RESEARCH-ARTICLE

Exploring Exploratory Querying

MARCELO ARENAS, Pontifical Catholic University of Chile, Santiago, RM, Chile
ENRICO FRANCONI, Free University of Bozen-Bolzano, Bolzano, BZ, Italy
JANIK HAMMERER, University of Bayreuth, Bayreuth, Bayern, Germany
OLAF HARTIG, Linköping University, Linkoping, Östergötland, Sweden
KATJA HOSE, Vienna University of Technology, Vienna, Vienna, Austria
LAURA KOESTEN, University of Vienna, Vienna, Vienna, Austria
View all

Published: 01 September 2025

Exploring Exploratory Querying

Marcelo Arenas
Universidad Católica de Chile
marenas@ing.puc.cl

Enrico Franconi
Free University of Bozen-Bolzano
franconi@inf.unibz.it

Janik Hammerer
University of Bayreuth
Janik.Hammerer@uni-bayreuth.de

Olaf Hartig
Linköping University
olaf.hartig@liu.se

Katja Hose
TU Wien
katja.hose@tuwien.ac.at

Laura Koesten
University of Vienna
laura.koesten@univie.ac.at

George Konstantinidis
University of Southampton
g.konstantinidis@soton.ac.uk

Leonid Libkin
RelationalAI and Univ. Edinburgh
l@libk.in

Wim Martens
University of Bayreuth
Wim.Martens@uni-bayreuth.de

Yuya Sasaki
University of Osaka
sasaki@ist.osaka-u.ac.jp

Stefanie Scherzinger
University of Passau
stefanie.scherzinger@uni-passau.de

Katherine Thornton
Yale University Library
katherine.thornton@yale.edu

Hsiang-Yun Wu
St. Pölten UAS
hsiang-yun.wu@fhstp.ac.at

ABSTRACT

We need to rethink how users understand and develop queries. The growing diversity of users, the increasing complexity of query languages and data architectures - now aided by tools like LLMs - are challenging the traditional view of a highly-trained user writing queries in a controlled environment. Query formulation has become a more exploratory endeavor that needs to be researched and supported: an iterative cycle of designing, debugging, and maintaining queries. To ground this vision, we present an empirical analysis of query logs from the Wikidata Query Service, revealing common patterns of iterative query modification. Based on these findings, we propose a concrete research program with hypotheses, user studies, and research questions for query languages, engines, and interfaces. Our contributions include a curated query session dataset, a classification of exploratory-query patterns, and a roadmap for building system-level support for exploratory querying.

PVLDB Reference Format:
Marcelo Arenas, Enrico Franconi, Janik Hammerer, Olaf Hartig, Katja Hose, Laura Koesten, George Konstantinidis, Leonid Libkin, Wim Martens, Yuya Sasaki, Stefanie Scherzinger, Katherine Thornton, and Hsiang-Yun Wu The Authors. Exploring Exploratory Querying. PVLDB, 18(13): 5731 - 5739, 2025.
doi:10.14778/3773731.3773746
PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/hartig/ExploratoryQueryingSessions/.

Proceedings of the VLDB Endowment, Vol. 18, No. 13 ISSN 2150-8097.
doi:10.14778/3773731.3773746

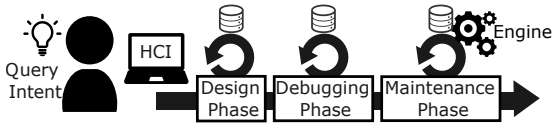

Figure 1: (Traditional) query lifecycle

1 INTRODUCTION

Today's data landscape contains an abundance of different data models and query languages, which are becoming more and more complex. The SQL standard alone consists of eleven parts, of which two (Framework and Foundations) are commonly taught at universities, while its total page number has grown to about four thousand. The premise of lightweight declarative query languages has been challenged by their increasing expressiveness — making query languages Turing-complete — and by the ongoing addition of complex features. At the same time, data architectures have evolved from closed, controlled systems to open, cloud-based architectures where development and testing are closer to production. On the one hand, this implies moving from inexpensive and complete access to data to higher testing and development costs in answering queries. On the other hand, users are no longer provided with a local, dedicated and protected testing and development environment.

The proliferation of different data models creates use cases where the schema is complex, unknown, or non-existent (e.g., graph databases or RDF). The advent of Large Language Models (LLMs) allows users to co-develop queries in a constant "vibe-dialogue" with an AI assistant. Additionally, users of databases have become much more diverse. Their cultural, socio-economic, professional, and technical backgrounds can directly affect the process of query construction, for example, in interpreting intermediate query results [46]. Public query interfaces, e.g., Wikidata [91], have an increasing number of users who aim to find information while having little or no formal training, while the labor market faces a deficit

in highly-skilled data and software engineers. Many users choose data models and query languages based on their preference and background rather than their suitability for the task at hand.

Due to the above, query development is now a more *exploratory process* where users actively engage in an iterative cycle of designing [16, 100], debugging, and maintaining queries in what we recognize as the *query lifecycle* phases (see Fig. 1 and Section 2). This paper is a call to action for developing (language, engine, and interface) support for exploratory querying throughout this lifecycle. To gather concrete evidence, we analyzed more than 24,000 user query sessions from the Wikidata Query Service (Section 3). We found numerous sessions exhibiting iterative query refinement, extension, and intent shifts—practical examples of exploratory behaviors. We develop a curated dataset of query sessions and a categorization of exploratory query patterns, providing a foundation for our research agenda in Section 4, where we present specific hypotheses, user studies, research questions, and evaluation metrics to support exploratory querying in languages, engines, and interfaces.

2 QUERY LIFECYCLE

Central to our investigation is the notion of a *query intent*, as the information seeking motive that users have when interacting with data. If there is no particular query intent in mind, this interaction falls under the remit of *data exploration* [28, 42, 54, 59]. On the other hand, the clearer the query intent becomes, the more we move into *exploratory querying*, where one tries to find the right query to express that intent. Figure 2 depicts this spectrum and the fact that there is no clear-cut separation between the two areas.

Execution of queries in database systems is well studied, and its various steps are extensively analyzed. Nevertheless, looking at queries as artifacts, there is a notion of query *lifecycle*, similar to the lifecycle of software products in software engineering. In a traditional setting, when a software developer creates a query for an application or when a business analyst creates a query for an analysis, the lifecycle of the query starts with the query intent. During the first phase of the lifecycle, the **design phase**, the query intent is formalized as an initial query in an available language.

As this initial query may not yet capture the query intent completely, the query designer may then engage in a form of debugging in which the query is tested and adapted. We refer to this phase as the **debugging phase**. The goal of this phase is to create a query that captures the query intent correctly and completely.

For the design and debugging phases, we assume that the environment in which the querying takes place (query processing system, user interface, schema, data, query language, etc.) does not change, or at least not to an extent that has an effect on the outcome of these phases. Eventually, however, parts of the environment may change such that a query that captured a query intent earlier does not capture that intent any longer or may even become invalid. Therefore, the end of the debugging phase marks the beginning of a **maintenance phase** in the life cycle of a query. During the maintenance phase, we may regularly check whether the query still captures the query intent and is valid. If not, we may decide to fix the query, and a new debugging or even design phase begins.

