

A Comprehensive Tutorial on over 100 Years of Diagrammatic Representations of Logical Statements and Relational Queries

Wolfgang Gatterbauer
Northeastern University
Boston, USA
 0000-0002-9614-0504

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 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 *diagrammatic representations of logical statements and relational expressions*, across both the relational database and the much older diagrammatic reasoning communities. We 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 investigations into diagrammatic languages, and organize the landscape by mapping the visual alphabets of diagrammatic representation systems to the syntax and semantics of Relational Algebra (RA) and Relational Calculus (RC). Tutorial website: <https://northeastern-datalab.github.io/diagrammatic-representation-tutorial/>

I. 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 [10], [12], [30], [35], [40], [52], [53], [64], [65]. A complementary and much-less-studied problem is that of helping users *read and understand existing queries*. With the proliferation of public data sources and associated queries, users increasingly have to read other people’s queries and scripts. At the same time, Large Language Models (LLMs) have become an effective way to generate “starter code” (including SQL queries [22], [49]) which still has to be checked for correctness and undergo refinement. Some even predict that “*all programs in the future will ultimately be written by AIs, with humans relegated to, at best, a supervisory role*” [63]. But while it is easier to modify a “starter query” than to write something from scratch, reading code is hard, and SQL is no exception. Any “human-AI-database interaction” relying on starter queries still requires users to understand written queries. For that reason alone, it is an opportune moment to study ways that help users understand queries, and visualization is one obvious route. While visual methods for composing queries have been studied

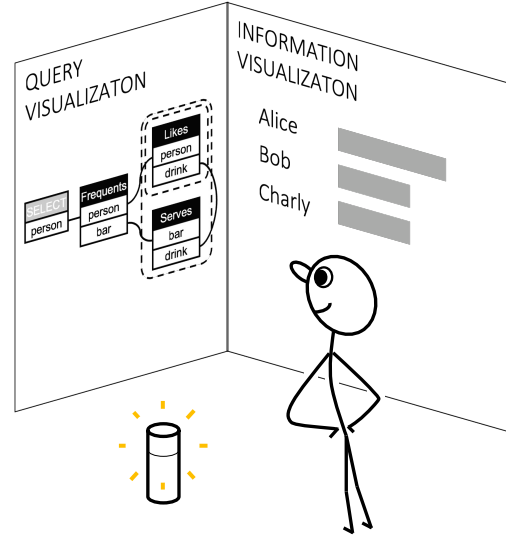


Fig. 1: An analyst dictates a query to her voice assistant which then shows the query as understood together with the query answers.

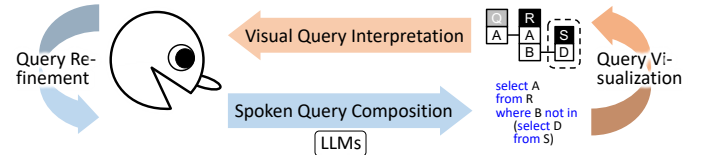


Fig. 2: The future user interaction with relational databases: a user dictates queries, and the system visualizes the queries back for the user to verify the correct interpretation.

extensively in the database literature under the topic of Visual Query Languages (VQLs) [10], the challenges for supporting the *reverse functionality* of automatically creating a visual representation of an existing query (“Query Visualization”) are different than the problem of composing a new query.

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

Outline. The lecture-style 3-hour tutorial consists of six 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 query visualization (see Figs. 1 and 2).

(2) *Principles of query visualization:* We discuss several proposed principles of query visualization [23], [27], rephrased in the terminology of “Algebraic Visualization Design” [37]. We use these principles as guides when later discussing different diagrammatic representations.

(3) *Logical foundations of relational query languages:* We discuss the logical foundations of relational query languages. We show a few example queries over the sailors-reserve-boats database in different relational query languages, which we later use when discussing and comparing visual representations.

(4) *Early diagrammatic representations:* Diagrammatic representations for logical statements were developed well before relational databases. We discuss several visual formalisms developed over more than 100 years of work on diagrammatic reasoning systems, in particular the influential beta existential graphs by Peirce [47] and their connection to the much later developed Domain Relational Calculus (DRC).

(5) *Modern visual query representations:* We use the earlier introduced queries over the sailors-reserve-boats database to discuss the main families of visual representations for relational queries proposed by the database community.

