A Formal Approach to Finding Explanations for Database Queries*

Sudeepa Roy
University of Washington
Seattle, WA, USA
sudeepa@cs.washington.edu

Dan Suciu
University of Washington
Seattle, WA, USA
suciu@cs.washington.edu

ABSTRACT

As a consequence of the popularity of *big data*, many users with a variety of backgrounds seek to extract high level information from datasets collected from various sources and combined using data integration techniques. A major challenge for research in data management is to develop tools to assist users in explaining observed query outputs. In this paper we introduce a principled approach to provide explanations for answers to SQL queries based on *intervention*: removal of tuples from the database that significantly affect the query answers. We provide a formal definition of intervention in the presence of multiple relations which can interact with each other through foreign keys. First we give a set of recursive rules to compute the intervention for any given explanation in polynomial time (data complexity). Then we give simple and efficient algorithms based on SQL queries that can compute the top-K explanations by using standard database management systems under certain conditions. We evaluate the quality and performance of our approach by experiments on real datasets.

Categories and Subject Descriptors

H.1.0 [**Models and Principles**]: General; H.2.8 [**Database Management**]: Database Applications

Keywords

Explanations; Causality; Intervention; Recursion; Data cube

1. INTRODUCTION

As a consequence of the popularity of "big data", many users with a variety of backgrounds seek to extract high level information from datasets. The typical scenario is this. A user integrates a number of data sets, computes some statistics, then seeks an explanation for what she sees. Why are these data values outliers? Or why are they *not* outliers? Why are two graphs similar (or dissimilar)? Why is this value so high (low)? A major challenge for research in both data management and AI is to develop tools to assist users in explaining observed query outputs.

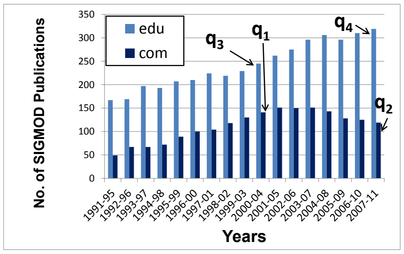


Figure 1: Number of SIGMOD publications in a five years windows, broken down into papers from industry ('com') and academia ('edu'). While both increase until 2000-2007, afterward the number of papers from academia continue to increase while that from industry decreases.

Consider the following example. Figure 1 shows the number of publications in SIGMOD during a moving five years window, broken down into papers published by authors from industry and papers published by authors from academia. The graph was generated using a three-way-join SQL query over the DBLP dataset integrated with an affiliation table[1]. Around 2000-2007 one can notice an interesting bump: while before that period both the number of academic and industrial papers increase over time, after that period the number of academic papers continues to increase, but that of industrial papers decreases. This is clearly an interesting phenomenon. For certain users (say, the director of a government funding agency), it is important to find explanations for this phenomenon, by digging deeper into the data.

The golden standard for an explanation is the *actual cause* of the observed outcome. Causality has been studied and defined algorithmically by Judea Pearl, in a highly influential work [11]. At the core, causality is defined in terms of intervention: an input is said to be a cause if we can affect the output by changing just the value of that input, while keeping all others unchanged. Causality can only be established in a controlled physical experiment, and cannot be derived solely from data. (Pearl, and others, circumvent that by assuming that causal paths are already known, allowing the

*This research was partially supported by NSF Awards IIS-0911036 and IIS-1115188.

SIGMOD'14, June 22–27, 2014, Snowbird, UT, USA.

ACM 978-1-4503-2376-5/14/06 ...$15.00.
http://dx.doi.org/10.1145/2588555.2588578.

