A comparative survey of ontology-based database querying solutions

Mohamed Amin Gaied, Christina Khnaisser
GRIIS, Université de Sherbrooke, Sherbrooke (Québec), Canada
Mohamed.Amin.Gaied@usherbrooke.ca, Christina.Khnaisser@usherbrooke.ca

May 6, 2025

Abstract

In an era of increasingly heterogeneous data ecosystems, ontologies provide a unified framework for semantic querying across both relational and RDF-based systems. This report presents a comparative survey of contemporary tools designed to support ontology-driven access to structured data. We introduce a structured set of functional and non-functional evaluation criteria and assess six representative systems according to their performance across these dimensions. Our findings identify OntoRelQuery as a notable advancement over existing solutions, particularly in terms of usability and query generation. This survey contributes to a deeper understanding of ontology-based querying architectures and offers strategic guidance for future developments in semantic data integration and access.

1 Introduction

In today's data-driven world, organizations across sectors struggle to extract coherent insights from increasingly fragmented and structurally diverse data sources. Despite the abundance of valuable information, discrepancies in schemas, terminologies and system architectures often hinder integrated analysis and complicate knowledge discovery. These disparities are especially problematic for analytical applications that rely on a unified data perspective to support informed decision-making.

Ontologies have emerged as a powerful response to this issue. By formally defining domain-specific concepts, relationships, and constraints, they provide a semantic layer that enables consistent interpretation across heterogeneous data systems. Ontologies act as bridges between differing structural representations, offering a shared vocabulary that can align disparate sources. However, connecting these abstract semantic models with the practical realities of data storage—particularly within relational database environments—remains a technically intricate endeavor.

Ontology-based database querying lies at this pivotal intersection, combining the expressive richness of ontologies with the mature infrastructure of relational database management systems (RDBMS). Although considerable progress has been made in this domain, current solutions still face critical shortcomings, including limited usability, restricted expressiveness, or complex technical configurations. Non-specialist users, in particular, often struggle to articulate advanced information needs across datasets without sacrificing semantic coherence.

This study presents a comparative assessment of state-of-the-art tools designed to facilitate ontology-based querying over relational and RDF-based systems. Central to our analysis is the ability of each system to support user-friendly query formulation while ensuring semantic

precision and operational efficiency. To structure our evaluation, we define a comprehensive framework of functional and non-functional requirements. The former includes criteria such as ontology processing, user request formulation, query generation, and result handling. The latter addresses broader concerns, including system availability, technological compatibility, multilingual query support, and extensibility.

We systematically evaluate seven prominent systems within this domain: OntoRelQuery[1], Ontop [1], OntoGrate [2], DAFO [3], VOn-QBE [4], ATHENA++[5], and OptiqueVQS[6]. Each tool is assessed in terms of its functional coverage and practical limitations, with a particular focus on how effectively it bridges the semantic-relational divide and empowers users to express information needs without requiring deep technical knowledge.

Our findings not only clarify the current capabilities and limitations of these tools but also highlight areas where future development is needed. In identifying persistent gaps—such as limited interoperability or insufficient support for complex query scenarios—we aim to inform ongoing research and guide the evolution of more accessible, semantically powerful querying systems capable of unlocking the full potential of heterogeneous data landscapes. Our contribution lies in providing a structured, comprehensive evaluation framework that enables objective comparison of ontology-based querying solutions across both functional and operational dimensions, offering practical guidance for both researchers and practitioners in this rapidly evolving field.

2 Requirements analysis

Evaluating systems for ontology-based data querying requires a structured framework that captures the challenges of bridging semantic knowledge representation with diverse data storage formats, including both relational databases and RDF triplestores. This section introduces our evaluation taxonomy, which considers both functional and non-functional dimensions essential to the lifecycle and viability of semantic querying tools.

Semantic querying systems operate at the junction of two complementary paradigms: the conceptual expressiveness of ontologies and the performance-oriented architectures of structured data repositories. The core challenge is translating semantically rich user intentions into executable queries—whether in SQL or SPARQL—while ensuring that the system remains usable for domain experts with limited technical backgrounds.

To enable a consistent and meaningful evaluation, we organize the requirements into two primary categories:

- **Functional requirements.** These refer to the key operational features needed throughout the query lifecycle, including ontology processing, query formulation, query translation, and output handling. A robust implementation ensures that conceptual queries are accurately interpreted and executed while preserving semantic integrity.
- **Non-functional requirements.** These encompass broader system attributes that influence usability and adoption, such as platform compatibility, maintainability, language interoperability, and adaptability to evolving architectures or data environments.

This classification enables a granular yet holistic assessment of both current capabilities and future adaptability—critical factors for organizations investing in scalable, semantically aware data integration solutions. Each requirement is further detailed through specific sub-criteria and

[1] `https://github.com/OpenLHS/OntoRelQuery`

measurable features, which form the foundation of our comparative analysis.

The selection of these particular criteria was informed by both a systematic literature review of semantic data integration challenges and consultations with industry practitioners facing real-world heterogeneous data access issues. While all criteria contribute to overall system effectiveness, we consider ontology processing (FR1) and query generation (FR3) to be of primary importance for functional requirements, as they directly impact semantic fidelity and query expressiveness. For non-functional aspects, compatibility (NFR2) and extensibility (NFR4) were prioritized, as they significantly affect long-term adoption in environments.

2.1 Functional requirements

Functional requirements define the core capabilities that an ontology-based database querying system must provide to effectively bridge the semantic and relational paradigms.

2.1.1 FR1. Ontology processing

Ontology processing encompasses the system's ability to effectively navigate, analyze, and leverage ontological structures as the foundation for query formulation. A proficient system must seamlessly traverse complex semantic relationships defined within ontologies while identifying structural patterns and optimizing query paths. This capability requires sophisticated graph traversal algorithms to explore variable-length paths, structural analysis mechanisms to detect ontological components and topological patterns, and reasoning capabilities to infer implicit relationships. By effectively processing ontological structures, the system establishes the semantic foundation upon which meaningful queries can be constructed, ensuring that the rich conceptual relationships defined in the ontology are preserved throughout the querying process.

Table 1 presents the definitions of the criteria used for the evaluation.

Criterion	**Sub-criterion**	**Description**	**Example**
Ontology graph traversal	Path traversal	Traverse ontological graph between concepts respecting hierarchy and associations.	Traversal via rdfs:subClassOf from Researcher to SeniorResearcher, JuniorResearcher, then PhD student respecting defined relationships.
	Traversal algorithms	Implement algorithms to traverse ontology graph efficiently per research objectives (depth, breadth, etc.).	DFS for taxonomy exploration; BFS for finding direct relationships with depth limitation options.
	Variable path exploration	Support paths of unknown length, discover accessible entities from start point without specifying end.	SPARQL using rdfs:subClassOf* for all subclasses at any depth; (owl:ObjectProperty)+ for connections without specifying steps.

Structural analysis	Detection of ontological components	Identify axiom types (class, property, individual) and structures (cardinality, data types).	Identifies: class axioms (Professor subClassOf Academic); property axioms (supervises Domain Professor Range Student); equivalent classes (FullProfessor EquivalentTo Professor and hasRank "Full").
	Pattern identification	Recognize topological structures: pivot entities, sparse graphs, thematic clusters; prioritize paths and key concepts.	Biomedical ontology analysis identifies "Disease" as pivot entity connected to Symptoms/Treatments; "SideEffects"-"ChemicalInteractions" as sparse subgraph requiring different strategies.
	Cycle detection	Identify circular dependencies in structure causing inference problems.	Detects cycles: Department employs Professor, Professor memberOf Department, Department subOrganizationOf Faculty, Faculty hasDepartment Department.
Traversal optimization	Customized traversal algorithms	Adapt graph algorithms (DFS, BFS, A*) based on ontological structure and semantic importance.	Algorithms detect structure type (hierarchy, network) and select optimal strategies automatically.
	Ontological graph reasoning	Exploit inference rules to deduce implicit relationships, identify shortcuts and optimize traversals.	Reasoning establishes "TenuredProfessor subClassOf Staff" directly from "TenuredProfessor subClassOf Professor" and "Professor subClassOf Staff" without full chain traversal.

Table 1: Ontology processing criteria definition

2.1.2 FR2. Query design

Query design focuses on enabling users to express complex information needs in intuitive ways without requiring expertise in formal query languages. This requirement emphasizes the human-system interface, providing diverse expression methods ranging from visual manipulation to natural language processing, while offering intelligent assistance during query construction. The system must guide users through the formulation process with contextual suggestions, error detection, and incremental refinement capabilities. By supporting progressive and iterative query development with immediate feedback, the system bridges the semantic gap between users' conceptual understanding and formal query structures, making sophisticated database interrogation accessible to domain experts regardless of their technical background.

Table 2 presents the definitions of the criteria used for the evaluation.