(6) *Lessons learned and open challenges:* We extract insights from our survey and discuss open challenges.

II. TUTORIAL INFORMATION

Audience and prerequisite. This 180 min tutorial targets researchers and practitioners who desire an intuitive, yet comprehensive survey of *diagrammatic representations of logical statements and relational queries*. Our focus is on the commonalities and differences between major design ideas. 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 short-paced summary of the characteristics of 5 relational query 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), 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 fast-paced overview of RA and RC and their connections to first-order logic.

Out-of-scope. The tutorial does not cover domain-specific visualizations, such as those for geographic information systems, time-series, and spatio-temporal data [14], [39], [42]. Neither does it cover dynamic interaction with queries or data [46].

Related other tutorials. A tutorial at SIGMOD’19 [59] (“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 [8] (“Graph Querying Meets HCI”) and SIGMOD’22 [7] (“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).

Contrast to prior offerings of this tutorial. A 90-min tutorial on the topic was presented at the “International Conference on the Theory and Application of Diagrams 2022” (DIAGRAMS-22) [24], the main international venue covering all aspects of research on the theory and application of diagrams. This conference attracts an audience with close to no intersection with the audience at database conferences. The emphasis of that tutorial was on the logical foundations of relational databases, the resulting different focus from the diagrammatic reasoning community, and the problems arising from visualizing logical disjunctions as diagrams. A 90-min version of the proposed tutorial was presented at VLDB 2023 [25]. The emphasis of that tutorial was on the principles guiding effective Query Visualization, and an overview of representations suggested by the database community. The tutorial is available as a 300-page slide deck on the tutorial web page [25].

The key novel parts of this 3-h tutorial will be as follows: ❶ Part 4 includes a comprehensive survey of early diagrammatic approaches for representing logical statements that largely predate attempts in the database community and are seemingly disconnected to them. We will cover Euler circles [18], Venn diagrams [62], Venn-Peirce diagrams [47], constraint diagrams [28], [36], Peirce’s beta existential graphs, and Sowa’s conceptual graphs [57]. We also discuss formalisms embodied by Higraphs [29] and UML notation [20]. ❷ Part 5 compares past approaches in the database community. It was previously (at VLDB 2023) only covered partially due to the limited time. This tutorial includes an additional query focusing on diagrammatic representations for disjunctions and unions, which are known to be the greatest challenge for diagrammatic representations (see e.g., discussion by Shin [55]). We will also cover two additional very recent diagrammatic formalisms. Furthermore, the historical comparisons (parts 4 and 5 together) will culminate in a new “lessons learned” synthesis (part 6). ❸ The new material grows the expected number of slides to over 400 pages. Slides (and possibly videos) of the tutorial will be made available afterward on the tutorial web page, similar to other recent tutorials by the presenter and collaborators on unrelated topics [60], [61].

III. TUTORIAL CONTENT

A. Part 1: Why visualizing queries and why now?

We give 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*. 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 [10], and many commercial database products offer some visual interface for users to write simple conjunctive queries. In parallel, there is a centuries-old history of the study of formal diagrammatic reasoning systems [33] with the goal of helping humans to reason in terms of logical statements.

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 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 either unchanged and still done in text, or alternatively replaced with natural language (NL) interfaces to personal assistants and learned models (Figs. 1 and 2). This composition is then augmented and *complemented with a visual interaction that helps interpretation and verification of correctness* [23].

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 [27]): The term query visualization refers to both (i) a graphical representation of a query (alternatively, “*query diagram*”) and (ii) the process of transforming a query into a graphical representation (alternatively, “*query diagramming*”). The goal of query visualization is to help users more quickly understand the intent of a query, as well as its relational query pattern.

B. Part 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 several earlier proposed principles of query visualization [23], [27], which are newly organized, extended, and rephrased in the terminology of “Algebraic Visualization Design” [37]. One important “correspondence principle” relies on a recently proposed notion of “**relational query pattern**” [26]. While we call them “principles”, they are not meant to be irrevocable axioms, but rather intuitive objectives, whose formulation helps us develop a shared vocabulary for later discussing the trade-offs among various visualizations. We also include them in order to spark a healthy debate during and after the tutorial.

