Specification of Complex SQL Queries using Visual Query Languages: A Systematic Literature Review

ANONYMOUS AUTHOR(S)*

Structured Query Language (SQL) is a widely-used language for accessing both relational and non-relational databases. In relational databases, SQL is the standard form of access, while in non-relational databases, SQL is becoming increasingly available and consolidating itself as an access interface for querying data in cluster environments (e.g., Spark and Spanner). Despite its simple syntax, the specification of SQL queries is not a trivial task, even for experts, because some queries demand complex constructs (i.e., subqueries, joins, set operations, conditional expressions, grouping restrictions, and recursion). Visual Query Languages (VQLs) are an alternative that seeks to reduce this complexity. However, although several VQLs have been proposed, they are not widely used in practice. The objective of this study is to understand why this occurs, by identifying and analyzing the support provided by the proposed VQLs to address complex SQL queries. For this purpose, a Systematic Literature Review was carried out. The results of this review point out to the need for more expressive VQLs and tools that support their use to be made available to end users.

CCS Concepts: • **Information systems → Query languages**; *Structured Query Language*; • **Software and its engineering → Visual languages**.

Additional Key Words and Phrases: SQL, query, conceptual modeling, visual query language

ACM Reference Format:
Anonymous Author(s). 2019. Specification of Complex SQL Queries using Visual Query Languages: A Systematic Literature Review. *ACM Trans. Datab. Syst.* 1, 1 (May 2019), 28 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Structured Query Language (SQL) [14] is a declarative language defined as the standard (both *de jure* and *de facto*) for accessing relational databases. Despite considerable hype regarding the NoSQL movement, SQL remains the *lingua franca* of data processing, including today's non-relational infrastructure platform (e.g., Spark [4] and Spanner [5]), which often involve some form of SQL-based querying. Because of this, SQL remains an important technology for both academia and industry. In academia, the most important computer science curricula [2, 3] recommend the teaching of SQL. On the other hand, in industry [41], SQL has become one of the most popular languages among Web developers, desktop developers, sysadmins/DevOps, and data scientists because of its ubiquity, ease of understanding, and maturity.

Although the basic syntax of SQL is easy to understand, specifying queries that use complex constructs is a non-trivial activity. That is, whenever a query has at least one of the following constructs, it can be considered complex, and therefore it will require greater intellectual and technical effort: 1) subquery (correlated or non-correlated) [12, 15, 25, 46]; 2) join (Cross, Natural, Inner, or Outer) [12];3) set operation (Union, Except, and Intersect) [18]; 4) conditional expression (Case clause) [21]; 5) grouping restriction (Having clause) [46]; and 6) recursion (With clause) [18].

0362-5915/2019/5-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Despite their advantages Visual Query Languages (VQL) bring to the reasoning and logic of complex queries, it can be observed in practice that these languages are rarely used when specifying queries related to critical business issues [6, 23]. In this context, a Systematic Literature Review (SLR) can be used to help identify how VQL addresses complex SQL queries. In this sense, the purpose of this study is to perform an SLR to investigate why these languages are not widely used for specifying complex queries. It is important to highlight that, although the data visualization tools (e.g., Tableau, Microsoft Power BI, or QlikView) allow for the specification of queries, these tools do not support the specification of complex SQL queries without the use of multilayered interfaces. This can compromise understanding and critical analysis, because the query details are not always visible once the query has been created using different and overlapping interfaces.

The remainder of this paper is organized as follows: Section 2 describes the background; Section 3 addresses the methodology; Section 4 presents an overview of the SLR execution; Section 5 presents an evaluation of the selected articles; Section 6 reports the analysis and discussion; Section 7 describes related studies; and Section 8 presents the final considerations and conclusions and points out directions for future work.

2 BACKGROUND

2.1 Complex SQL queries

There are different perceptions regarding what should be considered a complex SQL query [12, 18, 21, 25, 46]. In this study, a query was considered to be complex whenever it contained at least one of the following constructs: subquery, join, set operation, conditional expression, grouping restriction, or recursion. In the following paragraphs of this section, an overview of the most important aspects of these constructs [19] will be given.

There are two types of subqueries: (1) correlated, where the subquery is evaluated once for each row processed by the outer query, because the subquery references at least one column from its outer query; and (2) non-correlated, where the subquery is executed only once (before executing the outer query). Subqueries can be specified in the following clauses: Select, From, Where, Having, and Order By.

There are four types of joins: (1) Cross Join, which returns the Cartesian product of the table rows; (2) Inner Join, which returns only records that satisfy the join condition; (3) Natural Join, a type of inner join whose join condition is made implicitly from the equality of fields having the same names; and (4) Outer Join, which extends the results of an inner join by adding the rows from the left table (Left Outer Join), right table (Right Outer Join), or both tables (Full Outer Join) that do not match the join condition. Cross Joins, Natural Joins, and Outer Joins must be specified in the From clause, while Inner Joins can be specified in the From or Where clauses. Note that a Natural Join is an Equi Join (a join using an equality condition); while both Inner and Outer Joins can be Equi Joins, Self Joins (a join whose condition joins a table with itself), or Theta Joins (a join using an inequality condition); and Semi Join (returns one copy of each row in the first table with at least one match found in the second table) or Anti Join (returns rows from the first table having no matches found in the second table) are special ways of Inner Join. Therefore, the primary join types are Natural, Inner, and Outer.

There are three types of set operations: (1) Union, which combines the results from two queries, returning by default only distinct rows; (2) Intersect, which merges the results of two queries, returning by default only the distinct rows common to both; and (3) Except, which matches the results from two queries, returning by default only the rows of the first query that do not appear in the second query. It should be emphasized that the ALL operator can be used to return all rows, instead of only the distinct ones.

A conditional expression (i.e., Case-When clauses) allows a variable to be tested for equality against a list of values. This decision-making statement can be specified in the following clauses: Select, Where, Order By, and Having.

A grouping expression (i.e., Group By clause) groups rows with the same values into a single row. If a grouping restriction needs to be defined, it must be specified in the Having clause.

Recursion (i.e., With clause) allows for the definition of a view that can be recursively referenced as a datasource.

2.2 Visual Query Language

Visual Query Languages (VQL) [11] are an alternative to SQL. Unlike with textual representations, the visual abstractions of a VQL (i.e., forms, diagrams, icons, or a combination of these) allow the user to focus on semantic rather than syntactic aspects of the query [12, 22]. This should favor the development of critical and creative thinking, since VQL visual abstractions create conditions that improve analytical thinking and problem-solving skills. In addition, the VQL visual abstractions allow information to be interpreted faster than textual SQL notation, because they can exploit the ability of the human visual system to process information in parallel [35, 44]. Although it is possible to use a VQL with only pen and paper, a VQL is much more powerful when implemented by a Computer-Aided Software Engineering (CASE) tool. These tools facilitate interaction with VQL notation and have functionalities that can avoid or detect errors, as well as generate software artifacts (e.g., source code or documentation) [10].

3 METHODOLOGY

In this study, an SLR is conducted based on the guidelines defined by Kitchenham and Charters [29], which involves the following steps: 1) specification of the research questions (cf. Section 3.1); 2) development of the review protocol (cf. Sections 3.2–3.5); 3) selection of articles (cf. Section 4); 4) quality assessment (cf. Table 1); and 5) data analysis and synthesis (cf. Sections 5–6).

3.1 Research questions

The purpose of this SLR is to investigate why VQLs are not widely used to specify complex queries. To achieve this goal, the complex constructions supported by each VQL (i.e., its expressiveness) will be compiled in order to respond to the following three research questions (RQ):

- *RQ1: what is the frequency of each complex construction?*
  This research question investigates how many VQLs support each type of complex construction (i.e., subquery, join, set operation, conditional expression, grouping restriction, and recursion). This question is important because, besides showing how common each complex construct is, it also helps to determine the expressiveness of each VQL.
- *RQ2: what is the frequency of each SQL clause in supporting subqueries or conditional expressions?*
  This research question analyzes, for each SQL clause (i.e., Select, From, Where, Order by, and Having), how many VQLs allow a subquery or conditional expression to be specified within it. This question is important because, unlike other complex constructs, subqueries and conditional expressions can be specified within more than one SQL clause, and knowing what these clauses provide information about the expressiveness of the VQL.
- *RQ3: what is the availability of CASE tool?*
  This question investigates whether a CASE tool is available, unavailable, or does not exist for each VQL. This is important because the availability of a CASE tool directly impacts the use of the language by end users.

3.2 Search expression and source selection

The search string used to perform the SLR was:

> "(SQL AND ("complex query" OR "complex queries" OR subquery OR subqueries) AND (visual OR graphic OR graphical))"

The term *SQL* was used to restrict the search to this language. The terms *complex query*, *complex queries*, *subquery*, and *subqueries* focused the search on proposals that address complex constructs. The terms *visual*, *graphic*, and *graphical* were used to limit the search to visual approaches. The search string only considered the term *subquery*, because *join*, *having*, *case*, and *with* are very vague terms, whereas *conditional* and *recursive* are too restrictive. The research libraries searched were: ScienceDirect, SpringerLink, ACM Digital Library, and IEEE Xplore, because these include high-quality computer science journals and conferences.

3.3 Inclusion and exclusion criteria

This review included articles published between January 1, 1980 and February 28, 2019, using these inclusion criteria:

- *IC1* – articles that present graphical notation, syntax and/or semantic rules;
- *IC2* – articles that address subquery specification.

The following criteria were used to exclude articles:

- *EC1* – secondary articles that only revise a thematic area (e.g., an SLR);
- *EC2* – short articles (e.g., posters, abstracts or banners);
- *EC3* – redundant articles by the same author. In this case, only the most complete and recent article is maintained;
- *EC4* – articles not written in English;
- *EC5* – gray literature, including non-peer-reviewed articles (e.g., reports, books, theses, or dissertations).

The inclusion and exclusion criteria were evaluated as follows: each article was reviewed by a researcher, who read the title, keywords, and abstract to determine the relevance of the article according to each criterion. When necessary, the content of the article was also examined. Then, for each reviewer assessment, two other researchers independently performed sanity tests. Finally, any differences that arose were reconciled in a collaborative manner.

3.4 Quality criteria

During quality evaluation, the relevance of the articles was verified by analyzing two quality criteria: 1) the CORE Conference/Journal Ranking[1] and 2) the citation count (Google Scholar). In the CORE Ranking, a conference/journal is assigned to one of the following categories: A+, A, B, C, or Unranked. The article scores a 4 if its CORE Ranking is classified as A+, 3 if A, 2 if B, 1 if C, and 0 if Unranked. For citations, the article scores a 0 if it does not have any citations and scores 0.5 for every ten citations. For instance, an article receives a score of 0.5 if it has between 1 and 10 citations, and receives a score of 1 if it has between 11 and 20 citations. The quality score for a study corresponds to the sum of the scores obtained in the analysis of the CORE Ranking and citations. This score is important because it helps to rank the articles.

[1] http://www.core.edu.au/

3.5 Data extraction

After selecting the articles, data were extracted from the studies included using an extraction form. This form assisted in capturing general data (e.g., author, year, publication type) and in getting specific data for each research question. The criteria used to capture this data are listed below.

