Research Statement

Anna Fariha

1 Research Overview

The recent growth of data science expanded its reach to an ever-growing user base of non-experts, increasing the need for *democratization* and *transparency* in these systems. Democratization demands that a system can be used by people with different skills and backgrounds alike. Transparency requires explainability that helps the users understand and *trust* the system function, especially when unexpected behavior occurs. Unfortunately, most existing data systems offer limited usability and support for explanations: these systems are usable only by experts with sound technical skills, and even expert users are hindered by the lack of transparency into the inner workings and function of the systems. The aim of my research is to bridge the usability gap between non-expert users and complex data systems, and explain system behavior towards achieving trust while using these systems.

Research summary. Broadly, my research has three goals: (1) enhancing usability of data systems for non-experts, (2) explaining system behavior to enable understanding of unexpected outcomes, and (3) assisting users to achieve trust in data-driven systems. In my dissertation research, I developed usability-enhancing systems—following the *programming-by-example* [16] paradigm—in two different settings: querying relational databases [10] and personalized document summarization [8]. I developed two *explanation frameworks*—based on the principles of *causality*—to explain causes of data non-conformance in the context of trusted machine learning [13], and to explain root cause of a concurrent application's intermittent failure [11]. Finally, as a mechanism to achieve trust in data-driven machine learning, I developed a new data-profiling primitive called *conformance constraints* [14] that can characterize tuples for which a machine-learned model is likely to produce untrustworthy predictions. While my dissertation research establishes the foundation towards building accessible and transparent systems, the ultimate goal of my research is to expand these systems to support a larger and diverse class of problems towards a broader impact on the society.

Broader impact and recognition. While my primary area of research revolves around data management [10–12, 14], the application areas of my research have been interdisciplinary—spanning from program synthesis [10, 12] and software engineering [11] to machine learning [13, 14, 18, 22], natural-language processing [8], data mining [19, 27, 29, 30, 32, 33], and human-computer interaction [9]. The outcome of my research led to a number of top-tier publications, and has been recognized by several awards, including Microsoft research dissertation grant 2020 [1] and VLDB 2020 best demonstration runner-up award [2].

2 Research Accomplishments

2.1 Enhancing Usability of Data Systems

The proliferation of computational resources and data-sharing platforms has reached an ever-growing base of users without technical computing expertise, who wish to peruse, analyze, and understand data. The broad availability of data has the potential to fundamentally impact the way domain experts conduct their work. Unfortunately, while data is broadly available, data systems are seldom unfettered. Existing systems typically cater to users with sound technical computing and programming skills, posing significant hurdles to technical novices, who do not have strong technical background. Democratization of computational systems demands equal access to people of different skills and backgrounds. Therefore, the first goal of my research is to design and develop tools and mechanisms that can enhance usability of data systems.

While using a data system, the traditional setting requires the users to accurately specify the intended task in a language that the system understands. For example, a user has to compose a precise SQL query to retrieve data from a relational database, issue a well-formed natural-language query to extract relevant data from a large document in a question-answering system, or write a complex sampling script to sample data from a large dataset. Such rigid mechanisms pose hurdles for non-experts who lack technical expertise and are unfamiliar with the details of the data organization. However, a number of data-centric tasks can be simply expressed by *examples*. Such a *programming-by-example* interaction significantly enhances usability of these systems for non-experts. I investigate example-driven techniques to enhance usability of different data systems.

Example-driven query intent discovery. To bridge the usability gap between non-expert users and traditional relational data interfaces, I developed SQuID [10, 12], an *example-driven query intent discovery* framework. SQuID offers an alternative querying mechanism that effectively complements the traditional SQL-driven querying. It is an end-to-end system for semantic-similarity-aware query intent discovery: given a few examples of the desired result, SQuID automatically formulates a SQL query—with appropriate selection predicates and join conditions—that captures the user intent. The key idea is to express the problem of query intent discovery using a *probabilistic abduction model* that infers a query as the most likely explanation of the provided examples. An extensive empirical evaluation on three real-world datasets, including user-intent case studies, demonstrates that SQuID is efficient and effective, and outperforms machine learning methods [7], as well as the state of the art in the related query reverse engineering problem [34].