Traditionally, life cycle phases take place in the face of database queries being typically static and predefined. Developers often work

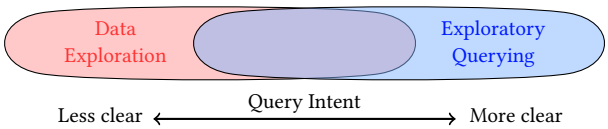

Figure 2: Exploratory querying versus data exploration

with a known database schema and a clear query intent that comes as a functional requirement of their implementation. The queries they design will later be embedded in the application code and executed at runtime. If software errors are detected, the query may need to be debugged, and if the application code or the database schema evolve over time, queries may require maintenance. This traditional view is rooted in all our query management practices.

Such static assumptions have not been made only in database management; in the domain of information seeking early research has assumed the information need to remain static [38, 89]. Yet, more recent information seeking models emphasize a more dynamic nature of this process, recognizing that information needs are often refined or redefined based on the information retrieved during the process [7]. For database querying, even if existing processes contain elements of exploration (of alternative query formulations), these processes have commonly considered a rather constrained search space, and both the schema and the query intent were traditionally fairly clear. In contrast, schemas today are not always as clear, assumptions radically change, and all phases of the query life cycle become more complex and exploratory in nature. For example, users may now use LLMs to transform natural language questions into database queries, and start from there.

During the design and debugging phases, a developer may create and run trial candidate queries, effectively exploring the space of all possible queries (in the given query language) that capture the query intent. A user may issue queries to better understand specific schema or data elements, thus performing schema or data exploration. A user not sufficiently experienced with the query language may write a number of queries to better understand relevant language features before or while attempting to write the actual query. In the maintenance phase, queries might be rewritten to explore changes in the logical environment, e.g., changes in data, schema, constraints, access control methods, or even cloud-hosting pricing policies, among others, or the physical environment, e.g., changes in the operating system, storage or execution environments.

3 EXPLORATORY QUERYING CHALLENGES

We performed an empirical study on Wikidata and identified numerous examples that illustrate exploratory querying.

An Empirical Study on Wikidata. We considered the *organic Wikidata query logs* (June 2017–March 2018) from [64], i.e., their heuristic classified them as "having human involvement". From these, we extracted streaks of similar queries using the method of [16] with added time interval constraints (at least 3 seconds and at most 5 minutes apart). From the original log file, which contains 3.57M queries, this resulted in 24,050 streaks of at least three queries each. We distributed the analysis effort among the authors of this paper. Each author manually inspected different sections of the collection to identify streaks that contain versions of a query that changes throughout the streak in exploratory directions. We manually ran the queries on the Wikidata Query Service to better

understand their semantics. We used a shared template to describe our streak analysis, leading to a collection of a total of 30 sessions with user stories that exhibit exploratory querying behavior, which is only a *tiny sample* due to the manual labor. This collection and related artifacts are available for further research [85].

Based on our collection of sessions, we observe that the sessions—and thus, exploratory querying behavior in general—can be grouped into at least six categories. These categories are neither exhaustive nor mutually exclusive; some sessions fit into multiple categories (to provide readers with a sense of the overall distribution, Table 1 lists how many sessions we found per category).

Category:	1	2	3	4	5	6
Number of Sessions:	15	12	4	7	5	7

Table 1: Number of sessions in our collection per category, where some of the sessions fit into multiple categories.

In ***Result Refinement (Cat. 1)*** sessions, the user modifies their query to limit the cardinalities of answers (rows) in the query result more and more. Typical query modifications in such sessions include the addition of further conditions, captured via FILTER expressions or via filtering triple patterns. In ***Result Generalization (Cat. 2)***, the user gradually generalizes their queries to retrieve additional entities in the result set, which is essentially the opposite of the previous category. Typical changes to the query within a result-generalization session are the inclusion of superclasses in the query pattern and an increase of the value of the LIMIT clause. In ***Result Extension (Cat. 3)***, the user expands their query to extend the query result with additional attributes of the selected entities. To this end, the user typically adds more triple patterns with predicates that correspond to the additional attributes and variables in the object position. In ***Query Refinement (Cat. 4)***, the user seems to have a specific query intent in mind from the beginning and starts with an initial query that goes in the direction of this intent. Then, throughout the session, the user works on this query to align it more accurately with their intent. In contrast to result refinement sessions (Cat. 1), query refinement sessions are not about simply restricting the result set (not only, at least), and also not about simply adding further properties to be retrieved for selected entities (as done in the result extension sessions, Cat. 3), but involve adding and tinkering with more complicated query features. In ***Bug Fixing (Cat. 5)***, the user starts with an incorrect query (e.g., containing incorrectly written URIs or incorrectly used features of the query language) and fixes the issues of this query throughout the session. Finally, in ***Shifting Query Intent (Cat. 6)***, the query intent shifts to different but related topics throughout a session. Typical changes to queries include adapting values used for filtering, changing the predicate or the class URIs mentioned in the query pattern, and also adding or dropping entire parts.

The Search for the Right Query. In each of our identified sessions, there is clear evidence of exploratory struggles. Evidently, users struggle with language familiarity. In session [83], the user seems to start from an overly constrained example query that returns nothing; they make changes until first results are found. Even if users are aided by simple tools (e.g., graphical tools that allow them to examine the schema as they are writing their queries) or more advanced ones such as LLMs or text-to-SQL tools, in the end they have a query which is just "draft zero" which needs to be debugged and reformulated until it corresponds to the query intent.

Query reformulation essentially tries to solve a **language and schema** problem of the user. In database and related literature, such as knowledge representation, there are principled approaches to study query reformulation. These approaches introduce a formal framework and tool to assist users in formulating precise queries that accurately reflect their information needs, even if users are completely unfamiliar with the underlying database schema. This exploratory querying process could be guided by automated reasoning tasks over a conceptual schema (an ontology) that describes the data domain of the queried database [30]. These tasks leverage formally defined basic operations available to the user, enabling them to modify the query, receive contextual feedback, and focus on relevant information during exploration. While logical AI approaches offer advantages, they restrict the expressiveness of query languages because they need to reason about queries, and they do not cover many features of traditional query languages. In Section 4 we expand our research proposal to study this problem.

In some of the exploratory categories identified, such as the Shifting Query Intent category, users are particularly constrained by the **query engine assumptions**. Consider for example session [84] where the user repeatedly needs to change the query until the engine stops timing out. Current query engines operate under assumptions of completeness, efficiency, and optimization, without accommodating iterative exploration or integration with the development environment. They execute queries in an agnostic manner, separate from the broader code base, and assume uniform treatment of data sources in federated queries, which may oversimplify complex or exploratory querying needs. Retrieval-augmented generation (RAG) is only beginning to be incorporated, hinting at a move towards dynamic query responses, but remains in early stages [41]. Approximate query-answering techniques [63] can provide faster, representative responses by relaxing precision; however, these approaches are commercially limited, with configurations that are often inaccessible to non-experts. Query auto-completeness [90, 97] is also underdeveloped, and provenance tracking [23, 40], which would help users understand data origins, is not yet standard, and applicable to restricted classes of queries only. Additionally, engines lack adaptability to the users' context, backgrounds, or skill levels, limiting accessibility for non-experts. We discuss in Section 4 that a query environment *in exploratory mode* could do its job by only showing a part of the results to the user. Knowing the user's skill level or background, the engine could customize which sample it will show. Materialized views, caching, and reuse can also be deployed [22, 37, 53] but these are mainly suited for independent queries rather than iterative exploratory processes. Section 4 suggests future developments that would be more effective for exploratory querying such as configurable approximations, contextualized querying, and richer provenance support, enabling a more adaptable and user-friendly experience.