- Criteria (i.e., complex constructs) to answer RQ1:
  - *C1* – non-correlated subquery;
  - *C2* – correlated subquery;
  - *C3* – cross join;
  - *C4* – natural join;
  - *C5* – inner join;
  - *C6* – outer join;
  - *C7* – set operation;
  - *C8* – grouping restriction;
  - *C9* – conditional expression;
  - *C10* – recursion.
- Criteria (i.e., SQL clauses) to answer RQ2:
  - *C11* – subquery in the Select clause;
  - *C12* – subquery in the From clause;
  - *C13* – subquery in the Where clause;
  - *C14* – subquery in the Order By clause;
  - *C15* – subquery in the Having clause;
  - *C16* – conditional expression in the Select clause;
  - *C17* – conditional expression in the Where clause;
  - *C18* – conditional expression in the Order By clause;
  - *C19* – conditional expression in the Having clause;
- Criterion to answer RQ3:
  - *C20* – availability of a CASE tool.

The criteria for RQ1 and RQ2 are binary and can score either 0 or 1 points. The criterion for RQ3, however, scores in the following manner: no CASE tool = 0 points; has a CASE tool that is unavailable = 0.5 points, or has a CASE tool that is available = 1 point. The outer join criterion (C6) scores a point if the VQL supports at least one outer join type (Full, Left, or Right) and the set operation criterion (C7) scores a point if the VQL supports at least one set operation type (Union, Except, or Intersect).

4 SLR EXECUTION OVERVIEW

This section presents an overview of the SLR execution. In Figure 1, the article selection results are presented. This figure shows the digital libraries, the steps performed, the articles resulting from each step, and the criteria that were used to execute the SLR. During identification (Step 1), 2,003 articles were found, which were then organized using the Covidence tool[2]. During duplicate removal (Step 2), using the same tool, 55 duplicate articles were automatically identified and excluded, leaving a set of 1,948 articles. In the first filter (Step 3), 1,833 articles were excluded following review of their titles, abstracts, and keywords (cf. reasons in Step 3 of Figure 1), using the inclusion and exclusion criteria presented in Section 3.3, leaving a set of 115 articles. Finally, in the second filter (Step 4), after a complete analysis of the text, 94 articles were excluded, because they were not in agreement with the inclusion criteria (cf. the reasons in Step 4 of Figure 1), leaving a total of 21 articles (cf., Table 1).

[2]https://www.covidence.org

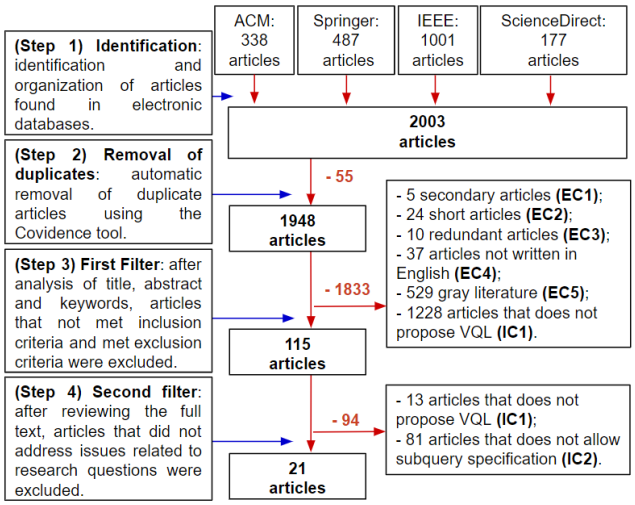


Fig. 1. Process for article selection.

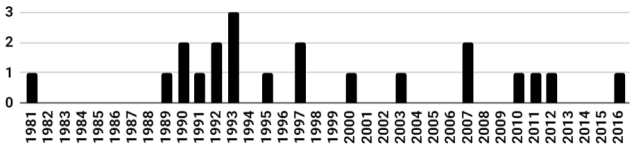


Fig. 2. Number of articles published by years.

In Table 1, the quality evaluation of the selected articles is shown. In this evaluation, the average quality score of 4.04% indicates that there is still room to develop studies in the VQL area. Additionally, most articles are relevant because they have been published in good Conferences/Journals (more than half of articles have CORE Rank A or A+) and they have an average of 31.1 citations per article.

In Figure 2, the frequency of publications per year is presented. The articles were published between 1981 and 2016 and, do not show consistency across time with regard to number of publications. It can be seen that 1993 had the most publications (i.e., three articles) and that, over the past ten years, six articles were published.

The articles included in this review come from conferences, journals, workshops, and symposiums. Most articles are published in journals (42.85%, 9 articles), followed by conferences (28.57%, 6 articles), workshops (19.04%, 4 articles), and symposiums (9.52%; 2 articles). In Table 2, the number of articles by publication source is shown. The 21 selected articles are distributed among 17 publication sources, suggesting that "specification of complex SQL queries using VQL" is a topic of interest for several publication vehicles, especially the Journal of Visual Languages and Computing, with three published articles.

5 EVALUATING THE SELECTED ARTICLES

In order to show evidence that helps answer the research questions, a structured summary highlighting the strengths of each article is presented below. For this, each summary is organized in such a way as to meet the criteria for data extraction shown in Section 3.5.

Table 1. Quality evaluation of selected articles.

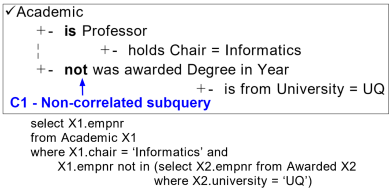
#	Author	Title	Year	CORE Rank	Citations Count.	Score
A01	Bloesch and Halpin [8]	Conceptual queries using ConQuer-II	1997	A	131	10
A02	Tansel et al. [43]	Time-by-example query language for historical databases	1989	A+	85	8.5
A03	Mohan and Kashyap [34]	A Visual Query Language for Graphical Interaction with Schema-Intensive Databases	1993	A+	61	7.5
A04	Whang et al. [45]	Two-dimensional specification of universal quantification in a graphical database query language	1992	A+	32	6
A05	Abouzied et al. [1]	DataPlay: Interactive Tweaking and Example-driven Correction of Graphical Database Queries	2012	A	43	5.5
A06	Murray et al. [37]	Kaleidoquery A Flow-based Visual Language and its Evaluation	2000	A	41	5.5
A07	Klug [30]	Abe: A Query Language for Constructing Aggregates-by-example	1981	A	42	5.5
A08	Danaparamita and Gatterbauer [16]	QueryViz: Helping Users Understand SQL Queries and Their Patterns	2011	A	36	5
A09	Epstein [20]	The TableTalk query language	1991	A	24	4.5
A10	Song et al. [40]	Using UML to model relational database operations	2007	B	38	4
A11	Jaakkola and Thalheim [24]	Visual SQL – High-Quality ER-Based Query Treatment	2003	A	20	4
A12	Sockut et al. [39]	GRAQULA: A graphical query language for entity-relationship or relational databases	1993	B	36	4
A13	Mun-Kew et al. [36]	The implementation of a visual language interface for an object-oriented multimedia database system	1990	A	3	3.5
A14	Bakke and Karger [7]	Expressive Query Construction Through Direct Manipulation of Nested Relational Results	2016	B	13	3
A15	Cerullo and Porta [13]	A System for Database Visual Querying and Query Visualization: Complementing Text and Graphics to Increase Expressiveness	2007	B	3	2.5
A16	Tan et al. [42]	A graphical knowledge level approach for user-database interaction	1990	B	4	2.5
A17	Borges and Macías [9]	Feasible Database Querying Using a Visual End-user Approach	2010	Unranked	14	1
A18	Keramopoulos et al. [27]	GOQL, A Graphical Query Language for Object-Oriented Database Systems	1997	Unranked	18	1
A19	Keim and Lum [26]	Visual Query Specification in a Multimedia Database System	1992	Unranked	16	1
A20	Ramos [38]	Design and Implementation of a Graphical SQL with Generic Capabilities	1993	Unranked	4	0.5
A21	Kwak and Moon [31]	Two-dimensional specification of queries in object-oriented databases	1995	Unranked	0	0

Firstly, evidence RQ1 is shown (highlight in blue), then RQ2, and finally RQ3. The answers to these criteria were based on statements or examples presented in the articles. Table 3 presents a summary of this evidence by article. This evidence is tabulated and compiled in Section 6, so that: in Table 4, the frequency of each complex construction (RQ1) is shown; in Table 4, the frequency of each SQL clause in supporting subqueries or conditional expressions (RQ2) is presented, and finally, in Table 6, the availability of a CASE tool for each VQL (RQ3) is shown. The overview each articleâĂŹs strengths is shown below. Table 3 summarizes the evidence that was observed by reading the articles. Therefore, a completely empty circle indicates that the criterion was not

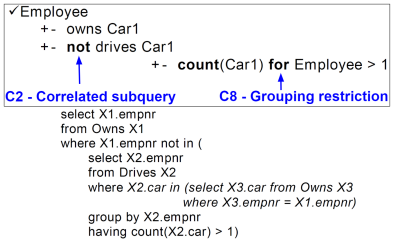
Table 2. Distribution of articles by publication source.

Source of publication	Articles	Count.
Journal of Visual Languages and Computing	A06, A07, A09	3
IEEE Transactions on Software Engineering	A15, A16	2
International Conference on Conceptual Modeling	A01, A03	2
Computer Standards & Interfaces	A12	1
Conference on Visualization	A05	1
Data & Knowledge Engineering	A13	1
IEEE Transactions on Knowledge and Data Engineering	A21	1
International Computer Software and Applications Conference	A10	1
International Conference on Extending Database Technology	A20	1
International Conference on Management of Data	A02	1
International Workshop on Database and Expert Systems Applications	A17	1
International Workshop on Information Technology	A11	1
International Workshop on Interfaces to Database Systems	A18	1
Microprocessing and Microprogramming	A19	1
Symposium on Engineering Interactive Computing Systems	A04	1
Symposium on User Interface Software and Technology	A08	1
Workshop on Statistical Database Management	A14	1

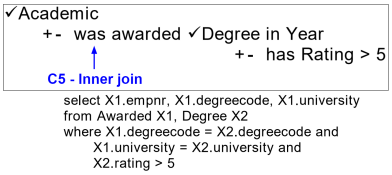
observed in the article (i.e., it could not be affirmed that the criterion is supported), a completely-filled circle indicates that the criterion was observed in the article (i.e., evidence was found that the criterion is supported by the work), and a partially-filled circle indicates that the criterion was partially supported by the article (i.e., evidence was found that the criterion is partially supported by the study). At last, the presented SQL codes are found in the selected articles.



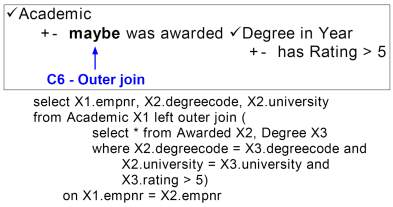
(a) Who are the academics who are professors of informatics and do not have a degree from UQ university?



(b) Which employees have more than one car and do not drive any of these cars?



(c) Who are the academics, as well as their degrees, with a rating greater than 5?



(d) Who are the academics, as well as their degrees (if any,) with a rating greater than 5?

Fig. 3. Main features of ConQuer-II [8].

5.1 ConQuer-II (A01)

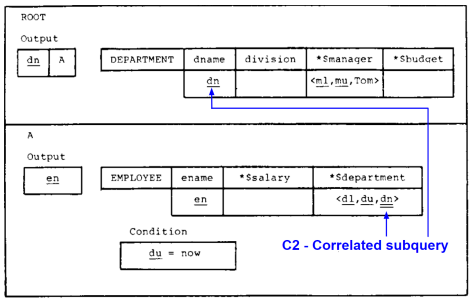
The main features of ConQuer-II [8] are shown in Figures 3a, 3b, 3c, and 3d. Figure 3a shows the use of a non-correlated subquery (C1) and addresses the query: *"who are the academics who are professors of informatics and do not have a degree from UQ university?"*. Figure 3b shows the use of a correlated subquery (C2) and grouping restriction (C8). This figure addresses the query: *"which employees have more than one car and do not drive any of these cars?"*. Figure 3c shows the use of an inner join (C5) and addresses the query: *"who are the academics, as well as their degrees, with a rating greater than 5?"*. Figure 3d shows the use of an outer join (C6) and addresses the query: *"who are the academics, as well as their degrees (if any), with a rating greater than 5?"*.