Criterion	**Sub-criterion**	**Description**	**Example**

Request expression methods	By graphical interface	Visual interface allowing query construction by directly manipulating graphical elements representing concepts and relationships without requiring knowledge of query languages.	Interface where users select "Scientific Article" class, add filter on publication year, connect with author and affiliation by drag-and-drop of elements and adding visual links between them.
	Keyword-based system	Mechanism allowing query formulation from domain-specific keywords automatically interpreted and mapped to ontological concepts and properties.	Input of "artificial intelligence articles 2022-2023 conferences high impact" converted into structured query filtering publications by domain, period, type and impact factor.
	By guided question-answer (Q&A)	Conversational system guiding users through targeted questions to progressively refine requests and build complete, coherent queries.	Dialogue: "What type of publication?" → "Scientific articles" → "What domain?" → "Artificial intelligence" → "What time period?" → "2020 to present" generating structured queries with these criteria.
	By natural language (NL)	Processing phrases in everyday language to transform them into formal queries by identifying entity relationships and conditions in free text.	Analysis of "Find all articles on machine learning published since 2021 by European researchers having impact greater than 4" to generate equivalent SPARQL/SQL query with appropriate filters.
	By custom language	Domain-specific simplified syntax serving as the intermediary between natural language and formal query languages with keywords adapted to business context.	Interpretation of command FIND Articles WHERE domain IS "machine learning" AND year >= 2021 AND authors.region IS "Europe" AND impact > 4 into formal SQL/SPARQL query.
Request formulation assistance	Contextual auto-completion	Suggestions of terms, concepts or relationships during input based on ontology and current context of query construction.	After typing "Aut", suggestion of "Author", "Authority", "Automation" corresponding to ontology concepts with priority given to the most relevant terms for the current context.
	Semantic suggestions	Recommendations of relevant properties or concepts based on ontology semantics and elements already selected in query.	After selecting "Researcher" class, suggesting applicable properties like "hasPublished", "worksIn", "specializedIn", "leads" based on relationships defined in ontology.
	Entity recognition	Automatic identification of references to ontological concepts in request text.	Detection that "papers", "publications", "articles" all refer to same ontological classes.
	Interactive visualizations during construction	Graphical representation of selected query elements allowing visualization of query structure and implications.	Graph showing selected classes as nodes, properties as edges, conditions as annotations with colour codes for different element types.

	Syntactic error detection and reporting	Mechanisms for automatic detection and correction of errors in request formulation at the syntactic level.	Correction of "Researcher publishes Article in 2022" to "Researcher hasPublished Article in 2022" to respect exact property name or suggestion if the user uses non-existent class.
	Semantic error detection and reporting	Proactive identification of potential problems like logical contradictions, impossible paths or overly restrictive filters.	Alert indicating filter "Researcher worksIn University AND Researcher worksIn Company" is not correct.
Progressive request formulation	Iterative formulation	Construction and refinement of queries through successive steps by progressively adding, modifying or removing elements with immediate feedback.	Interface allowing start with a simple query on "Researchers" then sequentially adding filters by domain and publication dates with results preview updated at each step.
	History management	System for storing queries and tracking versions during construction, allowing return to previous states or comparison of evolution.	Interactive timeline showing the history of query modifications with the ability to return to previous steps, compare versions or create alternative branches.

Table 2: Query design criteria definition

2.1.3 FR3. Query generation

Query generation represents the system's core technical ability to transform conceptual user requests into optimized, syntactically correct database queries. This requirement encompasses comprehensive capabilities for data filtering, selection, analysis, and combination, ensuring that queries capture the full semantic richness of user information needs while respecting the technical constraints of target database systems. From basic comparison operations to complex analytical functions, from elementary joins to sophisticated set operations, the system must generate queries that efficiently retrieve relevant information while maintaining semantic fidelity. Additionally, through abstraction and parameterization mechanisms, the system enables the reuse and adaptation of queries across different contexts, enhancing maintainability and consistency.

Table 3 presents the definitions of the criteria used for the evaluation.

Criterion	**Sub-criterion**	**Description**	**Example**
Data restriction capabilities	Simple restrictions	Data filtering according to elementary conditions using basic comparison operators (=, >, <, >=, <=, !=) on property values.	SQL: WHERE year = 2023 OR impact_factor > 5.0 SPARQL: FILTER(?year = 2023)
	Composite restrictions	Logical combination of multiple filtering conditions (AND, OR, NOT) enabling creation of complex and precise selection criteria.	SQL: WHERE (domain = 'AI' OR domain = 'ML') AND year >= 2020 AND NOT is_retracted = true SPARQL: FILTER((?domain = "AI"))

	Pattern-based restriction	Use of regular expressions or partial matching operators to perform advanced textual searches in the data.	SQL: WHERE title LIKE '%neural network%' OR abstract REGEXP 'deep[-]?learning' SPARQL: FILTER(REGEX(?title, "neural network", "i"))
	Set-based restriction	Evaluation of a value's membership in a predefined set of possible values or comparisons with data subsets (IN, ANY).	SQL: WHERE conference IN ('ICML', 'NeurIPS', 'ICLR') SPARQL: FILTER(?conference IN ("ICML", "NeurIPS", "ICLR"))
Data selection	Selective projection	Precise specification of attributes to include in the result allowing optimization of data volume and focus on relevant information.	SQL: SELECT author.name, author.institution, publication.title, publication.year, publication.doi SPARQL: SELECT ?authorName ?institution ?title ?year ?doi WHERE {...}
	Element renaming	Assignment of aliases to table attributes or variables to clarify the result avoiding ambiguities and facilitating the interpretation.	SQL: SELECT a.name AS author_name, i.name AS institution_name, p.title AS publication_title SPARQL: SELECT ?name AS ?author_name ?instName AS ?institution_name
	Ordering	Organization of results according to one or more sorting criteria with control of ascending or descending order for each criterion.	SQL: ORDER BY year DESC, impact_factor DESC, author_name ASC SPARQL: ORDER BY DESC(?year) DESC(?impact) ?authorName
	Result limitation	Restriction of the number of records returned facilitates pagination or limits the volume of data to process.	SQL: LIMIT 50 OFFSET 100 SPARQL: LIMIT 50 OFFSET 100 to get results 101 to 150 typically for pagination.
Analytical functions	Data aggregation	Application of statistical functions (COUNT, SUM, AVG, MIN, MAX) to calculate global or group metrics on data sets.	SQL: SELECT COUNT(id) AS total_publications, AVG(impact_factor) AS avg_impact, MAX(citations) AS max_citations SPARQL: SELECT (COUNT(?pub) AS ?total) (AVG(?impact) AS ?avg_impact)
	Grouping	Organization results in groups based on one or more common properties enabling comparative analysis by category.	SQL: SELECT domain, year, COUNT(*) AS num_publications, AVG(impact_factor) FROM publications GROUP BY domain, year SPARQL: SELECT ?domain ?year (COUNT(?pub) AS ?num)

	Conditions on aggregates	Application of filters on aggregation results allowing the selection of only groups meeting certain statistical criteria.	SQL: HAVING COUNT(publication) > 5 AND AVG(impact_factor) > 3 SPARQL: HAVING (COUNT(?pub) > 5 && AVG(?impact) > 3)
	Window function	Use of windowing functions to perform calculations on an ordered subset of rows.	SQL: ROW_NUMBER() OVER (PARTITION BY domain ORDER BY citations DESC) AS rank_in_domain SPARQL via extension: (RANK() AS ?domain_rank) OVER (PARTITION BY ?domain ORDER BY DESC(?citations))
Join operations	Joins	Association of data from different parts of the ontology based on corresponding values.	SQL: INNER JOIN authors ON publications.author_id = authors.id SPARQL: ?publication :hasAuthor ?author
	Outer joins	Inclusion of records from one source even if they have no match in the other source with null values for missing data.	SQL: LEFT JOIN citations ON publications.id = citations.cited_publication_id SPARQL: OPTIONAL { ?publication :isCitedBy ?otherPublication }
	Existential operators	Verification of existence or non-existence of results in a subquery allowing construction of conditions based on complex relationships.	SQL: WHERE EXISTS (SELECT 1 FROM citations WHERE citations.cited_publication_id = publications.id) SPARQL: FILTER EXISTS { ?publication :isCitedBy ?anyPublication }
	Subqueries	Nesting of a query inside another for multi-level operations or comparisons with dynamically calculated results.	SQL: WHERE citations > (SELECT AVG(citations) FROM publications WHERE domain = outer_pub.domain) SPARQL: { SELECT ?domain (AVG(?cites) AS ?avg_cites) WHERE {...} GROUP BY ?domain }
Set operations	Union	Combination of results from two structurally compatible queries with the option to eliminate duplicates (UNION) or preserve them (UNION ALL).	SQL: (SELECT * FROM ai_publications) UNION [ALL] (SELECT * FROM ml_publications) SPARQL: { ?pub a :AIPublication } UNION { ?pub a :MLPublication }
	Intersection	Extraction of records present in the results of both queries, allowing identification of elements common to two distinct criteria.	SQL: (SELECT author_id FROM ai_publications) INTERSECT (SELECT author_id FROM ml_publications) SPARQL: via combination of patterns or extensions

