SchenQL: Evaluation of a Query Language for Bibliographic Metadata

Christin K. Kreutz[0000−0002−5075−7699]1, Michael Wolz[0000−0002−9313−7131], Benjamin Weyers[0000−0003−4785−708X]1, and Ralf Schenkel[0000−0001−5379−5191]1

[1] Trier University, 54286 Trier, DE
{kreutzch, weyers, schenkel}@uni-trier.de

Abstract. Information access needs to be uncomplicated, as users may not benefit from complex and potentially richer data that may be less easy to obtain. A user's demand for answering more sophisticated research questions including aggregations could be fulfilled by the usage of SQL. However, this comes with the cost of high complexity, which requires for a high level of expertise even for trained programmers. A domain-specific query language could provide a straightforward solution to this problem. Although less generic, it is desirable that users not familiar with query construction are supported in the formulation of complex information needs.
In this paper, we extend and evaluate SchenQL, a simple and applicable query language that is accompanied by a prototypical GUI. SchenQL focuses on querying bibliographic metadata while using the vocabulary of domain-experts. The easy-to-learn domain-specific query language is suitable for domain-experts as well as casual users while still providing the possibility to answer complicated queries. Query construction and information exploration is supported by the prototypical GUI. Eventually, the complete system is evaluated: interviews with domain-experts and a bipartite quantitative user study demonstrate SchenQL's suitability and high level of users' acceptance.

Keywords: domain-specific query language, bibliographic metadata, digital libraries, graphical user interface.

1 Introduction

Scientific writing almost always starts with a thorough bibliographic research on relevant publications, authors, conferences, journals and institutions. While web search is excellent for question answering and intuitively performed, not all retrieved information is correct, unbiased and categorized [11]. The arising problem is people's tendency of rather using poor information sources that are easy to query than more reliable sources which might be harder to access [12]. This introduces the need for more formal and structured information sources such as digital libraries specialised in the underlying data, that at the same time need to be easy to query. Existing interfaces of digital libraries often provide

keyword search on metadata or to query attributes [1,3,6,22,30]. However, in many cases, these interfaces do not allow to directly express advanced queries such as *"Which are the five most cited articles written by person P about topic T after year Y?"*, but require complex interaction. Popular examples of such limited systems are dblp [30] or Semantic Scholar [6]. More complex tools such as GrapAL [2,15] are capable of answering said complex queries, but require specific and uncommon programming skills. Another option is to use structured query languages such as SQL, a widespread language for querying databases, which unfortunately tends to be difficult to master [39]. This is critical as in most cases domain-experts are familiar with the schema of the data but are not experienced in using all-purpose query languages such as SQL [9,31]. This is even worse for casual users of digital libraries who neither have knowledge of the structure of the data nor of SQL.

To close this gap, we presented the SchenQL Query Language for the domain of bibliographic metadata [28]. SchenQL is designed to be easily utilised by domain-experts as well as casual users as it uses the vocabulary of digital libraries in its syntax. While domain-specific query languages (DSLs) provide a multitude of advantages [17], the most important aspect in the conception of SchenQL was that no programming skills or database schema knowledge is required to use it. For SchenQL to be widely applicable, we introduce a prototypical graphical user interface (the SchenQL GUI) which supports the construction of queries, offers visualisations of query results and an additional dimension of retrieving information by exploring data and its relations through clicking. As an example of SchenQL, the aforementioned question can be formulated as follows: `MOST CITED (ARTICLES WRITTEN BY "P" ABOUT "T" AFTER Y) LIMIT 5`.

The major contribution of this paper is the empirical evaluation of SchenQL as domain-specific query language on bibliographic metadata including the investigation of a prototypical GUI that is designed to assist users in creating queries.SchenQL is evaluated two-fold: 1) interviews with domain-experts were conducted to identify applications as well as options for further development and 2) a quantitative user study consisting of two parts measured effectiveness, efficiency and users' satisfaction with our whole system: we first evaluated usage of command line SchenQL against SQL, followed by a study which compared usage of the SchenQL GUI to the previous results. Here, the User Experience Questionnaire [36] was conducted for assessment of user experience.

The remainder of this paper is structured as follows: Section 2 discusses related work. Section 3 introduces the structure and syntax of SchenQL including the presentation of the SchenQL GUI, which is evaluated in two parts in the following Section 4. The last Section 5 describes possible future research.

2 Related Work

Areas adjacent to the one we are tackling are *search on digital libraries*, *search interfaces on bibliographic metadata* and *domain-specific query languages*.

For *search on digital libraries*, the MARC format is a standard for information exchange [11]. While it is useful for known-item search, topical search might be problematic as contents of the corresponding fields can only be interpreted by domain-experts [11]. Most interfaces on digital libraries provide field-based Boolean search [35] which can lead to difficulties in formulating queries that require the definition and concatenation of multiple attributes. This might cause a substantial cognitive workload on the user [14]. In contrast, withholding or restriction of faceted search on these engines fails to answer complex search tasks [13]. Thus, we focus on a search of topical information that even casual users can utilise while also offering the possibility to clearly define search terms for numerous attributes in a single query.