C. Part 3: Logical foundations of relational query languages

We give a brief overview of the logical foundations of relational query languages by discussing 5 queries over a variant of the sailors-reserve-boats database from the “cow book” [51]. We use a consistent notation and give the queries in 5 textual query languages: **SQL**, **Domain Relational Calculus (DRC)**, **Tuple Relational Calculus (TRC)**, non-recursive **Datalog** with negation, and **Relational Algebra (RA)**. We use these queries and textual languages later in parts 4 and 5 where we establish the mappings between various visual formalisms and these 5 queries. By using a consistent set of queries throughout our survey we can give a unified comparison of visual alphabets and their “pattern expressiveness”. Our focus is on expressiveness equivalent to First-Order Logic (FOL), which allows us to make the connection to a century of research on formalisms for diagrammatic reasoning.

D. Part 4: Early diagrammatic representations

A query in Relational Calculus (RC) is a logical formula with free variables and as such a specialization of First-Order Logic (FOL). A *logical statement* has no free variables and is basically the same as a *Boolean query* that returns a truth value of TRUE or FALSE. Diagrammatic representations for logical statements [33] 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” [32].

An influential diagrammatic notation is the **Existential Graph (EG)** notation by Charles Sanders Peirce [47], [54], [56], who wrote on graphical logic as early as 1882 [38]. 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* (simple closed curves) to express negation, 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 Domain Relational Calculus (DRC). As we show, this imperfection has led to a lot of follow-up and confusion in various works on Peirce’s existential graphs.

We also cover **Euler circles** [18], **Venn diagrams** [62], and **Venn-Peirce diagrams** [47], following mainly the exposition by Shin [55]. We discuss **constraint diagrams** [28], [36], Sowa’s **conceptual graphs** [57], and formalisms embodied by **Higraphs** [29] and **UML notation** [20]. We may or may not cover **Frege’s two-dimensional conceptual notation** [21].

E. Part 4: 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 [10]) is that this tutorial shows original figures by using a consistent schema (the sailor-boat-database from the “cow book” [51]) 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) [66] is an influential early VQL that was influenced by DRC. QBE can express relational division by breaking the query into two logical steps and using a temporary relation [51, Ch. 6.9]. In doing so, QBE uses a query pattern from Datalog of implementing relational division (or universal quantification) in a dataflow-type, sequential manner, requiring multiple occurrences of the same table. We compare queries in QBE against equivalent Datalog queries and ask whether QBE is really more “visual” than Datalog.

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 [66]) 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 have a visual formalism 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 such as SQL Server Management Studio (SSMS) [58], Active Query Builder [2], QueryScope from SQLdep [50], MS Access [43], and PostgreSQL’s pgAdmin3 [48] lacks 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 [10], [13] by mapping its visual symbols to the operators of relational algebra. Following the same procedurality as RA, DFQL expresses the data flow 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 those listed in [10]) they are at their core visualizations of relational algebra operators.

Query By Diagram (QBD) [3], [4], [11] is based on an ER (Entity-Relationship) model of the data. **TableTalk** [17] visualizes the flow of a query top-down and displays logical conditions in tiles. **Object-Oriented VQL** [45] adds existential and universal quantifiers to attributes.

Visual SQL [34] is a visual query language that also supports 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** [44] places a strong focus on the actual syntax of SQL queries, and syntactic variants like nested EXISTS change the visualization.

QueryVis (earlier *QueryViz*) [6], [15], [23], [41] borrows the idea of a “default reading order” from diagrammatic reasoning systems [19] 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.

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.

SIEUFERD [5] is a direct manipulation spreadsheet-like interface that lets users manipulate the actual data. In SIEUFERD, a result header encodes “the structure” of the query. The query result is listed below that header.

String diagrams [9], [31] are essentially a variant of Peirce’s beta graphs that allow free variables in addition to bound variables. Both types of variables are represented by lines, yet bound “variable lines” end in a dot.

Relational Diagrams [26] are a recent variant inspired by QueryVis that indicates the nesting structure of table variables by using *nested negated bounding boxes* (instead of arrows) inspired by Peirce’s beta existential graphs. 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 Peirce’s beta graphs that have been the focus of intense research in the diagrammatic reasoning community.

