Hybrid Query and Instance Explanations and Repairs

Seokki Lee
University of Cincinnati
lee5sk@uc.edu

Boris Glavic
Illinois Institute of Technology
bglavic@iit.edu

Adriane Chapman
University of Southampton
Adriane.Chapman@soton.ac.uk

Bertram Ludäscher
University of Illinois, Urbana-Champaign
ludaesch@illinois.edu

ABSTRACT

Prior work on explaining missing (unexpected) query results identifies which parts of the query or data are responsible for the erroneous result or repairs the query or data to fix such errors. The problem of generating repairs is typically expressed as an optimization problem, i.e., a single repair is returned that is optimal wrt. to some criterion such as minimizing the repair's side effects. However, such an optimization objective may not concretely model a user's (often hard to formalize) notion of which repair is "correct". In this paper, we motivate hybrid explanations and repairs, i.e., that fix both the query and the data. Instead of returning one "optimal" repair, we argue for an approach that empowers the user to explore the space of possible repairs effectively. We also present a proof-of-concept implementation and outline open research problems.

ACM Reference Format:
Seokki Lee, Boris Glavic, Adriane Chapman, and Bertram Ludäscher. 2023. Hybrid Query and Instance Explanations and Repairs. In *Companion Proceedings of the ACM Web Conference 2023 (WWW '23 Companion), April 30-May 4, 2023, Austin, TX, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3543873.3587565

1 INTRODUCTION

Both why and why-not provenance (see [11] for a survey) have been widely used for explaining unexpected and missing query results. Explanations for missing answers typically fall into two categories: (i) instance-based explanations [12, 16] (the input database is considered the only source of error) and (ii) query-based explanations [4, 6] (explaining which parts of a query are problematic). Explanations aid users in understanding how errors in the data and query cause their queries to return an incorrect result. However, a user may also be interested in possible solutions for fixing such problems. How-to queries [18] compute an update to the input data that achieves the desired changes to the result of a query that maximizes a user-provided objective. Other work has studied the orthogonal problem of repairing the query instead, e.g., to return less spurious answers [19], to fix an empty query answer [15], or to construct a query from scratch that returns a desired result [13].

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
WWW '23 Companion, April 30-May 4, 2023, Austin, TX, USA
© 2023 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-9419-2/23/04…$15.00
https://doi.org/10.1145/3543873.3587565

However, both the query and input data may be erroneous and, thus, there is a need for *hybrid* explanations and repairs which consider both the input data as well as the query as potential causes for erroneous query results. Such hybrid repairs have been largely ignored in past work. Furthermore, most approaches model the problem of generating repairs as an optimization problem and return a single optimal (or approximately optimal) repair. Examples of optimization objectives that have been used are minimizing the side-effects on the query result (the updated query result minimally differs from the original query result), minimizing the changes to the query that are required to fix the user's complaints, or optimizing for a user-provided objective [18]. The advantage of such an approach is that the process of generating repairs can be fully automated. However, this only works if the objective of the optimization problem precisely models what constitutes a correct explanation / repair. Even though some systems allow the user to specify the objective, the user may not know enough about the unknown ground truth repair to specify an effective objective.

Example 1. *Consider the income database and Datalog query shown in Fig. 1. This query determines low-income residents that are underpaid for the type of work they are doing (earn less pre-tax income than a job-specific threshold). The user issuing this query wants to understand why no CA carpenters in TX teachers are in the result (highlighted tuples in* Q*'s result). In this example, both errors in the data as well as the query are responsible for the missing answers:*[1] *(i) the query uses pre-tax income, omitting several low-income individuals and (ii) the tax for TX residents only includes state income tax (which is 0% in TX) instead of federal plus state tax.*

Data-based explanations for missing answers [17] justify why the query did fail to return TX teachers and CA carpenters based on missing input data. How-to queries [18] can be used to determine a possible repair of the input database such that the expected result is returned by the query. However, data-based explanations and repairs are oblivious to errors with the query itself. Query-based explanations [6, 8] and query repairs [20] can identify issues with the query and propose repairs such as subtracting both federal and state taxes from the income ($I - T < 90,000$) or changing the threshold in the predicate $I < 90,000$. However, query-based explanations and repairs fail to recognize errors in the data as causes for the incorrect result. In summary, a hybrid of data- and query-based explanations (and repairs) is needed to address the missing answers in this example. Unfortunately, existing techniques cannot easily be combined as query and instance repairs can interfere. Furthermore,

