A Tutorial on Visual Representations of Relational Queries

Wolfgang Gatterbauer
Northeastern University
Boston, Massachusetts, USA
w.gatterbauer@northeastern.edu

ABSTRACT

Query formulation is increasingly performed by systems that need to guess a user's intent (e.g. via spoken word interfaces). But how can a user know that the computational agent is returning answers to the "right" query? More generally, given that relational queries can become pretty complicated, *how can we help users understand existing relational queries*, whether human-generated or automatically generated? Now seems the right moment to revisit a topic that predates the birth of the relational model: developing visual metaphors that help users understand relational queries.

This lecture-style tutorial surveys the key *visual metaphors developed for visual representations of relational expressions*. We will survey the history and state-of-the art of relationally-complete diagrammatic representations of relational queries, discuss the key visual metaphors developed in over a century of investigating diagrammatic languages, and organize the landscape by mapping their used visual alphabets to the syntax and semantics of Relational Algebra (RA) and Relational Calculus (RC).

PVLDB Reference Format:
Wolfgang Gatterbauer. A Tutorial on Visual Representations of Relational Queries. PVLDB, 16(12): 3890 - 3893, 2023.
doi:10.14778/3611540.3611578

1 INTRODUCTION

The design of relational query languages and the difficulty for users to compose relational queries have received much attention over the last 40 years [9, 12, 23, 27, 31, 41, 42, 48, 49]. A complementary and much-less-studied problem is that of helping users *read and understand an existing relational query*. Reading code is hard, and SQL is no exception. With the proliferation of public data sources, and associated queries, users increasingly have a need to read other people's queries and scripts. Furthermore, it is usually much easier to modify a draft than to write something from scratch. As such, modifying an already existing query could be an effective way to write new queries. However, modifying an existing query requires first to understand it. For that reason, it is valuable to help users understand queries, and visualization is one obvious route. While visual methods for expressing queries have been studied extensively in the database literature under the topic of Visual Query Languages (VQLs) [9], the challenges for supporting the explicit *reverse functionality* of creating a visual representation of an existing query ("Query Visualization") are on a whole different from the problem of composing a new query (Fig. 1).

Proceedings of the VLDB Endowment, Vol. 16, No. 12 ISSN 2150-8097.
doi:10.14778/3611540.3611578

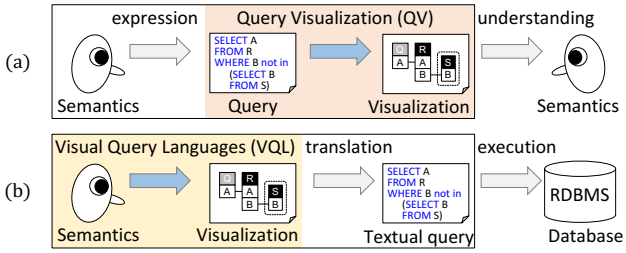

Figure 1: Contrasting Query visualization (on the top (a), in orange) with Visual Query Languages (VQL) (on the bottom (b) in yellow).

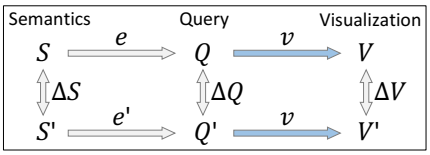

Figure 2: A simple algebraic framework will help us discuss various principles of query visualization (best understood with Fig. 1a).

The tutorial uses a few relational queries to survey and summarize over a history of *diagrammatic (thus visual) representations of first-order logic queries and statements*. The tutorial will contrast and highlight similarities and differences between approaches proposed across communities and use a mapping of the visual representations to equivalent expressions in Relational Algebra (RA) and Relational Calculus (RC) to guide the journey.

Outline. The lecture-style 1.5-hour tutorial consists of five parts:

(1) *Why visualizing queries and why now*: We contrast Query Visualization (QV) with Visual Query Languages (VQL) and give several usage scenarios for the use of the former.

(2) *Principles of Query Visualization*: We discuss 8 recently proposed principles of query visualization [18, 20], re-phrased in the terminology of "Algebraic Visualization Design" [28] (Fig. 2). We later refer to them when discussing different visualizations.