F. Part 6: Lessons Learned and Open Challenges

By extracting and synthesizing insights from our survey, we give design choices that avoid ambiguities resulting from overloading the meaning of lines as geometric marks (dubbed “the 3 abuses of the line”), and discuss open challenges.

IV. 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. He received an NSF Career award and – with his students and collaborators – a best paper award at EDBT 2021, best-of-conference mentions for PODS 2021, SIGMOD 2017, WALCOM 2017, and VLDB 2015, and two reproducibility awards for papers from SIGMOD 2020.

A. Acknowledgements

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

REFERENCES

- [1] A. Abouzied, J. M. Hellerstein, and A. Silberschatz, “Dataplay: interactive tweaking and example-driven correction of graphical database queries,” in *UIST*, 2012, pp. 207–218, <https://doi.org/10.1145/2380116.2380144>.
- [2] Active Query Builder, <https://www.activequerybuilder.com/>, 2019.
- [3] M. Angelaccio, T. Catarci, and G. Santucci, “QBD*: A graphical query language with recursion,” *IEEE Transactions on Software Engineering (TSE)*, vol. 16, no. 10, pp. 1150–1163, 1990, <https://doi.org/10.1109/32.60295>.
- [4] —, “Query by diagram: A fully visual query system,” *Elsevier Journal of Visual Languages & Computing*, vol. 1, no. 3, pp. 255–273, 1990, [https://doi.org/10.1016/S1045-926X\(05\)80009-6](https://doi.org/10.1016/S1045-926X(05)80009-6).
- [5] E. Bakke and D. R. Karger, “Expressive query construction through direct manipulation of nested relational results,” in *SIGMOD*, 2016, pp. 1377–1392, <https://doi.org/10.1145/2882903.2915210>.
- [6] S. D. Bartolomeo, M. Riedewald, W. Gatterbauer, and C. Dunne, “STRATISFIMAL LAYOUT: A modular optimization model for laying out layered node-link network visualizations,” *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, vol. 28, no. 1, pp. 324–334, 2022, <https://doi.org/10.1109/TVCG.2021.3114756>, Full version: <https://osf.io/qdyt9>.
- [7] S. S. Bhowmick and B. Choi, “Data-driven visual query interfaces for graphs: Past, present, and (near) future,” in *SIGMOD*, 2022, pp. 2441–2447, <https://doi.org/10.1145/3514221.3522562>.
- [8] S. S. Bhowmick, B. Choi, and C. Li, “Graph querying meets HCI: state of the art and future directions,” in *SIGMOD*, 2017, pp. 1731–1736, <https://doi.org/10.1145/3035918.3054774>.
- [9] F. Bonchi, A. D. Giorgio, N. Haydon, and P. Sobocinski, “Diagrammatic algebra of first order logic,” *arXiv:2401.07055*, 2024, <https://arxiv.org/abs/2401.07055>.
- [10] T. Catarci, M. F. Costabile, S. Levialdi, and C. Batini, “Visual query systems for databases: A survey,” *Elsevier Journal of Visual Languages & Computing*, vol. 8, no. 2, pp. 215–260, 1997, <https://doi.org/10.1006/jvlc.1997.0037>.
- [11] T. Catarci and G. Santucci, “Query by diagram: A graphical environment for querying databases,” in *SIGMOD*, 1994, p. 515, <https://doi.org/10.1145/191839.191976>.
- [12] H. C. Chan, K. K. Wei, and K. L. Siau, “User-database interface: The effect of abstraction levels on query performance,” *MIS Quarterly*, vol. 17, no. 4, pp. 441–464, 1993, <https://doi.org/10.2307/249587>.
- [13] G. J. Clark and C. T. Wu, “DFQL: Dataflow query language for relational databases,” *Information & Management*, vol. 27, no. 1, pp. 1–15, 1994, [https://doi.org/10.1016/0378-7206\(94\)90098-1](https://doi.org/10.1016/0378-7206(94)90098-1).
- [14] M. Correll and M. Gleicher, “The semantics of sketch: Flexibility in visual query systems for time series data,” in *VAST*, 2016, pp. 131–140, <https://doi.org/10.1109/VAST.2016.7883519>.
- [15] J. Danaparamita and W. Gatterbauer, “Queryviz: Helping users understand SQL queries and their patterns,” in *EDBT*, 2011, pp. 558–561, <https://doi.org/10.1145/1951365.1951440>, <https://queryvis.com/>.
- [16] dbForge, <https://www.devart.com/dbforge/mysql/querybuilder/>, 2019.