Comparative user studies. To understand how effective SQuID is for real users over real-world data exploration tasks from an HCI perspective, I further conducted comparative user studies contrasting SQuID with traditional SQL querying [9]. The result of the user studies demonstrates that users with varying expertise are significantly more effective and efficient with SQuID than SQL. The study also shows that SQuID eliminates the barriers in studying the database schema, formalizing task semantics, and writing syntactically-correct SQL queries, and, thus, substantially alleviates the need for technical expertise in data exploration.

Personalized document summarization. Inspired by SQuID's success, I applied the mechanism for example-driven user intent discovery to the area of text document summarization, which refers to the task of producing a brief representation of a document for easy human consumption. Automatic text document summarization is a key natural-language processing task. Existing text summarization techniques mostly focus on generic summarization, but users often require *personalized* summarization that targets their specific preferences. However, precisely expressing preferences is challenging, and current methods are often outside the user's control, or require costly training data. To this end, I developed SuDocu [8]—an *example-based personalized document summarization system*—which offers a novel and effective way to express summarization preferences via examples. In SuDocu, the user provides a few example summaries for a small number of documents in a collection, SuDocu learns the summarization intent from the examples, and produces summaries for new documents that reflect the user's summarization preferences. Under the hood, SuDocu combines topic modeling, semantic-similarity discovery [10], and in-database optimization [5] in a novel way to achieve efficient and scalable example-driven document summarization.

2.2 Explanation Frameworks

When data systems do not behave as expected (e.g., when we encounter tuples on which the predictions of a machine-learned model are untrustworthy), we need mechanisms that can help us understand what is causing such unexpected behavior: we need explanation tools. Furthermore, application development in distributed settings brings forth important challenges that also need explanation. For example, failures in distributed applications are often triggered by concurrency bugs, such as interference and coordination issues. Such concurrency bugs are difficult to reproduce; therefore, the root causes of concurrent program failures are hard to identify, and even harder to explain. Without succinct explanation of how the root cause eventually leads to failure, the developer might fail to draw the causal connection from the root cause to the failure. The second goal of my research is to develop explanation frameworks to facilitate understanding of outcomes involving data and data systems.

Explaining data non-conformance. In data-driven systems, we often encounter tuples on which the predictions of a machine-learned model are untrustworthy. A key cause of such untrustworthiness is *non-conformance* of a new tuple with respect to the training dataset. Conformance constraints [14] can detect whether a serving tuple is non-conforming; however, they do not provide interpretable explanations of non-conformance. To this end, I developed ExTuNe [13], a system for *explaining* causes of tuple non-conformance. Based on the principles of causality, ExTuNe assigns *responsibility* to the attributes for causing non-conformance. The key idea is to observe change in constraint violation under *intervention* on attribute-values. ExTuNe produces a ranked list of the test tuples based on their degree of non-conformance and visualizes tuple-level attribute responsibility for non-conformance through heat maps. Further, it visualizes attribute responsibility, aggregated over the test tuples. Empirically, ExTuNe is found to be able to effectively detect and explain tuple non-conformance, which can assist the users to make careful decisions towards achieving trusted machine learning.

Causality-guided adaptive interventional debugging. Runtime nondeterminism is ubiquitous in modern database applications. Previous research has shown that nondeterminism can cause applications to intermittently crash, become unresponsive, or experience data corruption. Debugging and understanding the root cause

of such intermittent failures have always posed a challenge even to the most expert developers. To this end, I developed Adaptive Interventional Debugging (AID) [11] for debugging such intermittent failures. AID combines existing statistical debugging [21], causal analysis [26], fault injection [3], and group testing [17] techniques in a novel way to pinpoint the root cause of an application's intermittent failure and to generate an explanation of how the root cause triggers the failure. AID works by first identifying a set of runtime behaviors (called predicates) that are strongly correlated to the failure. It then utilizes temporal properties of the predicates to over-approximate their causal relationships. Finally, it uses fault injection to execute a sequence of interventions on the predicates and discover their true causal relationships. AID was evaluated with six real-world applications—including some proprietary Microsoft applications—that intermittently fail under specific inputs. In each case, AID was able to identify the root cause and explain how the root cause triggered the failure, much faster than group testing and more precisely than statistical debugging.