(3) *Logical foundations of relational query languages*: We discuss the logical foundations of relational query languages. These concepts are also used when discussing visual representations.

(4) *Early diagrammatic representations*: Diagrammatic representations for logical statements were developed well before relational databases. We will discuss the influential beta existential graphs by Peirce [37] and their connection to the much later developed RC.

(5) *Modern Visual Query Representations and Design trade-offs*: We use a few queries over intuitive database schemas to discuss the main visual representations for relational queries proposed by the database community.

Slides and videos of the tutorial will be made available afterwards on the tutorial web page,[1] similar to other recent tutorials by the presenter and collaborators on unrelated topics.[2][3]

2 TUTORIAL INFORMATION

Audience and prerequisite. This 90 min tutorial targets researchers and practitioners who desire an intuitive introduction to the history of and approaches for visual representations of relational queries and logical statements, and desire to see major commonalities across past major designs of visual languages. The tutorial is best followed by being familiar with Relational Algebra (RA), Relational Calculus (RC) and the safety conditions to make them equivalent in expressiveness. However, the tutorial is self-contained and includes a concise short-paced introduction into overall characteristics of relational languages.

Scope of this tutorial. This tutorial surveys visual formalisms for representing *relational queries*. The focus is on relationally complete formalisms whose expressiveness is equivalent to Relational Algebra (RA) and Relational Calculus (RC) and non-recursive Datalog with stratified negation. In order to guide the discussion, the tutorial discusses mapping the visual alphabets of visual formalisms to expressions of RA and RC. It thus starts with a quick overview of RA and RC and their connection to first-order logic. It discusses various extensions to the relationally complete fragments (such as groupings and recursion) only at the end if time permits.

Out-of-scope. The tutorial does not discuss domain-specific visualizations, such as those developed for geographic information systems, time-series, and spatio-temporal data [5, 14, 30, 33]. Neither does it discuss dynamic interaction with queries or data [36].

Related other tutorials. A tutorial at SIGMOD'19 [47] ("Towards Democratizing Relational Data Visualizations") focused on ways to visualize data and languages that allow users to specify what visualizations they want to apply to data. The focus of this tutorial is instead of visual representations *of queries*. Two tutorials at SIGMOD'17 [7] ("Graph Querying Meets HCI") and SIGMOD'22 [6] ("Data-driven Visual Query Interfaces for Graphs") focused on visual composition of graph queries. The types of queries discussed in those tutorials basically correspond to conjunctive queries with inequalities over binary predicates, whereas our focus is on full-first order logic. Also the focus was on the human-interaction aspect of how to compose queries, while our focus is on the visual formalisms developed for relational queries over the last century (thus even predating the relational model).

Prior offerings of this tutorial. An early version of this tutorial was presented at the "*International Conference on the Theory and Application of Diagrams 2022*" (DIAGRAMS-22) [19], which is the main international venue covering all aspects of research on the theory an application of diagrams and attracts and audience complementary to the audience at VLDB. While that prior tutorial had a stronger focus on the 3rd (logical foundations of query languages) and 4th parts (diagrams predating relational databases), this tutorial will emphasize 2nd (principles of query visualization) and 5th parts (visual query languages developed in the database community).

3 TUTORIAL CONTENT

3.1 Why visualizing queries and why now?

The tutorial starts by giving several scenarios in which "appropriate" query visualizations could help users achieve new functionalities or increased efficiency in composing queries. An important detail is here that visualizations can be used as *complement* to query composition, *instead of substitution* for textual input. This contrasts with Visual Query Languages (VQLs) which allow users to express queries in a visual format. Visual methods for specifying relational queries have been studied extensively [9], and many commercial database products offer some visual interface for users to write simple conjunctive queries. In parallel, there is a centuries-old history on the study of formal diagrammatic reasoning systems [25] with the goal of helping humans to reason in terms of logical statements.[4]

Yet despite their intuitive appeal and extensive study, successful visual tools today mostly only *complement instead of replace* text for composing queries. We will discuss several reasons for why visual query composition *for general relational queries* have not yet widely replaced textual query composition and discuss a user-query interaction that separates the query composition from the visualization: Composition is unchanged and still done in text (or alternatively with exploratory input formats like natural language). But composition is augmented and *complemented with a visual that helps interpretation* [18]. With this motivation, the goal of this tutorial is to survey and highlight the key ideas behind major proposals for diagrammatic representations of relational statements and queries.