At the same time, the logs in our empirical study do not have information about the **interfaces** that generated the queries. In practice, query interactions are performed via command-line interfaces (CLI), graphical user interfaces (GUIs), natural language interfaces (NLIs) [39], voice user interfaces (VUIs), haptic interfaces, multi-modal interfaces, and so forth [26]. In addition to interfaces

requiring various hardware support, CLI, GUIs, and NLIs (i.e., ChatGPT) are expected to more often be used to support exploratory querying. Such applications include recommendation systems and visualizations [20, 31], which provide users with interactivity, as well as visual explanations. While CLIs are light and handy, they are not intuitive and require domain knowledge. A GUI facilitates human-computer communication through graphical components, e.g., menus and buttons, on desktop, Web, and mobile applications. Combining such modalities and defining the best interface features for exploratory querying requires sophisticated thinking. The next section defines a research agenda to address these challenges.

4 AN EXPLORATORY RESEARCH PROGRAM

To address the challenges of Section 3, we draw a number of research hypotheses and suggest a number of studies and metrics to form a sociotechnical research agenda around exploratory querying. We structure our agenda around the three main dimensions of database technology: language/schema, query engine, and human-computer interfaces, and consider how each of these dimensions has to support exploratory querying throughout the query lifecycle.

Research Hypotheses. We believe that there are systematic processes that people explore in using query languages and writing queries. Consequently, query languages need to provide functionalities that allow users to adjust the query writing process to reflect their natural habits, better express query intent, as well as enable easier designing, debugging, and maintenance.

Current query engine technology is not tailored to the iterative and investigative nature of exploratory querying. In this mode, users care more about quickly produced overviews of query answers [29, 60], rather than complete, precise answers. Exploratory querying involves a sequence of dependent queries, each one evolving from the previous, instead of a set of independent queries. Users might want to navigate their exploration based on the cost of the actual query execution in the cloud, or even based on knowledge or data that resides in other systems, LLMs, or the Web. While earlier studies have focused on understanding *how* users conduct exploratory querying, e.g. [87], less focus has been placed on their willingness to compromise for better system support.

Regarding HCI support for exploratory querying, we believe that effective use and combination of interaction modalities (such as voice, keyword-based, or natural language text inputs), both within and across different types of interfaces, can enhance query formulation efficiency, increase user satisfaction, and improve success rates in exploratory querying tasks. For example, offering different visual representations of complex datasets allows for a larger variety of entry points for users to develop an understanding of how data can be queried. Progressively showing or animating the results of queries in real-time can improve user comprehension and engagement. Modalities across GUIs and NLIs enable query suggestions or personalized guidance, possibly with better exploration outcomes. Interfaces can also be designed for collaboration, enabling iterative query design, debugging, and maintenance in teams.

Suggested Studies. To confirm our hypotheses, we propose a user study program including user surveys (interviews, questionnaires, etc.) and user studies (observing users, clicks, mouse-/key-/eye-tracking, etc.) that span query languages, engines, and interfaces, to gain an understanding of how people use exploratory querying: what languages and language features they use and how, what difficulties they face with traditional RDBMSes, what issues they identify on the underlying engines, and how they work with query systems across different interfaces and their combinations.

Most existing studies concentrate on one specific demographic group (CS students), and do so in one specific scenario: learning a language such as SQL [17]. While useful, this is just a segment of the population who writes database queries, and has a small intersection with the segment that writes real-life production queries. So, we should expand the surveyed sample to observe a more diverse user base who routinely write database queries for all different phases of the query lifecycle. These experiments require carefully designed user studies that consider varying levels of expertise from novice to advanced. Existing studies on database system support focus mainly on a *single* query language and dataset. At the same time, related research has examined user aspects of query formulation and refinement [52, 95], but often this has been in the context of web-search queries [45, 58]. Our empirical study (Section 3) suggests that many users tend to build queries in a sequential, pipeline-like style rather than crafting one monolithic SQL statement.

To explore this further, experiments could present equivalent queries written in these different styles (e.g., in a language like PRQL [77] that mimics data science libraries, or pipelined SQL syntax [86] as well as their translations to declarative SQL queries) and measure which is easier for users to compose and debug. To measure the effect of modularity, we can compare languages in which queries are easier to build bottom-up, such as Soufflé and Rel [44, 78]. Such studies will reveal whether simpler, more modular query paradigms improve learning and effectiveness and identify language features that would provide tailored support for exploratory querying.

In parallel, we should quantify the engine performance trade-offs (e.g., efficiency versus correctness or completeness) that users tolerate. Controlled tasks requiring iterative query refinement can measure how long people will wait for answers and whether they accept faster, approximate results instead of exact ones. These studies can pinpoint each user group's "latency tolerance" and boundaries for approximation or partial answers. In effect, we could derive concrete "pain thresholds" (maximum acceptable delay, minimum result completeness, etc.) that future systems should meet.

Finally, interface-focused studies should investigate how users interact with different tools and modalities. It remains largely unexplored how people combine keywords, graphical builders, and natural-language inputs to formulate queries. We should compare task performance and satisfaction across interface styles and query complexities and study how exploration patterns differ between experts, domain specialists, and lay users. For example, do advanced users leverage provenance tracking or query suggestions more effectively? With these lines of inquiry, our suggested studies will build an empirical foundation describing how real users formulate and refine queries across languages, engines, and interfaces.

Suggested Research Building on these studies, we can pursue database research in all dimensions. In query languages, one goal is to identify a "right-sized" sublanguage that captures common exploratory needs without the full complexity of a commercial language such as SQL, GQL or SPARQL. We should investigate modular, composable languages where blocks of a query can be

built and debugged *independently* (unlike SQL where correlated subqueries cannot be tested independently). In particular, blending declarative queries with simpler sequential (pipeline) operations (e.g., PRQL or Python notebooks) may reduce cognitive load. Research results on translating high-level user intent into a tailored query dialect could streamline design and reduce errors. Through this research path, we might even introduce new language features for exploration (for example, fuzzy quantifiers like "find five representative answers") or blur the line between data and schema by letting queries incorporate schema discovery as part of normal querying (as in, e.g., Data Hilog [81, 96]).

For query engines, the focus should be on meeting interactive exploration needs (such as work done in *data* exploration [29]). Already today, query optimizers strive to be highly responsive by returning the first tuple quickly [18]. We can extend existing work by engineering engines that obey user-set bounds on response time and accuracy. For instance, an optimizer might drop or replace expensive joins (e.g., using acyclic joins [34]) to speed up exploratory queries, or provide partial/streaming results by approximating aggregates or cardinalities [6, 56]. Techniques such as session-aware caching, pre-computation and dynamic allocation of resources can be developed so that frequently accessed data or intermediate results are ready when the user pivots, and lag is minimised. Another promising direction is federated execution of user-configurable query plans that would allow users to specify trade-offs (speed vs accuracy vs cost). This can be combined with RAG methods where LLMs augment results from multiple sources (e.g. across polystore [27] or even "polycloud" architectures). Using such innovations, latency and completeness can ultimately become tunable parameters in query environments.