[1] The affiliation information was available only for some authors, therefore the graph does not include all papers; papers with authors from both industry and academia are included in both bars.

rank	explanation
1	[affiliation = ibm.com]
2	[affiliation = bell-labs.com]
3	[author = Rajeev Rastogi]
4	[affiliation = ucla.edu]
5	[author = Hamid Pirahesh]
6	[affiliation = asu.edu]
7	[author = Rakesh Agrawal]
8	[affiliation = utah.edu]
9	[affiliation = gwu.edu]

Figure 2: Top explanations for the observation in Figure 1.

controlled experiment to be simulated algorithmically.) In contrast, an *explanation* lowers the bar of causality, and only requires that a change in the input affects the output [6, 9]: the more it affects the output, the better the explanation. All previous work on explanations have adopted the principle of intervention either explicitly or implicitly (for example, by using *influence* as a metric for explanation [19]).

In the context of databases, a major challenge in finding explanations is the difficulty of defining and computing interventions for complex datasets and queries. Users often examine multiple datasets, and the query is complex, involving joins and aggregates; often the question is the relationship of several queries rather than a single query. The challenge is to define an intervention, and compute all interventions in real time. Previous work on explaining data outcomes have been restricted to single table only, and to queries without joins [15, 2, 6, 19]. However, many interesting observations in data are made when we integrate multiple data sets and run complex queries, including joins. Any single dataset may be uninteresting, but once we join them we make interesting observations, which we wish to explain.

In this paper we introduce a principled approach to query explanation that applies to SQL queries. Our definition is based on intervention: an explanation is a predicate with the property that, when we remove from the database all tuples satisfying that predicate, then we affect significantly the outcome that we want to explain. For an illustration we show in Figure 2 the top explanations for the observation made in Figure 1. For example, `bell-labs` had a very active database groups in the mid 90's: if we intervene by removing `bell-labs` from the data then we flatten to some extent the bump in industrial publications in the late 90's (prior to 2000-2007) thus "explaining" the bump; similar explanations come from top researchers in industrial labs who published intensively in the 90's. A different kind of explanations are for the increase in academic publications: a large chunk comes from some highly prolific academic database groups established or increased during the 2000's, such as `asu.edu`, `utah.edu`, `gwu.edu`. Notice that all these are simply "explanations" and we do not claim to establish causality: in this particular example, the only way to prove causality would be to repeat the history. However, good explanations can go a long way in helping the user dig deeper in search for the ultimate cause[2]

In this paper we make the following contributions:

Framework In Section 2 we define a formal framework for explaining the outputs to SQL queries based on the notion of intervention. A key novel feature of our approach is to use foreign keys to model causal paths. We observe that SQL's cascade-delete rule already encodes a causal path: a record with a key `pk` is a cause for all records whose foreign key is `fk = pk`. We introduce a novel constraint that captures the reverse causal path, which often occurs in data, namely when the latter record causes the former record: we call this a back-and-forth foreign key, and show that it models naturally causal paths in databases. For example an author is cause for all her papers, hence if the intervention removes the author then it must also remove all her papers; in contrast the author is not a cause of her institution, hence an intervention may remove the author without removing her institution. In our framework, such causal paths can be modeled through a simple extension of foreign keys.

Theory In Section 3 we show that the minimal intervention associated to an explanation is uniquely defined. This is a key property for our framework, which allows us to measure the degree of explanation based on its unique minimal intervention. We further show that it can be defined as the minimal fixpoint of a recursive program that is monotone in its IDBs (hence has a unique minimal fixpoint), yet non-monotone in the input database (the intervention is not monotone in the input). We show some important properties of the recursive program: in particular we identify restrictions on the input schema under which the fixpoint is obtained in a constant number of iterations, thus eliminating the need for recursion.

Implementation and Optimization In Section 4 we show that, under certain restrictions, one can use SQL's data-cube operation to compute the degree of *all* candidate explanations, then return the top k to the user. While this technique works only under certain restrictions on the query and the database schema, when it can be applied, then it can compute and rank all explanations in real time. Furthermore, the entire computation can be pushed inside the database engine and, therefore, exploit all optimizations available for the data cube. Notice that, in contrast to our approach, the technique proposed recently by Wu and Madden [19] is based on a search algorithm on decision trees, which needs to be performed outside of the DBMS.