Definition 1 (Query Visualization [20]). *The term "query visualization" refers to both (i) a graphical representation of a query and (ii) the process of transforming a query into a graphical representation. The goal of query visualization is to help users more quickly understand the intent of a query, as well as its relational query pattern.*

3.2 Principles of Query Visualization

The challenge of query visualization is to find appropriate visual metaphors that (*i*) allow users to quickly understand a query's intent, even for complex queries, (*ii*) can be easily learned by users, and (*iii*) can be obtained from textual queries by automatic translation, including a visually-appealing automatic arrangement of elements of the visualization. We discuss 8 recently proposed principles of query visualization [20], however newly organized, extended, and rephrased in the terminology of "Algebraic Visualization Design" [28] (Fig. 2). We refer to them later extensively when discussing different visualizations. We also include those in order to spark a healthy debate during and after the tutorial.

3.3 Logical foundations of relational languages

We give a concise but comprehensive overview of *the logical foundations of relational query languages*. This overview uses a consistent

[1] https://northeastern-datalab.github.io/visual-query-representation-tutorial/
[2] https://northeastern-datalab.github.io/topk-join-tutorial/
[3] https://northeastern-datalab.github.io/responsive-dbms-tutorial/

[4] A relational query is a logical formula with free variables. A logical statement has no free variables and is intuitively the same as a Boolean query that returns a truth value of TRUE or FALSE.

notation that establishes shared concepts across relationally complete textual query languages. We will reuse those extensively later when discussing visual query representations where we establish direct mappings between a given visual formalisms and logically equivalent textual queries. These mappings allow us a unified comparison of visual alphabets and their "pattern expressiveness". Thus our focus is on expressiveness basically equivalent to first-order logic, which allows us to connect a century of research on formalisms for diagrammatic reasoning to our topic.

3.4 Early diagrammatic representations

Relational calculus is a specialization of First-Order Logic (FOL), namely expressions with free variables. Diagrammatic representations for logical statements [25] have been developed even before FOL, which was only clearly articulated in the 1928 first edition of David Hilbert and Wilhelm Ackermann's "Grundzüge der theoretischen Logik" [24]. An influential diagrammatic notation is the *existential graph notation* by Charles Sanders Peirce [37, 43, 45], who wrote on graphical logic as early as 1882 [29]. These graphs exploit topological properties, such as enclosure, to represent logical expressions and set-theoretic relationships. Peirce's graphs come in two variants: alpha and beta. Alpha graphs represent propositional logic, whereas beta graphs represent first-order logic (FOL). Both variants use so-called *cuts* to express negation (similar to our nesting boxes), and beta graphs use a syntactical element called the *Line of Identity* (LI) to denote *both the existence of objects and the identity between objects*. An important component of our discussions of Beta-existential graphs is showing their imperfect mapping to the Boolean fragment of restricted forms of DRC. As we will show, this imperfection has led to a lot of follow-up and confusions in various work on Peirce's existential graphs. We may also shortly cover the close connections to Euler diagrams, Venn diagramms, and Venn-Peirce diagrams, following the exposition by Shin [44].

3.5 Modern Visual Query Representations

We discuss the main proposed visual representations for relational queries. We will also include influential Visual Query Languages (VQLs) as long as those support (either directly or via simple additions) the inverse functionality of visualizing an existing relational query. A key difference of our tutorial in contrast to all prior surveys and overviews that we are aware of (like [9]) is that this tutorial shows original figures by using a consistent schema (the sailor-boat-database from the "cow book" [40]) and a few intuitive queries (such as "find sailors who have rented all red boats") to provide a consistent comparison across different past proposals.

Query-By-Example (QBE) [50] is an influential early VQL that was strongly influenced by DRC. QBE can express relational division breaking the query into two logical steps and using a temporary relation [40, Ch. 6.9]. But in doing so, QBE uses the query pattern from RA of implementing relational division (or universal quantification) in a dataflow-type, sequential manner, requiring multiple occurrences of the same table.