Query interface research should create more supportive, transparent tools. Future interfaces might offer real-time feedback as queries are built, e.g., highlighting errors or suggesting relevant predicates and filters on-the-fly. A unified visual framework could present the user's exploration history: visualized query paths, past results, and strategy annotations that can be reviewed and refined. Embedding best-practice examples or "how-to" hints in the interface may guide users along proven exploration patterns. Integrated LLMs could process and understand user intent across modalities (text, voice, or GUI) and enrich query results with contextual knowledge, thus aiding in sensemaking [51, 76].

Success in the research agenda presented here and defining the metrics that quantify how effectively a system supports exploratory querying is a research challenge in itself and will depend on the gaps that the studies outlined in Section 4 will reveal. Potential evaluation metrics are *session length* (total time of an exploration session), *time-to-insight* (duration until the user finds useful information), *query refinement count* (number of queries or adjustments the user makes), *result diversity* (breadth or variety of results returned). Applying them in benchmarks and user studies (alongside traditional accuracy and throughput metrics), we can empirically validate improvements in query languages, engines, and interfaces.

5 THE EXPLORATION LANDSCAPE

Data Exploration. The focus of data exploration is to form an idea of a dataset or to extract knowledge from it, even though we do not exactly know the query intent yet. There is a large body of research on data exploration, including recent work on interactive data exploration [28, 54, 59], data exploration with privacy guarantees [70], efficient evaluation of data exploration queries [12, 13, 25], or ML for data exploration [1, 47]. An overview of data exploration techniques is given by Idreos et al. [42].

Although data exploration and query exploration are different tasks, there is common ground. Both involve aspects of *schema exploration*, which comes in two forms. First, the schema already exists, and the goal is to explore data and schema together [98]. Second, schema information is absent. The task then includes inferring a schema from given data (e.g. for XML [8, 9], JSON [5, 48, 49, 88, 99], or graph data [14, 15, 35]), which typically involves classical learning algorithms [10]. Here, the inferred schema can provide an adequate summarization of the data and also guide the user in formulating valid queries. Another related aspect is data summarization [24]. Although a schema can be seen as a summary of a data set, data summarization techniques focus more on the data values. For example, an average column value can be seen as a summary of the column values. Data summarization aims at constructing and maintaining certain characteristics of the data using a drastically smaller structure, which can approximate but provides accuracy guarantees. So, data summaries can also be used in queries. Data exploration frequently involves searching for queries, where users often write their own queries to explore a database. However, they often face challenges in determining the right queries to ask [65].

Model Exploration. A crucial dimension of exploration, closely related to query exploration, is model exploration, which refers to the process of investigating and analyzing an ML model to understand its behavior, performance, and potential improvements. Model exploration arises from the increasing use of ML models, often perceived as "black boxes" [82]. Often, these models provide highly accurate predictions but offer limited transparency about their internal decision-making processes. This opacity is particularly problematic in high-stakes domains such as healthcare, finance, and legal systems, where accountability is fundamental. Lack of insight into why an ML model produces a particular output led to eXplainable Artificial Intelligence (XAI) [36, 67, 75], an area that focuses on producing human-understandable explanations for the output of the ML model. Model exploration is a critical activity of XAI methods and enables users to investigate, understand, and refine their interactions with ML models.

XAI methods have traditionally focused on producing localized explanations for specific predictions, such as feature importance scores [62, 67, 79, 80]. Although helpful, these explanations are not a silver bullet [68, 69], as they may be narrow in scope and do not reveal the broader logic behind an ML model. Consequently, there is growing interest in systematic approaches that enable a more comprehensive exploration of ML models, e.g., specialized query languages for explainability tasks [3, 4], which aim to provide users with the ability to pose queries that facilitate model exploration. Drawing inspiration from traditional database query languages, these explainability query languages are designed to offer structured and interactive means for users to probe and analyze ML models [2]. A key characteristic of these languages is their declarative nature, which allows users to specify the desired explanation without detailing the underlying computational process.

This approach not only has the potential to simplify the exploration process, but also to democratize access, allowing users with diverse expertise to engage with model exploration more effectively.

Exploratory Search. Exploratory search has also been examined from a user perspective in information seeking and HCI, focusing on how well intent is articulated prior to querying. The literature differentiates between simple look-up search and more complex exploratory tasks [30, 66, 94]. Exploratory queries involve iterative search processes in which users refine their queries based on intermediate results to discover new insights or patterns [38]. Unlike fact-based retrieval, exploratory queries focus on sensemaking, encouraging deeper engagement with the data and supporting tasks such as learning, investigation, and decision-making under uncertainty [11]. However, most studies on query categorizations stem from general search rather than being specific to data search [21], even though the unique characteristics of data as an information source from a user perspective have been highlighted [52]. Research on information seeking for data specifically has emphasized the inherently exploratory nature of most data search tasks, apart from direct lookup queries [50]. This is attributed to the nature of data search systems, where access to data is not direct, but is mediated through metadata or query mechanisms. These give insights into the underlying data structure (e.g., the database schema) and return partial information about a database or a corpus of datasets.

Sensemaking and Exploratory Information Seeking. Research on exploratory search and sensemaking in HCI and information science has long emphasized that users engage in iterative, interpretive cycles during information seeking. Naumer et al. [76] review methodological approaches across domains, highlighting key processes like collecting, organizing, and reframing information.

Similarly, classic models like Dervin's Sense-Making Methodology and Kuhlthau's Information Search Process model [55] describe general phases of search behavior from initiation and exploration to formulation and collection. These frameworks focus primarily on cognitive processes and static information needs, but not on the mechanism of iterative, exploratory query formulation, esp. in structured languages such as SQL or SPARQL.

Query Formulation in Search Interfaces and Interactive IR. Studies in interactive information retrieval have explored how users formulate and refine keyword-based queries. Wacholder [92] emphasizes the linguistic and cognitive challenges users face during interactive query formulation. Kato et al. [45] classify user reformulation actions (e.g., specialization, generalization, parallel movement) and suggest user interfaces with feedback mechanisms to steer these behaviors. Li et al. [58] show that users often synthesize multiple previous queries to meet evolving goals in exploratory search tasks. While valuable, these studies focus on unstructured queries and search tasks rather than formal query languages. GUIs aim to simplify user interaction by abstracting technical complexity; they enable users to perform advanced operations, such as generating complex queries, with minimal prior knowledge, due to built-in functionalities that require only a few clicks. One example is visual querying [19, 61], a paradigm that facilitates exploratory data analysis. Visualization tools, such as SQLVis [73], QueryVis [57], I-Rex [71], and Relational Diagrams [32], are designed to help formulate accurate SQL queries.