	Difference	Subtraction of results of one query from results of another allowing identification of elements unique to one set relative to another.	SQL: (SELECT * FROM researchers) EXCEPT (SELECT * FROM senior_researchers) SPARQL: via FILTER NOT EXISTS {...} or extensions
Advanced structures	Common table expressions (CTE)	Definition of reusable named subqueries within a main query improves modularity, readability and sometimes performance.	SQL: WITH recent_pubs AS (SELECT * FROM publications WHERE year > 2020), high_impact AS (SELECT * FROM publications WHERE impact > 5) SELECT * FROM recent_pubs JOIN high_impact...
	String manipulation	Application of text processing functions to transform extract or analyse textual data in queries.	SQL: WHERE UPPER(title) LIKE '%NEURAL%' OR SUBSTRING(abstract, 1, 10) = 'Deep learn' SPARQL: FILTER(UCASE(?title) = "NEURAL")
	Conditional expressions	Use of decision structures to define different values or behaviours according to specific conditions evaluated in the query.	SQL: SELECT CASE WHEN impact > 8 THEN 'Very high' WHEN impact > 5 THEN 'High' ELSE 'Standard' END AS impact_category SPARQL: BIND(IF(?impact > 8, "Very high", IF(?impact > 5, "High", "Standard")) AS ?impact_category)
Abstraction and reuse	Parameterizable query	Creation of generic queries with variables or parameters that can be defined at execution time allowing flexible reuse in different contexts.	Prepared SQL: WHERE domain = ? AND year BETWEEN ? AND ? AND author_institution = ? SPARQL parameterized via API: query with variables $domain, $startYear, $endYear
	Abstraction	Encapsulation of complex queries in reusable structures (views, stored procedures, named queries) that hide complexity and facilitate maintenance.	SQL: CREATE VIEW recent_high_impact_ai AS SELECT * FROM publications WHERE domain = 'AI' AND year > 2020 AND impact > 5 SPARQL: definition of named queries in a triplestore

Table 3: Query generation criteria definition

2.1.4 FR4. Result processing

Result processing focuses on transforming raw query results into meaningful, actionable information through effective visualization, documentation, and management capabilities. This requirement ensures that both the generated queries and their execution outcomes are communicated in ways that enhance understanding and facilitate further analysis. The system must provide clear representations of query structures, comprehensive export options, and interactive visualizations adapted to different data types. Furthermore, flexible execution capabilities enable deployment across diverse database architectures, from local single-source environments to

distributed heterogeneous systems, while maintaining a unified presentation of results. Through effective result processing, the system completes the knowledge cycle, transforming technical database queries into valuable insights for decision-making.

Table 4 presents the definitions of the criteria used for the evaluation.

Criterion	**Sub-criterion**	**Description**	**Example**
Visualization of generated queries	SQL/SPARQL code display	Formatted and structured presentation of the generated query code with syntax highlighting, automatic indentation and highlighting of main elements to facilitate reading and understanding.	Display of SQL code with formatting where each clause (SELECT, FROM, WHERE) is clearly identified by a distinct color, joins are indented and filtering conditions are logically structured with colored parentheses.
	Graphical representation	Conversion of the logical structure of the query into a visual diagram that shows the entities, their relationships, applied conditions and data flow. This helps users understand complex queries.	Diagram representing the "Publications" and "Authors" tables as nodes, their join as an edge with visual annotations.
Artefact export formats	Source code export	Saving generated queries in various textual formats adapted to different execution environments, database management systems or analysis tools.	Generation of .sql files for direct execution in MySQL, .rq for SPARQL endpoints, .yaml for system configuration or .json for API integration.
	Visual diagrams	Export of graphical query representations in vector or bitmap formats suitable for different documentation or presentation uses.	Saving query diagrams in SVG for vector editing, PNG for web integration, PDF for printing or DOT format for use with Graphviz.
	Associated metadata	Inclusion of contextual information with exported queries documenting their origin, purpose, parameters and relationship with the source ontology.	JSON file accompanying the query containing metadata such as creation date, creator, user, ontological concepts used and version of the reference ontology.
	Complete documentation	Automatic generation of comprehensive explanatory documents integrating query code, diagram explanations and contextual information necessary for understanding and use.	Structured PDF documents including commented SQL/SPARQL code, explanatory diagrams, a glossary of ontological terms used, and examples of expected results.
Visualization of execution results	Tabular display	Presentation of the query results in a tabular form with advanced features of pagination, dynamic sorting, filtering and display customization.	Tabular interface with resizable columns, sorting by clicking on headers, quick filters by column, configurable pagination and conditional highlighting of important values.

	Textual representations	Formatting of textual results with display options adapted to the nature of the data and hierarchical relationships between elements.	Tree presentation of results showing the hierarchy "Institution > Department > Researcher > Publications" with the ability to collapse/expand levels and rich text formatting.
	Interactive visualizations	Automatic transformation of tabular data into interactive graphical representations adapted to the nature of the data and facilitating visual analysis.	Automatic generation of charts adapted to the data: histograms for temporal distributions of publication, network graphs for collaborations between researchers or heat maps for co-citation matrices.
Result export options	Structured formats	Export of results in standardized data formats facilitating their reuse in other analysis tools or applications.	Export of results in CSV for analysis in spreadsheets, JSON for API integration, XML for interoperable exchange or RDF to preserve semantic links with the ontology.
	Document formats	Conversion of results into formatted documents ready to be shared, integrated into reports or presented to different audiences with a professional layout.	Generation of PDF reports with header, pagination and advanced formatting and Excel sheets with integrated formulas and charts.
Execution capabilities	Local execution - same DBMS	Ability to execute generated queries on a single locally installed database with direct access and complete connection management.	Interface allowing configuration and execution of queries on a locally installed PostgreSQL database or GraphDB with execution option parameters and direct result visualization.
	Local execution - multiple DBMS	Ability to execute queries on multiple locally installed databases or triplestores with automatic consolidation of results from these different sources.	System capable of simultaneously querying a local PostgreSQL database and a local GraphDB then merging the results into a unified presentation.
	Distributed execution - same DBMS	Support for the distribution and parallelism of a complex query across multiple nodes or clusters of the same DBMS.	Ability to execute queries on a distributed or remotely hosted database management system efficiently managing network connections and communication with the remote service.
	Distributed execution - multiple DBMS	Ability to simultaneously query different distributed or hosted database systems, translates queries into the respective languages of each system and integrates results from heterogeneous sources.	System executing an ontological query that simultaneously queries an SQL service hosted on AWS, a public SPARQL endpoint and a NoSQL database in the cloud combining the results in a coherent view.

Table 4: Result processing criteria definition

2.2 Non-Functional requirements

Non-functional requirements address the technical and operational characteristics that influence a system's adoption, integration, and long-term sustainability.

2.2.1 NFR1. Availability and maintenance

Availability and maintenance focus on ensuring the system remains accessible, reliable, and adaptable throughout its lifecycle. This requirement encompasses distribution models that balance openness and commercialization, comprehensive documentation and learning resources, consistent update cycles, and active community engagement. By providing clear implementation guidance, responsive support channels, and stable development technologies, the system establishes a foundation for sustainable growth and evolution. This technical sustainability is essential for organizations making strategic investments in semantic data integration, ensuring that the ontology-based querying solution delivers long-term value as both business needs and technological landscapes evolve.

Table 5 presents the definitions of the criteria used for the evaluation.

Criterion	**Sub-criterion**	**Description**	**Example**
Distribution mode	Open source	Source code is freely accessible, allowing modification, adaptation and redistribution according to license terms.	Project published on GitHub with MIT or Apache 2.0 license allowing commercial use and creation of derivatives.
	Proprietary with API	Solution exposing programming interfaces enabling integration with other systems.	System with a documented REST API allowing the integration of query generation functionalities into third-party applications.
	Commercial license	Proprietary solution requiring license purchase for use and deployment.	Commercial software with different license levels (enterprise, developer, academic) determining accessible features.
Support and maintenance	Technical documentation	Availability of comprehensive documentation covering architecture, API functionalities and deployment procedures.	Online documentation with installation guides, complete API references, architecture diagrams and commented code examples.
	Tutorials and examples	Presence of practical guides and concrete examples to facilitate onboarding and use of the solution.	Collection of progressive tutorials ranging from basic installation to advanced use cases with code examples and test ontologies.
	Regular updates	Frequency and quality of corrective and evolutionary updates of the solution.	Quarterly release cycle with monthly bug fixes, detailed release notes and long-term support policies.
Community support	Active community	Presence of a community of users and developers contributing to the support and evolution of the solution.	Active forum with more than 100 messages per month, regular code contributions and meetups organized by users.

