQL executed on the relational database. The graph is produced virtually at query time from the combination of the ontology, the mappings, and the data in the original source, so the user queries an RDF view without materializing it in a triple store, reducing replication and keeping the relational database as the primary source.

The fifth layer is the query interface, where users or applications submit SPARQL queries at the semantic level. Unlike SQL queries tied to the physical schema, SPARQL queries are formulated from domain concepts, allowing questions such as which players scored goals in a given World Cup edition, which matches took place in stadiums in the same city, or which national teams participated in the knockout stages.

The main advantage of the architecture is the separation between the conceptual representation and the physical structure of the data: it promotes maintainability, since database changes can be absorbed in the mappings without modifying user queries, and facilitates interoperability, since different applications access the same data through a shared vocabulary. The World Cup serves as a case study to demonstrate the applicability of the architecture in a domain rich in interrelated entities and analytical queries.

Table 3. Examples of mappings used to populate the Ontocup ontology.

Mapping	Description
`jogador-papel` target::jogador/{pc_id}a:jogador;bfo :has_role:pessoa/player-{player_id}.: pessoa/player-{player_id}bfo:has_me mber_part:selecao/{national_team_id }.	This mapping instantiates the *Player* role as a BFO Role borne by a *Person* individual, rather than as a subclass of *Person*. The same query also links the player, through the `has_member_part` relation, to the national team in which the role is exercised, illustrating how role instantiation and team membership are expressed jointly in a single mapping.
`gol` target::gol/{id}a:gol;:tipo{detail};:ma rcado_em{minuto};:marcado_por:jog ador/{pc_id};:registrado_em:partida/{ match_id}.	This mapping instantiates a *Goal* as an Information Content Entity generated by the underlying scoring process (modeled separately as a *Goal-Scoring* occurrent). It links the entity to the player who scored it and to the match in which it occurred, illustrating the pattern in which occurrents give rise to information content entities recording their outcome.

4. Queries and Results

In a traditional SQL query, the user—or the system responsible for automatic query generation—must deal directly with the physical schema: table names, columns, keys, and the joins required to retrieve the information, so that an apparently simple question such as retrieving players who scored goals in a given World Cup edition may require combining several tables. In the proposed approach, SPARQL queries are formulated over the domain ontology, using concepts closer to the semantic interpretation of the question: relationships such as *scoredBy* and *hasMemberPart* make the meaning of connections explicit, transferring part of the complexity of joins and physical details to the mappings and the virtualisation engine.

This suggests a potential benefit for integration with large language models. Rather than inferring, as in *Text-to-SQL*, how question terms relate to specific tables and columns, the model can convert natural language questions into SPARQL formulated over the ontology; since the ontology makes domain classes and properties explicit, query generation becomes more controlled and semantically oriented [D'Abramo et al. 2025, Zhu et al. 2024].

Four competency questions (CQs) were designed to exercise ontological constructs that have no direct counterpart in the relational schema. Each CQ was executed both as SPARQL over the virtual knowledge graph and as a hand-written SQL query returning the equivalent answer directly from PostgreSQL.

CQ-01 — Cross-type entity retrieval. *Which entities of any type in the domain have a given token in their name?* The property `:has_name` is asserted by 13 dis-

tinct mappings, so a single triple pattern simultaneously reaches people, stadiums, clubs, teams, cities and countries.

CQ-02 — Event taxonomy of a match. *Which kinds of record does a match accumulate?* The property `:recorded_in` spans seven mappings. The ontology further distinguishes the *process* (`:score_goal`, a subclass of `BFO_0000015`) from the *information record* of that process (`:goal`, a subclass of `IAO_0000027`)—a distinction absent from the relational schema, where both collapse into a single `match_event` row discriminated by a string column.

CQ-03 — Multi-granular membership. *How many distinct teams has each person been part of?* A single property returns membership at three levels of aggregation at once: the national team as a persistent entity, the squad of each individual edition, and each club-season.

CQ-04 — Role census by superclass. *How many people have borne each type of role?* Querying `?role a BFO_0000023` retrieves every role through T-Box subsumption, without enumerating the subclasses.

Table 4 contrasts the two formulations of each competency question. Repeated `UNION` branches in the SQL column are elided and their count reported, since they differ only in the source table and the literal type label; the full queries are available in the replication package.

4.1. Results

Both formulations of every competency question returned the same answer, which confirms that the semantic mappings faithfully expose the relational content. Table 5 summarizes the answers obtained.

Table 5. Answers returned by the four competency questions.