Experiments In Section 5 we run experiments on two real datasets. We show that explanations returned by our framework are interesting, and that they can be computed and returned in real time.

Related Work.

Pearl and others have explored various definitions of causality primarily using the concept of *counterfactuals* (see, e.g., [11, 12]); Meliou et al. adapted these notions to database queries [7, 8]. Silverstein et al. [15] study causal relationship in data mining settings for market basket data. Various approaches to explanations exists in the AI literature [10, 20]. In databases, Khoussainova et al. [6] proposed the system PerfXplain that explains systems performance. Fabbri and LeFevre studied explanation-based auditing of access log [3]. Das et al. [2] studied the problem of "Meaningful Ratings Interpretation (MRI)" based on the idea of data cube. Kanagal et al. [5] studied the problem of computing top-k influential variables and top-k explanations (to answer, *e.g.*, "why a tuple is in the output", "why the probability of an output tuple is greater than another one") in probabilistic databases. In recent work, Wu and Madden [19] discussed a

[2] We believe that the cause for the phenomenon observed in our example is a combination of a strategic hires by several Computer Science Departments of strong database faculty, combined with a hiring slowdown or even decline of database groups at some major industrial research labs.

framework for finding explanations for aggregate queries on a single relation based on the notion of influence. Related to explanation are Sarawagi and Sathe's techniques for efficient and automated data analysis in OLAP cubes (by RELAX, DIFF, and SURPRISE operators) [13, 14]. Our framework for explanation shares the same common philosophy of previous systems (measuring explanation by some sort of intervention or influence), but is the first comprehensive approach that can handle complex SQL queries over complex database schemas.

2. FRAMEWORK OF EXPLANATIONS

Let D be a database consisting of relations $R_1, \cdots, R_k$. For all[3] $i \in [k]$ we denote A_i the set of attributes of R_i. We assume that each relation R_i has a primary key, $\texttt{pk} \subseteq A_i$. A *numerical query* is an expression of the form:

$$Q = E(q_1, \ldots, q_m) \qquad (1)$$

Here each q_j is any SQL query where the `select` clause contains a single aggregate operator (q_j = `select agg(...) ...`), and E is an arithmetic expression, consisting of any numerical operators legal in SQL ($+, -, *, /, \log, \exp$ etc).

Definition 2.1. *A* user question *is a pair* (Q, dir)*, where (i)* Q *is a numerical query, and (ii)* $dir \in \{high, low\}$ *is a direction specifying whether the user thinks* Q *is higher or lower than the expected value on* D*.*

Example 2.2. *Our running example is a simplified version of the DBLP database[17] and has the following schema:*
$\texttt{Author}(\underline{\texttt{id}}, \texttt{name}, \texttt{inst}, \texttt{dom})$
$\texttt{Authored}(\underline{\texttt{id}, \texttt{pubid}})$
$\texttt{Publication}(\underline{\texttt{pubid}}, \texttt{year}, \texttt{venue})$
A small instance is in Figure 3. The "bump" around 2000-07 in Figure 1 is expressed as the following numerical query

$$Q = \frac{q_1}{q_2} \Big/ \frac{q_3}{q_4} = \frac{q_1}{q_2} \times \frac{q_4}{q_3}$$

which compares the ratios of SIGMOD publications from $\texttt{dom} =' com'$ *and* $\texttt{dom} =' edu'$ *in 2000-04 to that in 2007-11. The user question is* $(Q, high)$*. Here*

```
q1: select count(distinct z.pubid)
    from Author x, Authored y, Publication z
    where x.id = y.id and y.pubid = z.pubid
        and z.venue = 'SIGMOD' and x.dom = 'com'
        and 2000 <= z.year and z.year <= 2004
```

Similar queries can be written for

```
q2:   (2007 <= z.year <= 2011, x.dom = 'com'),
q3:   (2000 <= z.year <= 2004, x.dom = 'edu'), and
q4:   (2007 <= z.year <= 2011, x.dom = 'edu').
```

The bars for these four queries are indicated in Figure 1.