Concerning RQ2, Figures 3a and 3b show a subquery in the Where clause (C13). Regarding RQ3, the language is supported by the Conquer-II query engine tool, but it is not available for download. In summary, this study considers criteria C1, C2, C5, C6, C8, C13, and partially meets criterion C20.

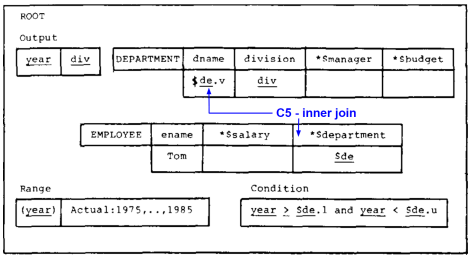
5.2 Time by Example (A02)

The main features of Time by Example (TBE) [43] are shown in Figures 4a and 4b. Figure 4a shows the use of a correlated subquery (C2) and addresses the query: *"which employees are present in each department managed by Tom?"*. Figure 4b shows the use of an inner join (C5) and addresses the query: *"which divisions has Tom worked for?"*.

Concerning RQ2, Figure 4a shows a subquery in the Where clause (C13). Regarding RQ3, the language has no CASE tool. In summary, this study considers criteria C2, C5, and C13.



(a) Which employees are present in each department managed by Tom?



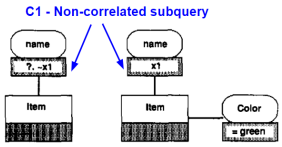
(b) Which divisions has Tom worked for?

Fig. 4. Main features of Time by Example (TBE) [43].

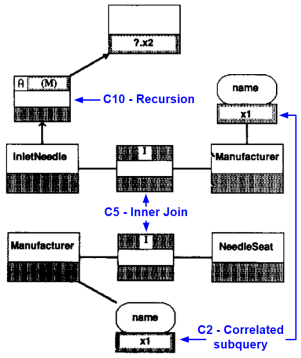
5.3 Mohan and Kashyap (A03)

The main features of the language proposed by Mohan and Kashyap [34] are shown in Figures 5a, 5b, and 5c. Figure 5a shows the use of a non-correlated subquery (C1) and addresses the query: *'what are the names of the items that are not green items?"*. Figure 5b shows the use of a correlated subquery (C2), an inner join (C5), and recursion (C10). This figure addresses the query: *'what are the components that use (or have components that use) inlet needles manufactured by manufacturers who also make needle seats?"*. Figure 5c shows the use of a grouping restriction (C8) and recursion (C10). This figure addresses the query: *"what are the components that, together with all the components that form them, have more than 20 components?"*.

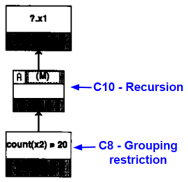
Concerning RQ2, Figures 5a and 5b show a subquery in the Where clause (C13). Regarding RQ3, the language is supported by a CASE tool, but it is not available for download. In summary, this study considers criteria C1, C2, C5, C8, C10, C13, and partially meets criterion C20.



(a) What are the names of the items that are not green items?



(b) What are the components that use (or have components that use) inlet needles manufactured by manufacturers who also make needle seats?



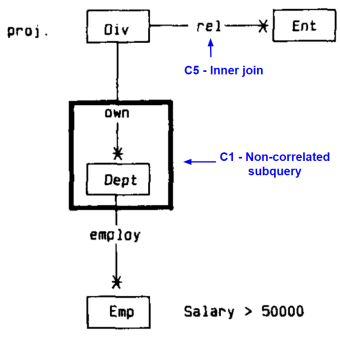
(c) What are the components that, together with all the components that form them, have more than 20 components?

Fig. 5. Main features of the language proposed by Mohan and Kashyap [34].

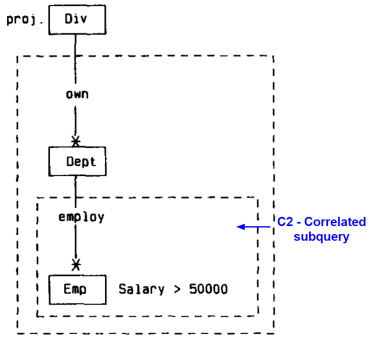
5.4 Whang et al. (A04)

The main features of the language proposed by Whang et al. [45] are shown in Figures 6a and 6b. Figure 6a shows the use of a non-correlated subquery (C1) and inner join (C5). This figure addresses the query: *'which divisions and their respective entities have departments where employees have a salary greater than 50,000?"*. Figure 6b shows the use of a correlated subquery (C2) and addresses the query: *"which divisions do not have a department where none of the employees has a salary greater than 50,000?"*.

Concerning RQ2, Figures 6a and 6b show a subquery in the Where clause (C13). Regarding RQ3, the language has no CASE tool. In summary, this study considers criteria C1, C2, C5, and C13.



(a) Which divisions and their respective entities have departments where employees have a salary greater than 50,000?



(b) Which divisions do not have a department where none of the employees has a salary greater than 50,000?

Fig. 6. Main features of the language proposed by Whang et al. [45].

5.5 DataPlay (A05)

The main features of DataPlay [1] are exemplified in Figure 7, which shows the use of a non-correlated subquery (C1) and an inner join (C5). This figure addresses the following query: *"which students are enrolled in grade A, receive marks above 90, and take courses CS11 and CS12?"*.

Concerning RQ2, Figure 7 shows a subquery in the Where clause (C13). Regarding RQ3, the language is supported by a CASE tool, but it is not available for download. In summary, this study considers criteria C1, C5, C13 and partially meets criterion C20.

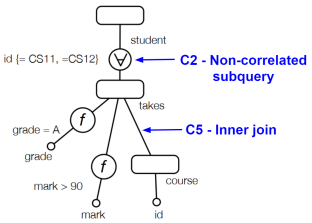


Fig. 7. Main features of DataPlay [1]. Which students are enrolled in grade A, receive marks above 90, and take courses CS11 and CS12?.

5.6 Kaleidoquery (A06)

The main features of Kaleidoquery [37] are shown in Figures 8a, 8b, 8c and 8d. Figure 8a shows the use of a non- correlated subquery (C1) and addresses the query: *"which people work in companies located in England?"*. Figure 8b shows the use of a grouping restriction (C8) and addresses the query: *"what is the name, age, and salary of persons named Smith who are older than 40 and younger than 65, who have an average salary greater than 50,000?"*. Figure 8c shows the use of a set operation (C7) and addresses the query: *"which people working at IBM intersected with people working at companies located in London?"*. Figure 8d shows the use of an inner join (C5) and addresses the query: *"what is the name of People that have the same salary as NewPeople?"*.

Concerning RQ2, Figure 8a shows a subquery in the Where clause (C13). Regarding RQ3, the language is supported by a CASE tool, but it is not available for download. In summary, this study considers criteria C1, C5, C7, C8, C13, and partially meets criterion C20.

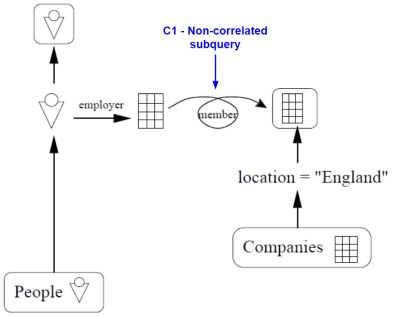
5.7 Abe (A07)

The main features of Abe [30] are exemplified in Figures 9a, 9b, and 9c. Figure 9a shows the use of a non-correlated subquery (C1) and a correlated subquery (C2). This figure addresses the query: *"who are the managers of departments selling no red items?"*. Figure 9b shows the use of a correlated subquery (C2) and a grouping restriction (C8). This figure addresses the query: *"what is the sum of the salaries of employees in departments having at least 50 employees?"*. Figure 9c shows the use of an inner join (C5) and addresses the query: *"who are the employees working in departments of the research division?"*.

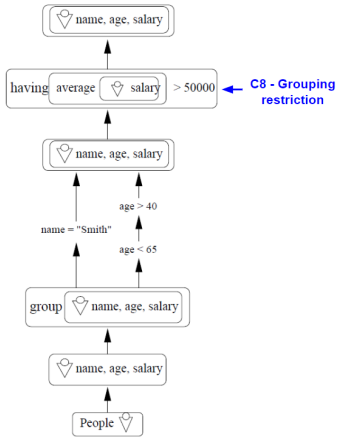
Concerning RQ2, Figures 9a and 9b show a subquery in the Where clause (C13). Regarding RQ3, the language is supported by a CASE tool, but it is not available for download. In summary, this study considers criteria C1, C2, C5, C8, C13, and partially meets criterion C20.

5.8 QueryViz (A08)

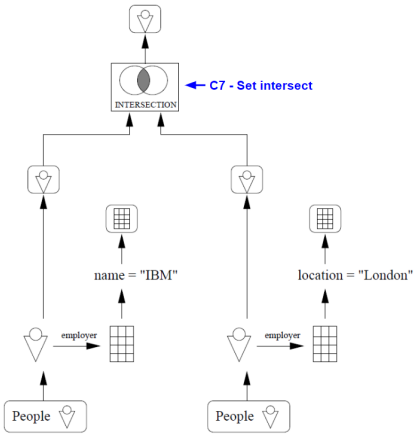
The main features of QueryViz [16] are shown in Figure 10, which shows the use of a non-correlated subquery (C1), a correlated subquery (C2), and an inner join (C5). This figure addresses the following



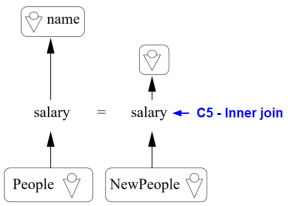
(a) Which people work in companies located in England?



(b) What is the name, age, and salary of persons named Smith who are older than 40 and younger than 65, who have an average salary greater than 50,000?



(c) Which people working at IBM intersected with people working at companies located in London?



(d) What is the name of People that have the same salary as NewPeople?

Fig. 8. Main features of Kaleidoquery [37].

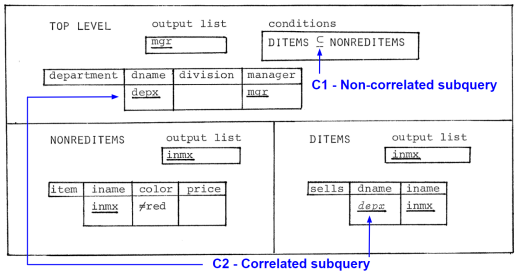
query: *"which actors have already worked with Kevin Bacon?"*. QueryViz uses the same notation to represent non-correlated and correlated subqueries.

Concerning RQ2, Figure 10 shows a subquery in the Where clause (C13). Regarding RQ3, the language is supported by a CASE tool available online[3]. In summary, this study includes criteria C1, C2, C5, C13, and C20.