Several *search interfaces on bibliographic metadata* exist such as dblp [26,30], Bibsonomy [22], Google Scholar [1], ResearchGate [3] or Semantic Scholar [6]. All of those systems allow for systematic refinement of result sets by application of filter options via facets to varying extends. Only dblp and Semantic Scholar (on a small scale) support search on venues. The formulation of complex queries with aggregations is not targeted by any of them. In contrast, SchenQL supported by a GUI specialises on these functionalities. GrapAL [2,15] actually provides all functions of SchenQL but is a complex tool utilising the Cypher [20] query language (QL).

Domain-specific query languages come in various shapes. They can be SQL-like [29], visual QLs [9,18] or use domain-specific vocabulary [38] but are typically specialised on a certain area. They also come in different complexities: for example MathQL [21] is a query language in markup style on RDF repositories but a user needs to be mathematician to be able to operate it. The DSL proposed by Madaan [31] stems from the medical domain and is designed to be used by inexperienced patients as well as medical staff. Some DSLs are domain-unspecific such as the aforementioned Cypher [20], BiQL [19] or SnQL [32] and depend on complicated SQL-like syntax. With SchenQL, we provide a QL which uses vocabulary from the domain of bibliographic metadata while being useful for experts as well as casual users and avoiding complicated syntax.

3 SchenQL: QL and GUI

For simplicity, we refer to SchenQL including its GUI as the *SchenQL system*. SchenQL was developed to access bibliographic metadata textually, which resembles natural language for casual as well as expert users of digital libraries [27]. The fundamental idea is to hide complex syntax behind plain domain-specific vocabulary. This enables usage from anyone versed in the vocabulary of the domain without experience in sophisticated query languages such as SQL. The prototypical GUI supports SchenQL: it helps in query formulation with auto-completion and keyword suggestion. Additionally, it provides visual exploration of query results supporting two standard visualisations: Ego Graph [34] and BowTie [25].

For our data model we assume bibliographic metadata consists of persons and publications they authored or edited. These persons can be affiliated with

	PUBLICATION	PERSON	CONFERENCE	JOURNAL	INSTITUTION
L	key, title	key, primary name, orcid	key, acronym	key, acronym	
S	MASTERTHESIS, BOOK, CHAPTER, PHDTHESIS, ARTICLE	AUTHOR, EDITOR			
F	PUBLISHED BY (I), ABOUT (keywords), WRITTEN BY (PE), EDITED BY (PE), APPEARED IN (C|J), BEFORE year, IN YEAR year, AFTER year, TITLED title, REFERENCES (PU), CITED BY (PU)	PUBLISHED IN (C|J), PUBLISHED WITH (I), WORKS FOR (I), NAMED name, ORCID orcid, AUTHORED (PU), REFERENCES (PU), CITED BY (PU)	ACRONYM acronym, ABOUT (keywords), BEFORE year, IN YEAR year, AFTER year	NAMED name, ACRONYM acronym, ABOUT (keywords), BEFORE year, IN YEAR year, AFTER year, VOLUME volume	NAMED name, CITY city, COUNTRY country, MEMBERS (PE)
V	title	primary name	acronym	acronym	primary name + location

Table 1: SchenQL base concepts `Publications` (`PU`), `persons` (`PE`), `conferences` (`C`), `journals` (`J`) and `institutions` (`I`) with their respective literals (L), specialisations (S), filters (F) and standard return values (V, relevant for the CLI).

certain institutions. Publications can be of multiple types and may be published in conferences or journals. Publications can reference previously published papers and might be cited themselves by more recent work building upon them.

3.1 Building Blocks

Base concepts are the basic return objects of SchenQL. A base concept is connected to an entity of the dataset and has multiple attributes. Those base concepts are `publications`, `persons`, `conferences`, `journals` and `institutions`. Upon these concepts, queries can be constructed. Base concepts can be specialised. For example `publications` can be refined by specialisations `books`, `chapters`, `articles`, `master` or `PhD theses`. A specialisation can be used instead of a base concept in a query.

Filters can restrict base concepts by extracting a subset of the data. Literals can be used as identifiers for objects from base concepts, they can be utilised to query for specific data. Attributes of base concepts can be queried, for a list of attributes see Kreutz et al. [28]. Table 1 gives an overview of literals, specialisations, filters and the standard return value for every base concept.

Functions are used to aggregate data or offer domain-specific operations. Right now, four functions are implemented in SchenQL: `MOST CITED`, `COUNT`, `KEYWORDS OF` and `COAUTHORS OF`. The function `MOST CITED (PUBLICATION)` can be applied on publications. It returns titles as well as numbers of citations of papers in the following set. By default, the top five results are returned. `COUNT` returns the number of objects contained in the following sub-query. `KEYWORDS OF (PUBLICATION | CONFERENCE | JOURNAL)` returns the keywords associated with the following base concept. `COAUTHORS OF (PERSON)` returns the coauthors of an author. The `LIMIT` x operator with $x \in \mathbb{N}$ can be appended at the end of any query to change the number of displayed results to at most x.