We envision a user interface[4] where the user would interact with the system by generating a graph (like Figure 1) using a group-by SQL query with one or several aggregates: each aggregate value represents a point in the graph. Using a graphical interface the user would select some points on the graph (like the four bars in Figure 1) and ask why a certain relationship holds; the system would convert this into a user question (Q, dir), as illustrated in our example. In this paper we assume that input to the system is the user question (Q, dir).

Author

id	name	inst	dom	
A1	JG	C.edu	edu	r_1
A2	RR	M.com	com	r_2
A3	CM	I.com	com	r_3

Publication

pubid	year	venue	
P1	2001	SIGMOD	t_1
P2	2011	VLDB	t_2
P3	2001	SIGMOD	t_3

Authored

id	pubid	
A1	P1	s_1
A2	P1	s_2
A1	P2	s_3
A3	P2	s_4
A2	P3	s_5
A3	P3	s_6

Figure 3: Tables for the running example, identifiers are shown next to the tuples for easy reference.

Given a database instance $D = (R_1, \ldots, R_k)$, we denote by $U(D) = R_1 \bowtie \ldots \bowtie R_k$ the universal relation, obtained by joining all tables on all foreign key constraints. The universal table for our running example in Figure 3 is shown in Figure 4. We assume that all relations are *semijoin-reduced*[5]; in other words, for every i, $R_i = \Pi_{A_i}(U(D))$, i.e. R_i can be obtained by projecting the universal relation on its attributes; if this does not hold, we simply replace R_i with $\Pi_{A_i}(U(D))$. In our running example, this means that every author has published at least one paper, and every paper has at least one author. All relations in Figure 3 are semijoin-reduced. We also assume that the schema is acyclic for simplicity[6].

2.1 Explanations

In response to a user question (Q, dir), the system returns a list of candidate explanations.

Definition 2.3. *A* candidate explanation ϕ *is a conjunction of predicates on attributes:* $\phi = \bigwedge_j \phi_j$*, where each* ϕ_j *is an atomic predicate of the form* $\phi_j = [R_i.A \ \texttt{op} \ c]$*, where* A *is an attribute of* R_i*,* c *is a constant, and* $\texttt{op} \in \{=, <, \leq, >, \geq\}$*.*

We note that our choice of defining explanations as predicates is similar to other frameworks for explanations [9, 19, 6], and is unlike frameworks for causality, where a cause is defined to consist of a single tuple [7, 8] (not predicates).

A simple example is $\phi = [\texttt{Author.name} = \texttt{JG}]$; another example is $\phi = [\texttt{Author.name} = \texttt{JG} \wedge \texttt{Publication.year} = 2001]$. The intuition is that the tuples in the database that satisfy the predicates in the explanation are major contributors to the value of Q being too high or too low. The system will rank the candidate explanations in *decreasing order* of their *degree of explanation*, which measures of how well ϕ explains the observed phenomenon. We will define two such degrees. The first one, called *degree of explanation by aggravation*, is simple but more limited: it simply measures the value of Q on the subset of the database that satisfies ϕ. Let $U = U(D)$ be the universal table, and denote $U_\phi = \sigma_\phi(U)$. Let D_ϕ be the database whose relations are $\Pi_{A_1}(U_\phi), \ldots, \Pi_{A_k}(U_\phi)$. Then:

Definition 2.4. *Given database* D *and a user question* (Q, dir)*, the* degree of a candidate explanation ϕ by aggra-

[3] For two integers $a \leq b$, $[a, b]$ denotes the set $\{a, a+1, \cdots, b-1, b\}$ and $[b] = [1, b]$.

[4] We are currently developing the user interface.