Human Factors in Structured Query Languages. Research on structured queries (e.g., SQL, SPARQL) often focuses on usability and learning. Early work (e.g., Welty [93]) highlights the role of human factors, while recent studies identify novice misunderstandings [72] and expert strategies during debugging [74]. These studies show that query formulation is challenging and iterative, yet they stop short of uncovering generalized patterns or reusable strategies. Tools such as visual builders and templates [33, 43] assist with writing but offer limited insight into how users conceptualize and construct queries in exploratory contexts.

Limitations of Existing Work and Our Contribution. Sensemaking and IR studies offer important general principles (e.g., iterative refinement, contextual reasoning), but do not detail the steps users follow when crafting structured queries. In contrast, SQL usability studies describe common difficulties but rarely address the structure of successful query development. We provide a missing link: an investigation of exploratory querying grounded in structured query logs (Section 3), identifying recurrent patterns (e.g., broadening, constraint tuning, schema probing). We propose to extend existing theories with domain-specific insights about *how users build structured queries iteratively* and what language, engine, and interface features support them. This bridges cognitive theory and database practice and sets the stage for adaptive exploratory systems.

6 CONCLUSIONS

The way we query data has evolved significantly, driven by increasing data complexity, the use of diverse data models and query languages, and the integration of generative AI to construct queries across models and languages. Traditional approaches, relying on assumptions of well-designed data and user familiarity with query languages no longer reflect the modern realities. Instead, querying has become an exploratory process that requires constant iteration within a query lifecycle that includes designing, debugging, and maintaining queries. We argue that to fully support this shift toward more exploratory query development, the way we design and execute queries must be rethought. In particular, we highlight three key areas where the support for exploratory querying needs improvement: query and schema languages, query engines, and user interfaces. Future data management systems should move toward more interactive and dynamic query design, enabling users to refine queries in real time, receive intelligent debugging assistance, and leverage AI-driven tools to bridge the gap between query intent and execution. By explicitly treating exploratory querying, we open new directions for research and system development. We hope that this discussion sparks further exploration into methodologies and frameworks that support more intuitive and adaptive querying in data management systems. Ultimately, this shift is more than just a technical advancement; it is a rethinking of how we engage with data in an increasingly complex and dynamic world.

ACKNOWLEDGMENTS

Work originated in No.183 NII Shonan Meeting, 2024. Hartig was funded by the Knut and Alice Wallenberg Foundation (KAW 2023.0111), Konstantinidis by HE projects UPCAST(101093216), RAISE(101058479), DataPACT(101189771), and RAISE Suite(101188337), and Sasaki by JST ASPIRE (JPMJAP2328).

REFERENCES

[1] Sihem Amer-Yahia. 2024. Intelligent Agents for Data Exploration. *Proc. VLDB Endow.* 17, 12 (2024), 4521–4530. https://www.vldb.org/pvldb/vol17/p4521-amer-yahia.pdf

[2] Marcelo Arenas. 2024. A Data Management Approach to Explainable AI. In *43rd Symposium on Principles of Database Systems, PODS 2024*. 1–3.

[3] Marcelo Arenas, Daniel Baez, Pablo Barceló, Jorge Pérez, and Bernardo Subercaseaux. 2021. Foundations of Symbolic Languages for Model Interpretability. In *NeurIPS 2021*. 11690–11701.

[4] Marcelo Arenas, Pablo Barceló, Diego Bustamante, Jose Caraball, and Bernardo Subercaseaux. 2024. A Uniform Language to Explain Decision Trees. In *21st International Conference on Principles of Knowledge Representation and Reasoning, KR 2024*. 60–70.

[5] Mohamed-Amine Baazizi, Dario Colazzo, Giorgio Ghelli, and Carlo Sartiani. 2019. Parametric schema inference for massive JSON datasets. *VLDB J.* 28, 4 (2019), 497–521. https://doi.org/10.1007/S00778-018-0532-7

[6] Pablo Barceló, Leonid Libkin, and Miguel Romero. 2014. Efficient Approximations of Conjunctive Queries. *SIAM J. Comput.* 43, 3 (2014), 1085–1130. https://doi.org/10.1137/130911731

[7] Marcia J Bates. 1989. The design of browsing and berrypicking techniques for the online search interface. *Online review* 13, 5 (1989), 407–424.

[8] Geert Jan Bex, Wouter Gelade, Frank Neven, and Stijn Vansummeren. 2010. Learning Deterministic Regular Expressions for the Inference of Schemas from XML Data. *ACM Trans. Web* 4, 4 (2010), 14:1–14:32. https://doi.org/10.1145/1841909.1841911

[9] Geert Jan Bex, Frank Neven, Thomas Schwentick, and Stijn Vansummeren. 2010. Inference of concise regular expressions and DTDs. *ACM Trans. Database Syst.* 35, 2 (2010), 11:1–11:47. https://doi.org/10.1145/1735886.1735890

[10] Henrik Björklund, Johanna Björklund, and Wim Martens. 2021. Learning algorithms. In *Handbook of Automata Theory*, Jean-Éric Pin (Ed.). European Mathematical Society Publishing House, Zürich, Switzerland, 375–409. https://doi.org/10.4171/AUTOMATA-1/11

[11] Ann Blandford and Simon Attfield. 2010. *Interacting with Information*. Morgan & Claypool Publishers. https://doi.org/10.2200/S00227ED1V01Y200911HCI006

[12] Angela Bonifati, Stefania Dumbrava, George Fletcher, Jan Hidders, Matthias Hofer, Wim Martens, Filip Murlak, Joshua Shinavier, Slawek Staworko, and Dominik Tomaszuk. 2022. Threshold Queries in Theory and in the Wild. *Proc. VLDB Endow.* 15, 5 (2022), 1105–1118. https://doi.org/10.14778/3510397.3510407

[13] Angela Bonifati, Stefania Dumbrava, George Fletcher, Jan Hidders, Matthias Hofer, Wim Martens, Filip Murlak, Joshua Shinavier, Slawek Staworko, and Dominik Tomaszuk. 2023. Threshold Queries. *SIGMOD Rec.* 52, 1 (2023), 64–73. https://doi.org/10.1145/3604437.3604452

[14] Angela Bonifati, Stefania-Gabriela Dumbrava, Emile Martinez, Fatemeh Ghasemi, Malo Jaffré, Pacome Luton, and Thomas Pickles. 2022. DiscoPG: Property Graph Schema Discovery and Exploration. *Proc. VLDB Endow.* 15, 12 (2022), 3654–3657. https://doi.org/10.14778/3554821.3554867

[15] Angela Bonifati, Stefania Dumbrava, and Nicolas Mir. 2022. Hierarchical Clustering for Property Graph Schema Discovery. In *Proceedings of the 25th International Conference on Extending Database Technology, EDBT 2022, Edinburgh, UK, March 29 - April 1, 2022*, Julia Stoyanovich, Jens Teubner, Paolo Guagliardo, Milos Nikolic, Andreas Pieris, Jan Mühlig, Fatma Özcan, Sebastian Schelter, H. V. Jagadish, and Meihui Zhang (Eds.). OpenProceedings.org, 2:449–2:453. https://doi.org/10.48786/EDBT.2022.39

[16] Angela Bonifati, Wim Martens, and Thomas Timm. 2020. An analytical study of large SPARQL query logs. *VLDB J.* 29, 2-3 (2020), 655–679. https://doi.org/10.1007/S00778-019-00558-9

[17] Stefan Brass and Christian Goldberg. 2006. Semantic errors in SQL queries: A quite complete list. *Journal of Systems and Software* 79, 5 (2006), 630–644.