3.2 Syntax and Implementation

The syntax of SchenQL follows simple rules resulting in queries similar to natural language which are aiming at simple construction. Sub-queries have to be surrounded by parentheses. It is possible to write singular or plural when using base concepts or specialisations (e.g. `JOURNAL` or `JOURNALS`). Filters following base concepts or their specialisations can be in arbitrary order and get connected via conjunction if not specified otherwise (`OR` and `NOT` are also possible). Most filters expect a base concept as parameter (e.g. `WRITTEN BY (PERSONS)`), however some filters anticipate a string as parameter (e.g. `COUNTRY "de"`). Specialisations can be used in place of base concepts. Instead of a query `PERSON NAMED "Ralf Schenkel"` a specialisation like `AUTHOR NAMED "Ralf Schenkel"` would be possible. If a filter requires a base concept, parentheses are needed except for the case of using literals for identifying objects of the base concept. For example `PUBLICATIONS WRITTEN BY "Ralf Schenkel"` is semantically equivalent to `PUBLICATIONS WRITTEN BY (PERSONS NAMED "Ralf Schenkel")`. Attributes of base concepts can be accessed by putting the queried for attribute(s) in front of a base concept and connecting both parts with an `OF` (e.g. `"name", "acronym" OF CONFERENCES ABOUT KEYWORDS ["DL", "QLs"]`).

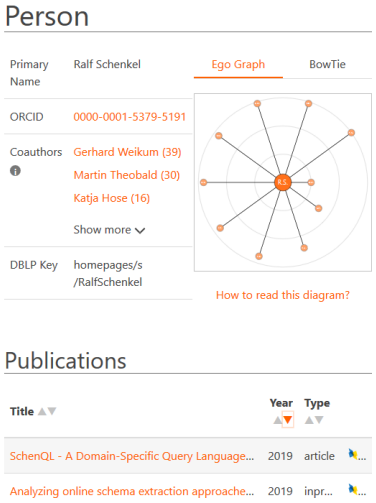
For implementation, lexer and parser of the compiler for SchenQL were built using ANTLR with Java as target language. The compiler translates queries from SchenQL to SQL and runs them against a MySQL 8.0.16 database holding the data. Data on references and citations is contained in a single table. SchenQL can be used in a terminal client similar to the MySQL shell.

3.3 GUI

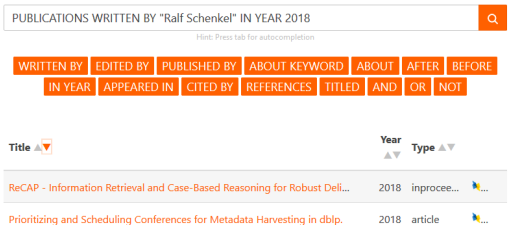
The GUI is inspired by results from the qualitative study described in Section 4.2. It provides access to information by supporting the construction of queries including the interactive navigation with the GUI. It also offers autocompletion of SchenQL query keywords and suggestions for the formulation of queries. Results of queries can be sorted for every column of the result table. In Figure 1b query formulation with suggested keywords and result representation in the SchenQL GUI is depicted. If a search result is selected by clicking on it, detail views open (see Figure 1a) which offer all information available for the respective element of a base concept. Furthermore we incorporated two already established visualisations: *Ego Graph* [34] and *BowTie* [25]. The *Ego Graph* for persons (see Figure 1a top) supports the analysis of persons' most important co-authorships. The *BowTie* visualisation can be used for easy estimation of a person's, publication's or venue's influence in terms of gained citations and its actuality (see Figure 1c).

4 Evaluation

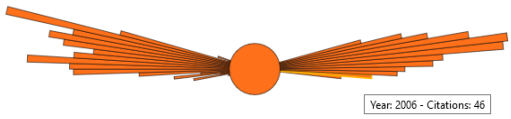
Our evaluation of the SchenQL system consists of a qualitative and a quantitative investigation. In a first qualitative study, we examine domain experts' use-cases



(a) Person detail view with Ego Graph. Nodes symbolise authors, the further an author is from the middle, the less publications he shares with the person in focus.



(b) SchenQL GUI for a search with suggested language components and search result.



(c) BowTie view of referenced and citing papers of a person with numbers of referenced(bows left of knot)/citing(bows right of knot) papers per year in single slices. The higher the number of citations or references, the longer the bow.

Figure 1: SchenQL GUI (top right), detail view of a person with Ego Graph (left) and BowTie view (bottom right).

and desired functionality of a DSL such as SchenQL as well as an accompanied GUI. The major goal of this first investigation was to check SchenQL for completeness and suitability for the addressed use cases. In a subsequent step, we conducted a quantitative study in which we first compared SchenQL with SQL, both used through a command line interface (CLI) to ensure comparability. The goal was to measure the effectiveness, efficiency and users' satisfaction with SchenQL as query language. As a follow-up, we evaluated the web-based GUI of the SchenQL system using the same queries and compared the results with those received from usage of the SchenQL CLI. We additionally investigated the SchenQL system's user experience using the User Experience Questionnaire (UEQ) [36].

Considering the overall goals for SchenQL, we derived the following three hypotheses to be investigated:

H_1 Utilisation of the SchenQL CLI achieves better results in terms of higher correctness, lower perceived difficulty of query construction as well as lower required time for query formulation than usage of SQL.

H_2 SchenQL is suitable for domain-experts as well as non-experts.

H_3 The SchenQL system provides high suitability and user experience (indicated by values $> .8$ for all six quality dimensions assessed with the UEQ [7]) for users not familiar with structured queries.

For the studies, we used a dataset from the area of computer science: our structures were filled with data from dblp [30] mapped on data from Semantic Scholar [6] and enriched with information about institutions from Wikidata [8].

4.1 Qualitative Study: Interviews

To get a comprehensive picture of SchenQL's completeness and suitability, we conducted semi-structured one-on-one interviews with four employees of the dblp team to discover realistic use-cases as well as desirable functionalities and potential extensions. Leading questions were which queries they would like to answer with the data and which functions or visualisations they envisioned in a GUI. The participants do work daily on digital libraries and are thus considered highly experienced in the area. They were only aware of the domain of interest and the underlying dataset but did not know anything about SchenQL.