	Community platforms	Existence of communication and mutual aid channels between users and developers.	Dedicated Stack Overflow tag, Discord/Slack channel, mailing list and GitHub repository with active issue tracking.
Development technology	Programming language	Programming language(s) used to implement the solution, which may influence performance, portability and ease of extension.	Java, Python, C++, JavaScript, Scala, etc.

Table 5: Availability and maintenance criteria definition

2.2.2 NFR2. Compatibility

Compatibility and interoperability determine the system's ability to function effectively across diverse technological environments and integrate seamlessly with existing infrastructure. This requirement ensures support for varied database management systems—from traditional relational databases to modern NoSQL and semantic stores—while maintaining consistent behavior across different operating platforms and deployment architectures. Through standardized interfaces, protocols, and data formats, the system facilitates smooth communication with external applications and services. This technological flexibility enables organizations to implement ontology-based querying without disrupting established data ecosystems, protecting existing investments while enhancing analytical capabilities through semantic enrichment.

Table 6 presents the definitions of the criteria used for the evaluation.

Criterion	**Sub-criterion**	**Description**	**Example**
Supported DBMS	Relational databases	Support for major relational database management systems on the market.	Native connectors for MySQL, PostgreSQL, Oracle, SQL Server, SQLite, etc., with automatic handling of specific SQL dialects.
	NoSQL databases	Ability to interact with document-oriented, key-value, columns or graph databases.	Support for MongoDB, Cassandra, Neo4j, etc., with automatic query adaptation to the specificities of each data model.
	Triplestores	Support for RDF-oriented databases for the storage of ontologies and data.	Native integration with Apache Jena, GraphDB, Virtuoso, AllegroGraph with full support for SPARQL 1.1 standards.
Portability	Multi-platform	Ability to function on different operating systems without major modification.	Identical installation and execution on Windows, Linux (various distributions), macOS with adapted installation scripts.

	Cloud deployment	Support for cloud deployment architectures and containers.	Available as Docker images with CloudFormation templates for AWS and Kubernetes configurations for scalable deployment.
Integration	Available APIs	Programming interfaces allowing the integration of the solution into existing environments.	REST API with OpenAPI/Swagger documentation, SDK client for Java, Python, JavaScript, and GraphQL support for advanced queries.

Table 6: Compatibility and interoperability criteria definition

2.2.3 NFR3. Interoperability of query languages

Interoperability of query languages addresses the linguistic interfaces through which users and systems communicate information needs and retrieve results. This requirement encompasses both input languages that capture user requests through various paradigms (natural, visual, domain-specific) and output languages that generate executable queries for different database technologies. The translation mechanisms connecting these linguistic interfaces represent the critical intelligence that preserves semantic fidelity across representational boundaries. By supporting diverse query languages and formats, the system accommodates users with varying technical backgrounds while enabling integration with heterogeneous data storage systems, serving as a universal semantic interpreter across the enterprise data landscape.

Table 7 presents the definitions of the criteria used for the evaluation.

Criterion	**Sub-criterion**	**Description**	**Example**
Input languages	Formulation languages	Diversity of languages accepted for the initial formulation of user requests.	Support for natural language, SQL-like syntax, SPARQL expressions, custom DSL and visual interfaces for query construction.
Output languages	Generated languages	Variety of query languages that can be produced by the solution for different target systems.	Generation of standard SQL and specific dialects (T-SQL, PL/SQL), SPARQL 1.1 compatible, Cypher for Neo4j and API calls for REST systems.

Table 7: Query language support criteria definition

2.2.4 NFR4. Extensibility

Extensibility determines the system's capacity to adapt to evolving requirements without requiring a fundamental redesign. This requirement ensures the system can incorporate new functionalities through modular extension mechanisms, adjust behavior through configuration rather than coding, and customize operations for specific domains or use cases. By providing well-defined extension points, plugin architectures, and declarative configuration options, the system enables organizations to tailor capabilities to their unique needs while maintaining core stability. This architectural adaptability is essential for long-term viability, allowing the ontology-based

querying solution to evolve alongside changing business requirements and emerging technologies.

Table 8 presents the definitions of the criteria used for the evaluation.

Criterion	Sub-criterion	Description	Example
Functional extensions	Addition of features	Ability to extend the basic capabilities of the system with new functionalities.	Framework allowing the addition of new traversal algorithms, visualization methods or export formats without modification of the main source code.
	Custom configuration	Ability to adapt system behaviour to specific needs without development.	YAML/JSON configuration files allowing the definition of business rules, traversal priorities, custom mappings and query templates.

Table 8: Flexibility and extensibility criteria definition

3 Methodology

3.1 Search procedure

To construct a representative overview, we conducted a structured, multi-phase literature review. Our objective was to identify research that directly addresses the challenges of query formulation over structured data sources, particularly those involving ontologies, relational databases, RDF representations, and semantic data access layers.

The process began with the formulation of a set of thematic keyword group. These were designed to reflect the conceptual domains relevant to the study. Specifically, the group of querying concepts included terms such as Query, Queries, and Querying combined with variants like Generator, Model, Builder, Creation, Rewriting, or Faceted. For the query language group, we focused on SQL, SPARQL, and RDF, while the ontology frameworks group captured terms such as Semantic Web, Ontology, OWL, Knowledge Graph, and DL (Description Logic). In parallel, a database context group targeted terminology around RDBMS and Relational Database, and a final tools and systems axis included generic descriptors like Tool, System, Algorithm, Interface, Comparison, Benchmark, Software, or Product. These categories are illustrated in Figure 1.
To ensure comprehensive coverage, we consulted both established academic repositories and AI-powered discovery platforms. Primary sources included Scopus, IEEE Xplore, ScienceDirect, and the ACM Digital Library. In parallel, we leveraged specialized semantic discovery tools such as Elicit, Connected Papers, Litmaps, and Research Rabbit to uncover emerging or interdisciplinary works. Additionally, Google Scholar was used to broaden the scope and capture literature that might not be indexed in traditional domain-specific databases.

The results of this search were imported into Covidence, a systematic review platform adopted by Sherbrooke University. This tool facilitated the deduplication, tracking, and collaborative screening of references. All retrieved entries were evaluated based on predefined inclusion criteria to ensure consistency and relevance.

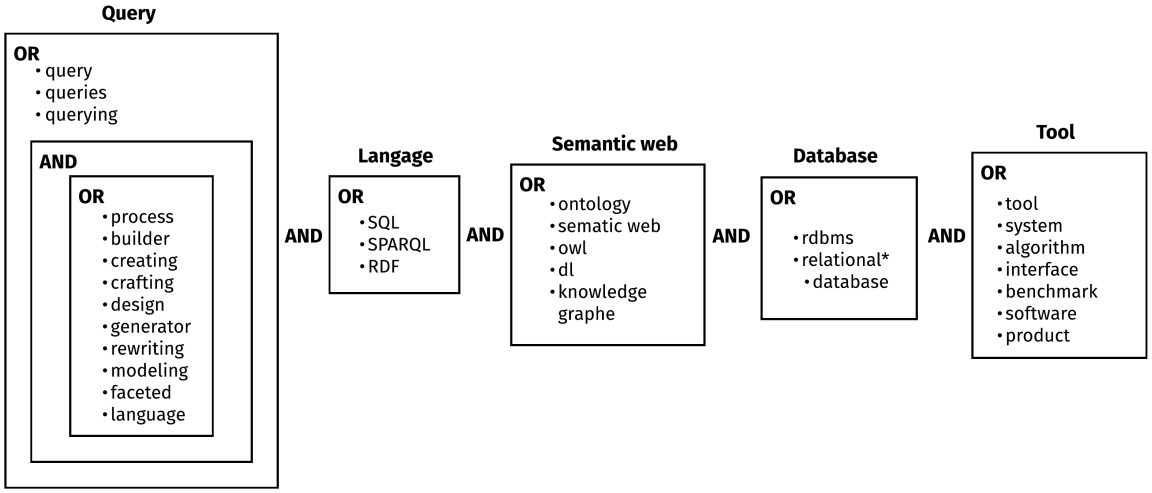

Figure 1: Keywords and thematic categories

3.2 Inclusion criteria

To maintain both methodological rigor and topical relevance, a set of clear inclusion filters was applied. First, only works published between 2013 and 2023 were retained, ensuring alignment with contemporary developments in semantic technologies and query interfaces.

Second, we prioritized publications that demonstrated scholarly reliability, with preference given to peer-reviewed journal articles and conference proceedings. This focus on scientific quality was intended to filter out speculative or low-impact contributions.