2.3 Trust and Fairness in Machine Learning

Data is central to modern systems in a wide range of domains, including healthcare, transportation, and finance. The core of modern data-driven systems comprises of models learned from large datasets, and they are usually optimized to target particular data and workloads. While these data-driven systems have seen wide adoption and success, their reliability and proper function hinge on the data's continued conformance to the systems' initial settings and assumptions. If the serving data (on which the system operates) deviates from the profile of the initial data (on which the system was trained), then system performance degrades and system behavior becomes unreliable. A mechanism to assess the trustworthiness of a system's inferences is paramount, especially for systems that perform safety-critical or high-impact operations. Since data is central to machine learning, it can guide us to determine when a prediction made by a machine-learned model can be trusted and when not. Furthermore, data plays a crucial role towards fairness of the learned model: if the data contains bias, the model will also learn to discriminate. The third question that my research aims to answer is how to achieve *trust* in predictions made by machine-learned models and how to ensure *fairness* in machine learning.

Conformance constraints for trusted machine learning. The problem of *Trusted Machine Learning* refers to quantifying trust in the prediction made by a machine-learned model on a previously unseen input tuple. To solve the problem, I developed *conformance constraints* [14]: a new data-profiling primitive tailored towards quantifying the degree of non-conformance of a tuple with respect to the training data that can effectively characterize if inference over that tuple is untrustworthy. Conformance constraints are constraints over certain arithmetic expressions (called projections) involving the numerical attributes of a dataset, which existing data-profiling primitives such as functional dependencies [25] and denial constraints [6] cannot model. The key finding is that projections that incur low variance on a dataset construct effective conformance constraints. This principle yields the surprising result that low-variance components of a principal component analysis, which are usually discarded for dimensionality reduction, generate stronger conformance constraints than the high-variance components. Beyond trusted machine learning, conformance constraints can also quantify data drift more accurately than the state of the art.

Benchmarking fair classification approaches. Predictions made by classifiers are often used to make high-stake decisions such as mortgage approval, job-applicant filtering, pre-trial risk assessment of criminals, etc. However, similar to all data-driven approaches, classifiers also suffer from the general phenomenon of "garbage-in, garbage-out": if the data contains some inherent bias, the model will also reflect or even exacerbate it. While classifiers generally try to maximize their correctness, correctness alone is insufficient to characterize the desirable qualities of a classifier; we need to ensure that the classifier is *fair* as well. In recent years, fairness in classification has emerged as a multidisciplinary field, receiving significant interest from both the machine learning and the data management communities. It is important for research communities, application developers, and other stakeholders alike to understand the distinction among the available fair approaches, especially because the choice of a fair approach affects performance issues such as the correctness-fairness trade-off, efficiency, scalability, etc. To this end, I worked on a project where the goal was to empirically evaluate a total of 18 fair classification approaches over four real-world datasets from five important aspects: correctness, fairness, efficiency, scalability, and stability [18].

3 Future Research Agenda

I intend to continue working on building tools that will ease the interactions between different user groups—end users, data scientists, and developers—and systems, using the techniques of data management, program synthesis, and machine learning. Specifically, my goal is to aid users build data-driven systems and help them reason about both data and system functionalities. This motivation guides my future research directions.

3.1 Facilitating End Users

In continuation with my prior work, the first type of users I aim to facilitate are the end users. To this end, I aim to develop systems that provide the end users insights about their data.

Data change summarization. Data is everywhere, and, thus, the task of data exploration and understanding is of paramount importance. Existing tools for data summarization enable users to interactively understand the database content [20]. However, the fact—"change is the only constant in life"—is also true for databases. Therefore, we need a mechanism to understand change in databases effectively and efficiently as well. Whenever there is a notion of change, there are always accompanying questions regarding mechanism, quantification, and cause of the change. When data has changed, the next questions are: *how*, *how much*, and *why*. To answer these questions, I envision a technique that will explain changes in two snapshots of a database, usually associated with two different time-stamps or generated by two different mechanisms from a parent database. My key observation here is that change in databases are often systematic and is rarely random. My goal is to produce *explanation of change*—a set of interpretable transformations—–that will ensure that all or most of the changes are sufficiently explained and the explanation is of reasonable complexity for human consumption. In other words, my goal is to explain how two databases differ *semantically*.

3.2 Facilitating Data Scientists

Data science is a key area in today's era of machine learning and artificial intelligence. Therefore, the second type of users I aim to facilitate are the data scientists. I want to help the data scientists understand the disconnect between data and systems, and, also, help them write codes within the machine-learning pipeline.