The interviews showed that the dblp staff wished to formulate queries to compute keywords of other publications that were published in the same journal as a given publication, the determination of the most productive or cited authors, as well as the most cited authors with few co-authors. Furthermore, a GUI should support numerous visualisations: colour coded topics of publications or co-author-groups were explicitly asked for. Another participant requested intermateable components for the visualisation of graphs to display co-publications, co-institutions or connections between venues. Other desired functionalities were a fault-tolerant person name search and sophisticated ranking methods.

As expected, the experts' suggestions were quite specific and strongly shaped by their daily work with dblp, which may not fit classic non-expert use of digital libraries. SchenQL is able to formulate several of the desired questions, however it needs to be evaluated by non-power-users as we have done in the quantitative evaluation described below to ensure usability for casual users as well. Comments on visualisation drove the design of the GUI's visual analysis components.

4.2 Quantitative Study: SchenQL CLI vs. SQL, GUI and UEQ

Our quantitative study consists of two parts: the SchenQL CLI is compared to SQL, then the usability of the GUI and thus the SchenQL system as a whole is assessed. For the first part, it is not feasible to compare a specialised system such as the SchenQL CLI to a commercial search engine, differences between the compared systems should be minor [24]. Additionally, as stated above, search interfaces in this domain [1,3,6,22,30] do not provide as many functionalities as SchenQL. We also refrained from evaluating the CLI against other DSLs such as Cypher [20] as test users would have been required to learn two new query languages. Comparing our CLI against SPARQL would have required the definition of classes, properties and labels for the dataset and was therefore also disregarded in favour of the comparison against SQL.

Users participated voluntarily in the study, they were aware of being able to quit any time without negative consequences. They actively agreed on their data being collected anonymously and their screens being captured. We assume gender

Q_1	What are the titles of publications written by author A?
Q_2	What are the names of authors which published at conference C?
Q_3	What are the titles of the publications referenced by author A in year Y?
Q_4	What are the titles of the five most cited publications written by author A?

Table 2: Templates of all queries used in the qualitative evaluations. A are different authors, C is a conference and Y is a year.

does not influence the measured values so it is not seen as additional factor in the evaluation [24]. We assume domain-experts are versed in the vocabulary and connections between bibliographic objects, non-experts might have their first encounter with bibliographic metadata.

For significance tests, we used an independent two-sample t-test in case data is normally distributed (checked with Shapiro-Wilk test) and variances are homogeneous (checked with Levene's test). Otherwise and if we do not specify differently we applied Mann-Whitney U tests. We consider a p-value of .05 as significance level.

Queries In both parts of the study, we asked the participants to find answers to the queries given in Table 2 using either SchenQL CLI/SQL (part I) or the GUI (part II). The used queries are inspired by everyday search tasks of users of digital libraries [16,33]. We formulated four different types of queries targeting core concepts found in the domain. Variables were switched between query languages to prevent learning effects based on query results. Q_1, Q_3 and Q_4 are publication searches while Q_2 targets person search. Q_1 and Q_2 can be answered by using dblp [30] alone. Except for Q_3, Semantic Scholar [6] could technically be used to find answers for the queries. The following formulation of Q_3 in SQL intends to show the complexity of those queries:

```
SELECT DISTINCT title
FROM publication p, publication_references pr
WHERE p.publicationKey = pr.pub2Key AND pr.pub1Key IN (
  SELECT publicationKey
  FROM person_authored_publication NATURAL JOIN person_names
    NATURAL JOIN publication
  WHERE person_names.name = "A" AND year = Y);
```

In SchenQL, the query could be formulated as follows (for all queries see [27]):

```
PUBLICATIONS CITED BY (PUBLICATIONS WRITTEN BY "A" IN YEAR Y);
```

We refrained from evaluating more complex queries to keep the construction time for SQL queries feasible.

Part I: SchenQL CLI vs. SQL

With this first part of the quantitative study, we assess the usability, suitability as well as user satisfaction of usage of the SchenQL CLI compared to SQL for queries typically answered with an information retrieval system operating

	SQL			SchenQL CLI		
	CORR	DIFF	time	CORR	DIFF	time
Q_1	90.48	2.86	4:57	90.48	1.57	2:57
Q_2	90.48	3.	4:35	100.	2.1	3:11
Q_3	23.81	4.86	8:55	47.62	2.71	3:33
Q_4	23.81	5.91	10:36	95.24	1.71	1:53

Table 3: Correctness (CORR) in percent, assessed average difficulty (DIFF) and average time in minutes for the four queries for SQL and the SchenQL CLI.

	SchenQL GUI		
	CORR	DIFF	time
Q_1	90	1.3	1:05
Q_2	90	2.2	1:41
Q_3	40	3.6	2:56
Q_4	90	2.4	2:18

Table 4: Correctness (CORR) in percent, assessed average difficulty (DIFF) and average time in minutes for the four queries for the GUI.

on bibliographic metadata. Additionally, the need for a DSL in the domain of bibliographic metadata is analysed as we try to verify or falsify hypotheses H_1 and H_2. Participants of this evaluation needed to be familiar with SQL.