Third, accessibility was essential: only studies available in full text were eligible for in-depth examination, as abstract-only entries or inaccessible works could not be reliably assessed.

Lastly, we adopted a bilingual approach, accepting contributions written in either English or French. This decision reflects the linguistic diversity of our research context and ensures inclusivity in the sources considered.

3.3 System selection criteria

From an initial pool of 42 relevant articles identified through our literature review, we selected six representative solutions for detailed evaluation. This selection was guided by four primary criteria: (1) scientific impact, measured by citation count and publication venue prestige; (2) technological diversity, ensuring representation of different architectural approaches to ontology-based querying; (3) active development status, with preference given to systems with updates within the past three years; and (4) documentation completeness, enabling thorough assessment of capabilities and limitations.

The selected systems Ontop, OntoGrate, DAFO, VOn-QBE, ATHENA++, and OptiqueVQS collectively represent the state of the art across different paradigms of ontology-based database access, from visual query interfaces to natural language processing approaches and purely semantic integration frameworks.

3.4 Evaluation methodology

To ensure systematic and objective assessment, we developed a multi-phase evaluation process:

1. **Criteria definition phase.** Each functional and non-functional requirement was decomposed into measurable criteria and sub-criteria with explicit definitions and examples, as detailed in Section 2.

2. **Evidence collection phase.** For each system, we gathered documentation from three sources: (a) peer-reviewed publications by system authors, (b) official technical documentation and user manuals, and (c) publicly available source code repositories when accessible.

3. **Scoring phase.** Each sub-criterion was evaluated on a binary scale (0/1) indicating whether the system fully implements the capability. This evaluation was conducted independently by two researchers, with disagreements resolved through consensus discussion.

4. **Validation phase.** Where possible, we validated documentary findings through practical testing of available system implementations.

This methodology ensures that our comparative analysis provides an accurate and defensible assessment of each system's capabilities while minimizing subjective bias in the evaluation process.

3.5 Overview of the solutions

This section presents a synthesis of the main approaches developed to facilitate access and querying of relational databases. The reviewed systems exemplify different paradigms—visual interfaces, faceted navigation, natural language interfaces, and ontology-based data access—each addressing distinct challenges in terms of usability, query expressiveness, and integration with ontologies. We summarize the core characteristics of each solution below:

– **Ontop** [1]: An OBDA system translating SPARQL queries into SQL via precompiled T-mappings from OWL 2 QL ontologies. It ensures performant access to relational data without RDF materialization, but struggles with query complexity growth and partial support for SPARQL 1.1 features such as aggregations and property paths.

– **OntoGrate** [2]: An ontology-based data integration framework supporting automated mapping and query translation between relational databases and the semantic web. It enables reasoning-based SPARQL-to-SQL transformation but faces scalability and alignment challenges when source schemas lack overlap.

– **DAFO** [3]: A faceted query interface that maps ontological structures into relational schemas. It enables users to formulate queries via interaction with a tree-based interface, which is then transformed into First-Order Faceted Queries and rewritten into SQL. The system prioritizes logical rigor but does not support disjunction, negation, or highly expressive ontologies.

– **VOn-QBE** [4]: A schema-driven QBE system that constructs SPARQL queries based on ontology TBox without relying on instance data. It is suitable for OBDA contexts and large-scale RDF stores but offers limited expressiveness—unable to support instance-specific queries, disjunction, or SPARQL aggregates.

– **ATHENA++** [5]: A natural language interface designed to parse and translate complex nested queries into executable SQL. It uses semantic annotation and classification of query types to handle constructs like aggregation and correlation. Its effectiveness declines in ambiguous or linguistically implicit scenarios.

– **OptiqueVQS** [6]: A visual query system based on ontologies, designed for non-technical users. It employs a multi-widget interface guiding the user through query construction using semantic navigation and visual representation. While it supports linear and tree-shaped conjunctive queries, it currently lacks support for negation, universal quantification, and complex aggregations.

3.6 Functional criteria evaluation

The functional evaluation assesses how effectively each solution implements the core capabilities required for ontology-based database querying. We examine them across four essential functional areas: ontology processing, user request formulation, query generation, and result processing.

3.6.1 FR1. Ontology processing

Criterion	Sub-criterion	OntoRelQuery	ATHENA++	VON-QBE	Optique-VQS	DAFO	OntoGrate	Ontop
Ontology graph traversal	Path traversal	✓	✓	✓	✓	✓	✓	✓
	Traversal algorithms	✓	✓	✓	-	-	✓	✓
	Variable path exploration	✓	✓	✓	✓	✓	✓	✗
Structural analysis	Detection of ontological components	✓	✓	✓	✓	✓	✓	✓
	Pattern identification	✗	✗	✗	?	?	?	?
	Cycle detection	✓	✓	✗	✓	?	?	?
Traversal optimization	Customized traversal algorithms	✓	✗	✓	?	-	✓	✓
	Ontological graph reasoning	✗	✓	✓	✓	✓	✓	✓
Nb. Total (/8)		**6**	**6**	**6**	**5**	**4**	**6**	**5**

Figure 2: Ontology processing capabilities evaluation

As shown in Table 2, most solutions demonstrate strong capabilities in basic ontology traversal, particularly path traversal and detection of ontological components. However, more advanced features like pattern identification and customized traversal algorithms show greater variation. OntoRelQuery, ATHENA++, VON-QBE, and OntoGrate all achieved scores of 6 out of 8, indicating comprehensive support for ontology processing, while DAFO scored lowest at 4 points, suggesting limitations in its ability to handle complex ontological structures.

3.6.2 FR2. User request formulation

The ability to effectively capture user information needs through intuitive interfaces represents an important dimension of system usability:

Criterion	Sub-criterion	OntoRelQuery	ATHENA++	VON-QBE	Optique-VQS	DAFO	OntoGrate	Ontop
Request expression methods	By graphical interface	✗	✗	✗	✓	✓	✗	✓
	Keyword-based system	✗	✗	✓	✗	✓	✗	✗
	By guided question-answer (Q&A)	✓	✗	✗	✗	✗	✗	✗
	By natural language (NL)	✗	✓	✓	✗	✗	✗	✗
	By custom language	✓	?	✗	✗	✗	✓	✗
Request formulation assistance	Contextual auto-completion	✓	Ø	Ø	✓	Ø	Ø	Ø
	Semantic suggestions	✗	Ø	✓	✓	✓	Ø	Ø
	Entity recognition	✓	✓	✓	✓	✓	✓	✓
	Interactive visualizations during	✓	✗	✗	✓	✗	?	✗
	Syntactic error detection and reporting	✓	✗	?	?	✗	?	✓
	Semantic error detection and reporting	✓	✓	?	?	✗	?	✓
Progressive request formulation	Iterative formulation	✓	✗	✓	✓	✓	?	Ø
	History management	✗	-	✗	✓	?	?	Ø
Nb. Total (/13)		**8**	**3**	**5**	**7**	**5**	**2**	**4**

Figure 3: User request formulation capabilities evaluation

Table 3 demonstrates considerable diversity in approaches across the evaluated systems. OntoRelQuery and OptiqueVQS offer the most comprehensive request formulation support (scoring 8 and 7 points respectively), but through different mechanisms—OntoRelQuery excels in guided question-answer and assistance features, while OptiqueVQS focuses on graphical interfaces with strong visualization support. ATHENA++ specializes in natural language processing but lacks broader formulation assistance capabilities. This diversity highlights the different philosophical approaches to bridging the gap between user information needs and formal query languages.