[18] Michael J. Carey and Donald Kossmann. 1997. On Saying "Enough Already!" in SQL. In *SIGMOD 1997, Proceedings ACM SIGMOD International Conference on Management of Data, May 13-15, 1997, Tucson, Arizona, USA*, Joan Peckham (Ed.). ACM Press, 219–230. https://doi.org/10.1145/253260.253302

[19] Tiziana Catarci, Maria Francesca Costabile, Stefano Levialdi, and Carlo Batini. 1997. Visual Query Systems for Databases: A Survey. *Journal of Visual Languages & Computing* 8, 2 (1997), 215–260. https://doi.org/10.1006/jvlc.1997.0037

[20] Kerry Shih-Ping Chang and Brad A Myers. 2016. Using and exploring hierarchical data in spreadsheets. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. 2497–2507.

[21] Adriane Chapman, Elena Simperl, Laura Koesten, George Konstantinidis, Luis-Daniel Ibáñez, Emilia Kacprzak, and Paul Groth. 2020. Dataset search: a survey. *The VLDB Journal* 29, 1 (2020), 251–272.

[22] Surajit Chaudhuri, Ravi Krishnamurthy, Spyros Potamianos, and Kyuseok Shim. 1995. Optimizing queries with materialized views. In *Proceedings of the Eleventh International Conference on Data Engineering*. IEEE, 190–200.

[23] James Cheney, Laura Chiticariu, and Wang Chiew Tan. 2009. Provenance in Databases: Why, How, and Where. *Found. Trends Databases* 1, 4 (2009), 379–474. https://doi.org/10.1561/1900000006

[24] Graham Cormode and Ke Yi. 2020. *Small Summaries for Big Data*. Cambridge University Press.

[25] Binyang Dai, Xiao Hu, and Ke Yi. 2024. Reservoir Sampling over Joins. *Proc. ACM Manag. Data* 2, 3 (2024), 118. https://doi.org/10.1145/3654921

[26] Alan Dix, Janet E. Finlay, Gregory D. Abowd, and Russell Beale. 2003. *Human-Computer Interaction (3rd Edition)*. Prentice-Hall, Inc.

[27] Jennie Duggan, Aaron J Elmore, Michael Stonebraker, Magda Balazinska, Bill Howe, Jeremy Kepner, Sam Madden, David Maier, Tim Mattson, and Stan Zdonik. 2015. The bigdawg polystore system. *ACM Sigmod Record* 44, 2 (2015), 11–16.

[28] Philipp Eichmann, Emanuel Zgraggen, Carsten Binnig, and Tim Kraska. 2020. IDEBench: A Benchmark for Interactive Data Exploration. In *Proceedings of the 2020 International Conference on Management of Data, SIGMOD Conference 2020, online conference [Portland, OR, USA], June 14-19, 2020*, David Maier, Rachel Pottinger, AnHai Doan, Wang-Chiew Tan, Abdussalam Alawini, and Hung Q. Ngo (Eds.). ACM, 1555–1569. https://doi.org/10.1145/3318464.3380574

[29] Philipp Eichmann, Emanuel Zgraggen, Carsten Binnig, and Tim Kraska. 2020. IDEBench: A Benchmark for Interactive Data Exploration. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data* (Portland, OR, USA) *(SIGMOD '20)*. Association for Computing Machinery, New York, NY, USA, 1555–1569. https://doi.org/10.1145/3318464.3380574

[30] Enrico Franconi, Paolo Guagliardo, Marco Trevisan, and Sergio Tessaris. 2011. Quelo: an Ontology-Driven Query Interface. In *Proceedings of the 24th International Workshop on Description Logics (DL 2011), Barcelona, Spain, July 13-16, 2011 (CEUR Workshop Proceedings)*, Riccardo Rosati, Sebastian Rudolph, and Michael Zakharyaschev (Eds.), Vol. 745. CEUR-WS.org. http://www.inf.unibz.it/~franconi/quelo/

[31] Sneha Gathani, Peter Lim, and Leilani Battle. 2020. Debugging database queries: A survey of tools, techniques, and users. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–16.

[32] Wolfgang Gatterbauer and Cody Dunne. 2024. On The Reasonable Effectiveness of Relational Diagrams: Explaining Relational Query Patterns and the Pattern Expressiveness of Relational Languages. *Proc. ACM Manag. Data* 2, 1 (2024), 61:1–61:27. https://doi.org/10.1145/3639316

[33] Wolfgang Gatterbauer, Cody Dunne, H. Jagadish, and Mirek Riedewald. 2022. Principles of Query Visualization. https://doi.org/10.48550/arXiv.2208.01613

[34] Georg Gottlob, Nicola Leone, and Francesco Scarcello. 2001. The complexity of acyclic conjunctive queries. *J. ACM* 48, 3 (2001), 431–498. https://doi.org/10.1145/382780.382783

[35] Benoît Groz, Aurélien Lemay, Slawek Staworko, and Piotr Wieczorek. 2022. Inference of Shape Graphs for Graph Databases. In *25th International Conference on Database Theory, ICDT 2022, March 29 to April 1, 2022, Edinburgh, UK (Virtual Conference) (LIPIcs)*, Dan Olteanu and Nils Vortmeier (Eds.), Vol. 220. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 14:1–14:20. https://doi.org/10.4230/LIPICS.ICDT.2022.14

[36] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. 2019. A Survey of Methods for Explaining Black Box Models. *ACM Comput. Surv.* 51, 5 (2019), 93:1–93:42.

[37] Alon Y Halevy. 2001. Answering queries using views: A survey. *The VLDB Journal* 10 (2001), 270–294.

[38] Marti Hearst. 2009. *Search user interfaces*. Cambridge University Press.

[39] Gary G Hendrix, Earl D Sacerdoti, Daniel Sagalowicz, and Jonathan Slocum. 1978. Developing a natural language interface to complex data. *ACM Transactions on Database Systems (TODS)* 3, 2 (1978), 105–147.

[40] Melanie Herschel, Ralf Diestelkämper, and Houssem Ben Lahmar. 2017. A survey on provenance: What for? What form? What from? *The VLDB Journal* 26 (2017), 881–906.

[41] Yuntong Hu, Zhihan Lei, Zheng Zhang, Bo Pan, Chen Ling, and Liang Zhao. 2024. GRAG: Graph Retrieval-Augmented Generation. *CoRR* abs/2405.16506 (2024).