[5] They are called *globally consistent* in [1]

[6] We can handle cyclic schemas but with the standard restrictions for the universal relation [16], sec 17.8, page 1056.

vation *is defined as:*

$$\mu_{aggr(D,Q,dir)}(\phi) = \begin{cases} -Q(D_\phi) & \text{if } dir = low \\ \ \ Q(D_\phi) & \text{if } dir = high \end{cases}$$

We will omit D, Q, dir in the subscript where they are clear from the context. Intuitively, if we restrict the database to tuples that satisfy the explanation predicate, it will further *aggravate* the situation by changing the answer to Q in the same direction as dir (*e.g.*, if $dir = low$, explanations with even lower values of $Q(D_\phi)$ will have top ranks).

The explanation by aggravation is appealing through its simplicity, but it ignores any causality path. Our second definition is called *degree of explanation by intervention* and is based on causality relations in the data, which we define next.

2.2 Causal Paths and Foreign Keys

An *intervention* is a set of tuples to be deleted from D. If $\Delta_i \subseteq R_i$ for $i \in [k]$, then we denote the intervention as $\Delta = (\Delta_1, \ldots, \Delta_k)$ and write $D' = D - \Delta$ for the residual database instance after deleting all tuples in Δ. Each candidate explanation ϕ defines a certain intervention Δ^ϕ; its *degree of explanation by intervention* is the amount by which the query on the residual database $Q(D')$ moves in the direction expected by the user. Unlike aggravation, intervention Δ^ϕ includes all tuples reachable through a *causal path* (discussed further in Section 3.3). Intuitively, there exists a causal path from a tuple t to another tuple t' if t is a necessary cause for t'. In other words, if we delete t from the database then the semantics of the data is such that we must delete t' as well. Pearl defines causal paths in terms of causal networks [10]. In this paper we propose a simpler definition of a causal path by exploiting the foreign-key relationships in the database.

A standard foreign key constraint $R_j.\texttt{fk} \to R_i.\texttt{pk}$ means that for every tuple $t_j \in R_j$ there exists a tuple $t_i \in R_i$ such that $t_j.\texttt{fk} = t_i.\texttt{pk}$. For a simple example, in Figure 3 we have the foreign key `Authored.id` → `Author.id`. Every foreign key automatically induces a causal relation between the two tuples, which we denote $t_i \to t_j$. The precise meaning that we give to this causal relation is that, whenever we delete t_i from the database, we must delete t_j as well. We call this causal relation a *cascade* causal relation, because it corresponds precisely to the cascade rule in SQL.

In some datasets, however, the reverse causal relation holds too, $t_j \to t_i$. This happens when every member of a collection is necessary for the existence of the collection; for example, every author is necessary for the existence of a paper, every part is needed for the existence of an electronic gadget. In that case we denote the foreign key constraint as $R_j.\texttt{fk} \hookrightarrow R_i.\texttt{pk}$. The meaning is that we have two causal relations, a cascade relation $t_i \to t_j$ and *backward cascade* relation, which we denote by $t_j \dashrightarrow t_i$: if either t_i or t_j is deleted then the other tuple must be deleted as well. We say that $R_j.\texttt{fk} \hookrightarrow R_i.\texttt{pk}$ is a *back-and-forth* foreign key.

Definition 2.5. *Let $D = (R_1, \ldots, R_k)$ be a database with a set of foreign keys (standard and back-and-forth). Let $\Delta = (\Delta_1, \ldots, \Delta_k)$, where $\Delta_i \subseteq R_i$ for all $i \in [k]$. We call Δ* closed *if it satisfies the following conditions:*