3.6.3 FR3. Query generation

Query generation represents the technical core of these systems, translating conceptual information needs into executable database queries:

Criterion	Sub-criterion	OntoRelQuery	ATHENA++	VON-QBE	Optique-VQS	DAFO	OntoGrate	Ontop
Data restriction capabilities	Simple restrictions	✓	✓	✓	✓	✓	✓	✓
	Composite restrictions	✓	✓	?	✓	✓	✓	✓
	Pattern-based restriction	✓	✓	?	✓	✓	-	Ø
	Set-based restriction	✓	✓	?	✓	✓	?	✓
Data selection	Selective projection	✓	✓	✓	✓	✓	✓	✓
	Element renaming	✓	✓	✓	✗	✓	✗	✓
	Ordering	✓	✓	✗	✗	✗	✗	✓
	Result limitation	✓	✗	✗	✗	✗	✗	✓
Analytical functions	Data aggregation	✓	✓	✗	✗	?	?	✓
	Grouping	✓	✓	✗	✗	?	?	✓
	Conditions on aggregates	✓	✓	✗	✗	?	?	✓
	Window function	✗	✗	✗	✗	?	?	Ø
Join operations	Joins	✓	✓	✓	✓	✓	✓	✓
	Outer joins	✓	✓	✗	?	?	?	✓
	Existential operators	✓	✓	✗	✗	✓	?	Ø
	Subqueries	✓	✓	✗	✗	?	✓	✓
Set operations	Union	✓	✓	✗	✗	✓	✓	✓
	Intersection	✓	✓	✗	✗	?	?	Ø
	Difference	✓	✓	✗	✗	?	?	-
Advanced structures	Common table expressions (CTE)	✓	-	✗	Ø	✗	Ø	Ø
	String manipulation	✗	-	✗	Ø	?	Ø	-
	Conditional expressions	✗	✗	✗	Ø	?	Ø	Ø
Abstraction and reuse	Parameterizable query	✗	✗	Ø	Ø	?	-	Ø
	Abstraction	✗	✗	Ø	✓	?	✗	Ø
Nb. Total (/24)		**19**	**17**	**4**	**7**	**9**	**6**	**14**

Figure 4: Query generation capabilities evaluation

Table 4 reveals OntoRelQuery and ATHENA++ as the most comprehensive solutions for query generation, scoring 19 and 17 points respectively out of 24 possible points. Both systems demonstrate robust capabilities across the spectrum of query operations, including advanced data restrictions, analytical functions, join operations, and set operations. In contrast, VON-QBE shows significant limitations, supporting only basic query operations with a score of just 4 points. Ontop represents a middle ground with solid implementation of core functionality (14 points) but limited support for advanced query features.

3.6.4 FR4. Result processing

The final functional dimension examines how systems present query results to support further analysis:

Criterion	Sub-criterion	OntoRelQuery	ATHENA++	VON-QBE	Optique-VQS	DAFO	OntoGrate	Ontop
Visualization of generated queries	SQL/SPARQL code display	✓	✓	✓	✓	✓	?	✓
	Graphical representation	✓	-	✗	✓	✓	?	?
Artifact export formats	Source code export	✓	✓	?	✓	?	?	✓
	Visual diagrams	✓	✗	?	?	?	?	Ø
	Associated metadata	✓	✗	?	?	?	?	Ø
	Complete documentation	✓	✗	?	?	?	?	Ø
Visualization of execution results	Tabular display	✓	✓	?	?	?	?	✓
	Textual representations	✗	✗	?	?	?	?	✓
	Interactive visualizations	✗	✗	?	?	?	?	Ø
Result export options	Structured formats	✓	?	?	?	?	?	Ø
	Document formats	✓	?	?	?	?	?	Ø
Execution capabilities	Local execution - same DBMS	✓	✓	?	✓	✓	✓	✓
	Local execution - multiple DBMS	✗	?	?	✓	?	✓	✓
	Distributed execution - same DBMS	✓	?	?	?	?	?	✓
	Distributed execution - multiple DBMS	✗	?	?	?	?	?	✓
Nb. Total (/15)		**11**	**4**	**1**	**5**	**3**	**2**	**8**

Figure 5: Result processing capabilities evaluation

Table 5 shows OntoRelQuery leading in result processing capabilities with 11 out of 15 possible points, particularly excelling in artefact export formats and result visualization. Ontop follows with 8 points, showing strength in execution capabilities but limited support for visualization and export options. Most other solutions demonstrate significant limitations in this area, with VON-QBE scoring just 1 point, suggesting minimal attention to how users interact with query results.

3.7 Non-functional criteria evaluation

The non-functional evaluation examines the technical qualities that influence adoption, integration, and long-term sustainability of ontology-based database querying solutions. We assess these solutions across three key dimensions: availability and maintenance, compatibility of technologies, and flexibility and extensibility.

3.7.1 NFR1. Availability and maintenance

Criterion	Sub-criterion	OntoRelQuery	ATHENA++	VON-QBE	Optique-VQS	DAFO	OntoGrate	Ontop
Distribution mode	Open source	✓	✗	✓	✓	✗	✗	✓
	Proprietary with API	✗	✗	✗	✗	✗	✗	✓
	Commercial license	✗	✓	✗	✗	✗	✗	✗
Support and maintenance	Technical documentation	✓	✓	✓	✓	✓	✓	✓
	Tutorials and examples	✓	?	✓	✓	✓	✓	✓
	Regular updates	-	?	?	?	✗	?	✓
Community support	Active community	-	✗	-	-	✗	✗	✓
	Community platforms	GitHub	✗	GitHub	GitHub	✗	✗	GitHub
Development technology	Programming language	Java	Java	Java	Java	C#	Java	Java
Nb. Total* (/9)		**5**	**3**	**5**	**5**	**3**	**3**	**8**

Figure 6: Availability and maintenance capabilities evaluation

As shown in Table 6, Ontop demonstrates the highest score in availability and maintenance with 8 out of 9 possible points, providing comprehensive technical documentation, regular updates, and strong community support. OntoRelQuery, Von-QBE, and OptiqueVQS each scored 5 points, primarily due to their open-source distribution model and adequate documentation resources. However, all three lack evidence of regular updates and active community engagement.

ATHENA++, DAFO, and OntoGrate show significant limitations in this area, scoring just 3 points each, primarily due to restricted distribution modes and minimal community support infrastructure.

3.7.2 NFR2. Compatibility of technology

Criterion	Sub-criterion	OntoRelQuery	ATHENA++	VON-QBE	Optique-VQS	DAFO	OntoGrate	Ontop
Supported DBMS	Relational databases	✓	✓	-	✗	✓	✗	✓
	NoSQL databases	✗	✗	✗	✗	✗	✗	✗
	Triplestores	✗	✗	✓	✓	✗	✓	✓
Portability	Multi-platform	✓	✓	✓	✓	?	?	✓
	Cloud deployment	-	ø	ø	ø	ø	ø	ø
Integration	Available APIs	✗	✗	✗	✗	✗	✗	✗
Nb. Total*(/6)		**2**	**2**	**2**	**2**	**1**	**1**	**3**

Figure 7: Technology compatibility capabilities evaluation

Table 7 reveals Ontop as the most versatile solution in terms of technology compatibility, scoring 3 out of 6 points with support for both relational databases and triplestores along with multi-platform capabilities. Most other solutions demonstrate limited compatibility, scoring 2 points each with support primarily for relational databases and basic multi-platform functionality. DAFO and OntoGrate exhibit the most restricted compatibility profiles with scores of only 1 point, suggesting significant integration challenges in heterogeneous environments. Notably, none of the evaluated solutions provide adequate support for NoSQL databases, cloud deployment, or comprehensive API integration, indicating an industry-wide gap in addressing modern distributed data architectures.

3.7.3 NFR3. Interoperability of query languages

Table 8 presents the language capabilities of each evaluated solution, documenting both the input interfaces through which users can express queries and the output formats generated for execution.

Criterion	Sub-criterion	OntoRelQuery	ATHENA++	VON-QBE	Optique-VQS	DAFO	OntoGrate	Ontop
Input languages	Formulation languages	Questions & Answers and ORQL*	Natural Language	Natural Language and Keyword-based Search	Faceted search	Keyword-based Search	SPARQL	SPARQL
Output languages	Generated languages	SQL	SQL	SPARQL	SPARQL	SQL	SQL	SQL

Figure 8: Interoperability of query languages evaluation

[1]ORQL (OntoRelQuery Language) is a custom domain-specific language developed specifically for OntoRelQuery, based on a formal grammar that enables precise query formulation while abstracting database complexity. The complete language specification and grammar definitions are available in the project's documentation repository at `https://github.com/OPENLHS/OntoRelQuery/doc/grammar.adoc`.

3.7.4 NFR4. Extensibility

Criterion	Sub-criterion	OntoRelQuery	ATHENA++	VON-QBE	Optique-VQS	DAFO	OntoGrate	Ontop
Functional extensions	Addition of features	✓	?	✓	✓	?	?	✓
	Custom configuration	✓	?	?	✓	?	?	✓
Nb. Total* (/2)		**2**	**0**	**1**	**2**	**0**	**0**	**2**

Figure 9: Flexibility and extensibility capabilities evaluation

The assessment of flexibility and extensibility in Table 9 shows that OntoRelQuery, OptiqueVQS, and Ontop lead with perfect scores of 2 out of 2 points, providing both feature extension mechanisms and custom configuration capabilities. Von-QBE demonstrates partial flexibility with support for feature additions but limited configuration options, scoring 1 point. ATHENA++, DAFO, and OntoGrate show concerning limitations in this area, offering neither modular extension capabilities nor significant configuration options. This rigidity suggests these solutions may struggle to adapt to evolving requirements or specialized use cases without substantial redevelopment.

4 Results discussion

This section presents a comprehensive analysis of our comparative evaluation of the different solutions. We examine both functional and non-functional aspects, identify strengths and limitations, and position each solution strategically within the current technological landscape.