[42] Stratos Idreos, Olga Papaemmanouil, and Surajit Chaudhuri. 2015. Overview of Data Exploration Techniques. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. ACM, 277–281. https://doi.org/10.1145/2723372.2731084

[43] Matthias Jarke and Yannis Vassiliou. 1985. A Framework for Choosing a Database Query Language. *ACM Comput. Surv.* 17 (09 1985), 313–340. https://doi.org/10.1145/5505.5506

[44] Herbert Jordan, Bernhard Scholz, and Pavle Subotic. 2016. Soufflé: On Synthesis of Program Analyzers. In *Computer Aided Verification - 28th International Conference, CAV 2016, Toronto, ON, Canada, July 17-23, 2016, Proceedings, Part II (Lecture Notes in Computer Science)*, Swarat Chaudhuri and Azadeh Farzan (Eds.), Vol. 9780. Springer, 422–430. https://doi.org/10.1007/978-3-319-41540-6_23

[45] Makoto P Kato, Tetsuya Sakai, and Katsumi Tanaka. 2012. Structured query suggestion for specialization and parallel movement: effect on search behaviors. In *Proceedings of the 21st international conference on World Wide Web*. 389–398.

[46] Cecilia Katzeff. 1988. The effect of different conceptual models upon reasoning in a database query writing task. *International Journal of Man-Machine Studies* 29, 1 (1988), 37–62.

[47] Moe Kayali, Anton Lykov, Ilias Fountalis, Nikolaos Vasiloglou, Dan Olteanu, and Dan Suciu. 2024. CHORUS: Foundation Models for Unified Data Discovery and Exploration. *Proc. VLDB Endow.* 17, 8 (2024), 2104–2114. https://www.vldb.org/pvldb/vol17/p2104-kayali.pdf

[48] Stefan Klessinger, Meike Klettke, Uta Störl, and Stefanie Scherzinger. 2022. Extracting JSON Schemas with tagged unions. In *Proceedings of the First International Workshop on Data Ecosystems co-located with 48th International Conference on Very Large Databases (VLDB 2022) (CEUR Workshop Proceedings)*, Vol. 3306. CEUR-WS.org, 27–40. https://ceur-ws.org/Vol-3306/paper4.pdf

[49] Meike Klettke, Uta Störl, and Stefanie Scherzinger. 2015. Schema Extraction and Structural Outlier Detection for JSON-based NoSQL Data Stores. In *Datenbanksysteme für Business, Technologie und Web (BTW) (LNI)*, Vol. P-241. GI, 425–444. https://dl.gi.de/handle/20.500.12116/2420

[50] Laura Koesten. 2019. *A user centred perspective on structured data discovery*. Ph.D. Dissertation. University of Southampton.

[51] Laura Koesten, Kathleen Gregory, Paul Groth, and Elena Simperl. 2021. Talking datasets–understanding data sensemaking behaviours. *International journal of human-computer studies* 146 (2021), 102562.

[52] Laura M Koesten, Emilia Kacprzak, Jenifer FA Tennison, and Elena Simperl. 2017. The Trials and Tribulations of Working with Structured Data: -a Study on Information Seeking Behaviour. In *Proceedings of the 2017 CHI conference on human factors in computing systems*. 1277–1289.

[53] George Konstantinidis and José Luis Ambite. 2011. Scalable query rewriting: a graph-based approach. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*. 97–108.

[54] Tim Kraska. 2018. Northstar: An Interactive Data Science System. *Proc. VLDB Endow.* 11, 12 (2018), 2150–2164. https://doi.org/10.14778/3229863.3240493

[55] Carol Collier Kuhlthau. 1999. The role of experience in the information search process of an early career information worker: perceptions of uncertainty, complexity, construction, and sources. *J. Am. Soc. Inf. Sci.* 50, 5 (April 1999), 399–412.

[56] Viktor Leis, Andrey Gubichev, Atanas Mirchev, Peter A. Boncz, Alfons Kemper, and Thomas Neumann. 2015. How Good Are Query Optimizers, Really? *Proc. VLDB Endow.* 9, 3 (2015), 204–215. https://doi.org/10.14778/2850583.2850594

[57] Aristotelis Leventidis, Jiahui Zhang, Cody Dunne, Wolfgang Gatterbauer, H.V. Jagadish, and Mirek Riedewald. 2020. QueryVis: Logic-based Diagrams help Users Understand Complicated SQL Queries Faster. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data (SIGMOD '20)*. 2303–2318.

[58] Pengfei Li, Yin Zhang, and Bin Zhang. 2022. Understanding Query Combination Behavior in Exploratory Searches. *Applied Sciences* 12, 2 (2022). https://doi.org/10.3390/app12020706

[59] Dandan Liu and Zhaonian Zou. 2023. gCore: Exploring Cross-layer Cohesiveness in Multi-layer Graphs. *Proc. VLDB Endow.* 16, 11 (2023), 3201–3213. https://doi.org/10.14778/3611479.3611519

[60] Zhicheng Liu and Jeffrey Heer. 2014. The Effects of Interactive Latency on Exploratory Visual Analysis. *IEEE Transactions on Visualization and Computer Graphics* 20, 12 (2014), 2122–2131. https://doi.org/10.1109/TVCG.2014.2346452

[61] Jorge Lloret-Gazo. 2016. A Survey on Visual Query Systems in the Web Era. In *Database and Expert Systems Applications*. Springer International Publishing, 343–351.

[62] Scott M. Lundberg and Su-In Lee. 2017. A Unified Approach to Interpreting Model Predictions. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*. 4765–4774.

[63] Qingzhi Ma, Ali Mohammadi Shanghooshabad, Mehrdad Almasi, Meghdad Kurmanji, and Peter Triantafillou. 2021. Learned Approximate Query Processing: Make it Light, Accurate and Fast. In *CIDR*. www.cidrdb.org.

[64] Stanislav Malyshev, Markus Krötzsch, Larry González, Julius Gonsior, and Adrian Bielefeldt. 2018. Getting the Most out of Wikidata: Semantic Technology Usage in Wikipedia's Knowledge Graph. In *Proceedings of the 17th International Semantic Web Conference (ISWC'18) (LNCS)*, Denny Vrandečić, Kalina Bontcheva, Mari Carmen Suárez-Figueroa, Valentina Presutti, Irene Celino, Marta Sabou, Lucie-Aimée Kaffee, and Elena Simperl (Eds.), Vol. 11137. Springer, 376–394.

[65] Antonis Mandamadiotis, Georgia Koutrika, and Sihem Amer-Yahia. 2024. Guided SQL-Based Data Exploration with User Feedback. In *40th IEEE International Conference on Data Engineering, ICDE 2024, Utrecht, The Netherlands, May 13-16, 2024*. IEEE, 4884–4896. https://doi.org/10.1109/ICDE60146.2024.00372

[66] Gary Marchionini. 2006. Exploratory search: from finding to understanding. *Commun. ACM* 49, 4 (2006), 41–46.

[67] João Marques-Silva. 2024. Logic-Based Explainability: Past, Present & Future. *CoRR* abs/2406.11873 (2024).

[68] João Marques-Silva and Xuanxiang Huang. 2024. Explainability Is *Not* a Game. *Commun. ACM* 67, 7 (2024), 66–75.

[69] João Marques-Silva and Alexey Ignatiev. 2023. No silver bullet: interpretable ML models must be explained. *Frontiers Artif. Intell.* 6 (2023).