- [17] R. G. Epstein, “The tabletalk query language,” *Elsevier Journal of Visual Languages & Computing*, vol. 2, no. 2, pp. 115–141, 1991, [https://doi.org/10.1016/S1045-926X\(05\)80026-6](https://doi.org/10.1016/S1045-926X(05)80026-6).
- [18] L. Euler, *Letters of Euler to a German princess, on different subjects in physics and philosophy addressed to a German Princess, Letters 103-106*, 2nd ed. Translated by Henry Hunter, 1802, <https://archive.org/details/letterseulertoa00eulegoog/page/396/mode/2up>.
- [19] A. Fish and J. Howse, “Towards a default reading for constraint diagrams,” in *3rd International Conference on Theory and Application of Diagrams (DIAGRAMS)*. Springer, 2004, pp. 51–65, https://doi.org/10.1007/978-3-540-25931-2_8.
- [20] M. Fowler, *UML Distilled: A Brief Guide to the Standard Object Modeling Language*, 3rd ed. Addison-Wesley Longman, 2003, <https://dl.acm.org/doi/10.5555/861282>.
- [21] G. Frege, *Begriffsschrift, eine der arithmetischen nachgebildete Formelsprache des reinen Denkens*. Verlage von Louis Nebert, 1879, <https://gdz.sub.uni-goettingen.de/download/pdf/PPN538957069/PPN538957069.pdf>.
- [22] D. Gao, H. Wang, Y. Li, X. Sun, Y. Qian, B. Ding, and J. Zhou, “Text-to-SQL empowered by large language models: A benchmark evaluation,” *arXiv:2308.15363*, 2023, <https://doi.org/10.48550/arXiv.2308.15363>.
- [23] W. Gatterbauer, “Databases will visualize queries too,” *PVLDB*, vol. 4, no. 12, pp. 1498–1501, 2011, <https://doi.org/10.14778/3402755.3402805>.
- [24] —, “Interpreting and understanding relational database queries using diagrams,” *International Conference on Theory and Application of Diagrams (DIAGRAMS) – Tutorials*, 2022, <http://www.diagrams-conference.org/2022/index.php/program/tutorials/>.
- [25] —, “A tutorial on visual representations of relational queries,” *PVLDB*, vol. 16, no. 12, pp. 3890–3893, 2023, <https://doi.org/10.14778/3611540.3611578>, <https://northeastern-datalab.github.io/visual-query-representation-tutorial/>, <https://northeastern-datalab.github.io/visual-query-representation-tutorial/slides/VLDB2023-VisualRepresentationsOfRelationalQueries.pdf>.
- [26] W. Gatterbauer and C. Dunne, “On the reasonable effectiveness of Relational Diagrams: Explaining relational query patterns and the pattern expressiveness of relational languages,” *PACMMOD (SIGMOD’24)*, vol. 2, no. 1, pp. 61:1–61:27, 2024, <https://doi.org/10.1145/3639316>, <https://relationaldiagrams.com>, Full version: <https://arxiv.org/pdf/2401.04758>.
- [27] W. Gatterbauer, C. Dunne, H. Jagadish, and M. Riedewald, “Principles of query visualization,” *Bulletin of the Technical Committee on Data Engineering (DEBull)*, vol. 45, no. 3, pp. 47–67, 2022, <http://sites.computer.org/debull/A22sept/p47.pdf>.
- [28] J. Y. Gil, J. Howse, and S. Kent, “Constraint diagrams: A step beyond UML,” in *Proc. of the Technology of Object-Oriented Languages and Systems (TOOLS)*, 1999, p. 453, <https://doi.ieeecomputersociety.org/10.1109/TOOLS.1999.10066>.
- [29] D. Harel, “On visual formalisms,” *Communications of the ACM (CACM)*, vol. 31, no. 5, pp. 514–530, 1988, <https://doi.org/10.1145/42411.42414>.
- [30] E. C. Harel and E. R. McLean, “The effects of using a nonprocedural computer language on programmer productivity,” *MIS Quarterly*, vol. 9, no. 2, pp. 109–120, jun 1985, <https://doi.org/10.2307/249112>.
- [31] N. Haydon and P. Sobocinski, “Compositional diagrammatic first-order logic,” in *11th International Conference on the Theory and Application of Diagrams (DIAGRAMS)*, ser. LNCS, vol. 12169. Springer, 2020, pp. 402–418, https://doi.org/10.1007/978-3-030-54249-8_32.
- [32] D. Hilbert and W. Ackermann, *Grundzüge der theoretischen Logik*. By. Berlin, J. Springer, 1928, <https://doi.org/10.2307/2018808>.
- [33] J. Howse, “Diagrammatic reasoning systems,” in *International Conference on Conceptual Structures (ICCS)*, ser. LNCS, vol. 5113. Springer, 2008, pp. 1–20, https://doi.org/10.1007/978-3-540-70596-3_1.