Interactive query builders employ visual diagrams that users can manipulate (most often in order to select tables and attributes) while using *a separate query configurator* (similar to QBE's condition boxes [50]) to specify selection predicates, attributes, and sometimes nesting between queries. They work mainly for constructing conjunctive queries but limited forms of negation and union can be incorporated into the condition part of such queries. For more general forms of negation and union, however, views as intermediate relations need to be used, resulting in multiple screens. dbForge [16] is the most advanced and commercially supported tool we found for interactive query building. Yet it does not show any visual indication for non-equi joins between tables and the actual filtering values and aggregation functions can only be added in a separate query configurator. Moreover, it has limited support for nested queries: the inner and outer queries are built separately, and the diagram for the inner query is *presented separately and disjointly* from the diagram for the outer query. Thus *no visual depiction of correlated subqueries is possible*. Other graphical SQL editors like SQL Server Management Studio (SSMS) [46], Active Query Builder [2], QueryScope from SQLdep [39], MS Access [34], and PostgreSQL's pgAdmin3 [38] lack in even more aspects of visual query representations: most do not allow nested queries, none has a single visual element for the logical quantifiers `NOT EXISTS` or `FOR ALL`, and all require specifying details of the query in SQL or across several tabbed views *separate from a visual diagram*.

Dataflow Query Language (DFQL) is an example visual representation that is relationally complete [9, 13] by mapping its visual symbols to the operators of relational algebra. Following the same procedurality as RA, DFQL expresses the dataflow in a top-down tree-like structure. Like most visual formalisms that we are aware of and that were proven to be relationally complete (including [3] and those listed in [9]) they are at their core visualizations of relational algebra operators. This applies even to the more abstract *graph data structures (GDS)* from [8] and the later *graph model (GM)* from [10]. The key difference is that GDS and GM are formulated inductively based on mappings onto operators of relational algebra. They thus mirror dataflow-type languages where visual symbols (directed hyperedges) represent operators like *set difference* connecting two relational symbols, leading to a new third symbol as output.

DataPlay [1] uses a nested universal relation data model and allows a user to compose their query by interactively modifying a *query tree with quantifiers* and observing changes in the matching/non-matching data. *Visual SQL* [26] is a visual query language that also support query visualization. With its focus on query specification, it maintains the one-to-one correspondence to SQL, and syntactic variants of the same query lead to different representations. Similarly, *SQLVis* [35] places a strong focus on the actual syntax of a SQL query and syntactic variants like nested EXISTS queries change the visualization. *GraphSQL* [11] uses visual metaphors that are different from typical relational schema notations and visualizations, even simple conjunctive queries can look unfamiliar. The *Query Graph Model (QGM)* developed for Starburst [22] helps users understand query plans, not query intent. QueryVis (earlier QueryViz) [4, 15, 18, 32] borrows the idea of a "default reading order" from diagrammatic reasoning systems [17] and uses *arrows* to indicate an implicit reading order between different nesting levels. Without the arrows, there would be no natural order placed on the existential quantifiers and the visualization would be ambiguous. QueryVis focuses on the non-disjunctive fragment of relational calculus and is guaranteed to represent connected nested queries unambiguously up to nesting level 3. Relational

Diagrams [21] is a more recent variant that indicates the nesting structure of table variables by using *nested negated bounding boxes* (instead of arrows) inspired by Peirce's influence beta existential graphs [37, 43, 45]. Interestingly, because Relational Diagrams are based on Tuple Relational Calculus (instead of Domain Relational Calculus which is closer to First-Order Logic) they solve interpretation problems of beta graphs that have been the focus of intense research in the diagrammatic reasoning communities.

4 AUTHOR INFORMATION

Wolfgang Gatterbauer is an Associate Professor at the Khoury College of Computer Sciences at Northeastern University. His research interests lie in the intersection of theory and practice of data management. With co-authors he got the EDBT 2021 best paper award, "best of conference" mentions for PODS 2021, SIGMOD 2017, WALCOM 2017, and VLDB 2015, and two SIGMOD 2021 reproducibility awards. Prior to joining Northeastern, he was an Assistant Professor at Carnegie Mellon's Tepper School of Business, and before that a PostDoc at University of Washington. He received his PhD in Computer Science at Vienna University of Technology.