Setting We defined the evaluation process of our archetypical interactive information retrieval study [24] as follows: every user performed the evaluation alone in presence of a passive investigator on a computer with two monitors. The screens were captured in order to measure times used to formulate the queries. A query language was assigned with which a user was going to start the evaluation to compensate for learning effects. Users were permitted to use the internet at any stage of the evaluation. A SchenQL cheat sheet, the ER diagram and examples for the database schema provided to test subjects can be found in Kreutz et al. [27].

At first, a video tutorial [4] for the introduction and usage of SQL and the SchenQL CLI was shown, afterwards subjects were permitted to formulate queries using the system they were starting to work with. Following this optional step, users were asked to answer a first online questionnaire to assess their familiarity with the domain of bibliographic metadata. Participants were asked to submit the queries in SQL and SchenQL respectively. This part of the first quantitative evaluation was concluded with a second online questionnaire regarding the overall impression of SchenQL, the rating of SchenQL and SQL for the formulation of queries as well as several open questions targeting possible advantages and improvements of SchenQL. We evaluated 21 participants from the area of computer science with SQL knowledge. In total, ten subjects started by using SQL, eleven participants began the evaluation using SchenQL.

Analysis of H_1 To assess validity of hypothesis H_1 of SchenQL leading to better results than using SQL, we observe the *number of correctly formulated queries*, the *rated difficulty* and the *required time* for the formulation of queries with the SchenQL CLI and SQL. For each of these values, we first conducted significance tests on all four queries together, here the two languages SchenQL and SQL were regarded as groups, afterwards we performed significance tests on

each of the four queries. Table 3 gives an overview of correctness, average rated difficulty and average time for all four queries for both languages. Difficulty was rated on a scale from 1 (very easy) to 7 (very difficult) to allow neutral ratings.

Correctness 57.14% of queries were correctly formulated using SQL whereas 83.33% of queries were correctly formulated using SchenQL. This result clearly shows the significantly superior effectiveness of SchenQL compared to SQL in terms of overall correctness. While Q_1 and Q_2 were answered correctly by most participants, the number of correctly formulated queries for Q_3 and Q_4 highly depends on the system. Q_4 was correctly answered by a quarter of subjects using SQL while more than 95% of users were able to formulate the query in SchenQL, this difference is significant. These observations support the partial verification of H_1 in terms of higher number of correctly formulated queries with the SchenQL CLI compared to SQL.

Rated Difficulty The mean rating of difficulty of the formulation of queries with SQL was 4.16 ($\sigma = 1.94$), with SchenQL the mean rating was significantly lower (2.02, $\sigma = 1.11$). On average, query construction using SQL is rated more difficult for every query. The averaged highest rated difficulty for a query in SchenQL is still lower than the averaged lowest rated difficulty of a query in SQL. We found significantly lower ratings of difficulties of queries for all four queries (for Q_3 t-test) when using SchenQL compared to utilisation of SQL. These observations support the partial verification of H_1 in terms lower perceived difficulty in query formulation with the SchenQL CLI compared to SQL.

Time Average construction of queries in SQL took 7:15 minutes ($\sigma = 4{:}47$ min.), with the CLI the construction was significantly quicker and took 2:52 minutes ($\sigma = 1{:}51$ min.) on average. This documents the efficiency of SchenQL. We found significantly lower required times for query formulation all four queries (for Q_1 t-test) when using SchenQL compared to utilisation of SQL. These observations support the partial verification of H_1 in terms of lower required time for query formulation with the SchenQL CLI compared to SQL.

General Results The queries Q_3 and Q_4 in SQL are assumed to be complex which is supported by the low percentage of correct formulations using SQL. They are also much longer than the respective SchenQL ones so the time required to write them down is higher and there is more opportunity to make mistakes which causes query reformulation [35]. The overall rating of suitability of SchenQL for constructing the queries resulted in an average of 6.43 ($\sigma = .6$) while the rating was significantly lower (3.14, $\sigma = 1.2$) for SQL on a scale from 1 (very bad) to 7 (very good). While SQL was rated below mediocre, SchenQL was evaluated as excellent which shows users' satisfaction with it. These results lead to the conclusion of SchenQL being highly suitable for solving the given tasks which represent everyday queries of users of digital libraries and a high user acceptance of SchenQL.

In summary, utilisation of SchenQL achieves higher correctness of queries, lower perceived difficulty and requires less time than using SQL, which together verifies hypothesis H_1.

Analysis of H_2 To assess validity of hypothesis H_2 of SchenQL being suitable for experts and non-experts, we conduct significance tests on all queries independent by system, all queries dependent on system and each separate query for the three aforementioned values (correctness, rated difficulty and required time). The 21 participants from before form the two user groups: nine participants are non-experts and twelve participants are familiar with bibliographic metadata.

Correctness In general, 75% of queries were correctly formulated by domain-experts whereas non-experts achieved only 63.89% in both QLs. (Non-)Experts were able to solve 65.58% (47.22%) of queries in SQL and 85.42% (80.56%) in SchenQL. Tian et al. [38] stated that for a domain-expert, it would be easier to write queries in a DSL than in SQL.We found no significant differences between the two groups for correctness (separated by system, by query and in general).

Rated Difficulty We found no significant differences between the two groups for rated difficulty (separated by system, by query and in general).