Exposing disconnect between data and systems. Due to the recent growth of data-driven systems, data is now one of the key components of these systems. We often observe that certain properties of the data cause these systems to malfunction, primarily because the systems were not designed to handle those data properties in the first place. Hence, similar to software debugging that aims to find bugs in the mechanism (source code or runtime conditions), we must also debug data to identify the *disconnect* between the assumptions over the data and the design of the systems that operate on that data. Specifically, we need to find out what *properties* of the data cause a data-driven system to malfunction. To this end, I aim to build a technique that can expose certain data properties, or *profiles*, as root causes for a system's performance degradation. This identification, subsequently, will provide a clear guideline towards fixing the design of the system that operates on the data. The key technique I envision is *causal reasoning* through *intervention*—a mechanism that alters the profiles of a dataset for which the system malfunctions—and subsequent *observation* of the change in the system's behavior due to the intervention. Unlike statistical observational analysis, which reports mere correlations, such a technique will report causally-verified root causes—in terms of data profiles—of the system malfunction.

Example-driven data sampling. Evaluating ML pipelines with large training datasets is time-consuming and inefficient, and, thus, data scientists use samples instead. However, skewed samples—generated by incorrect sampling scripts—may provide misleading insights about the data and may lead to unexpected performance anomalies when a model learned on that sample is deployed over the entire data. Unfortunately, correctly writing sampling scripts is a challenging task. I aim to develop an *example-driven sampling* technique, where the users will express their sampling criteria via example-driven interactions, and the system will synthesize the desired sampling script. Such a system will alleviate the burden of writing complex sampling scripts for data scientists.

3.3 Facilitating Developers

The third type of users I want to help are the developers. To this end, I envision two kinds of tools: the first one involves multimodal interaction for program synthesis, and the second one involves automatic program repair to expedite debugging.

Data integration by program synthesis over a dual-specification model. The information required for a data-analysis task is often spread across multiple heterogeneous data sources. For example, information regarding different aspects of a product (technical specification, price, user ratings and reviews, etc.) may lie across different web pages and databases. To make effective use of such data, we need to consolidate them into meaningful information. To achieve this, one needs to design *data-integration* pipelines and write codes that can blend data from disparate sources. Data integration is a key data-management task that involves data querying [10], data extraction [28], data transformation [31], semantic-type detection, entity matching, etc. It is the very first step for data analytics and must be completed before downstream analysis steps. However, writing scripts for components of the data-integration pipeline is mundane, tedious, and often repetitive. Thus, I aim to investigate ways to automatically synthesize scripts for data-integration programs. The key observation here is that data can give us significant hints about the programs that operate on the data, allowing more insights about how to connect the user intents to the desired data-integration programs.

Beyond traditional single-interaction models, I aim to exploit *context-aware conversational interaction* along with *example* or *demonstration*. Such a dual-specification model [4] would significantly advance the state of the art in program synthesis that relies on single-interaction models, which involve either natural-language interaction (NLI) or by-example interaction. Further, a key advantage here is that for data-integration tasks, such a paradigm can exploit two additional source of information—*context* and the *data*—beyond the user-provided specifications (NL and examples). The ultimate goal of such a dual-specification model is to synthesize the desired data-integration programs, where the developers can specify their data-integration intents through a series of natural-language interactions (conversational) and a few examples or demonstrations. Towards solving this problem, my idea is to exploit neural networks to first generate a *sketch* of the program, and, then, input-output examples can guide to fill out the holes of the sketch to concretize it to a valid program.

Support in debugging: automatic program repair. Some studies claim that developers spend 75% of their time on debugging, which makes debugging challenges an industry-wide issue. Debugging involves repairing a program by augmenting it with code segments or *patches*. Recently, automatic program repair has been an active area of research both in academia [15, 24] and in industry [23]. I aim take a step further towards building automatic program repair mechanisms based on four types of information: (1) syntactic and semantic analysis of the buggy program, (2) test cases (input-output examples), provided either by the developer or generated automatically, (3) knowledge of prior debugging solutions collected from historical debugging data, and (4) the techniques of interventional causality which will guarantee that applying the synthesized patches will remove the bugs. My goal is to design automated program repair tools that are scalable and efficient—allowing interactive speed—which will significantly boost the developers' productivity.