ACKNOWLEDGMENTS

This work was supported in part by NSF under award numbers IIS-1762268 and IIS-1956096.

REFERENCES

[1] Azza Abouzied, Joseph M. Hellerstein, and Avi Silberschatz. 2012. DataPlay: interactive tweaking and example-driven correction of graphical database queries. In *UIST*. 207–218. https://doi.org/10.1145/2380116.2380144
[2] Active Query Builder. 2019. https://www.activequerybuilder.com/.
[3] Eirik Bakke and David R. Karger. 2016. Expressive Query Construction through Direct Manipulation of Nested Relational Results. In *SIGMOD*. ACM, 1377–1392. https://doi.org/10.1145/2882903.2915210
[4] Sara Di Bartolomeo, Mirek Riedewald, Wolfgang Gatterbauer, and Cody Dunne. 2022. STRATISFIMAL LAYOUT: A modular optimization model for laying out layered node-link network visualizations. *IEEE Transactions on Visualization and Computer Graphics* 28, 1 (2022), 324–334. https://doi.org/10.1109/TVCG.2021.3114756, Full version: https://osf.io/qdyt9.
[5] Leilani Battle, Danyel Fisher, Robert DeLine, Mike Barnett, Badrish Chandramouli, and Jonathan Goldstein. 2016. Making Sense of Temporal Queries with Interactive Visualization. In *CHI*. 5433–5443. 10.1145/2858036.2858408
[6] Sourav S. Bhowmick and Byron Choi. 2022. Data-driven Visual Query Interfaces for Graphs: Past, Present, and (Near) Future. In *SIGMOD*. 2441–2447. https://doi.org/10.1145/3514221.3522562
[7] Sourav S. Bhowmick, Byron Choi, and Chengkai Li. 2017. Graph Querying Meets HCI: State of the Art and Future Directions. In *SIGMOD*. 1731–1736. https://doi.org/10.1145/3035918.3054774
[8] Tiziana Catarci. 1991. On the Expressive Power of Graphical Query Languages. In *Visual Database Systems, II. Proceedings of the IFIP TC2/WG 2.6 Second Working Conference on Visual Database Systems. (IFIP Transactions)*, Vol. A-7. North-Holland, 411–421. https://dblp.org/rec/conf/vdb/Catarci91
[9] Tiziana Catarci, Maria Francesca Costabile, Stefano Levialdi, and Carlo Batini. 1997. Visual Query Systems for Databases: A Survey. *Journal of Visual Languages and Computing* 8, 2 (1997), 215–260. https://doi.org/10.1006/jvlc.1997.0037 https://doi.org/10.1006/jvlc.1997.0037.
[10] Tiziana Catarci, Giuseppe Santucci, and Michele Angelaccio. 1993. Fundamental Graphical Primitives for Visual Query Languages. *Inf. Syst.* 18, 2 (1993), 75–98. https://doi.org/10.1016/0306-4379(93)90006-M
[11] Claudio Cerullo and Marco Porta. 2007. A System for Database Visual Querying and Query Visualization: Complementing Text and Graphics to Increase Expressiveness. In *DEXA*. IEEE, 109–113. https://doi.org/10.1109/DEXA.2007.91
[12] Hock Chuan Chan, Kwok Kee Wei, and Keng Leng Siau. 1993. User-Database Interface: The Effect of Abstraction Levels on Query Performance. *MIS Quarterly* 17, 4 (1993), 441–464. https://doi.org/10.2307/249587
[13] Gard J. Clark and C. Thomas Wu. 1994. DFQL: Dataflow query language for relational databases. *Information & Management* 27, 1 (1994), 1–15. https://doi.org/10.1016/0378-7206(94)90098-1
[14] Michael Correll and Michael Gleicher. 2016. The semantics of sketch: Flexibility in visual query systems for time series data. In *VAST*. 131–140. https://doi.org/10.1109/VAST.2016.7883519
[15] Jonathan Danaparamita and Wolfgang Gatterbauer. 2011. QueryViz: Helping Users Understand SQL queries and their patterns. In *EDBT*. 558–561. https://doi.org/10.1145/1951365.1951440, https://queryvis.com/.