[1] Here we focus on expected, but missing answers, but the same arguments apply to unexpected answers which should be removed.

residents

Name	State	Income	Jobtitle	Tax
Peter	IL	45,000	teacher	8,400
Alice	IL	253,000	engineer	69,300
Bob	CA	183,000	carpenter	73,000
Fred	TX	90,000	teacher	0
Fred	TX	120,000	carpenter	0

minincome

Job	Threshold
teacher	55,000
engineer	100,000
carpenter	90,000

$r_1 : \mathtt{Q}(S, J) :- \mathtt{Q_{lowin}}(S, J, I), \mathtt{minincome}(J, T), I < T$

$r_2 : \mathtt{Q_{lowin}}(S, J, I) :- \mathtt{residents}(N, S, I, J, T),$ I < 90,000

Q

State	Jobtitle
IL	teacher
CA	carpenter
TX	teacher

Figure 1: Query that determines which jobs are underpaid in a particular state (low income residents with this job earn less pre-tax income than a job-specific threshold).

even if we ignore the lack of support for hybrid explanations, identifying the correct ground truth repair requires a solid understanding of the application domain as well as the semantics of the intended query. This type of information is often hard to formalize for a user as it requires extensive knowledge about the ground truth repair. At the other end of the spectrum, simply enumerating all possible repairs is also not an option as the number of candidate solution is already too large for data-based repairs [17].

In this paper, we argue that these are fundamental problems of approaches that treat explanations and repairs as an optimization problem or enumerate all candidates. We envision an approach that combines user-directed exploration of the search space and techniques for summarizing large sets of repairs and explanations to provide guidance to the user. Furthermore, we argue that for a middle ground between enumerating all solutions and optimizing for a single solution by filtering solutions that are obviously inferior. To provide an example, we may group input data changes based on which attributes that are updated or what are common characteristics of rows that are effected, e.g., *"this class of repairs updates the tax column of rows where state is TX"*. As a proof of concept, we demonstrate how an extension of an existing approach for explaining missing and existing answers for non-recursive Datalog [16, 17] can generate hybrid explanations and how hybrid repairs can be extracted from explanations. However, as this is just a first step towards realizing our vision, we also discuss open research problems and how they relate to existing work in the field.

2 HYBRID QUERY-INSTANCE REPAIRS

We now formalize the problem of hybrid query-instance repairs. We assume as input a set Δ^+ of tuples that should occur in the result of the query, but are currently missing, as well as a set Δ^- of tuples that are currently in the query result, but should be removed. Updates can be modeled using deletion and insertion. We use set semantics here, but an extension to bag semantics should be feasible. Instead of specifying a single objective, we define thresholds on side effects for the input database, query results, and query to exclude repairs that are trivial (e.g., rewriting Q as $Q - \Delta^- \cup \Delta^+$). Furthermore, we assume a distance function d_Q for queries and d_D for instances (e.g., the size of the symmetric difference).

Definition 2.1 (Hybrid Repair). We are given as input a query Q, database D, and side effect thresholds. Given a hypothetical update to $\mathcal{A} = Q(D)$ expressed as $C = (\Delta^+, \Delta^-)$ such that $\Delta^- \subseteq \mathcal{A}$ and $\Delta^+ \cap \mathcal{A} = \emptyset$, a solution to the hybrid repair problem is a pair (Q', D') such that:

$$d_Q(Q, Q') \leq \lambda_Q \qquad d_D(D \cup \Delta^+ - \Delta^-, D') \leq \lambda_{output}$$

$$d_D(D, D') \leq \lambda_{input} \qquad \frac{|\ \Delta^- - Q'(D')\ | + |\ \Delta^+ \cap Q'(D')\ |}{|\ \Delta^+\ | + |\ \Delta^-\ |} \geq \lambda_\Delta$$

where Q' is the modified query and D' is the updated database.