- *For every standard foreign key $R_j.\texttt{fk} \to R_i.\texttt{pk}$ and for all tuples $t_i \in R_i, t_j \in R_j$ such that $t_j.\texttt{fk} = t_i.\texttt{pk}$, if $t_i \in \Delta_i$ then $t_j \in \Delta_j$.*
- *For every back-and-forth foreign key $R_j.\texttt{fk} \hookrightarrow R_i.\texttt{pk}$ and for all tuples $t_i \in R_i, t_j \in R_j$ such that $t_j.\texttt{fk} = t_i.\texttt{pk}$ (a) if $t_i \in \Delta_i$ then $t_j \in \Delta_j$ (forth), and (b) if $t_j \in \Delta_j$ then $t_i \in \Delta_i$ (back).*

id	pubid	name	inst	dom	year	venue	
A1	P1	JG	C.edu	edu	2001	SIGMOD	u_1
A2	P1	RR	M.com	com	2001	SIGMOD	u_2
A1	P2	JG	C.edu	edu	2011	VLDB	u_3
A3	P2	CM	I.com	com	2011	VLDB	u_4
A2	P3	RR	M.com	com	2001	SIGMOD	u_5
A3	P3	CM	I.com	com	2001	SIGMOD	u_6

Figure 4: Universal table $U(D)$ for the running example.

We illustrate on our running example Figure 3. Here we have a natural causal path from authors to papers: the presence of an author is necessary for their papers. In other words, if we delete an author from the database, then we should delete all her papers too. By contrast, there exists no causal path from papers to authors: we can delete a paper without deleting its authors. This is captured by a standard foreign key and a back-and-forth foreign key:

$$\begin{aligned} &\texttt{Authored.id} \to \texttt{Author.id} \\ &\texttt{Authored.pubid} \hookrightarrow \texttt{Publication.pubid} \end{aligned} \tag{2}$$

Referring to the tuples in Figure 3, we have the causal relations $r_1 \to s_1 \dashrightarrow t_1$ and also $r_1 \to s_3 \dashrightarrow t_2$. This means that if we delete the author r_1 then we must also delete s_1, s_3, t_1, t_3, then also delete s_2 and s_4 by the standard cascade rule.

The *intervention* associated to a candidate explanation ϕ is a set of tuples Δ^ϕ to be deleted.

Definition 2.6. *Let $D = (R_1, \cdots, R_k)$ be a database and ϕ be a candidate explanation. A set $\Delta = (\Delta_1, \cdots, \Delta_k) \subseteq D$ is called a* valid intervention *for ϕ if:*

1. *Δ is closed (Definition 2.5).*
2. *The residual database $D - \Delta$ is semi-join reduced.*
3. *For each tuple t in the residual universal relation $U(D - \Delta)$, $\phi(t) = \texttt{false}$.*

The intervention associated to ϕ, denoted Δ^ϕ, is the minimal valid intervention for ϕ, in the sense that, for every valid Δ' we have $\Delta^\phi \subseteq \Delta'$.

The trivial intervention $\Delta = D$ is always valid, but in general, is not minimal. We show in the next section that there always exists a unique minimal intervention. Our second (and main) definition of degree of explanation is based on intervention:

Definition 2.7. *Given database D and a user question (Q, dir), the* degree of a candidate explanation ϕ by intervention *is defined as:*

$$\mu_{interv(D,Q,dir)}(\phi) = \begin{cases} \ \ Q(D - \Delta^\phi) & \text{if } dir = low \\ -Q(D - \Delta^\phi) & \text{if } dir = high \end{cases}$$

Note that the sign for μ_{interv} is opposite to that for μ_{aggr} (Definition 2.4), since unlike μ_{aggr} here we want to intervene or inhibit the situation by changing the value of Q in the opposite direction of dir.