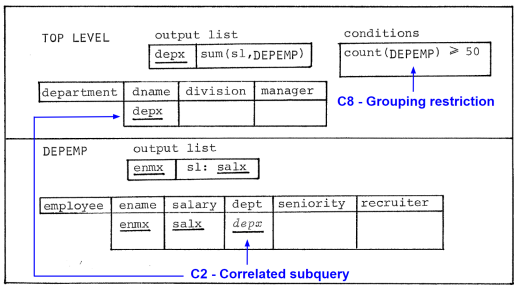
5.9 TableTalk (A09)

The main features of TableTalk [20] are exemplified in Figures 11a and 11b. Figure 11a shows the use of a non-correlated subquery (C1) and addresses the query: *"what are the numbers, titles, authors, and prices of books published by Addison-Wesley that are more expensive than the most expensive*

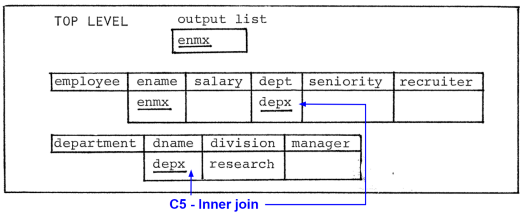
[3]http://queryviz.com/online



(a) Who are the managers of departments selling no red items?



(b) What is the sum of the salaries of employees in departments having at least 50 employees?



(c) Who are the employees working in departments of the research division?

Fig. 9. Main features of Abe [30].

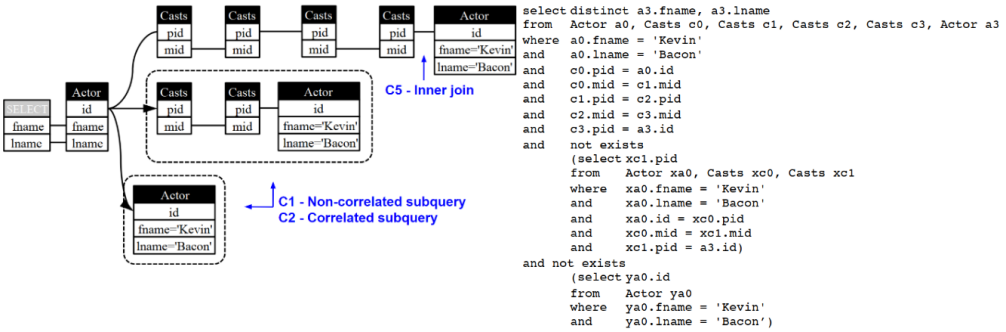


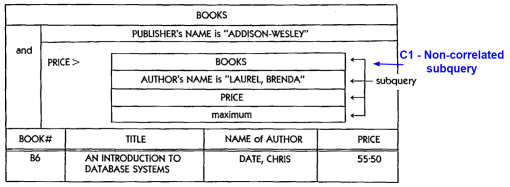
Fig. 10. Main features of QueryViz [16]. Which actors have already worked with Kevin Bacon?.

book written by Brenda Laurel?". Figure 11b shows the use of an inner join (C5) and addresses the query: *"what are the order numbers, book numbers, titles, and prices for all books that were ordered on 18 December, 1990?"*.

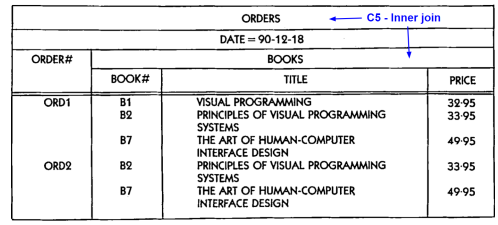
Concerning RQ2, Figure 11a shows a subquery in the Where clause (C13). Regarding RQ3, the language is supported by a CASE tool, but it is not available for download. In summary, this study considers criteria C1, C5, C13, and partially meets criterion C20.

5.10 Song et al. (A10)

The main features of the language proposed by Song et al. [40] are exemplified in Figures 12a, 12b, 12c, and 12d. Figure 12a shows the use of a non-correlated subquery (C1) and addresses the query: "*what is the concentration of pollutants whose names begin with A?*". Figure 12b shows the use of a

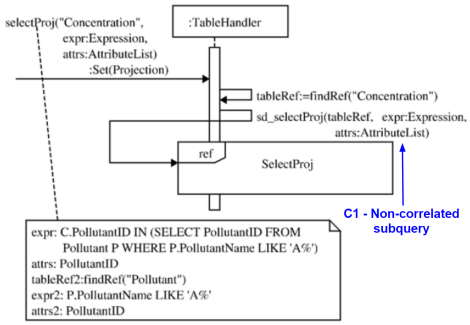


(a) What are the numbers, titles, authors, and prices of books published by Addison-Wesley that are more expensive than the most expensive book written by Brenda Laurel?

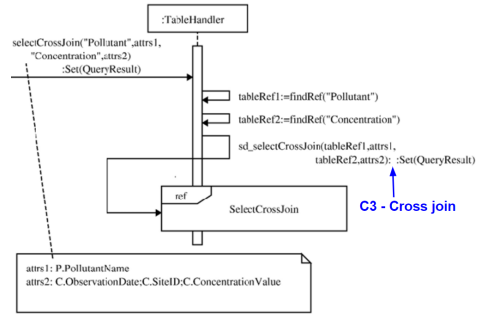


(b) What are the order numbers, book numbers, titles, and prices for all books that were ordered on 18 December, 1990?

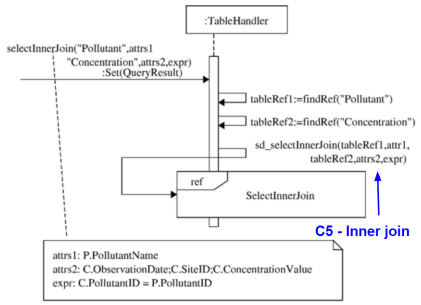
Fig. 11. Main features of TableTalk [20].



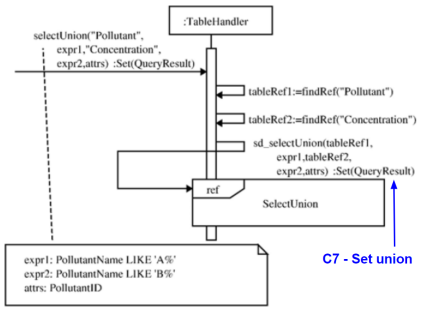
(a) What is the concentration of pollutants whose names begin with A?



(b) What are the names of the pollutants, the observation dates, and concentration values observed?



(c) What are the names, dates of observation, and concentration values of the pollutants observed?



(d) Which pollutants names begin with A coupled with pollutants whose names begin with B?

Fig. 12. Main features of the language proposed by Song et al. [40].

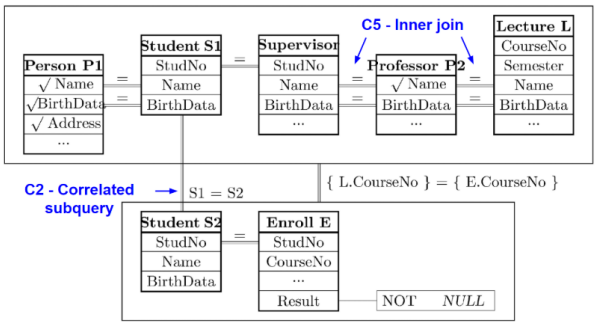
cross join (C3) and addresses the query: *"what are the names of the pollutants, the observation dates, and concentration values observed?"*. Figure 12c shows the use of an inner join (C5) and addresses the query: *"what are the names, dates of observation, and concentration values of the pollutants observed?"*. Figure 12d shows the use of a set operation (C7) and addresses the query: *"which pollutants names begin with A coupled with pollutants whose names begin with B?"*.

Concerning RQ2, Figure 12b shows a subquery in the Where clause (C13). Regarding RQ3, although it is based on UML flow diagrams, the language has no CASE tool. In summary, this study includes criteria C1, C3, C5, C7, and C13.

5.11 Visual SQL (A11)

The main features of VisualSQL [24] are shown in Figure 13, which shows the use of a correlated subquery (C2) and inner join (C5). This figure addresses the following query: *"which students have successfully completed only courses that have been or are currently administered by the student supervisor?"*.

Concerning RQ2, Figure 13 shows a subquery in the Where clause (C13). Regarding RQ3, the language is supported by a CASE tool, but it is not available for download. In summary, this study considers criteria C2, C5, C13, and partially meets criterion C20.




```
SELECT P1.Name, P1.BirthData, P1. Address,
       P2.Name AS "Name of supervisor"
FROM Person P1, Professor P2, Student S1, Supervisor,
     Lecture L, Enroll E
WHERE P1.Name = S1.Name
    AND P1.BirthData = S1.BrirthData
    AND S1.StudNo = E.StudNo
    AND S1.Result IS NOT NULL
    AND S1.StudNo = Supervisor.StudNo
    AND Supervisor.Name = P2.Name
    AND Supervisor.BirthData = P2.BirthData
    AND L.Name = Professor.Name
    AND L.BirthData = P2.BirthData
    AND L.CourseNo IN ( SELECT E2.CourseNo
        FROM Enroll E2
        WHERE S1.StudNo = E2.StudNo
            AND E2.Result IS NOT NULL)
    AND E.CourseNo IN ( SELECT L2.CourseNo
        FROM Lecture L2
        WHERE L2.Name = P2.Name
        AND L2.BirthData = L2.BirthData);
```

Fig. 13. Main features of VisualSQL [24]. Which students have successfully completed only courses that have been or are currently administered by the student supervisor?.

5.12 GRAQULA (A12)

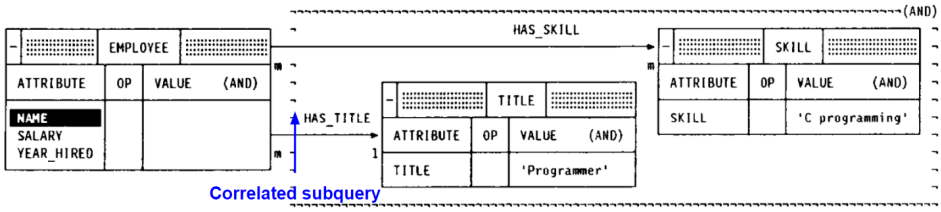
The main features of GRAQULA [39] are shown in Figures 14a, 14b, and 14c. Figure 14a shows the use of a correlated subquery (C2) and addresses the query: *"what are the names of the employees who do not have a programmer's title and have C programming skills?"*. Figure 14b shows a non-correlated subquery (C1) and a grouping restriction (C8). This figure addresses the query: *"what are the years of employment for which the average salary of employees hired in that year exceeds one-tenth of the average budget of all departments?"*. Figure 14c shows the use of an inner join (C5) and addresses the query: *"what is the name and salary of the research division employees earning over 50,000?"*.

Concerning RQ2, Figures 14a and 14b show a subquery in the Where clause (C13). Regarding RQ3, the language has no CASE tool. In summary, this study includes criteria C1, C2, C5, C8, and C13.

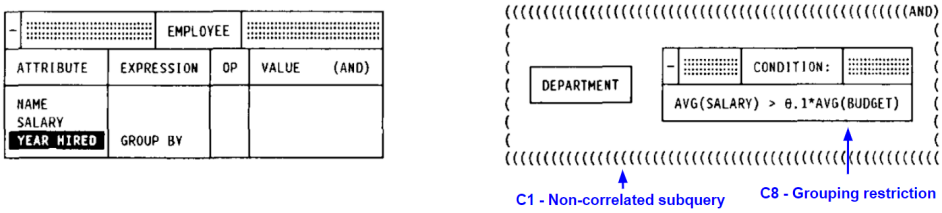
5.13 VILD (A13)

The main features of Visual Language for Database (VILD) [36] are shown in Figures 15a and 15b. Figure 15a shows the use of a non-correlated subquery (C1) and addresses the query: *"which employees have a salary higher than Emmanuel's commission?"*. Figure 15b shows the use of an inner join (C5) and addresses the query: *"which employees work in the same department as John?"*.