[16] dbForge. 2019. https://www.devart.com/dbforge/mysql/querybuilder/.
[17] Andrew Fish and John Howse. 2004. Towards a Default Reading for Constraint Diagrams. In *International Conference on Theory and Application of Diagrams (DIAGRAMS)*. Springer, 51–65. https://doi.org/10.1007/978-3-540-25931-2_8
[18] Wolfgang Gatterbauer. 2011. Databases will Visualize Queries too. *PVLDB* 4, 12 (2011), 1498–1501. https://doi.org/10.14778/3402755.3402805, http://www.vldb.org/pvldb/vol4/p1498-gatterbauer.pdf, http://www.youtube.com/watch?v=kVFnQRGAQls.
[19] Wolfgang Gatterbauer. 2022. Interpreting and understanding relational database queries using diagrams. International Conference on Theory and Application of Diagrams (DIAGRAMS) – Tutorials. http://www.diagrams-conference.org/2022/index.php/program/tutorials/.
[20] Wolfgang Gatterbauer, Cody Dunne, H.V. Jagadish, and Mirek Riedewald. 2022. Principles of Query Visualization. *Bulletin of the Technical Committee on Data Engineering (DEBull)* 45, 3 (2022), 47–67. http://sites.computer.org/debull/A22sept/p47.pdf
[21] Wolfgang Gatterbauer, Cody Dunne, and Mirek Riedewald. 2022. Relational Diagrams: a pattern-preserving diagrammatic representation of non-disjunctive Relational Queries. *Arxiv preprint arXiv:2203.07284* (2022). https://arxiv.org/abs/2203.07284, https://relationaldiagrams.com.
[22] Laura M. Haas, Johann Christoph Freytag, Guy M. Lohman, and Hamid Pirahesh. 1989. Extensible Query Processing in Starburst. *SIGMOD Record* 18, 2 (1989), 377–388. https://doi.org/10.1145/67544.66962
[23] Elie C. Harel and Ephraim R. McLean. 1985. The Effects of Using a Nonprocedural Computer Language on Programmer Productivity. *MIS Quarterly* 9, 2 (jun 1985), 109–120. https://doi.org/10.2307/249112
[24] David Hibert and Wilhelm Ackermann. 1928. *Grundzüge der theoretischen Logik. By.* Berlin, J. Springer. https://doi.org/10.2307/2018808
[25] John Howse. 2008. Diagrammatic Reasoning Systems. In *International Conference on Conceptual Structures (ICCS) (LNCS)*, Vol. 5113. Springer, 1–20. https://doi.org/10.1007/978-3-540-70596-3_1
[26] Hannu Jaakkola and Bernhard Thalheim. 2003. Visual SQL – High-Quality ER-Based Query Treatment. In *Workshops @ International Conference on Conceptual Modeling (ER)*. 129–139. https://doi.org/10.1007/978-3-540-39597-3_13
[27] Matthias Jarke and Yannis Vassiliou. 1985. A Framework for Choosing a Database Query Language. *Comput. Surveys* 17, 3 (1985), 313–340. https://doi.org/10.1145/5505.5506
[28] Gordon L. Kindlmann and Carlos Eduardo Scheidegger. 2014. An Algebraic Process for Visualization Design. *IEEE Trans. Vis. Comput. Graph.* 20, 12 (2014), 2181–2190. https://doi.org/10.1109/TVCG.2014.2346325
[29] Christian J. W. Kloesel, Max H. Fisch, Nathan Houser, Ursula Niklas, Marc Simon, Don D. Roberts, and Aleta Houser (Eds.). 1989. *Writings of Charles S. Peirce: A Chronological Edition, Volume 4: 1879–1884.* Indiana University Press. http://www.jstor.org/stable/j.ctt16gz8j1
[30] Doris Jung Lin Lee, John Lee, Tarique Siddiqui, Jaewoo Kim, Karrie Karahalios, and Aditya G. Parameswaran. 2020. You can't always sketch what you want: Understanding Sensemaking in Visual Query Systems. *IEEE Trans. Vis. Comput. Graph.* 26, 1 (2020), 1267–1277. https://doi.org/10.1109/TVCG.2019.2934666