Intuitively, the above definition allows for any repair where the side effects on the input database and query are within the thresholds (λ_{input} and λ_Q). Furthermore, we also limit the side effect on the query result (changes other than the ones requested by the user) by threshold λ_{output} and require that the repair implements at least a certain fraction λ_Δ of the changes Δ^- and Δ^+. By defining repairs using thresholds we can exclude trivial repairs without the pitfalls of optimizing for a single "optimal" repair: maximizing an objective function that may be a poor substitute for repair correctness.

Example 2 (Running Example Repair). *The user's complaint from Ex. 1 is* $\Delta^+ = \{(CA, carpenter), (TX, teacher)\}$ *and* $\Delta^- = \emptyset$. *For example, a possible data-based repair (assuming appropriate thresholds) is to reduce the income of teacher Fred (TX) and of carpenter Bob (CA) such that both* $I < 90,000$ *and* $I < T$ *evaluate to true for these two rows (e.g., by setting them to 42,999 and 89,999, respectively). One query-based repair is to increase the constant in the condition* $I < 90,000$ *and relax the condition* $I < T$*, e.g., by replacing it with* $I < T + c$ *for a sufficiently large constant* c.

Both repairs from the example above are minimal in terms of their result and input data (or query[2]) side-effects. The issue is not just that the correct repair for this example is a hybrid repair, but also that it does not have minimal side-effects! Furthermore, even if we optimize for a user-provided metric, it is questionable whether the user would be able to formalize a metric that would cause an optimization-based algorithm to select the correct repair as this fundamentally requires already a solid understanding of what caused the error (tax values are incorrectly recorded for TX, and the predicate in r_2 should use $I - T < 90,000$ instead of $I < 90,000$). A user that has understood the nature of the problem to that extend would likely be able to determine the correct repair without further guidance. Thus, this example motivates the need for user-guided repair search. We will expand on that in Sec. 5.

3 RELATED WORK

Data- and Instance-based Explanations for Existing and Missing Answers. Provenance has been the basis of most approaches for instanced-based [16, 21] (tracing errors to the input data) and query-based explanations [8, 12] of existing and missing answers.
Hybrid Explanation for Missing Answers. [10] presents the *Conseil* system which computes hybrid explanations for missing answers over non-monotonic queries. The instance-part of an explanation produced by Conseil encodes multiple possible repairs compactly using c-tables and supports user-provided side-constraints. However, generating repairs with Conseil is still a one-shot process instead of an interactive exploration of the search space. This requires the development of new techniques that can dynamically

[2]If we assume query side-effects are measured as the total number of modified goals and restrict repairs to modifying predicate goals which of course is only one possible meaningful choice.

react to changing constraints on allowable repairs during the user's exploration of the search space.

Query Repairs. Techniques that repair a query typically change selected operators such that the missing result is returned: query refinement targets the use case where a query returns too many (unexpected) answers [20] and query relaxation addresses the issue of an empty query result [19]. Query reverse engineering (QRE) [13] reconstructs a query based on data examples. Given a database D and the result $Q(D)$ of an unknown query Q, QRE generates a query Q' such that $Q'(D) = Q(D)$.

Instance Repairs. Repairing the input data to achieve a desired query result is an instance of the view update problem [5]. How-to queries [18] phrase this problem as an optimization problem, enabling the user to specify the objective function.

Data + Constraint Repair. The problem of hybrid repairs is also related to the problem of simultaneously repairing a set of integrity constraints and a dataset violating the constraints (e.g., see [3]).

Summarizing, Querying, and Exploring Provenance. We envision interactive exploration and querying to play an important role in exploring hybrid repairs. We expect work on filtering [7] and querying provenance [2, 14], on an interactive exploration of provenance, and on (approximate) provenance summarization [1, 17] to play an important role. However, more work is needed to support efficient interactive exploration of the search space.

4 HYBRID EXPLANATIONS AND REPAIRS

As a proof of concept, we extend our past work [16] which generates instance-based explanations for missing and existing answers of Datalog queries to support hybrid explanations and discuss how to "read-out" repairs from these explanations.