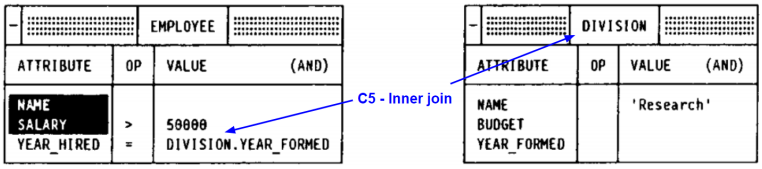
Concerning RQ2, Figure 15a shows a subquery in the Where clause (C13). Regarding RQ3, the language is supported by a CASE tool, but it is not available for download. In summary, this study considers criteria C1, C5, C13 and partially meets criterion C20.



(a) What are the names of the employees who do not have a programmer's title and have C programming skills?

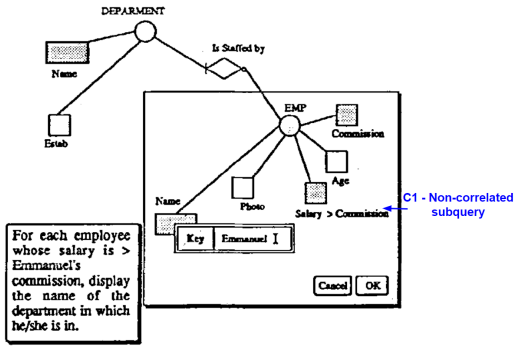


(b) What are the years of employment for which the average salary of employees hired in that year exceeds one-tenth of the average budget of all departments?

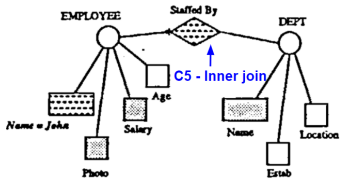


(c) What is the name and salary of the research division employees earning over 50,000?

Fig. 14. Main features of GRAQULA [39].



(a) Which employees have a salary higher than Emmanuel's commission?



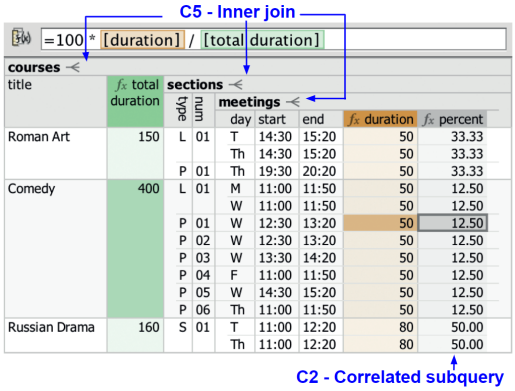
(b) Which employees work in the same department as John?

Fig. 15. Main features of VILD [36].

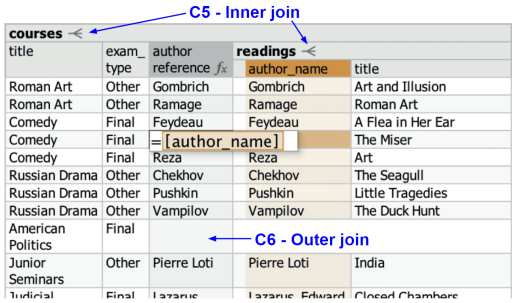
5.14 SIEUFERD (A14)

The main features of SIEUFERD [7] are exemplified in Figures 16a, 16b, and 16c. Figure 16a shows the use of a correlated subquery (C2) and an inner join (C5). This figure addresses the query: *"what is the percentage of minutes for each meeting out of all meetings held during the course?"*. Figure 16b

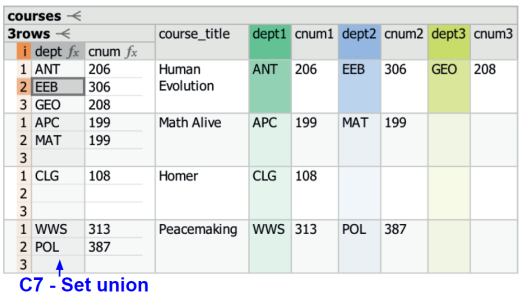
shows the use of an inner join (C5) and an outer join (C6). This figure addresses the query: *"what are the recommended readings for each course? (If the course has no recommendations, it must be returned in the search)"*. Figure 16c shows the use of a set operation (C7) and addresses the query: *"what are the departments by course?"*. In addition to the queries presented, the authors state that the "Cartesian product x is an inner join with C = true" [7]. Therefore, it is also possible to specify a cross join(C3).



(a) What is the percentage of minutes for each meeting out of all meetings held during the course?



(b) What are the recommended readings for each course? (If the course has no recommendations, it must be returned in the search)



(c) What are the departments by course?

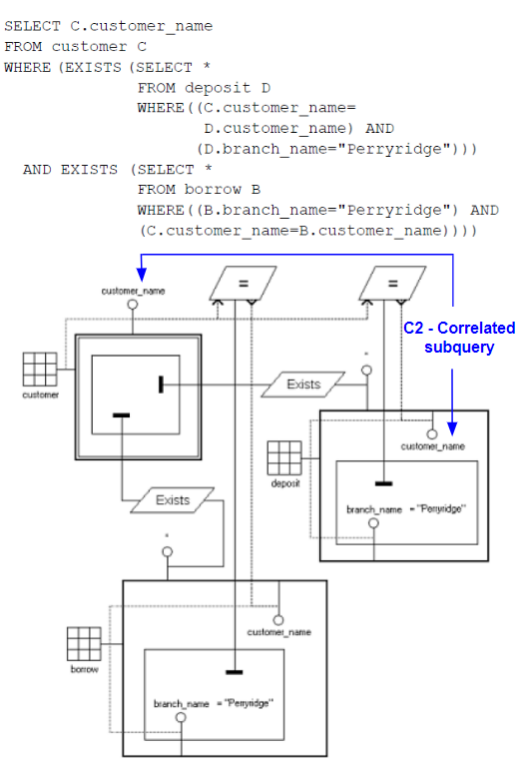
Fig. 16. Main features of SIEUFERD [7].

Concerning RQ2, Figure 16b shows a subquery in the Select clause (C11). Regarding RQ3, the language is supported by a CASE tool, but it is not available for download. In summary, this study considers criteria C2, C3, C5, C6, C7, C11, and partially meets criterion C20.

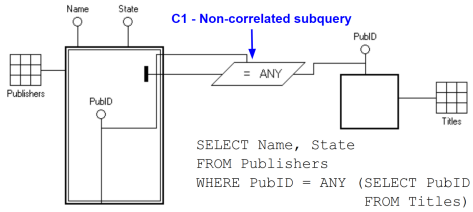
5.15 GraphSQL (A15)

The main features of GraphSQL [13] are shown in Figures 17a, 17b, 17c, and 17d. Figure 17a shows the use of a correlated subquery (C2) and addresses the query: *"what are the names of the customers who deposit and borrow from a Perryridge branch?"*. Figure 17b shows the use of a non-correlated subquery (C1) and addresses the query: *"which publishers have titles published?"*. Figure 17c shows the use of an outer join (C6), but the same construct can be used to express an inner join (C5). This figure addresses the query: *"what are the author names and ISBN codes of all authors who have or do not have a publication? (If they are not published, the ISBN should return null)"*. Figure 17d shows the

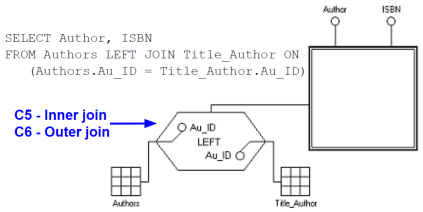
use of a grouping restriction (C8) and addresses the query: *"which publishers have more than 100 published titles?"*. In addition to the queries presented, the authors state that GraphSQL also has some other symbols used to express several aspects of relational algebra querying (e.g., intersection and union among the results of different queries) [13]. Therefore, it is also possible to specify set operations (C7).



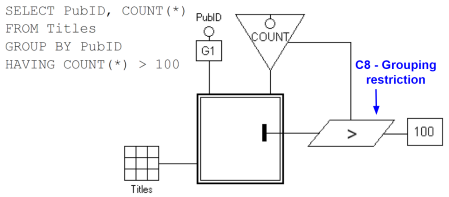
(a) What are the names of the customers who deposit and borrow from a Perryridge branch?



(b) Which publishers have titles published?



(c) What are the names and ISBN codes of all authors who have or do not have a publication? (If they are not published, the ISBN should return null)



(d) Which publishers have more than 100 published titles?

Fig. 17. Main features of GraphSQL [13].

Concerning RQ2, Figures 17a and 17b show a subquery in the Where (C13) clause. However, subqueries in the Having clause are possible (C15) since the comparison operator presented in Figure 17d also accepts a subquery as argument (cf. Figure 17a). Regarding RQ3, the language is supported by the GraphSQL Builder tool, but it is not available for download. In summary, this study considers criteria C1, C2, C5, C6, C7, C8, C13, C15, and partially meets criterion C20.

5.16 GKQL (A16)

The main features of the Graphical Knowledge Level Query Language (GKQL) [42] are shown in Figure 18, which shows a non-correlated subquery (C1) and an inner join (C5). This figure addresses the following query: *"which suppliers provide at least one part also supplied by the red part supplier?"*.

Concerning RQ2, Figure 18 shows a subquery in the Where clause (C13). Regarding RQ3, the language is supported by a CASE tool, but it is not available for download. In summary, this study considers criteria C1, C5, C13, and partially meets criterion C20.

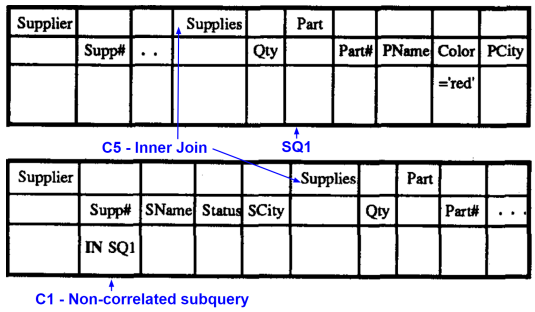


Fig. 18. Main features of GKQL [42]. Which suppliers provide at least one part also supplied by the red part supplier?.

5.17 Visque (A17)

The main features of Visque [9] are exemplified in Figure 19, which shows the use of a non-correlated subquery (C1) and a set union (C7). This Figure addresses the following query: *"how many movies are rented in the first quarter, grouped by quarter and year?"*.

Concerning RQ2, Figure 19 shows a subquery in the Where clause (C13). Regarding RQ3, the language is supported by a CASE tool, but it is not available for download. In summary, this study includes criteria C1, C7, C13, and partially meets criterion C20.

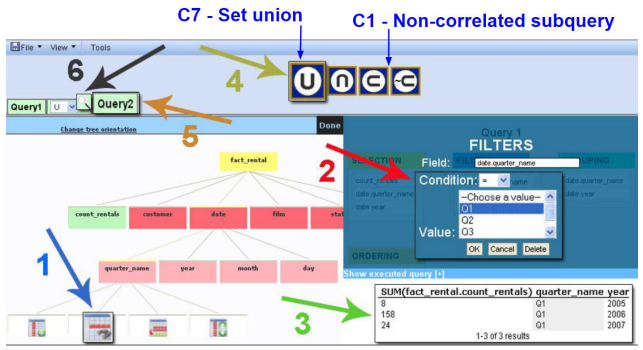
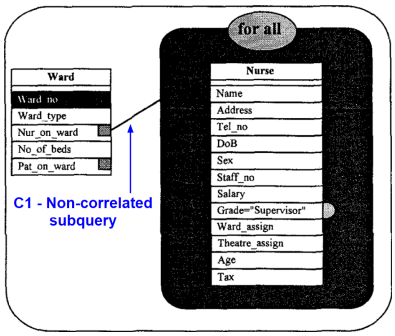


Fig. 19. Main features of Visque [9]. How many movies are rented in the first quarter, grouped by quarter and year? [9].