CQ	Answer
CQ-01	person 13; stadium 8; team 7; club 4; organization 4; city 2
CQ-02	make_substitution 2,744; score_goal 1,222; goal 1,189; issue_card 809; score 352; result 352
CQ-03	Ochoa, Messi and Ronaldo with 13 distinct teams each
CQ-04	player 4,309; midfielder 1,410; defender 1,403; forward 1,028; goalkeeper 519; coach 217

The clearest difference lies in how much of the physical schema each formulation must encode. Table 6 contrasts the number of `UNION` branches and distinct source tables the SQL formulation requires against the single triple pattern that suffices in SPARQL.

Table 6. Structural cost of each formulation.

CQ	SPARQL triple patterns	SQL `UNION` branches	SQL source tables
CQ-01	1	9	9
CQ-02	1	6	2
CQ-03	1	3	1
CQ-04	1	6	2

Table 4. The four competency questions in SPARQL and in equivalent SQL. Namespace prefixes and the filter restricting `?type` to the domain namespace are omitted for brevity.

	SPARQL	SQL equivalent

CQ-01	`SELECT ?type (COUNT(DISTINCT ?e) AS ?n)`
	`WHERE {`
	`  ?e :has_name ?name ; a ?type .`
	`  FILTER(CONTAINS(LCASE(?name),"santos"))`
	`} GROUP BY ?type ORDER BY DESC(?n)`
		`SELECT type, count(DISTINCT id) AS n FROM (`
		`  SELECT 'person' AS type,id,name FROM player`
		`  UNION ALL SELECT 'stadium',id,name FROM stadium`
		`  -- + 7 branches: coach, national_team, club_team,`
		`  --  city, country, competition, stage`
		`) t WHERE lower(name) LIKE '%santos%'`
		`GROUP BY type ORDER BY n DESC`

CQ-02	`SELECT ?type (COUNT(*) AS ?n)`
	`WHERE {`
	`  ?x :recorded_in ?match ; a ?type .`
	`} GROUP BY ?type ORDER BY DESC(?n)`
		`SELECT type, count(*) AS n FROM (`
		`  SELECT 'score_goal' AS type FROM match_event`
		`  WHERE event_type='Goal'`
		`  UNION ALL SELECT 'make_substitution'`
		`  FROM match_event WHERE event_type='subst'`
		`  -- + 4 branches: goal, issue_card, result, score`
		`) t GROUP BY type ORDER BY n DESC`

CQ-03	`SELECT ?name (COUNT(DISTINCT ?team) AS ?teams)`
	`WHERE {`
	`  ?p a :person ; :has_name ?name ;`
	`  bfo:BFO_0000129 ?team .`
	`} GROUP BY ?p ?name`
	`ORDER BY DESC(?teams) LIMIT 15`
		`SELECT p.name, count(DISTINCT e.team) AS teams`
		`FROM player p JOIN (`
		`  SELECT player_id,'nt-'||national_team_id AS team`
		`  FROM player_convocation`
		`  UNION SELECT player_id,'squad-'||national_team_id`
		`  ||'-'||season_id FROM player_convocation`
		`  -- + 1 branch: club-level membership`
		`) e ON e.player_id=p.id`
		`GROUP BY p.id,p.name ORDER BY teams DESC LIMIT 15`

CQ-04	`SELECT ?type (COUNT(DISTINCT ?person) AS ?people)`
	`WHERE {`
	`  ?role a bfo:BFO_0000023 , ?type ;`
	`  bfo:BFO_0000197 ?person .`
	`} GROUP BY ?type ORDER BY DESC(?people)`
		`SELECT type, count(DISTINCT person) AS people FROM (`
		`  SELECT 'player' AS type, player_id::text AS person`
		`FROM player_convocation`
		`  UNION ALL SELECT 'coach','coach-'||id::text FROM coach`
		`  -- + 4 branches: goalkeeper, defender,`
		`  --  midfielder, forward`
		`) t GROUP BY type ORDER BY people DESC`

This asymmetry is not merely syntactic. In every case the SQL formulation enumerates by hand a list that the ontology already holds—which tables carry a name, which event types a match may record, which roles exist—and the enumeration is silently incomplete whenever the schema grows: a new table with a `name` column does not appear in CQ-01 until the query is edited, whereas the SPARQL formulation picks it up as soon as a mapping is declared. CQ-03 illustrates the practical consequence: our first SQL formulation modelled only two levels of aggregation (national team and club) and returned seven teams for L. Messi, but enumerating the SPARQL result revealed a third level, the per-edition squad, raising the correct answer to thirteen. The ontology exposed a distinction that the author of the SQL query did not know to look for.

Execution time was measured from a single process on the same host: SPARQL over HTTP against the Ontop endpoint, SQL over the native PostgreSQL protocol, both reusing the connection across runs. One warm-up execution was discarded, since the engine is configured with lazy initialisation and the first query pays for building the OBDA plan; five subsequent executions were measured. The reported value is the arithmetic mean, and the dispersion remained below 4% in every case.

Table 7. Mean execution time over five runs, in milliseconds.