Given an input query Q and database D, a hybrid explanation HyExpl justifies the existence/absence of a result tuple t of Q based on the success/failure to derive t using the rules of Q. Such explanations are generated in the form of graphs whose nodes correspond to existing and missing tuples as well as successful/failed goals in the body of grounded Datalog rules (rules instantiated with values from the database). These graphs contain four types of nodes: *tuple nodes* (oval) represent EDB and IDB facts (tuples of the input database and tuples produced by rules, respectively); *rule nodes* (rectangle) represent a grounded rule and are labeled with an identifier for the rule (e.g., r_1) and the arguments it was instantiated with; *instance goal nodes* (superscript I) represent goals of an instantiated rule and are labeled with their arguments and an identifier for the rule and their position in the rule's body; *predicate goal nodes* (superscript P) represent comparison predicates. The color of a tuple node indicates whether the tuple exists (green) or not (dark red). The same color scheme is used to indicate success / failure of rule and goal nodes.

Similar to the explanations in PUG [16], existing IDB tuples (query result tuples) are connected to the successful grounded rules that derive the tuple while missing tuples are connected to all grounded rules that could have derived the tuple (but failed). A grounded rule succeeds if all of its goals succeeds and, thus, the success of the goals justifies the success of the rule. That is, successful rule nodes are connected to successful goal nodes for each goal of the rule. A grounded rule evaluates to false if at least one of its goal

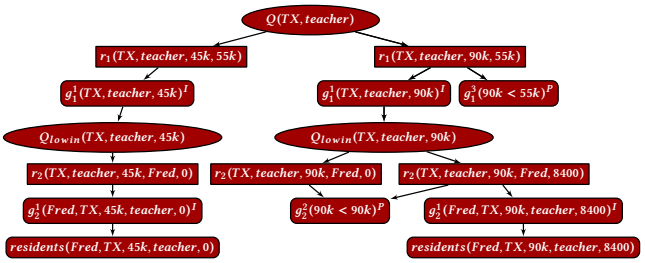

Figure 2: Partial hybrid explanations for (TX,teacher) $\in \Delta^-$

evaluates to false. Thus, the failed goals of a grounded rule justify the failure of the rule. As shown in [16], ignoring the predicate goals, this type of graphs are expressive enough to encode provenance polynomials and their extension for negation [9], but also encode information about the query structure. By adding predicate goal nodes, hybrid explanations also model how the success and failure of comparison predicates affect the success/failure of the grounded rules of a query.

Example 3. *Fig. 2 shows a partial hybrid explanation for the missing result* Q(*TX, teacher*)*, i.e., why no teachers in TX are underpaid. We show three example possible causes: (i) Data-based (left-most) - the tuple node* Q(*TX, teacher*) *connects to the failed rule grounding which derives the instance goal through* r_1*. This derivation fails based on the absence of the tuple* $(Fred, TX, 45k, teacher, 0)$ *in the relation* `residents`*; (ii) Query-based (middle) - connects the tuple node to the failed predicate goals* $g_1^3(90k < 55k)^P$ *and* $g_2^2(90k < 90k)^P$ *through the failed grounded rule* $r_1(TX, teacher, 90k, 55k)$ *and* $r_2(TX, teacher, 90k, Fred, 0)$*, respectively; (iii) Hybrid (right-most) - shows an example combination of a data-based and query-based explanation: tax should include, e.g., the federal tax for Fred that is* 8400 *(encoded in the failed tuple node* $residents(Fred, TX, 90k, teacher, 8400)$ *and the failed instance goal node* $g_2^1(Fred, TX, 90k, teacher, 8400)^I$*). Also the comparison predicates of* r_1 *and* r_2 *are too strict as shown by the failed predicate goals in the graph.*

Hybrid Repairs. We now show how to compute repairs based on a hybrid explanation. In [16], we proved that the PUG's explanations generalize the dual provenance polynomials of the semiring framework extension for first-order logic [10] by assigning a provenance token for each tuple node in the leaves and considering rule nodes as conjunctions ($\cdot$) and goal nodes and rooted tuple nodes as alternatives (+) and dealing with negation as in [9] by separate tokens for positive (existing) and negated (missing) tuples. Then, the hybrid repair problem can be recast as a constraint optimization problem by translating HyExpl to a set of linear constraints based on the expected query result. The problem is solvable using off-the-shelf solvers. However, the cost of running a constraint solver over a problem whose size is polynomial in data is in general not an option because of the high computational complexity of constraint solving. Furthermore, constraint solvers typically return only a single result and, thus, do not directly allow us to involve the user in the search for the correct repair. We will discuss possible solutions in Sec. 5.