4 Service and Outreach

I am a program committee member of SIGMOD 2021 (research track) and EDBT 2021 (demonstration track). I also served as an external reviewer at SIGMOD 2018, 2019, and 2020, and CHI 2021. I advised several Ph.D. and masters students during my Ph.D. In a *Research Experience for Undergraduates (REU)* program at UMass, I mentored a junior undergraduate student. I also volunteered in the *Girls Inc. Eureka! Summer Workshop*—a STEM-intensive program for 8th–12th grade girls.

5 Conclusions

My research targets fundamental problems in data platforms and analytics: example-driven techniques complement traditional task specification mechanisms in various data platforms; trust and fairness in machine learning boost reliability and acceptability; and explanation frameworks offer understanding of complex data systems. I firmly believe that the outcome of my research can result into industry-standard products to make data systems more accessible and transparent to a much broader set of users in all data platforms, and, thus, provide a significant boost in human productivity, system transparency, and trustworthiness in AI.

References

[1] Dissertation Grant - Microsoft Research. `https://www.microsoft.com/en-us/research/academic-program/dissertation-grant/#!grant-recipients`, Nov 2020.

[2] VLDB Awards - VLDB 2020 Tokyo. `https://vldb2020.org/vldb-2020-awards.html`, Nov 2020.

[3] Peter Alvaro, Joshua Rosen, and Joseph M. Hellerstein. Lineage-driven fault injection. In *SIGMOD*, pages 331–346, 2015.

[4] Christopher Baik, Zhongjun Jin, Michael J. Cafarella, and H. V. Jagadish. Duoquest: A dual-specification system for expressive SQL queries. In *SIGMOD*, pages 2319–2329, 2020.

[5] Matteo Brucato, Juan Felipe Beltran, Azza Abouzied, and Alexandra Meliou. Scalable package queries in relational database systems. *PVLDB*, 9(7):576–587, 2016.

[6] Xu Chu, Ihab F. Ilyas, and Paolo Papotti. Discovering denial constraints. *PVLDB*, 6(13):1498–1509, 2013.

[7] Charles Elkan and Keith Noto. Learning classifiers from only positive and unlabeled data. In *SIGKDD*, pages 213–220. ACM, 2008.

[8] **Anna Fariha**, Matteo Brucato, Peter J. Haas, and Alexandra Meliou. SuDocu: Summarizing Documents by Example. *PVLDB*, 13(12):2861–2864, 2020. [VLDB Best Demonstration Runner-up Award]

[9] **Anna Fariha**, Lucy Cousins, Narges Mahyar, and Alexandra Meliou. Example-driven User Intent Discovery: Empowering Users to Cross the SQL Barrier Through Query by Example. *CoRR*, abs/2012.14800, 2020.

[10] **Anna Fariha** and Alexandra Meliou. Example-Driven Query Intent Discovery: Abductive Reasoning using Semantic Similarity. *PVLDB*, 12(11):1262–1275, 2019.

[11] **Anna Fariha**, Suman Nath, and Alexandra Meliou. Causality-Guided Adaptive Interventional Debugging. In *SIGMOD*, pages 431–446, 2020.

[12] **Anna Fariha**, Sheikh Muhammad Sarwar, and Alexandra Meliou. SQuID: Semantic Similarity-Aware Query Intent Discovery. In *SIGMOD*, pages 1745–1748, 2018.

[13] **Anna Fariha**, Ashish Tiwari, Arjun Radhakrishna, and Sumit Gulwani. ExTuNe: Explaining Tuple Non-conformance. In *SIGMOD*, pages 2741–2744, 2020.

[14] **Anna Fariha**, Ashish Tiwari, Arjun Radhakrishna, Sumit Gulwani, and Alexandra Meliou. Conformance Constraint Discovery: Measuring Trust in Data-Driven Systems. In *SIGMOD (to appear)*, 2021. `http://people.cs.umass.edu/afariha/papers/ConformanceConstraints.pdf`.

[15] Claire Le Goues, Michael Pradel, and Abhik Roychoudhury. Automated program repair. *Communications of the ACM*, 62(12):56–65, 2019.

[16] Sumit Gulwani and Prateek Jain. Programming by examples: PL meets ML. In *APLAS*, pages 3–20, 2017.