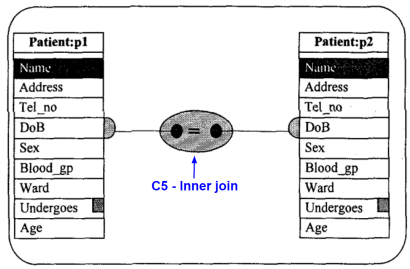
5.18 GOQL (A18)

The main features of the Graphical Object-Oriented Query Language (GOQL) [27] are exemplified in Figures 20a, 20b, and 20c. Figure 20a shows a non-correlated subquery (C1) and addresses the query: *"what are the numbers of the wards where only nurses supervise?"*. Figure 20b shows an inner join (C5) and addresses the query: *"what are the names of the patients who have the same birthdate?"*. Figure 20c shows the use of a set operation (C7) and addresses the query: *"what are the names of male patients or patients whose telephone number begins with 0181?"*.

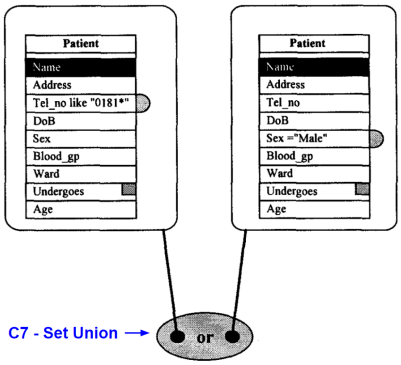
Concerning RQ2, Figure 20a shows a subquery in the Where clause (C13). Regarding RQ3, the language has no CASE tool. In summary, this study includes criteria C1, C5, C7, and C13.



(a) What are the numbers of the wards where only nurses supervise?

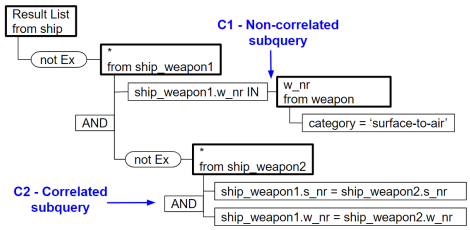


(b) What are the names of the patients who have the same birthdate?

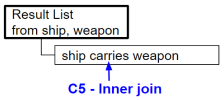


(c) What are the names of male patients or patients whose telephone number begins with 0181?

Fig. 20. Main features of GOQL [27].



(a) Which ships carry unique surface-to-air category weapons that no other ship carries?



(b) Which ships carry weapons?

Fig. 21. Main features of the language proposed by Keim and Lum [26].

5.19 Keim and Lum (A19)

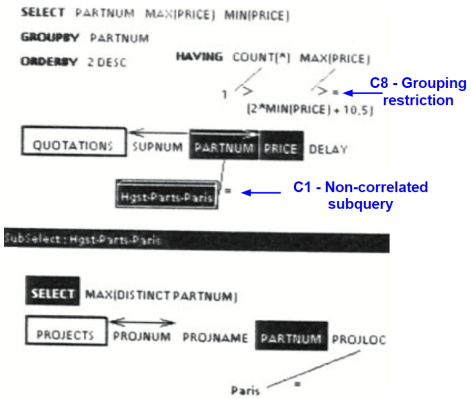
The main features of the language proposed by Keim and Lum [26] are exemplified in Figures 21a and 21b. Figure 21a shows the use of a non-correlated subquery (C1) and a correlated subquery (C2). This figure addresses the query: *"which ships carry unique surface-to-air category weapons that no other ship carries?"*. Figure 21b shows the use of an inner join (C5) and addresses the query: *"which ships carry weapons?"*. In addition to the presented queries, the authors state that the functions are

divided into five groups: logical operators, comparison operators, nesting operators, set operators and aggregate operators [26]. Therefore, it is also possible to specify set operations (C7).

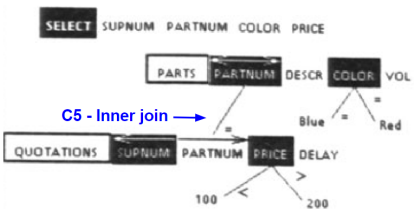
Concerning RQ2, Figure 21a shows a subquery in the Where clause (C13). Regarding RQ3, the language is supported by a CASE tool, but it is not available for download. In summary, this study considers criteria C1, C2, C5, C7, C13, and partially meets criterion C20.

5.20 IQL (A20)

The main features of the Interactive Query Language (IQL) [38] are shown in Figures 22a and 22b. Figure 22a shows the use of a non-correlated subquery (C1) and a grouping restriction (C8). This figure addresses the query: *"what are the serial number, maximum price, and minimum price of quotations with: 1) serial number equal to the highest serial number of the parts used by project pr1002 or by projects not allocated in Paris; 2) at most two quotations; and 3) maximum price greater than twice the minimum price plus 10.5?"*. Figure 22b shows the use of an inner join (C5) and addresses the query: *"what are the supplier numbers, part numbers, color, and price of parts with: 1) red parts and a price less than 100; or 2) blue parts and a price not greater than 200?"*.



(a) What are the serial number, maximum price, and minimum price of quotations with: 1) serial number equal to the highest serial number of the parts used by project pr1002 or by projects not allocated in Paris; 2) at most two quotations; and 3) maximum price greater than twice the minimum price plus 10.5?



(b) What are the supplier numbers, part numbers, color, and price of parts with: 1) red parts and a price less than 100; or 2) blue parts and a price not greater than 200?

Fig. 22. Main features of IQL [38].

Concerning RQ2, Figure 22a shows a subquery in the Where clause (C13). Regarding RQ3, the language is supported by a CASE tool, but it is not available for download. In summary, this study includes criteria C1, C5, C8, C13, and partially meets criterion C20.

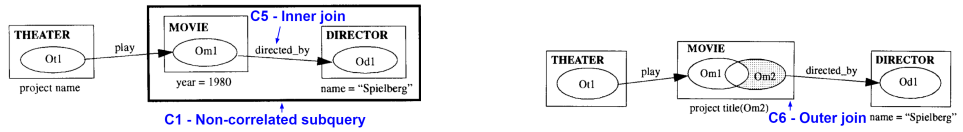
5.21 OQD (A21)

The main features of the Object Query Diagram (OQD) [27] are shown in Figures 23a and 23b. Figure 22 shows the use of a non-correlated subquery (C1) and an inner join (C5). This Figure addresses the query: *"which theaters presented all films directed by Spielberg in 1980?"*. Figure 22 shows the use of an outer join (C6) and addresses the query: *"which theaters presented films directed by Spielberg?"*.

Table 3. Overview of selected articles.

	RQ1										RQ2									RQ3
#	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	C17	C18	C19	C20
A01	●	●	○	○	●	●	○	●	○	○	○	○	●	○	○	○	○	○	○	◐
A02	○	●	○	○	●	○	○	○	○	○	○	○	●	○	○	○	○	○	○	○
A03	●	●	○	○	●	○	○	●	○	●	○	○	●	○	○	○	○	○	○	◐
A04	●	●	○	○	●	○	○	○	○	○	○	○	●	○	○	○	○	○	○	○
A05	●	○	○	○	●	○	○	○	○	○	○	○	●	○	○	○	○	○	○	◐
A06	●	○	○	○	●	○	●	●	○	○	○	○	●	○	○	○	○	○	○	◐
A07	●	●	○	○	●	○	○	●	○	○	○	○	●	○	○	○	○	○	○	◐
A08	●	●	○	○	●	○	○	○	○	○	○	○	●	○	○	○	○	○	○	●
A09	●	○	○	○	●	○	○	○	○	○	○	○	●	○	○	○	○	○	○	◐
A10	●	○	●	○	●	○	●	○	○	○	○	○	●	○	○	○	○	○	○	○
A11	○	●	○	○	●	○	○	○	○	○	○	○	●	○	○	○	○	○	○	◐
A12	●	●	○	○	●	○	○	●	○	○	○	○	●	○	○	○	○	○	○	○
A13	●	○	○	○	●	○	○	○	○	○	○	○	●	○	○	○	○	○	○	◐
A14	○	●	●	○	●	●	●	○	○	○	●	○	○	○	○	○	○	○	○	◐
A15	●	●	○	○	●	●	●	●	○	○	○	○	●	○	●	○	○	○	○	◐
A16	●	○	○	○	●	○	○	○	○	○	○	○	●	○	○	○	○	○	○	◐
A17	●	○	○	○	○	○	●	○	○	○	○	○	●	○	○	○	○	○	○	◐
A18	●	○	○	○	●	○	●	○	○	○	○	○	●	○	○	○	○	○	○	○
A19	●	●	○	○	●	○	●	○	○	○	○	○	●	○	○	○	○	○	○	◐
A20	●	○	○	○	●	○	○	●	○	○	○	○	●	○	○	○	○	○	○	◐
A21	●	○	○	○	●	●	○	○	○	○	○	○	●	○	○	○	○	○	○	○

○ - evidence not found; ◐ - partial supported; ● - supported.
C1 – non-correlated subquery; C2 – correlated subquery; C3 – cross join; C4 – natural join; C5 – inner join; C6 – outer join; C7 – set operation; C8 – grouping restriction; C9 – conditional expression; C10 – recursion; C11 – subquery in Select; C12 – subquery in From; C13 – subquery in Where; C14 – subquery in Order By; C15 – subquery in Having; C16 – conditional expression in Select; C17 – conditional expression in Where; C18 – conditional expression in Order By; C19 – conditional expression in Having; C20 – availability of CASE tool.



(a) Which theaters presented all films directed by Spielberg in 1980?

(b) Which theaters presented films directed by Spielberg?

Fig. 23. Main features of OQD [27].

Concerning RQ2, Figure 23a shows a subquery in the Where clause (C13). Regarding RQ3, the language has no CASE tool. In summary, this study includes criteria C1, C5, C6, and C13.

6 ANALYSIS AND DISCUSSION

In this section, the answers to the research questions listed in Section 3.1 are presented.

6.1 RQ1: what is the frequency of each complex construction?

In this research question, the number of VQLs supporting each complex construction (i.e., subquery, join, set operation, conditional expression, grouping restriction, and recursion) was investigated.

This question is important because, beyond showing how common each complex constructor is, it also helps to determine the expressiveness of each VQL.

The results are shown in Table 4. Note that: inner joins are available in almost all articles (95.23%, 20 articles); non-correlated subqueries are available in most articles (85.71%, 18 articles); correlated subqueries are present in more than half of the articles (52.38%, 11 articles); set operations and grouping restrictions are available in a third of the articles (33.33%, 7 articles); outer joins, cross joins, and recursion are present in few articles, 19.04% (4 articles), 9.52% (2 articles) and 4.76% (1 article), respectively; and no article supports natural joins or conditional expressions. The presence of two or more tables in the From clause, with or without a join condition were considered as the inner join and cross join constructs, respectively.

Table 4. Frequency of each complex construction.