[31] John Leggett and Glen Williams. 1984. An empirical investigation of voice as an input modality for computer programming. *International Journal of Man-Machine Studies* 21, 6 (1984), 493–520. https://doi.org/10.1016/S0020-7373(84)80057-7
[32] 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 *SIGMOD*. 2303–2318. https://doi.org/10.1145/3318464.3389767, https://queryvis.com/, Full version: https://osf.io/btszh/.
[33] Miro Mannino and Azza Abouzied. 2018. Expressive Time Series Querying with Hand-Drawn Scale-Free Sketches. In *CHI*. 388. https://doi.org/10.1145/3173574.3173962
[34] Microsoft Access. 2019. https://products.office.com/en-us/access.
[35] Daphne Miedema and George Fletcher. 2021. SQLVis: Visual Query Representations for Supporting SQL Learners. In *VL/HCC*. 1–9. https://doi.org/10.1109/VL/HCC51201.2021.9576431
[36] Arnab Nandi, Lilong Jiang, and Michael Mandel. 2013. Gestural Query Specification. *PVLDB* 7, 4 (2013), 289–300. https://doi.org/10.14778/2732240.2732247
[37] Charles Sanders Peirce. 1933. Collected Papers of Charles Sanders Peirce. Vol. 4. *The ANNALS of the American Academy of Political and Social Science* (1933). https://doi.org/10.1177/000271623417400185
[38] pgAdmin. 2019. https://www.pgadmin.org/.
[39] QueryScope. 2019. https://sqldep.com/.
[40] Raghu Ramakrishnan and Johannes Gehrke. 2000. *Database management systems* (2nd ed.). McGraw-Hill. https://dl.acm.org/doi/book/10.5555/556863
[41] Phyllis Reisner. 1981. Human Factors Studies of Database Query Languages: A Survey and Assessment. *Comput. Surveys* 13, 1 (1981), 13–31. https://doi.org/10.1145/356835.356837
[42] Phyllis Reisner, Raymond F. Boyce, and Donald D. Chamberlin. 1975. Human Factors Evaluation of Two Data Base Query Languages: Square and Sequel. In *Proceedings of the May 19-22, 1975, national computer conference and exposition (AFIPS)*. ACM, 447–452. https://doi.org/10.1145/1499949.1500036
[43] Don D. Roberts. 1992. The existential graphs. *Computers & Mathematics with Applications* 23, 6 (1992), 639–663. https://doi.org/10.1016/0898-1221(92)90127-4
[44] Sun-Joo Shin. 1995. *The Logical Status of Diagrams.* Cambridge University Press. https://doi.org/10.1017/CBO9780511574696
[45] Sun-Joo Shin. 2002. *The Iconic Logic of Peirce's Graphs.* The MIT Press. https://doi.org/10.7551/mitpress/3633.001.0001
[46] SQL Server Management Studio. 2019. https://www.microsoft.com/en-us/sql-server/sql-server-downloads.
[47] Nan Tang, Eugene Wu, and Guoliang Li. 2019. Towards Democratizing Relational Data Visualization. In *SIGMOD*. 2025–2030. https://doi.org/10.1145/3299869.3314029
[48] Charles Welty and David W. Stemple. 1981. Human Factors Comparison of a Procedural and a Nonprocedural Query Language. *ACM Transactions on Database Systems (TODS)* 6, 4 (1981), 626–649. https://doi.org/10.1145/319628.319656
[49] M.Y.-M. Yen and R.W. Scamell. 1993. A human factors experimental comparison of SQL and QBE. *IEEE Transactions on Software Engineering* 19, 4 (1993), 390–409. https://doi.org/10.1109/32.223806
[50] Moshé M. Zloof. 1977. Query-by-Example: A Data Base Language. *IBM Systems Journal* 16, 4 (1977), 324–343. https://doi.org/10.1147/sj.164.0324