Example 4. *Using the hybrid explanation in Ex. 3, a user can obtain valid repairs that cause* Q(*TX, teacher*) *to appear in the result of* r_1*. However, not all of these repairs are correct and meaningful. For example, the data-based explanation (left-most in Fig. 2) leads to a*

repair that inserts tuple $(Fred, TX, 45k, teacher, 0)$ *which causes* r_2 *to succeed which in turns causes* r_1 *to derive the expected result. This explanation yields a valid repair and is side-effect free. However,* $45k$ *is not the correct income for Fred. Based on the query-based explanation (middle derivation in the figure), changing the comparisons in* r_1 *and* r_2 *is a valid repair and is minimal in terms of query modifications. However, it may introduce a large number of side-effects on the query result. Importantly, both repairs do not fix the error with the missing federal tax for TX residents. The hybrid derivation (right-most) guides the user to obtain the correct repair: the instance goal that describes the tax amount should be changed as well as the comparisons of the query need to be changed. However, there are still many possible options for how to repair this part of the data and query.*

Preliminary Evaluation. We conducted a preliminary evaluation to measure the runtime in seconds for computing HyExpl using a real-world dataset[3] (varying size from 270 to 270K rows). We use a simple conjunctive query with a comparison and compute HyExpl for a particular missing result. The experiments are conducted on a machine with 2×3.30 Ghz 8 cores Intel Xeon CPU and 128GB memory. Computing HyExpl for the missing result takes about 10s of seconds. Even for this simple case for a selective comparison, the number of failed derivations is already between $3 \cdot 10^5$ and $2.2 \cdot 10^6$, demonstrating the infeasibility of enumerating all possible repairs.

5 CONCLUSIONS AND DISCUSSION

In this paper, we introduced our vision for hybrid repairs and motivated why modeling the problem as an unsupervised optimization problem is often insufficient. Moving from providing a single explanation/candidate repair to a larger selection of options, there can be a set of conflicting requirements for which an optimal solution does not exist or where it is virtually impossible to specify a metric that captures the characteristics of the ground truth repair. In this case, it is essential to provide users with the necessary tools to control the search for the correct repair by providing information about the search space and about specific repairs as well as allow users to state and update requirements such a repair should fulfill. Thus, to realize our vision, we outline several open research questions that have to be addressed in the following.

Restricting the Space of Viable Repairs. In Definition 2.1, we did only slightly restrict the space of possible repairs. However, this is obviously too general for many use cases and may still result in a very large space of repairs. Hence, one of the first challenges that needs to be addressed is to study restrictions of the search space to exclude further meaningless repairs without being too restrictive (e.g., only allowing repairs with minimal side-effects). For example, such restrictions may come in the form of requiring specific parts of the query or input database to not be modified by the repair, limiting the percentage of changed tuples for subsets of the input data, or restrict repairs to changes of predicate goals [8].

User-guided Exploration. Even when significantly restricting the search space, full exploration of the search space will not be an option. To empower the user to efficiently navigate the search space, we need effective visualizations of repairs at flexible levels of abstraction (e.g., the user may want to first see a summary of the changes that constitute a repair such as a compact description of the rows that would be modified), the ability to sort a set of repairs and the ability to change the sort order dynamically, e.g., sorting the repairs based on their side-effect size or based on a user-defined query (the data of how many high-value customers will be affected by the repair?). Furthermore, the user should be able to pose and update restrictions on the search space based on new knowledge uncovered during the search process. Such restrictions should include positive feedback (e.g., the user may identify a subset of updates made by a repair as correct and may only want to explore repairs that apply these updates) as well as negative feedback (e.g., exclude all repairs that change the income of persons).

Summarizing Repairs to Guide Exploration. In addition to presenting information about a single repair at flexible levels of details, we should also aid the user to understand sets of repairs that share common characteristics. This requires summarizing sets of repairs compactly and to compute statistics about such sets (e.g., all repairs that update the tax rate in TX will at least have x side effects). As enumerating large sets of repairs is likely not an option, this necessitates the development of approximation techniques to ensure that such statistics can be computed efficiently.