[70] Miti Mazmudar, Thomas Humphries, Jiaxiang Liu, Matthew Rafuse, and Xi He. 2022. Cache Me If You Can: Accuracy-Aware Inference Engine for Differentially Private Data Exploration. *Proc. VLDB Endow.* 16, 4 (2022), 574–586. https://doi.org/10.14778/3574245.3574246

[71] Zhengjie Miao, Tiangang Chen, lexander Bendeck, Kevin Day, Sudeepa Roy, and Jun Yang. 2020. I-Rex: an interactive relational query explainer for SQL. *Proc. VLDB Endow.* 13, 12 (2020), 2997–3000.

[72] Daphne Miedema, Efthimia Aivaloglou, and George Fletcher. 2021. Identifying SQL Misconceptions of Novices: Findings from a Think-Aloud Study. In *Proceedings of the 17th ACM Conference on International Computing Education Research* (Virtual Event, USA) *(ICER 2021)*. Association for Computing Machinery, New York, NY, USA, 355–367. https://doi.org/10.1145/3446871.3469759

[73] Daphne Miedema and George Fletcher. 2021. SQLVis: Visual Query Representations for Supporting SQL Learners. In *2021 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. 1–9.

[74] Randy Minas, Binny Samuel, Lingyao Yuan, and Vijay Khatri. 2014. Exploring the iterative nature of corrective SQL maintenance: an eye tracking study. In *24th Workshop on Information Technologies and Systems, Auckland, New Zealand*.

[75] Christoph Molnar. 2022. *Interpretable Machine Learning*. https://christophm.github.io/interpretable-ml-book

[76] C Naumer, K Fisher, and B Dervin. 2008. SenseMaking: A Methodological Perspective. Conference paper. In *CHI Conference on Human Factors in Computing Systems, Florence, Italy. https://tinyurl. com/3zx9r39a*.

[77] PRQL. 2024. *Pipelined Relational Query Language*. https://prql-lang.org.

[78] RelationalAI. 2024. https://learn.relational.ai/.

[79] Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*. ACM, 1135–1144.

[80] Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. 2018. Anchors: High-Precision Model-Agnostic Explanations. In *AAAI*. 1527–1535.

[81] Kenneth A. Ross. 1992. Relations with Relation Names as Arguments: Algebra and Calculus. In *Proceedings of the Eleventh ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 2-4, 1992, San Diego, California, USA*, Moshe Y. Vardi and Paris C. Kanellakis (Eds.). ACM Press, 346–353. https://doi.org/10.1145/137097.137905

[82] Cynthia Rudin. 2019. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nat. Mach. Intell.* 1, 5 (2019), 206–215.

[83] Session 10 [n.d.]. https://github.com/hartig/ExploratoryQueryingSessions/tree/main/sessions/session10.

[84] Session 11 [n.d.]. https://github.com/hartig/ExploratoryQueryingSessions/tree/main/sessions/session11.

[85] Sessions of Exploratory Querying [n.d.]. https://github.com/hartig/ExploratoryQueryingSessions.

[86] Jeff Shute, Shannon Bales, Matthew Brown, Jean-Daniel Browne, Brandon Dolphin, Romit Kudtarkar, Andrey Litvinov, Jingchi Ma, John D. Morcos, Michael Shen, David Wilhite, Xi Wu, and Lulan Yu. 2024. SQL has problems. We can fix them: Pipe syntax in SQL. *Proc. VLDB Endow.* 17, 12 (2024), 4051–4063. https://www.vldb.org/pvldb/vol17/p4051-shute.pdf

[87] Tarique Siddiqui, Albert Kim, John Lee, Karrie Karahalios, and Aditya Parameswaran. 2016. Effortless data exploration with zenvisage: an expressive and interactive visual analytics system. *Proc. VLDB Endow.* 10, 4 (Nov. 2016), 457–468. https://doi.org/10.14778/3025111.3025126

[88] William Spoth, Oliver Kennedy, Ying Lu, Beda Christoph Hammerschmidt, and Zhen Hua Liu. 2021. Reducing Ambiguity in Json Schema Discovery. In *SIGMOD '21: International Conference on Management of Data, Virtual Event, China, June 20-25, 2021*, Guoliang Li, Zhanhuai Li, Stratos Idreos, and Divesh Srivastava (Eds.). ACM, 1732–1744. https://doi.org/10.1145/3448016.3452801

[89] Alistair G. Sutcliffe and Mark Ennis. 1998. Towards a cognitive theory of information retrieval. *Interacting with Computers* 10, 3 (1998), 321–351. https://doi.org/10.1016/S0953-5438(98)00013-7

[90] Saedeh Tahery and Saeed Farzi. 2020. Customized query auto-completion and suggestion—A review. *Information Systems* 87 (2020), 101415.

[91] Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Commun. ACM* 57 (October 2014), 78–85. https://doi.org/10.1145/2629489

[92] Nina Wacholder. 2011. Interactive query formulation. *Annual review of information science and technology* 45, 1 (2011), 157–196.

[93] Charles Welty. 1990. Human factors studies of database query languages: SQL as a metric. *Journal of Database Management (JDM)* 1, 1 (1990), 2–11.

[94] Ryen W White and Resa A Roth. 2009. *Exploratory search: Beyond the query-response paradigm*. Number 3. Morgan & Claypool Publishers.

[95] Barbara M. Wildemuth, Diane Kelly, Emma Boettcher, Erin Moore, and Gergana Dimitrova. 2018. Examining the impact of domain and cognitive complexity on query formulation and reformulation. *Information Processing & Management* 54, 3 (2018), 433–450.

[96] Peter T. Wood. 1993. Bottom-Up Evaluation of DataHiLog. In *Rules in Database Systems. Proceedings of the 1st International Workshop on Rules in Database*

Systems, Edinburgh, Scotland, 30 August - 1 September 1993 (Workshops in Computing), Norman W. Paton and M. Howard Williams (Eds.). Springer, 401–415. https://doi.org/10.1007/978-1-4471-3225-7_24

[97] Peipei Yi, Byron Choi, Sourav S Bhowmick, and Jianliang Xu. 2016. Autog: A visual query autocompletion framework for graph databases. *Proceedings of the VLDB Endowment* 9, 13 (2016), 1505–1508.

[98] Cong Yu and H. V. Jagadish. 2006. Schema Summarization. In *Proceedings of the 32nd International Conference on Very Large Data Bases, Seoul, Korea, September 12-15, 2006*, Umeshwar Dayal, Kyu-Young Whang, David B. Lomet, Gustavo Alonso, Guy M. Lohman, Martin L. Kersten, Sang Kyun Cha, and Young-Kuk Kim (Eds.). ACM, 319–330. http://dl.acm.org/citation.cfm?id=1164156

[99] Joohyung Yun, Byungchul Tak, and Wook-Shin Han. 2024. ReCG: Bottom-Up JSON Schema Discovery Using a Repetitive Cluster-and-Generalize Framework. *Proc. VLDB Endow.* 17, 11 (2024), 3538–3550. https://doi.org/10.14778/3681954.3682019

[100] Xinyue Zhang, Meng Wang, Muhammad Saleem, Axel-Cyrille Ngonga Ngomo, Guilin Qi, and Haofen Wang. 2020. Revealing Secrets in SPARQL Session Level. In *The Semantic Web - ISWC 2020 - 19th International Semantic Web Conference, Athens, Greece, November 2-6, 2020, Proceedings, Part I (Lecture Notes in Computer Science)*, Vol. 12506. Springer, 672–690.