[17] F. K. Hwang. A method for detecting all defective members in a population by group testing. *Journal of the American Statistical Association*, 67(339):605–608, 1972.

[18] Maliha Tashfia Islam, **Anna Fariha**, and Alexandra Meliou. A Data Management Perspective on Fair Classification: An Experimental Analysis and Evaluation. *(under review in VLDB)*, 2021. `https://people.cs.umass.edu/afariha/papers/FairnessAnalysis.pdf`.

[19] Shariful Islam, **Anna Fariha**, Chowdhury Farhan Ahmed, and Byeong-Soo Jeong. EGDIM: evolving graph database indexing method. In *ICUIMC*, pages 56:1–56:10, 2012.

[20] Manas Joglekar, Hector Garcia-Molina, and Aditya G. Parameswaran. Interactive data exploration with smart drill-down. *IEEE Trans. Knowl. Data Eng.*, 31(1):46–60, 2019.

[21] Ben Liblit, Mayur Naik, Alice X. Zheng, Alexander Aiken, and Michael I. Jordan. Scalable Statistical Bug Isolation. In *PLDI*, pages 15–26, 2005.

[22] Amit Mandal, Mehedi Hasan, **Anna Fariha**, and Chowdhury Farhan Ahmed. GSCS - graph stream classification with side information. In *APWeb*, pages 389–400, 2015.

[23] Alexandru Marginean, Johannes Bader, Satish Chandra, Mark Harman, Yue Jia, Ke Mao, Alexander Mols, and Andrew Scott. Sapfix: Automated end-to-end repair at scale. In *ICSE-SEIP*, pages 269–278. IEEE, 2019.

[24] Sergey Mechtaev, Jooyong Yi, and Abhik Roychoudhury. Angelix: Scalable multiline program patch synthesis via symbolic analysis. In *ICSE*, pages 691–701, 2016.

[25] Thorsten Papenbrock, Jens Ehrlich, Jannik Marten, Tommy Neubert, Jan-Peer Rudolph, Martin Schönberg, Jakob Zwiener, and Felix Naumann. Functional dependency discovery: An experimental evaluation of seven algorithms. *PVLDB*, 8(10):1082–1093, 2015.

[26] Judea Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, New York, NY, USA, 2000.

[27] Quazi Marufur Rahman, **Anna Fariha**, Amit Mandal, Chowdhury Farhan Ahmed, and Carson K. Leung. A Sliding Window-Based Algorithm for Detecting Leaders from Social Network Action Streams. In *IEEE/WIC/ACM WI-IAT*, pages 133–136, 2015.

[28] Mohammad Raza and Sumit Gulwani. Web data extraction using hybrid program synthesis: A combination of top-down and bottom-up inference. In *SIGMOD*, pages 1967–1978, 2020.

[29] Md. Samiullah, Chowdhury Farhan Ahmed, Manziba Akanda Nishi, **Anna Fariha**, S. M. Abdullah, and Md. Rafiqul Islam. Correlation mining in graph databases with a new measure. In *APWeb*, pages 88–95, 2013.

[30] Md. Samiullah, Chowdhury Farhan Ahmed, **Anna Fariha**, Md. Rafiqul Islam, and Nicolas Lachiche. Mining frequent correlated graphs with a new measure. *Expert Syst. Appl.*, 41(4):1847–1863, 2014.

[31] Rishabh Singh and Sumit Gulwani. Transforming spreadsheet data types using examples. In Rastislav Bodík and Rupak Majumdar, editors, *POPL*, pages 343–356. ACM, 2016.

[32] **Anna Fariha**, Chowdhury Farhan Ahmed, Carson K. Leung, Md. Samiullah, Suraiya Pervin, and Longbing Cao. A new framework for mining frequent interaction patterns from meeting databases. *Eng. Appl. Artif. Intell.*, 45:103–118, 2015.

[33] **Anna Fariha**, Chowdhury Farhan Ahmed, Carson Kai-Sang Leung, S. M. Abdullah, and Longbing Cao. Mining frequent patterns from human interactions in meetings using directed acyclic graphs. In *PAKDD*, pages 38–49, 2013.

[34] Quoc Trung Tran, Chee Yong Chan, and Srinivasan Parthasarathy. Query reverse engineering. *VLDB J.*, 23(5):721–746, 2014.