Time We found no significant differences between the two groups for required time (separated by system and in general). We found significant differences for times needed to complete Q_3 and Q_4 with SQL when applying t-tests for the two user groups. With Q_3 the twelve participants versed in the domain were much slower to complete the task (inexperienced: 6:04 min., $\sigma = 2{:}51$ min.; experienced: 11:02 min., $\sigma = 5{:}41$ min.) while with Q_4 users with experience in bibliographic metadata were faster (inexperienced: 13:26 min., $\sigma = 6.17$ min.; experienced: 8:28 min., $\sigma = 4{:}27$ min.). We observed that domain-experts tend to review the result of their query online and therefore need more time to answer Q_3 than non-experts. Another explanation could be that since they are experienced with the principle of citations they were more confused with the one needed table as it contains publications and their references instead of two tables for papers, one which holds its citations and one which holds its references. Domain-experts are faster in formulating Q_4 as the query might be familiar and they already took more time to understand the database layout of references and citations while solving Q_3 which had to be tackled beforehand.

Result No user group is consistently better than the other, the SchenQL CLI is suitable for domain-experts as well as non-experts, thus, H_2 is verified.

Open Questions and Discussion In the open questions, the short, easy and intuitive SchenQL queries were complimented by many participants. Users noted the comprehensible syntax was suitable for non-computer scientists as it resembles natural language. Some noted their initial confusion due to the syntax and their incomprehension of usage of literals or limitations. Others asked for auto-completion, syntax highlighting, a documentation and more functions such as most cited with variable return values. No participant wished for visualisations which could be caused by design fixation [23] or generally lower requirements for such a system compared to the experts from the qualitative study.

The average overall impression of SchenQL was rated by the subjects as 5.05 ($\sigma = .74$) on a scale from 1 (very bad) to 6 (very good), enforcing a non-neutral rating. Assessed difficulty and required times to formulate the four queries were

significantly lower when utilising SchenQL compared to SQL, the overall correctness of all queries was significantly higher for SchenQL as well. This verified hypothesis H_1 of the CLI leading to generally better results than SQL. Our hypothesis H_2 of the SchenQL system being suitable for domain-experts as well as casual users is also verified. No user group was found to be consistently significantly better than the other one.

This evaluation lead to the construction of the prototypical GUI with its syntax suggestion as well as auto-completion features. Additionally, although they were not mentioned by participants in this evaluation, some visualisations were included following suggestions from the qualitative evaluation.

Part II: SchenQL GUI vs. CLI and User Experience Questionnaire

This second part of the quantitative study focused on evaluating the GUI and, thus, the SchenQL system as a whole. We assessed how usage of the web interface compared to users' impressions and performance when utilising the SchenQL CLI. Beside a part where test users answered queries with the GUI, we conducted the User Experience Questionnaire [36] to measure user experience with the SchenQL system. To resemble our target audience we did not pose the precondition of users being familiar with SQL or formulation of structured queries. Here, we intend to assess the hypothesis H_3.

Setting This evaluation is performed analogous to the previous part: every user performed the evaluation alone but in presence of a passive investigator on a computer with two monitors. We measured times used to find answers by capturing screens. The same SchenQL cheat sheet as in the first part was provided to the test subjects. At first, a video tutorial [5] introduced the usage of the SchenQL GUI. The next part was the formulation or the navigation towards solutions of the four queries introduced in Table 2 using the GUI. Afterwards, the subjects completed the User Experience Questionnaire [36] followed by questions regarding the overall impression of the GUI as well as possible improvements.

We evaluated ten participants from the area of computer science and adjacent fields which did not yet take part in a previous evaluation of the SchenQL system.

Partial Analysis of H_3: users unfamiliar with query formulation To assess partial validity of hypothesis H_3 in terms of the GUI's suitability for users unfamiliar with query formulation, we conduct significance tests on all queries together and each separate query for correctness, rated difficulty and required time. We observe the results from usage of the SchenQL CLI from the previous evaluation and participants' results from utilisation of the GUI as the two groups. Table 4 gives an overview of correctness, average rated difficulty and average required time for all four queries when using the SchenQL system.

Correctness Except for Q_3, participants mostly solved the queries correctly, resulting in an overall correctness of 77.5% (-10.83% compared to CLI, difference

not significant). We found no significant differences between the two groups for correctness in any of the four queries.

Rated Difficulty Users rated the difficulty of queries as 2.38 (+.35 compared to CLI, difference not significant) on average. We found no significant differences between the two groups for rated difficulty in any of the four queries.

Time Users took about 2:15 minutes for retrieval of the solution (-0:37 minutes compared to CLI, difference is significant) on average. We found significant differences in times required to solve queries Q_1 and Q_2. Times required for formulating the queries with the GUI were significantly lower than those resulting from using the CLI. As these queries were relatively simple, we assume the auto-completion and suggestion-feature of the GUI is especially helpful in the fast construction of straightforward queries or the GUI offering other suitable ways of quickly obtaining simple bibliographic information. Usage of the GUI might be more intuitive compared to writing simple queries in the SchenQL CLI.

General Results We want to point out that participants from the first part of the quantitative study who were familiar with query formulation, but were not offered help in the construction, did not significantly differ in rating of difficulty and correctness from users of this user study. In case of the GUI, subjects were supported in the formulation of queries but were not necessarily familiar with this kind of task. Hence, we assume the system's suggestion and auto-completion feature is useful for redemption of unequal prior knowledge in this case.