- [34] H. Jaakkola and B. Thalheim, “Visual SQL – high-quality er-based query treatment,” in *Workshops @ International Conference on Conceptual Modeling (ER)*, 2003, pp. 129–139, https://doi.org/10.1007/978-3-540-39597-3_13.
- [35] M. Jarke and Y. Vassiliou, “A framework for choosing a database query language,” *ACM Computing Surveys (CSUR)*, vol. 17, no. 3, pp. 313–340, 1985, <https://doi.org/10.1145/5505.5506>.
- [36] S. Kent, “Constraint diagrams: Visualizing invariants in object-oriented models,” *SIGPLAN Not.*, vol. 32, no. 10, pp. 327–341, oct 1997, <https://doi.org/10.1145/263700.263756>.
- [37] G. L. Kindlmann and C. E. Scheidegger, “An algebraic process for visualization design,” *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, vol. 20, no. 12, pp. 2181–2190, 2014, <https://doi.org/10.1109/TVCG.2014.2346325>.
- [38] C. J. W. Kloesel, M. H. Fisch, N. Houser, U. Niklas, M. Simon, D. D. Roberts, and A. Houser, Eds., *Writings of Charles S. Peirce: A Chronological Edition, Volume 4: 1879–1884*. Indiana University Press, 1989, <http://www.jstor.org/stable/j.ctt16gz8j1>.
- [39] D. J. L. Lee, J. Lee, T. Siddiqui, J. Kim, K. Karahalios, and A. G. Parameswaran, “You can’t always sketch what you want: Understanding sensemaking in visual query systems,” *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, vol. 26, no. 1, pp. 1267–1277, 2020, <https://doi.org/10.1109/TVCG.2019.2934666>.
- [40] J. Leggett and G. Williams, “An empirical investigation of voice as an input modality for computer programming,” *International Journal of Man-Machine Studies*, vol. 21, no. 6, pp. 493–520, 1984, [https://doi.org/10.1016/S0020-7373\(84\)80057-7](https://doi.org/10.1016/S0020-7373(84)80057-7).
- [41] A. Leventidis, J. Zhang, C. Dunne, W. Gatterbauer, H. V. Jagadish, and M. Riedewald, “Queryvis: Logic-based diagrams help users understand complicated SQL queries faster,” in *SIGMOD*, 2020, pp. 2303–2318, <https://doi.org/10.1145/3318464.3389767>, <https://queryvis.com/>, Full version: <https://osf.io/btszh/>.
- [42] M. Mannino and A. Abouzied, “Expressive time series querying with hand-drawn scale-free sketches,” in *CHI*, 2018, p. 388, <https://doi.org/10.1145/3173574.3173962>.
- [43] Microsoft Access, <https://products.office.com/en-us/access>, 2019.
- [44] D. Miedema and G. Fletcher, “SQLVis: Visual query representations for supporting SQL learners,” in *VL/HCC*, 2021, pp. 1–9, <https://doi.org/10.1109/VL/HCC51201.2021.9576431>.
- [45] L. Mohan and R. L. Kashyap, “A visual query language for graphical interaction with schema-intensive databases,” *EEE Transactions on Knowledge and Data Engineering (TKDE)*, vol. 5, no. 5, pp. 843–858, 1993, <https://doi.org/10.1109/69.243513>.
- [46] A. Nandi, L. Jiang, and M. Mandel, “Gestural query specification,” *PVLDB*, vol. 7, no. 4, pp. 289–300, 2013, <https://doi.org/10.14778/2732240.2732247>.
- [47] C. S. Peirce, “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>.
- [48] pgAdmin, <https://www.pgadmin.org/>, 2019.
- [49] B. Qin, B. Hui, L. Wang, M. Yang, J. Li, B. Li, R. Geng, R. Cao, J. Sun, L. Si, F. Huang, and Y. Li, “A survey on text-to-SQL parsing: Concepts, methods, and future directions,” *arXiv:2208.13629*, 2022, <https://doi.org/10.48550/arXiv.2208.13629>.
- [50] QueryScope, <https://sqldep.com/>, 2019.
- [51] R. Ramakrishnan and J. Gehrke, *Database management systems*, 2nd ed. McGraw-Hill, 2000, <https://dl.acm.org/doi/book/10.5555/556863>.
- [52] P. Reisner, “Human factors studies of database query languages: A survey and assessment,” *ACM Computing Surveys (CSUR)*, vol. 13, no. 1, pp. 13–31, 1981, <https://doi.org/10.1145/356835.356837>.
- [53] P. Reisner, R. F. Boyce, and D. D. Chamberlin, “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, 1975, pp. 447–452, <https://doi.org/10.1145/1499949.1500036>.
- [54] D. D. Roberts, “The existential graphs,” *Computers & Mathematics with Applications*, vol. 23, no. 6, pp. 639–663, 1992, [https://doi.org/10.1016/0898-1221\(92\)90127-4](https://doi.org/10.1016/0898-1221(92)90127-4).