CQ	SPARQL (ms)	SQL (ms)	Ratio
CQ-01	18.7	1.7	11.0× slower
CQ-02	9.9	1.6	6.2× slower
CQ-03	26.8	8.1	3.3× slower
CQ-04	11.5	10.9	1.1× slower

SPARQL was slower in all four cases. This is expected and follows from the architecture, not from any deficiency of the queries: an OBDA engine does not execute SPARQL but rewrites it into SQL and submits it to the same relational engine that answers the hand-written query. The measured difference is therefore the cost of rewriting plus the HTTP round trip, on top of an execution that is, at best, identical to the SQL baseline. This overhead is largely fixed: it does not scale with the result set, so it dominates on queries that return few rows, precisely the profile of aggregate competency questions. The gap narrows as the hand-written SQL becomes more elaborate—in CQ-04, whose relational formulation requires six `UNION` branches over two tables, the two approaches are within 6% of each other—suggesting that the semantic layer is paid for not in throughput but in the conciseness and stability of the query.

We therefore do not claim a performance benefit for the virtual knowledge graph. The contribution of the semantic layer is the ability to formulate a question in terms of the domain rather than the schema, which also makes it an attractive target for language-model-generated queries: a model emitting SPARQL over an explicit vocabulary operates in a far smaller, better-specified search space than one inferring joins over a physical schema.

Three limitations were observed and should be reported. First, the virtualisation engine does not support SPARQL property paths of arbitrary length: queries of the form `?s :locatedIn+ ?o` are rejected, since an unbounded path cannot be unfolded into non-recursive SQL. Transitive closure over the geographic or mereological hierarchies—arguably where SPARQL would have the clearest advantage over SQL—is therefore unavailable, and such traversals must be written with a fixed number of hops. Second, mapping design directly determines identity: in one mapping the referee individual is minted from the match identifier rather than the person, so a referee officiating 285 matches yields 285 distinct individuals instead of 116—an error invisible in the relational data that surfaces only when the graph is queried. Third, because `:has_name` is shared by entities of every type, aggregating by name rather than by IRI conflates homonyms across the domain, which is why CQ-03 required grouping by the person IRI.

5. Final Remarks

This paper presented an architecture for semantic virtualization of sports data through a Virtual Knowledge Graph, using World Cup data as a case study, combining the Ontocup ontology, semantic mappings, and the Ontop engine to query a PostgreSQL relational database at the conceptual level through SPARQL. The work demonstrates that knowledge graph virtualization can support the integration and semantic querying of sports data without full materialization in RDF, reducing replication, preserving the original sources, and allowing users and applications to query information based on domain concepts.

As future work, we intend to extend the domain ontology, incorporate new sports data sources, explore more complex queries, and evaluate the architecture's performance on larger-scale databases. We also plan to investigate integration with natural language interfaces, enabling users to ask questions in Portuguese or English and obtain responses directly from the virtual knowledge graph.

Declaration on Generative AI

During the preparation of this work, the author(s) used ChatGPT 4.1 and Claude Sonnet 4.5 in order to: Grammar and spelling check, Paraphrase and reword. After using these tools, the author(s) reviewed and edited the content as needed and take(s) full responsibility for the publication's content.

References

D'Abramo, J., Zugarini, A., and Torroni, P. (2025). Investigating large language models for text-to-SPARQL generation. In *Proceedings of the 4th International Workshop on Knowledge-Augmented Methods for Natural Language Processing*, pages 66–80, Albuquerque, New Mexico, USA. Association for Computational Linguistics.

Gruber, T. R. (1993). A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220.

Hogan, A., Blomqvist, E., Cochez, M., et al. (2021). Knowledge graphs. *ACM Computing Surveys*, 54(4):1–37.

Poggi, A., Lembo, D., Calvanese, D., De Giacomo, G., Lenzerini, M., and Rosati, R. (2008). Linking data to ontologies. In *Journal on Data Semantics X*, pages 133–173. Springer.

Studer, R., Benjamins, V. R., and Fensel, D. (1998). Knowledge engineering: Principles and methods. *Data and Knowledge Engineering*, 25(1):161–197.

W3C (2012). R2rml: Rdb to rdf mapping language. W3C Recommendation.

W3C (2013). Sparql 1.1 query language. W3C Recommendation.

W3C (2014). Rdf 1.1 concepts and abstract syntax. W3C Recommendation.

Xiao, G. et al. (2019). Virtual knowledge graphs: An overview of systems and use cases. *Data Intelligence*, 1(3):201–223.

Xiao, G., Lanti, D., Kontchakov, R., et al. (2020). The virtual knowledge graph system ontop. In *International Semantic Web Conference*. Springer.

Zhu, X., Li, Q., Cui, L., and Liu, Y. (2024). Large language model enhanced text-to-SQL generation: A survey.