Correctness and rating of difficulty did not differ significantly between usage of CLI and GUI, but users were significantly faster in finding answers for simple queries with the GUI which underlines the suitability of the interface for everyday usage. Participants from this study resemble SchenQL's target audience, which additionally emphasizes its usefulness and partly verifies hypothesis H_3 in terms of the GUI being suitable for users not versed with structured query formulation.

Partial Analysis of H_3: UEQ Attractiveness, perspicuity, efficiency, dependability, simulation and novelty of interactive products can be measured with the UEQ [36] even at small sample sizes. Here, we want to conclude the assessment of validity of hypothesis H_3 in terms of rating of user experience.

Participants of this study answered the 26 questions of the UEQ regarding usage of the SchenQL system. Ratings on pairs of contrasting stances (-3 to 3) such as *complicated-easy* or *boring-exciting* were then grouped to the six dimensions mentioned before. Values above .8 are generally considered as positively evaluated equalling high user experience, values above 2 are rarely encountered [7].

In general, users seem to enjoy using the SchenQL system (attractiveness = 2.07, $\sigma = .25$). The handling of our system is extremely easy learned (perspicuity = 2.3, $\sigma = .19$). Tasks can be solved without unnecessary effort (efficiency = 2.03, $\sigma = .49$) and users feel in control of the system (dependability = 1.83, $\sigma = .63$). They seem exited to use the SchenQL system (stimulation = 1.73, $\sigma = .33$) and rate the system as innovative and interesting (novelty = 1.58, $\sigma = .68$).

As all six quality dimensions achieved ratings well over .8, the system is positively evaluated which equals high user experience and partially verifies H_3.

Open Questions and Discussion In the open questions, participants praised the intuitive usability, the auto-completion and the suggestion feature. For future development, suggestions for literals were requested and two participants wished for a voice input. Remarkably, not a single user mentioned the need for more or other visualisations, this is possibly attributed to design fixation [23] but might also stem from the advanced needs of power users from the expert interviews.

Users were significantly faster in solving simple queries when using the GUI compared to the CLI. As we found no significant impairments from usage of the GUI, we assume its usefulness and usability for query formulation. Participants from this study were less familiar with construction of structured queries compared to those of the previous study but seemed to be adequately supported by the GUI in retrieval of information. Together with the UEQ which showed users' high ratings ($> .8$) for all six quality dimensions (which proves high user experience [7]), hypothesis H_3 could be partially verified.

5 Conclusion and Future Work

We evaluated SchenQL, a domain-specific query language operating on bibliographic metadata from the area of computer science with accompanying GUI supporting query formulation. Our thorough evaluation against SQL showed the need for such a DSL. Test subjects' satisfaction with the SchenQL system was assessed with application of the UEQ. The introduction of a GUI and its evaluation with users resembling our target audience did not significantly change the correctness of answers or the users' rating of difficulty of the queries compared to the CLI but instead the time needed to formulate simple queries was reduced significantly. Missing prior knowledge with structured query formulation seems to be compensated by using a GUI with a suggestions and auto-completion feature. As the CLI and the GUI proved to be viable tools for information retrieval on bibliographic metadata, users' preferences should decide which one to use.

Using SchenQL lead to generally better results compared to utilisation of SQL (H_1). The system is suitable for domain-experts and non-experts (H_2). Our GUI has high usability for users not familiar with structured query formulation (H_3).

Enhancements of functionalities could include more visualisations such as color-coded topics or graph visualisation as the experts from the qualitative study requested. Furthermore, more specific query options such as a filter for papers with few co-authors or most cited with variable return values could be included. As visualisations were not relevant for users in our quantitative evaluation, future efforts could focus on supporting more advanced query options: algorithms for social network analysis as PageRank, computation of mutual neighbours, hubs and authorities or connected components [37] would fit. Centrality of authors, the length of a shortest path between two authors and the introduction of aliases for finding co-citations [19] would also be useful query building blocks. Incorporation of social relevance in the search and result representation process as shown in [10] could also be an extension. User profiles could store papers and keywords, which in terms influence results of search and exploration.