- [55] S.-J. Shin, *The Logical Status of Diagrams*. Cambridge University Press, 1995, <https://doi.org/10.1017/CBO9780511574696>.
- [56] —, *The Iconic Logic of Peirce's Graphs*. The MIT Press, 2002, <https://doi.org/10.7551/mitpress/3633.001.0001>.
- [57] J. F. Sowa, "Conceptual graphs for a data base interface," *IBM Journal of Research and Development*, vol. 20, pp. 336–357, Jul. 1976, <https://doi.org/10.1147/rd.204.0336>.
- [58] SQL Server Management Studio, <https://www.microsoft.com/en-us/sql-server/sql-server-downloads>, 2019.
- [59] N. Tang, E. Wu, and G. Li, "Towards democratizing relational data visualization," in *SIGMOD*, 2019, pp. 2025–2030, <https://doi.org/10.1145/3299869.3314029>.
- [60] N. Tziavelis, W. Gatterbauer, and M. Riedewald, "Optimal join algorithms meet topk," in *SIGMOD*, 2020, pp. 2659–2665, <https://doi.org/10.1145/3318464.3383132>, <https://northeastern-datalab.github.io/topk-join-tutorial/>.
- [61] —, "Toward responsive DBMS: optimal join algorithms, enumeration, factorization, ranking, and dynamic programming," in *ICDE*, 2022, pp. 3205–3208, <https://doi.org/10.1109/ICDE53745.2022.00299>, <https://northeastern-datalab.github.io/responsive-dbms-tutorial/>.
- [62] J. Venn, "I. on the diagrammatic and mechanical representation of propositions and reasonings," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 10, no. 59, pp. 1–18, 1880, <https://doi.org/10.1080/14786448008626877>.
- [63] M. Welsh, "The end of programming," *Communications of the ACM (CACM)*, vol. 66, no. 1, pp. 34–35, dec 2022, <https://doi.org/10.1145/3570220>.
- [64] C. Welty and D. W. Stemple, "Human factors comparison of a procedural and a nonprocedural query language," *ACM Transactions on Database Systems (TODS)*, vol. 6, no. 4, pp. 626–649, 1981, <https://doi.org/10.1145/319628.319656>.
- [65] M.-M. Yen and R. Scamell, "A human factors experimental comparison of SQL and QBE," *IEEE Transactions on Software Engineering (TSE)*, vol. 19, no. 4, pp. 390–409, 1993, <https://doi.org/10.1109/32.223806>.
- [66] M. M. Zloof, "Query-by-example: A data base language," *IBM Systems Journal*, vol. 16, no. 4, pp. 324–343, 1977, <https://doi.org/10.1147/sj.164.0324>.