Complex construction	Articles	Count	%
C1 – non-correlated subquery	A01, A03, A04, A05, A06, A07, A08, A09, A10, A12, A13, A15, A16, A17, A18, A19, A20, A21	18	85.71
C2 – correlated subquery	A01, A02, A03, A04, A07, A08, A11, A12, A14, A15, A19	11	52.38
C3 – cross join	A10, A14	2	9.52
C4 – natural join	∅	0	0
C5 – inner join	A01, A02, A03, A04, A05, A06, A07, A08, A09, A10, A11, A12, A13, A14, A15, A16, A18, A19, A20, A21	20	95.23
C6 – outer join	A01, A14, A15, A21	4	19.04
C7 – set operation	A06, A10, A14, A15, A17, A18, A19	7	33.33
C8 – grouping restriction	A01, A03, A06, A07, A12, A15, A20	7	33.33
C9 – conditional expression	∅	0	0
C10 – recursion	A03	1	4.76

The analysis of the results shows that the lack of support for conditional expressions and the poor support for recursion, cross joins, outer joins, set operations, and grouping restrictions significantly affect the expressiveness of VQLs because the semantics of these complex constructions cannot be replaced by other complex constructions; the use of conditional expressions is necessary to create real-world business queries that depend on the conditional logic of procedural languages [21]; and because the use of recursion is mandatory for specifying queries with transitive closure (i.e., queries that allow chaining based on the previous value of an attribute). Without recursion, this type of query needs to use a procedure in which a table represents the transitive closure [18]. Cross joins are useful to obtain a Cartesian product and reshape and create data dynamically (e.g., periods intersecting calendar or unpivot columns), especially when the creation of tables in the database is not allowed [28]. Outer joins are commonly used, and there are queries that can only be specified using this constructor (e.g., obtaining optional data), and the use of grouping restrictions is important for specifying queries with conditions on summarized groups. Despite the fact that natural joins were not available in any article, the lack of support for this construct does not affect the expressiveness of the VQL because its use is not recommended and its semantics can be replaced with either an inner or outer join.

6.2 RQ2: what is the frequency of each SQL clause in supporting subqueries or conditional expressions?

For this research question, the number of VQLs that allows the specification of a subquery or conditional expression within the various SQL clauses was investigated. That is, each VQL was examined to see if it supports the definition of subqueries in the Select, From, Where, Order by, or

Having clauses and the definition of conditional expressions in Select, Where, Order by, or Having clauses.

The results are shown in Table 5. Almost all VQLs allow subqueries in the Where clause (95.23%, 20 articles), only one VQL allows subqueries in the Having clause (4.76%; 1 article), only one VQL allows subqueries in the Select clause (4.76%, 1 article), while no VQL allows subqueries in the From or Order By clauses, nor does any VQL allow conditional expressions in the SQL clauses.

Table 5. Frequency of each SQL clause in supporting subqueries or conditional expressions.

SQL clauses	Articles	Count	%
C11 – subquery in the Select clause	A14	1	4.76
C12 – subquery in the From clause	∅	0	0
C13 – subquery in the Where clause	A01, A02, A03, A04, A05, A06, A07, A08, A09, A10, A11, A12, A13, A15, A16, A17, A18, A19, A20, A21	20	95.23
C14 – subquery in the Order By clause	∅	0	0
C15 – subquery in the Having clause	A15	1	4.76
C16 – conditional expression in the Select clause	∅	0	0
C17 – conditional expression in the Where clause	∅	0	0
C18 – conditional expression in the Order By clause	∅	0	0
C19 – conditional expression in the Having clause	∅	0	0

The analysis of the results shows that, although all articles support subquery, almost all support this constructor only in the Where clause. The lack of support for a subquery specification in the Select, From, Having, and Order By clauses significantly affect the expressiveness of the VQL, because: subqueries in the Select clause are helpful in calculating aggregate function values (e.g., COUNT, AVG, MIN, MAX) from a subquery in the projection of the main query; and subqueries in the From clause are important because they allow for the application of more than one aggregate function to a column. The first aggregate function is applied to the subquery in the From clause, and the second aggregate function is applied to the main query. It is also useful for creating optimized queries because it can restrict table rows and columns before performing a Cartesian product or join [17]. Subqueries in the Having clause are useful in grouping restrictions to avoid having to compare against hardcoded values. Note that a comparison with an aggregate value can only be specified in the Having clause, because it will generate an SQL syntax error if specified in the Where clause. Subqueries in the Order By clause are helpful in determining sort order and OFFSET or FETCH values from the rows based on a given subquery result. Note that no article supports conditional expressions. The lack of this constructor in the Select, Where, Having, and Order By clauses also affects the expressiveness of the VQLs because it is not possible to specify conditional logic without the use of procedures, making the SQL code proprietary.

6.3 RQ3: what is the availability of CASE tool?

This research question investigated whether the CASE tool of a VQL was available, unavailable, or did not exist. This feature is critical for the practical use of the VQL. That is, without this feature, it is impossible to perform forward or reverse engineering, making the VQL merely a simple artifact for documentation.

The results are shown in Table 6. Note that more than half of the VQLs (66.66%, 14 articles) have a tool that is not available for download, most of the rest did not have a tool (28.57%, 6 articles), and only one had a tool available to the end user (i.e., 5.76%; 1 article).

Table 6. Availability of CASE tool.

Availability (C20)	Articles	Count	%
CASE tool available	A08	1	4.76
CASE tool unavailable	A01, A03, A05, A06, A07, A09, A11, A13, A14, A15, A16, A17, A19, A20	14	66.66
There is no CASE tool	A02, A04, A10, A12, A18, A21	6	28.57

The analysis of results shows that most of the VQL proposals supposedly developed a CASE tool, but only article A08 provides access to this tool to end users. The unavailability of the CASE tools makes it difficult to use the VQL in either academia or industry. In the academic sphere, a CASE tool is a proof of concept for implementation of the VQL and can be used as a form of empirical evaluation. In the industry, without a CASE tool, there is no reason to use the VQL.

6.4 Validity threats

This section presents concerns related to validity threats to this SLR. That is, the possibility that the results obtained are not in accordance with reality is discussed. According to Maxwell Maxwell [33], validity threats can be classified into four categories:

- **Construct validity**: this threat is related to scenarios in which the conclusions obtained are valid, but where it must be verified that the factors leading to such conclusions correspond to reality. A threat to construct validity would be the search terms and their organization within the search string. It is possible that the constructed string might not be including an important term in the search. To reduce this threat, the search terms were designed to focus on the objectives of the search questions and cover the maximum possible results (i.e., plural and singular terms were used);
- **Internal validity**: this threat refers to unplanned situations that may have occurred during the review process and, to reduce this threat, the studies included were peer reviewed;
- **External validity**: this threat is related to the generalization of the search results and, to reduce this threat, validations were performed on the search terms used. That is, the search was performed and the results evaluated to see if they satisfied the objectives;
- **Conclusion validity**: this threat refers to the verification of randomness in the results obtained with regard to the planned scenario and, to reduce this threat, the inclusion, exclusion, and quality criteria were planned and applied with the assistance of specialists.

7 RELATED WORK

In this section, two papers [11, 32] are presented because they are also reviews of VQL literature. These studies were found during the review process in the selection phase of this SLR.

In [11], the authors reviewed 80 articles presenting VQLs published between 1975 and 1996. The objective of the article was to classify the visual representation and interaction strategy of VQLs for relational databases. The visual representation was analyzed based on the formulation of queries and display of results, which can be based on forms, diagrams, icons or hybrids. The results of this analysis showed that diagrams were the most commonly used visual representations for query specification and that forms were the most common visual representation for displaying the results of those queries. With regard to interaction strategies, the VQL was classified according to comprehension strategy (top-down, navigation, or simplification of schema) and the query formulation strategy (by schema navigation, by subqueries, by correspondence, or by interval selection). According to the results, the most commonly used strategy for understanding queries

was navigation, whereas the most commonly used strategy for formulating queries was by schema navigation. In addition to these classifications, the authors analyzed the category of users, usability, and formal VQL aspects, reporting five points: 1) non-expert users preferred icon-based VQLs; 2) diagram-based VQLs were better for formulating high complexity queries with; 3) usability was most commonly evaluated by measuring the time to specify and understand the query; 4) few VQLs had formally defined syntax or semantics; and 5) few VQLs were formally analyzed for expressivity. Finally, the article emphasized that increasing expressiveness and specification of semantics using a formal approach are open questions.

The work of Lloret-Gazo [32] extended the revision of [11], increasing the period of research from 1997 to 2016, including 34 VQLs. The purpose of the article was to search for VQLs that make query specification a simpler task for non-specialist users. The VQLs found were evaluated according to the availability of the tool on the internet (i.e., accessible through a Web browser) and the maturity level of the tool (i.e., tool-less, prototype, user tested, tested in a real environment, or commercial tool). The results showed that, with regard to internet availability, only five could be accessed through the Web, indicating that this feature was rarely considered; and with regard to tool maturity, 13 were prototypes, 10 had been tested by users, 3 had been tested in a real environment, and only 1 had become a commercial tool. It was concluded that few studies described actual implementations, which could explain the fact that VQLs are not a widely-adopted solution.

It is important to highlight that the studies of [11, 32] did not create an SLR through the specification of a search string, which hinders the reproduction or verification of the presented results. Furthermore, since the purpose of this paper is to investigate why VQLs are not widely-used to specify complex queries, it presents scientific contributions different from these related proposals.

8 CONCLUSION

This paper presents the results of an SLR that investigates why VQLs are not widely-used for specifying complex SQL queries. In this review, the search was performed automatically and the evaluation was carried out by the authors using well-defined inclusion, exclusion, and quality criteria. Initially, an overview of the selected articles was presented. The expressiveness of VQL for specifying complex SQL queries (RQ1 and RQ2) and the availability of CASE tools (RQ3) were investigated. Finally, a discussion of the results was presented. Twenty-one articles proposing VQLs were analyzed, and these provided the evidence to answer the three research questions. The research questions addressed in this article and the respective answers are summarized below.

RQ1: what is the frequency of each complex construction? It was found that the inner join, non-correlated subquery, and correlated subquery constructs are available in most articles; the set operation, outer join, cross join, and recursion are present in a few articles; while no article supports natural joins or conditional expressions. Beyond that, it was observed that the lack of support for conditional expressions and the poor support for recursion, cross joins, outer joins, set operations, and grouping restrictions significantly affected the expressiveness of the VQLs.

RQ2: what is the frequency of each SQL clause in supporting subqueries or conditional expressions? It was found that almost all VQLs support subqueries only in the Where clause and that no VQL allows conditional expressions in any SQL clause. It was observed that the lack of support for the specification of subqueries in the Select, From, Having, and Order By clauses and the lack conditional expression support in any SQL clause significantly affected the expressiveness.

RQ3: what is the availability of CASE tool? Only one tool was found to be available for the end user and more than half of the VQLs have a tool that is unavailable for download.

Based on these results, this SLR is useful in assisting researchers in the development of a new VQL that focuses on the power of expression, since a VQL that supports the highlighted complex constructions and that has a CASE tool available to the end users can be a solution that will

encourage the use of VQLs for specifying real-world business queries over both relational and non-relational databases. A future study should propose a VQL that supports all investigated complex constructions and a CASE tool. In this sense, the specification of a Domain Specific Modeling Language (DSML) [10] and its CASE tool could be an interesting proposal.

REFERENCES

[1] Azza Abouzied, Joseph Hellerstein, and Avi Silberschatz. 2012. DataPlay: Interactive Tweaking and Example-driven Correction of Graphical Database Queries. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology (UIST '12)*. ACM, New York, NY, USA, 207–218. https://doi.org/10.1145/2380116.2380144

[2] ACM/IEEE. 2013. *Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science*. ACM, NY, USA. 999133.