References

1. Google Scholar, https://scholar.google.com/
2. GrapAL, https://grapal.allenai.org/
3. ResearchGate, https://www.researchgate.net
4. SchenQL Evaluation CLI vs. SQL - Tutorial, https://youtu.be/g7J64wzbE5I
5. SchenQL Evaluation GUI - Tutorial, https://youtu.be/56-23zyUDPQ
6. Semantic Scholar, https://www.semanticscholar.org
7. User Experience Questionnaire Handbook, https://www.ueq-online.org/Material/Handbook.pdf
8. Wikdata, https://www.wikidata.org/wiki/Wikidata:Main_Page
9. Amaral, V., Helmer, S., Moerkotte, G.: A visual query language for HEP analysis. In: IEEE NSS 2003. vol. 2, pp. 829–833. IEEE Computer Society (2003)
10. Amer-Yahia, S., Lakshmanan, L.V.S., Yu, C.: SocialScope: Enabling Information Discovery on Social Content Sites. In: CIDR 2009. www.cidrdb.org (2009)
11. Baeza-Yates, R., Ribeiro-Neto, B.A.: Modern Information Retrieval - the concepts and technology behind search, Second edition. Pearson Education Ltd., Harlow, England (2011)
12. Bates, M.: Task Force Recommendation 2.3 Research and Design Review: Improving User Access to Library Catalog and Portal Information: Final Report (version 3) (2003)
13. Beall, J.: The weaknesses of full-text searching. The Journal of Academic Librarianship **34**(5), 438 – 444 (2008)
14. Berget, G., Sandnes, F.E.: Why textual search interfaces fail: a study of cognitive skills needed to construct successful queries. Inf. Res. **24**(1) (2019)
15. Betts, C., Power, J., Ammar, W.: GrapAL: Connecting the Dots in Scientific Literature. In: ACL 2019. pp. 147–152. ACL (2019)
16. Bloehdorn, S., Cimiano, P., Duke, A., Haase, P., Heizmann, J., Thurlow, I., Völker, J.: Ontology-Based Question Answering for Digital Libraries. In: ECDL 2007. vol. 4675, pp. 14–25. Springer (2007)
17. Borodin, A., Kiselev, Y., Mirvoda, S., Porshnev, S.: On Design of Domain-Specific Query Language for the Metallurgical Industry. In: BDAS 2015. vol. 521, pp. 505–515. Springer (2015)
18. Collberg, C.S.: A Fuzzy Visual Query Language for a Domain-Specific Web Search Engine. In: Diagrams 2002. vol. 2317, pp. 176–190. Springer (2002)
19. Dries, A., Nijssen, S., Raedt, L.D.: BiQL: A Query Language for Analyzing Information Networks. In: Bisociative Knowledge Discovery 2012, vol. 7250, pp. 147–165. Springer (2012)
20. Francis, N., Green, A., Guagliardo, P., Libkin, L., Lindaaker, T., Marsault, V., Plantikow, S., Rydberg, M., Selmer, P., Taylor, A.: Cypher: An Evolving Query Language for Property Graphs. In: SIGMOD 2018. pp. 1433–1445. ACM (2018)
21. Guidi, F., Schena, I.: A Query Language for a Metadata Framework about Mathematical Resources. In: MKM 2003. vol. 2594, pp. 105–118. Springer (2003)
22. Hotho, A., Jäschke, R., Benz, D., Grahl, M., Krause, B., Schmitz, C., Stumme, G.: Social Bookmarking am Beispiel BibSonomy. In: Social Semantic Web 2009, pp. 363–391. Springer (2009)
23. Jansson, D.G., Smith, S.M.: Design fixation. Design Studies **12**(1), 3 – 11 (1991)
24. Kelly, D.: Methods for Evaluating Interactive Information Retrieval Systems with Users. Found. Trends Inf. Ret. **3**(1-2), 1–224 (2009)

25. Khazaei, T., Hoeber, O.: Supporting academic search tasks through citation visualization and exploration. Int. J. on Digital Libraries **18**(1), 59–72 (2017)
26. Klink, S., Ley, M., Rabbidge, E., Reuther, P., Walter, B., Weber, A.: Browsing and visualizing digital bibliographic data. In: VisSym 2004. pp. 237–242. Eurographics Association (2004)
27. Kreutz, C.K., Wolz, M., Schenkel, R.: SchenQL - A Domain-Specific Query Language on Bibliographic Metadata. CoRR **abs/1906.06132** (2019)
28. Kreutz, C.K., Wolz, M., Schenkel, R.: SchenQL: A Concept of a Domain-Specific Query Language on Bibliographic Metadata. In: ICADL 2019. vol. 11853, pp. 239–246. Springer (2019)
29. Leser, U.: A query language for biological networks. In: ECCB/JBI 2005. p. 39 (2005)
30. Ley, M.: DBLP - Some Lessons Learned. PVLDB **2**(2), 1493–1500 (2009)
31. Madaan, A.: Domain Specific Multi-stage Query Language for Medical Document Repositories. PVLDB **6**(12), 1410–1415 (2013)
32. Martín, M.S., Gutiérrez, C., Wood, P.T.: SNQL: A Social Networks Query and Transformation Language. In: AMW 2011. vol. 749. CEUR-WS.org (2011)
33. Pirolli, P.: Powers of 10: Modeling Complex Information-Seeking Systems at Multiple Scales. IEEE Computer **42**(3), 33–40 (2009)
34. Reitz, F.: A Framework for an Ego-centered and Time-aware Visualization of Relations in Arbitrary Data Repositories. CoRR **abs/1009.5183** (2010)
35. Schaefer, A., Jordan, M., Klas, C., Fuhr, N.: Active Support for Query Formulation in Virtual Digital Libraries: A Case Study with DAFFODIL. In: ECDL 2005 (2005)
36. Schrepp, M., Hinderks, A., Thomaschewski, J.: Applying the User Experience Questionnaire (UEQ) in Different Evaluation Scenarios. In: HCI 2014. vol. 8517, pp. 383–392. Springer (2014)
37. Seo, J., Guo, S., Lam, M.S.: SociaLite: An Efficient Graph Query Language Based on Datalog. IEEE Trans. Knowl. Data Eng. **27**(7), 1824–1837 (2015)
38. Tian, H., Sunderraman, R., Calin-Jageman, R.J., Yang, H., Zhu, Y., Katz, P.S.: NeuroQL: A Domain-Specific Query Language for Neuroscience Data. In: EDBT Workshops 2006. vol. 4254, pp. 613–624. Springer (2006)
39. Xu, B., Cai, R., Zhang, Z., Yang, X., Hao, Z., Li, Z., Liang, Z.: NADAQ: Natural Language Database Querying Based on Deep Learning. IEEE Access **7**, 35012–35017 (2019)