[3] https://www.kaggle.com/leomauro/argodatathon2019

REFERENCES

[1] E. Ainy, P. Bourhis, SB. Davidson, D. Deutch, and Tova Milo. 2015. Approximated Summarization of Data Provenance. In *CIKM*. 483–492.
[2] Bahareh Arab, Su Feng, Boris Glavic, Seokki Lee, Xing Niu, and Qitian Zeng. 2018. GProM - A Swiss Army Knife for Your Provenance Needs. *IEEE Data Eng. Bull.* 41, 1 (2018), 51–62.
[3] George Beskales, Ihab F. Ilyas, Lukasz Golab, and Artur Galiullin. 2013. On the relative trust between inconsistent data and inaccurate constraints. In *ICDE*. 541–552.
[4] Nicole Bidoit, Melanie Herschel, Katerina Tzompanaki, et al. 2014. Query-Based Why-Not Provenance with NedExplain. In *EDBT*. 145–156.
[5] Aaron Bohannon, Benjamin C Pierce, and Jeffrey A Vaughan. 2006. Relational lenses: a language for updatable views. In *PODS*. 338–347.
[6] Adriane Chapman and H. V. Jagadish. 2009. Why Not?. In *SIGMOD*. 523–534.
[7] Daniel Deutch, Amir Gilad, and Yuval Moskovitch. 2015. Selective Provenance for Datalog Programs Using Top-K Queries. *PVLDB* 8, 12 (2015), 1394–1405.
[8] Ralf Diestelkämper, Seokki Lee, Melanie Herschel, and Boris Glavic. 2021. To Not Miss the Forest for the Trees - A Holistic Approach for Explaining Missing Answers over Nested Data. In *SIGMOD*. 405–417.
[9] Erich Grädel and Val Tannen. 2017. Semiring Provenance for First-Order Model Checking. *arXiv preprint arXiv:1712.01980* (2017).
[10] M. Herschel. 2015. A Hybrid Approach to Answering Why-Not Questions on Relational Query Results. *JDIQ* 5, 3 (2015), 10.
[11] Melanie Herschel, Ralf Diestelkämper, and Houssem Ben Lahmar. 2017. A survey on provenance: What for? What form? What from? *VLDBJ* (2017), 1–26.
[12] M. Herschel and M. Hernandez. 2010. Explaining Missing Answers to SPJUA Queries. *PVLDB* 3, 1 (2010), 185–196.
[13] Dmitri V Kalashnikov, Laks VS Lakshmanan, and Divesh Srivastava. 2018. Fastqre: Fast query reverse engineering. In *SIGMOD*. 337–350.
[14] G. Karvounarakis, Z.G. Ives, and V. Tannen. 2010. Querying data provenance. In *SIGMOD*. 951–962.
[15] Nick Koudas, Chen Li, Anthony K. H. Tung, and Rares Vernica. 2006. Relaxing Join and Selection Queries. In *VLDB*. 199–210.
[16] Seokki Lee, Bertram Ludäscher, and Boris Glavic. 2018. PUG: a framework and practical implementation for why and why-not provenance. *VLDBJ* (2018), 1–25.
[17] Seokki Lee, Bertram Ludäscher, and Boris Glavic. 2020. Approximate Summaries for Why and Why-not Provenance. *PVLDB* 13, 6 (2020), 912–924.
[18] A. Meliou and D. Suciu. 2012. Tiresias: The database oracle for how-to queries. In *SIGMOD*. 337–348.
[19] Chaitanya Mishra and Nick Koudas. 2009. Interactive Query Refinement. In *EDBT*. 862–873.
[20] Quoc Trung Tran and Chee-Yong Chan. 2010. How to ConQueR why-not questions. In *SIGMOD*. 15–26.
[21] Jane Xu, Waley Zhang, Abdussalam Alawini, and Val Tannen. 2018. Provenance Analysis for Missing Answers and Integrity Repairs. *IEEE Data Eng. Bull.* 41, 1 (2018), 39–50.