[3] ACM/IEEE. 2017. *Information Technology Curricula 2017: Curriculum Guidelines for Baccalaureate Degree Programs in Information Technology*. ACM, NY, USA.

[4] Michael Armbrust, Reynold S Xin, Cheng Lian, Yin Huai, Davies Liu, Joseph K Bradley, Xiangrui Meng, Tomer Kaftan, Michael J Franklin, Ali Ghodsi, et al. 2015. Spark sql: Relational data processing in spark. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. ACM, 1383–1394.

[5] David F Bacon, Nathan Bales, Nico Bruno, Brian F Cooper, Adam Dickinson, Andrew Fikes, Campbell Fraser, Andrey Gubarev, Milind Joshi, Eugene Kogan, et al. 2017. Spanner: Becoming a SQL System. In *Proceedings of the 2017 ACM International Conference on Management of Data*. ACM, 331–343.

[6] Eirik Bakke and David R. Karger. 2016. Expressive Query Construction through Direct Manipulation of Nested Relational Results. *Proceedings of the 2016 International Conference on Management of Data - SIGMOD '16* (2016), 1377–1392. https://doi.org/10.1145/2882903.2915210

[7] Eirik Bakke and David R. Karger. 2016. Expressive Query Construction Through Direct Manipulation of Nested Relational Results. In *Proceedings of the 2016 International Conference on Management of Data (SIGMOD '16)*. ACM, New York, NY, USA, 1377–1392. https://doi.org/10.1145/2882903.2915210

[8] Anthony C Bloesch and Terry A Halpin. 1997. Conceptual queries using ConQuer-II. In *International Conference on Conceptual Modeling*. Springer, 113–126.

[9] Clemente Rafael Borges and José Antonio Macías. 2010. Feasible Database Querying Using a Visual End-user Approach. In *Proceedings of the 2Nd ACM SIGCHI Symposium on Engineering Interactive Computing Systems (EICS '10)*. ACM, New York, NY, USA, 187–192. https://doi.org/10.1145/1822018.1822047

[10] Marco Brambilla, Jordi Cabot, and Manuel Wimmer. 2017. Model-Driven Software Engineering in Practice: Second Edition. *Synthesis Lectures on Software Engineering* 3, 1 (2017), 1–207. https://doi.org/10.2200/S00751ED2V01Y201701SWE004

[11] Tiziana Catarci, Maria F Costabile, Stefano Levialdi, and Carlo Batini. 1997. Visual query systems for databases: A survey. *Journal of Visual Languages & Computing* 8, 2 (1997), 215–260.

[12] Tiziana Catarci and Giuseppe Santucci. 1995. *Diagrammatic Vs Textual Query Languages: A Comparative Experiment*. Springer US, Boston, MA, 69–83. https://doi.org/10.1007/978-0-387-34905-3_5

[13] C. Cerullo and M. Porta. 2007. A System for Database Visual Querying and Query Visualization: Complementing Text and Graphics to Increase Expressiveness. In *18th International Workshop on Database and Expert Systems Applications (DEXA 2007)*. 109–113. https://doi.org/10.1109/DEXA.2007.91

[14] Don Chamberlin. 2009. *SQL*. Springer US, Boston, MA, 2753–2760. https://doi.org/10.1007/978-0-387-39940-9_1091

[15] B. D. Czejdo, R. P. Tucci, D. W. Embley, and S. W. Liddle. 1993. Graphical query specification with participation constraints. In *Computing and Information, 1993. Proceedings ICCI '93., Fifth International Conference on*. 433–437. https://doi.org/10.1109/ICCI.1993.315334

[16] Jonathan Danaparamita and Wolfgang Gatterbauer. 2011. QueryViz: Helping Users Understand SQL Queries and Their Patterns. In *Proceedings of the 14th International Conference on Extending Database Technology (EDBT/ICDT '11)*. ACM, New York, NY, USA, 558–561. https://doi.org/10.1145/1951365.1951440

[17] Djoni Darmawikarta. 2016. *Oracle SQL, A BeginnerâĂŹs Tutorial*. Brainy Software Inc.

[18] Sami El-Mahgary and Eljas Soisalon-Soininen. 2015. A Form-based Query Interface for Complex Queries. *J. Vis. Lang. Comput.* 29, C (Aug. 2015), 15–53. https://doi.org/10.1016/j.jvlc.2015.03.001

[19] Ramez Elmasri and Shamkant B. Navathe. 2016. *Fundamentals of Database Systems, Seventh Edition*. Person, Boston, MA, USA.

[20] Richard G. Epstein. 1991. The TableTalk query language. *Journal of Visual Languages & Computing* 2, 2 (1991), 115 – 141. https://doi.org/10.1016/S1045-926X(05)80026-6

[21] Jarek Gryz, Qiong Wang, Xiaoyan Qian, and Calisto Zuzarte. 2008. SQL Queries with CASE Expressions. In *Foundations of Intelligent Systems*. Springer Berlin Heidelberg, Berlin, Heidelberg, 351–360.

[22] Jozef HvoreckÃ¡, Martin DrlÃŋk, and Michal Munk. 2010. The effect of visual query languages on the improvement of information retrieval skills. *Procedia - Social and Behavioral Sciences* 2, 2 (2010), 717 – 723. https://doi.org/10.1016/j.sbspro.2010.03.090 Innovation and Creativity in Education.

[23] Yannis E. Ioannidis. 1996. Visual User Interfaces for Database Systems. *ACM Comput. Surv.* 28, 4es, Article 137 (Dec. 1996). https://doi.org/10.1145/242224.242399

[24] Hannu Jaakkola and Bernhard Thalheim. 2003. *Visual SQL – High-Quality ER-Based Query Treatment.* Springer Berlin Heidelberg, Berlin, Heidelberg, 129–139. https://doi.org/10.1007/978-3-540-39597-3_13

[25] J. Kawash. 2004. Complex quantification in Structured Query Language (SQL): A tutorial using relational calculus. *Journal of Computers in Mathematics and Science Teaching* 23, 2 (2004), 169–190.

[26] Daniel A. Keim and Vincent Lum. 1992. Visual Query Specification in a Multimedia Database System. In *Proceedings of the 3rd Conference on Visualization '92 (VIS '92)*. IEEE Computer Society Press, Los Alamitos, CA, USA, 194–201. http://dl.acm.org/citation.cfm?id=949685.949722

[27] Euclid Keramopoulos, Philippos Pouyioutas, and Chris Sadler. 1997. GOQL, A Graphical Query Language for Object-Oriented Database Systems. In *Proceedings of the 3rd Basque International Workshop on Information Technology (BIWIT '97) (BIWIT '97)*. IEEE Computer Society, Washington, DC, USA, 35–. http://dl.acm.org/citation.cfm?id=523989.791426

[28] Johan Kettan. 2013. The Cross Join Collection. https://community.tableau.com/docs/DOC-5247

[29] Barbara Kitchenham and Stuart Charters. 2007. *Guidelines for performing Systematic Literature Reviews in Software Engineering.* Technical Report. Keele University and Durham University Joint Report.

[30] Anthony C. Klug. 1981. Abe: A Query Language for Constructing Aggregates-by-example. In *Proceedings of the 1st LBL Workshop on Statistical Database Management (SSDBM'81)*. Lawrence Berkeley Laboratory, Berkeley, CA, US, 190–205. http://dl.acm.org/citation.cfm?id=1115950.1115976

[31] Jae-Cheol Kwak and Songchun Moon. 1995. Two-dimensional specification of queries in object-oriented databases. *Microprocessing and Microprogramming* 41, 3 (1995), 227 – 244. https://doi.org/10.1016/0165-6074(95)00005-9

[32] Jorge Lloret-Gazo. 2016. A Survey on Visual Query Systems in the Web Era. In *Database and Expert Systems Applications*, Sven Hartmann and Hui Ma (Eds.). Springer International Publishing, Cham, 343–351.

[33] Joseph A Maxwell. 2012. *Qualitative research design: An interactive approach.* Vol. 41. Sage publications.

[34] L. Mohan and R. L. Kashyap. 1993. A Visual Query Language for Graphical Interaction with Schema-Intensive Databases. *IEEE Trans. on Knowl. and Data Eng.* 5, 5 (Oct. 1993), 843–858. https://doi.org/10.1109/69.243513

[35] D. Moody. 2009. The Physics of Notations: Toward a Scientific Basis for Constructing Visual Notations in Software Engineering. *IEEE Transactions on Software Engineering* 35, 6 (Nov 2009), 756–779. https://doi.org/10.1109/TSE.2009.67

[36] Leong Mun-Kew, Choo Boon-Siong, Kok Chun-Hong, Lim Jyh-Jang, and Desai Narasimhalu. 1990. The implementation of a visual language interface for an object-oriented multimedia database system. *Journal of Visual Languages & Computing* 1, 3 (1990), 275 – 289. https://doi.org/10.1016/S1045-926X(05)80010-2

[37] Norman S. Murray, Norman W. Paton, Carole A. Goble, and Joanne Bryce. 2000. KaleidoqueryâĂŤA Flow-based Visual Language and its Evaluation. *Journal of Visual Languages & Computing* 11, 2 (2000), 151 – 189. https://doi.org/10.1006/jvlc.1999.0150

[38] H. B. Ramos. 1993. Design and Implementation of a Graphical SQL with Generic Capabilities. In *Interfaces to Database Systems (IDS92)*, Richard Cooper (Ed.). Springer London, London, 74–91.

[39] Gary H. Sockut, Luanne M. Burns, Ashok Malhotra, and Kyu-Young Whang. 1993. GRAQULA: A graphical query language for entity-relationship or relational databases. *Data & Knowledge Engineering* 11, 2 (1993), 171 – 202. https://doi.org/10.1016/0169-023X(93)90004-9

[40] Eunjee Song, Shuxin Yin, and Indrakshi Ray. 2007. Using UML to model relational database operations. *Computer Standards & Interfaces* 29, 3 (2007), 343 – 354. https://doi.org/10.1016/j.csi.2006.05.006

[41] StackOverflow. 2018. Stack Overflow Developer Survey Results 2018. https://insights.stackoverflow.com/survey/2018. Accessed: 2018-12-14.

[42] K. P. Tan, H. C. Chan, and K. L. Siau. 1990. A graphical knowledge level approach for user-database interaction. In *Proceedings., Fourteenth Annual International Computer Software and Applications Conference*. 453–458. https://doi.org/10.1109/CMPSAC.1990.139408

[43] A. U. Tansel, M. E. Arkun, and G. Ozsoyoglu. 1989. Time-by-example query language for historical databases. *IEEE Transactions on Software Engineering* 15, 4 (April 1989), 464–478. https://doi.org/10.1109/32.16597

[44] Colin Ware. 2004. Chapter 9 - Images, words, and gestures. In *Information Visualization (Second Edition)* (second edition ed.), Colin Ware (Ed.). Academic Press, San Diego, 297 – 316. https://doi.org/10.1016/B978-155860819-1/50012-1

[45] K. Y. Whang, A. Malhotra, G. H. Sockut, L. Burns, and K. S. Choi. 1992. Two-dimensional specification of universal quantification in a graphical database query language. *IEEE Transactions on Software Engineering* 18, 3 (Mar 1992), 216–224. https://doi.org/10.1109/32.126770

[46] G. Zhang, W. W. Chu, F. Meng, and G. Kong. 1999. Query formulation from high-level concepts for relational databases. In *Proceedings User Interfaces to Data Intensive Systems*. 64–74. https://doi.org/10.1109/UIDIS.1999.791463