4.1 Analysis of functional requirements

Our evaluation of functional requirements reveals significant variations in implementation approaches and capabilities across the assessed solutions. Table 9 presents the detailed scores for each solution across the four functional dimensions.

Table 9: Functional requirements scores

Req.	OntoRelQuery	ATHENA++	Von-QBE	OptiqueVQS	DAFO	OntoGrate	Ontop
FR1 (/8)	6	6	6	5	4	6	5
FR2 (/13)	8	3	5	7	5	2	4
FR3 (/24)	19	17	4	7	9	6	14
FR4 (/15)	11	4	1	5	3	2	8
Total	**44**	**30**	**16**	**24**	**21**	**16**	**31**

4.1.1 FR1. Ontology processing

For ontology processing capabilities (FR1), the scores reveal a notably consistent pattern among several solutions. OntoRelQuery, ATHENA++, Von-QBE, and OntoGrate all achieved identical scores of 6/8 points, demonstrating robust capabilities in ontology traversal. A detailed

examination of sub-criteria reveals that all four solutions excel at basic path traversal and traversal algorithm implementation, with OntoRelQuery and Von-QBE additionally providing strong support for variable path exploration. OptiqueVQS and Ontop follow closely with 5/8 points, while DAFO lags with 4/8 points.

The detailed criteria analysis reveals that OntoRelQuery achieves its high score through balanced implementation across all ontology processing dimensions, particularly excelling in cycle detection and customized traversal algorithms—capabilities critical for navigating complex ontological structures efficiently. This contrasts with ATHENA++ which compensates for weaker pattern identification with robust ontological reasoning capabilities, and Von-QBE which excels at traversal algorithms but lacks effective pattern identification.

4.1.2 FR2. User request formulation

The query design dimension (FR2) reveals a more substantial differentiation between solutions. OntoRelQuery leads with 8/13 points, closely followed by OptiqueVQS with 7/13 points. Both solutions, however, achieve these scores through markedly different approaches: OntoRelQuery excels in guided question-answer interaction and advanced request formulation assistance (specifically in contextual auto-completion, interactive visualizations, and error detection), while OptiqueVQS prioritizes graphical interfaces complemented by strong semantic suggestions. Von-QBE and DAFO occupy the middle tier with 5/13 points each, though they emphasize different capabilities—Von-QBE leverages keyword-based systems and natural language processing, while DAFO focuses on graphical interfaces with limited assistance features.

The substantial gap between the leading solutions and ATHENA++ (3/13), Ontop (4/13), and particularly OntoGrate (2/13) indicates a significant limitation in user interface capabilities that could restrict their adoption in contexts where non-technical users are primary stakeholders. The sub-criteria reveal that OntoGrate's particularly low score stems from supporting only custom languages while lacking both graphical interfaces and natural language capabilities, severely limiting its accessibility to non-specialist users.

4.1.3 FR3. Query generation

Query generation (FR3) demonstrates the most substantial performance differentiation, with OntoRelQuery leading at 19/24 points and ATHENA++ following closely at 17/24 points . The detailed criterion analysis reveals that OntoRelQuery achieves comprehensive implementation across all query operation categories: data restriction (supporting all four restriction types), data selection (implementing selective projection, element renaming, ordering, and result limitation), analytical functions (providing full aggregation, grouping, and aggregate condition capabilities), join operations (supporting all join types and subqueries), and set operations (implementing union, intersection, and difference operations). ATHENA++ shows similar breadth but with slightly reduced capabilities in advanced structures.

Ontop occupies a middle position with 14/24 points, demonstrating solid implementation of basic query operations but limited support for advanced analytical functions and set operations. DAFO (9/24), OntoGrate (6/24), OptiqueVQS (7/24), and particularly Von-QBE (4/24) show significant limitations in query generation capabilities. Von-QBE's particularly low score stems from supporting only basic data restrictions and joins while lacking the features for analytical functions, set operations, and advanced structures entirely—severely limiting its utility for complex analytical queries.

4.1.4 FR4. Result processing

Result processing capabilities (FR4) reveal the most significant gap between solutions, with OntoRelQuery substantially outperforming all competitors at 11/15 points. Detailed sub-criteria analysis shows OntoRelQuery's comprehensive implementation of visualization capabilities (SQL/SPARQL code display and graphical representation), artefact export formats (supporting all four export types), and structured result formats. Ontop follows at a distance with 8/15 points, showing strength primarily in execution capabilities across different database systems but limited support for visualization and export options.

OptiqueVQS (5/15), ATHENA++ (4/15), DAFO (3/15), and OntoGrate (2/15) demonstrate substantial limitations in this dimension, while Von-QBE scores just 1/15 point, supporting only basic SQL/SPARQL code display without any capabilities for result visualization, export, or flexible execution. This pattern indicates that most solutions have prioritized query formulation and generation over the equally important post-execution phases of the query lifecycle—a critical gap considering that effective result presentation is essential for knowledge discovery and decision-making.

4.1.5 Overall

The aggregated functional scores position OntoRelQuery as the clear leader with 44/60 points, demonstrating not only the highest total score but also the most balanced performance profile across all functional dimensions. OntoRelQuery ranks first in user request formulation (tied with OptiqueVQS), query generation, and result processing, while achieving a joint-first position in ontology processing. This consistent excellence across the entire query lifecycle distinguishes OntoRelQuery from solutions that excel in specific functional areas but demonstrate significant limitations in others.

Ontop (31/60) and ATHENA++ (30/60) form a second tier with complementary strengths: ATHENA++ excels in query generation (17/24) but shows substantial limitations in user interface design (3/13) and result processing (4/15), while Ontop provides more balanced functionality across all dimensions without exceptional performance in any specific area. OptiqueVQS (24/60), DAFO (21/60), Von-QBE (16/60), and OntoGrate (16/60) demonstrate significant functional limitations that restrict their utility for comprehensive ontology-based database querying workflows.

4.2 Analysis of non-functional requirements

The evaluation of non-functional requirements examines critical technical qualities that determine practical deployment viability and long-term sustainability. Table 10 presents the detailed scores for each solution across the three assessed non-functional dimensions.[2]

4.2.1 NFR1. Availability and maintenance

In the availability and maintenance dimension (NFR1), Ontop demonstrates exceptional performance with 8/9 points. The detailed sub-criteria analysis reveals that Ontop achieves this through comprehensive implementation of all assessed capabilities: open-source distribution, thorough technical documentation and tutorials, regular updates, and active community support through both formal and informal channels. This robust maintenance infrastructure suggests a

[2]The evaluation omits NFR3 (Query language support) from the formal scoring and analysis, as this requirement primarily represents a descriptive classification of input and output language capabilities rather than a comparative quality metric. While these capabilities were documented during our assessment, they did not contribute to the quantitative evaluation.

Table 10: Non-functional requirements scores

Req.	OntoRelQuery	ATHENA++	Von-QBE	OptiqueVQS	DAFO	OntoGrate	Ontop
NFR1 (/9)	5	3	5	5	3	3	8
NFR2 (/6)	2	2	2	2	1	1	3
NFR4 (/2)	2	0	1	2	0	0	2
Total	**9**	**5**	**8**	**9**	**4**	**4**	**13**

mature, stable solution with strong potential for long-term sustainability.

OntoRelQuery, Von-QBE, and OptiqueVQS form a second tier with identical scores of 5/9 points. All three solutions employ open-source distribution models and provide adequate technical documentation and tutorials. However, they show limitations in regular update cycles (OntoRelQuery and OptiqueVQS) and community support mechanisms (all three). The sub-criteria analysis indicates that OntoRelQuery provides GitHub-based community platforms but lacks evidence of active community engagement beyond this basic infrastructure.

ATHENA++, DAFO, and OntoGrate score just 3/9 points, indicating significant concerns regarding their long-term viability. ATHENA++ employs a commercial license model with limited documentation and no evidence of community support, while DAFO and OntoGrate demonstrate similar limitations despite offering technical documentation. This pattern suggests that most solutions have not yet established the robust maintenance infrastructure and active community engagement necessary for sustainable evolution in a rapidly changing technological landscape.

4.2.2 NFR2. Technology compatibility

The technology compatibility assessment (NFR2) reveals an industry-wide limitation in supporting diverse database technologies and deployment architectures. Ontop leads narrowly with 3/6 points, supporting both relational databases and triplestores along with multi-platform capabilities, but lacking NoSQL database support, cloud deployment options, and comprehensive API integration.

OntoRelQuery, ATHENA++, Von-QBE, and OptiqueVQS all score 2/6 points, typically supporting relational databases (except Von-QBE, which focuses on triplestores) and offering basic multi-platform functionality, but showing limited or no support for other database types, cloud deployment, or API integration. DAFO and OntoGrate score just 1/6 points, demonstrating the most restricted compatibility profiles.

The sub-criteria analysis for OntoRelQuery shows support for relational databases and multi-platform deployment but reveals limitations in NoSQL and triplestore integration—a significant gap considering the increasing importance of diverse database technologies in contemporary data architectures. This limited compatibility could restrict OntoRelQuery's utility in heterogeneous data environments that leverage multiple database paradigms.

4.2.3 NFR3. Interoperability of query languages

The most used languages are SQL and SPARQL.

4.2.4 NFR4. Extensibility

In the flexibility and extensibility dimension (NFR4), OntoRelQuery, OptiqueVQS, and Ontop achieve perfect scores of 2/2 points, demonstrating comprehensive implementation of both extension mechanisms and configuration capabilities. This architectural flexibility enables these solutions to adapt to specialized domain requirements and evolving technological landscapes without requiring substantial redevelopment.

Von-QBE demonstrates partial flexibility with 1/2 points, supporting feature additions but offering limited configuration options. ATHENA++, DAFO, and OntoGrate show concerning rigidity with 0/2 points, providing neither modular extension mechanisms nor significant configuration capabilities.

The detailed assessment of OntoRelQuery's extensibility reveals robust support for both functionality extensions through modular architecture and comprehensive configuration options through structured configuration files. This flexibility represents a significant strategic advantage, enabling adaptation to diverse use cases and incorporation of new capabilities as requirements evolve—essential characteristics for maintaining relevance in rapidly evolving technological environments.

4.2.5 Overall

The aggregated non-functional scores position Ontop as the clear leader in technical sustainability with 13/17 points. OntoRelQuery and OptiqueVQS follow with identical scores of 9/17 points, demonstrating reasonable technical viability with balanced performance across availability, compatibility, and flexibility dimensions, but showing room for improvement in maintenance infrastructure and technical ecosystem integration.

Von-QBE achieves 8/17 points, while ATHENA++ (5/17), DAFO (4/17), and OntoGrate (4/17) demonstrate significant limitations in non-functional qualities that could impact their practical utility in enterprise environments. This distribution reveals that technical sustainability considerations—critical for long-term deployments—have been unevenly addressed across the evaluated solutions.

4.3 Strategic positioning of OntoRelQuery

The comprehensive evaluation positions OntoRelQuery as the most balanced and capable solution in the current landscape of ontology-based database querying systems. With the highest functional score (44/60) and strong non-functional performance (9/17), it demonstrates excellence across the complete query lifecycle while providing the technical sustainability necessary for practical deployment.

OntoRelQuery's functional strengths lie particularly in query generation (19/24) and result processing (11/15)—capabilities directly impacting query effectiveness and knowledge discovery. In query generation, OntoRelQuery supports comprehensive data restriction capabilities, analytical functions, join operations, and set operations—significantly outperforming all competitors except ATHENA++ in this crucial dimension. Similarly, in result processing, OntoRelQuery offers sophisticated visualization, export formats, and execution capabilities that facilitate meaningful interaction with query results.

From a non-functional perspective, OntoRelQuery demonstrates a particular strength in flexibility and extensibility (2/2), providing robust extension mechanisms and configuration capabilities that enable adaptation to specialized domain requirements. Its open-source distribution model, combined with adequate documentation and community platforms, establishes a foundation for sustainable evolution, though there remains room for improvement in regular update cycles and active community engagement.

The analysis identifies several strategic improvement opportunities for OntoRelQuery:

- **Maintenance infrastructure enhancement**: Establishing more systematic update cycles and fostering a more active user community would strengthen long-term sustainability and accelerate feature evolution through community contributions.
- **Technical ecosystem integration**: Expanding support for NoSQL databases, triplestores, and cloud deployment architectures would enhance utility in contemporary heterogeneous data environments and improve integration with existing enterprise data infrastructure.
- **Advanced ontology processing capabilities**: While OntoRelQuery demonstrates strong basic ontology traversal capabilities, enhancing its pattern identification mechanisms would enable more intelligent navigation of complex ontological structures and potentially improve query efficiency.

OntoRelQuery's primary competitive advantages over alternative solutions lie in its balanced excellence across the complete query lifecycle, its comprehensive query generation capabilities, and its superior result processing features. Unlike specialized solutions such as ATHENA++ (focused on query generation) or OptiqueVQS (specialized in graphical interfaces), OntoRelQuery provides integrated end-to-end functionality without significant weaknesses in any specific dimension—a critical characteristic for practical deployment in diverse application contexts.

4.4 Future development directions

The comprehensive evaluation identifies several consistent gaps across the assessed solutions that represent important opportunities for future development in the field of ontology-based database querying:

- **Technical ecosystem integration**: None of the evaluated solutions demonstrate comprehensive compatibility with contemporary data ecosystems. Developing robust connectors for diverse database technologies (particularly NoSQL systems) and cloud deployment architectures would enhance the practical utility of these solutions in modern heterogeneous data environments.
- **Integrated user interfaces**: The current landscape shows a clear division between solutions that prioritize graphical interfaces (OptiqueVQS) and those emphasizing assistance features (OntoRelQuery). Future solutions could benefit from integrating these complementary approaches, combining intuitive visual query construction with intelligent contextual assistance to provide a more comprehensive user experience.
- **Advanced ontology processing**: While basic ontology traversal is well supported across most solutions, advanced capabilities like pattern identification remain underdeveloped. Enhancing these capabilities would enable more intelligent navigation of complex ontological structures, potentially improving both query efficiency and semantic accuracy.

– **Maintenance infrastructure standardization**: The significant variation in maintenance practices suggests an opportunity for establishing more standardized approaches to documentation, community engagement, and update processes that could enhance the sustainability of solutions in this domain.

These development directions, combined with the specific improvement opportunities identified for OntoRelQuery, provide a roadmap for advancing the field of ontology-based database querying toward more comprehensive, accessible, and sustainable solutions.

5 Conclusion

This survey has undertaken a comprehensive evaluation of ontology-based querying systems, bridging the theoretical promise of semantic technologies with the practical challenges of structured data access. Through a detailed framework encompassing both functional and non-functional requirements, we assessed seven representative solutions OntoRelQuery, Ontop, ATHENA++, OptiqueVQS, DAFO, Von-QBE, and OntoGrate—each embodying different design philosophies and technical trade-offs.

Our analysis reveals that while progress in the field has been substantial, key challenges persist. Most systems provide solid foundations for ontology processing and basic query generation, yet few offer an end-to-end solution that combines intuitive user interfaces, comprehensive query capabilities, and robust result processing. Among the evaluated systems, OntoRelQuery distinguishes itself through its balanced excellence across all functional dimensions, particularly excelling in query generation and result visualization, while also maintaining strong architectural flexibility and extensibility.

Nonetheless, OntoRelQuery and the field at large would benefit from deeper integration with contemporary data, including support for NoSQL databases, cloud native deployments, and enhanced API capabilities. Additionally, improvements in user community engagement, update cycles, and ontology pattern identification mechanisms are needed to ensure long-term sustainability and greater semantic precision.

This work contributes to a structured and reproducible framework for the evaluation of semantic query systems and offers a roadmap for future developments in ontology-based access to relational data. As organizations increasingly seek to unify their heterogeneous data landscapes, the evolution of such systems will be pivotal in enabling semantically rich, accessible, and scalable data integration and analysis solutions.

References

[1] Calvanese, D., Cogrel, B., Komla-Ebri, S., Kontchakov, R., Lanti, D., Rezk, M., Rodriguez-Muro, M., and Xiao, G. Ontop: Answering sparql queries over relational databases. *Semantic Web 8*, 3 (Dec. 2016), 471–487.

[2] Dou, D., Qin, H., and Lependu, P. Ontograte: Towards automatic integration for relational databases and the semantic web throught an ontology-based framework. *International Journal of Semantic Computing 04*, 01 (Mar. 2010), 123–151.

[3] Pankowski, T., and Bąk, J. *DAFO: An Ontological Database System with Faceted Queries*. Springer International Publishing, 2019, p. 152–155.

[4] Peres, L., L. Coelho da Silva, T., Macedo, J., and Araujo, D. *Ontology-Schema Based Query by Example*. Springer International Publishing, 2019, p. 204–212.

[5] Sen, J., Lei, C., Quamar, A., Özcan, F., Efthymiou, V., Dalmia, A., Stager, G., Mittal, A., Saha, D., and Sankaranarayanan, K. Athena++: natural language querying for complex nested sql queries. *Proceedings of the VLDB Endowment 13*, 12 (Aug. 2020), 2747–2759.

[6] Soylu, A., Kharlamov, E., Zheleznyakov, D., Jimenez-Ruiz, E., Giese, M., Skjæveland, M. G., Hovland, D., Schlatte, R., Brandt, S., Lie, H., and Horrocks, I. Optiquevqs: A visual query system over ontologies for industry. *Semantic Web 9*, 5 (Aug. 2018), 627–660.