Example 2.8. *For our running example, consider the explanation $\phi = [\texttt{Author.name} = \texttt{JG} \wedge \texttt{Publication.year} = \texttt{2001}]$. The associated intervention is:*

$$\Delta_{\texttt{Author}} = \varnothing \qquad \Delta_{\texttt{Authored}} = \{s_1, s_2\} \qquad \Delta_{\texttt{Publication}} = \{t_1\}$$

To see this, note that we must delete s_1*. If we don't then any residual database* $D' \subseteq D$ *that contains* s_1 *must contain* r_1, t_1*, and therefore its universal relation contains* (r_1, s_1, t_1)*, which satisfies* ϕ*; but this violates item 3 of Definition 2.6. But once we delete* s_1 *we must also delete* t_1 *(because of the back-and-forth* $\texttt{Authored.pubid} \hookrightarrow \texttt{Publication.pubid}$*), then cascade delete* s_2*. Notice how the causal path from* `Author` *to* `Publication` *lead to an asymmetric intervention* Δ^{ϕ}*, where we delete the paper with* $\texttt{Publication.year} = \texttt{2001}$ *but do not delete the author with* $\texttt{Author.name} = \texttt{JG}$*.*

In contrast, if both foreign keys in Eq.(2) were standard, then intervention Δ^{ϕ} *will be symmetric for the same* ϕ*:*

$$\Delta_{\texttt{Author}} = \varnothing \qquad \Delta_{\texttt{Authored}} = \{s_1\} \qquad \Delta_{\texttt{Publication}} = \varnothing$$

because if we delete only s_1 *then all foreign keys still hold, and no tuple in the universal relation satisfies* ϕ*.*

We explain now the critical role of item (2) of Definition 2.6, requiring the residual database $D - \Delta^{\phi}$ to be semijoin reduced. This is necessary in order to ensure that every explanation has a unique, minimal, and valid intervention. To see why the condition is necessary, consider this example:

Example 2.9. *The schema is* $R_1(\underline{x})$*,* $S_1(x, y)$*,* $R_2(\underline{y})$*,* $S_2(y, z)$*,* $R_3(\underline{z})$*. There are four standard FK's (from* S_1 *to* R_1, R_2*, and from* S_2 *to* R_2, R_3*), the database instance is*

$$D = \{R_1(a), S_1(a, b), R_2(b), S_2(b, c), R_3(c)\} \qquad (3)$$

and the explanation $\phi = [R_1.x = a \wedge R_2.y = b \wedge R_3.z = c]$*. If we don't require the residual database to be semi-join reduced, then there are two minimal interventions,* $\Delta = \{S_1(a, b)\}$ *and* $\Delta = \{S_2(b, c)\}$*: in both cases the residual database* $D - \Delta$ *has dangling tuples and its universal relation is empty, thus satisfying all conditions in Definition 2.6, except item (2).*

Since we require $D - \Delta$ to be semijoin reduced, there is always a unique minimal intervention Δ^{ϕ} (we prove this in the next section), and this is an important property that makes our framework work, since we can rank explanations based on degree of their *unique* minimal intervention[7]. Continuing Example 2.9, the minimal intervention is $\Delta^{\phi} = D$, because after semijoin reduction the residual database becomes $\varnothing$.

Finally, our last remark is that the minimal intervention Δ^{ϕ} is a non-monotone function of the input database[8]. We show this by continuing Example 2.9 further:

Example 2.10. *Suppose we insert the tuples* $S_1(a, b')$*,* $R_2(b')$*,* $S_2(b', c)$ *into* D *(Eq.(3)): the minimal intervention decreases to* $\Delta^{\phi} = \{S_1(a, b), R_1(b), S_2(b, c)\}$*, in other words the tuples* $R_1(a)$*,* $R_3(c)$ *remain in the residual database. Notice that this is a valid intervention – the universal table of the residual database has a single tuple* $(R_1(a)$*,* $S_1(a, b')$*,* $R_2(b')$*,* $S_2(b', c)$*,* $R_3(c))$*, which does not satisfy* ϕ*.)*

We show in Section 3 how to compute Δ^{ϕ} efficiently, then discuss in Section 4 how to compute efficiently *all* degrees of intervention $\mu(D, Q, dir, \phi)$, for all candidate explanations ϕ, using SQL's data cube feature.

3. COMPUTATION OF INTERVENTION

In this section we show how to compute the intervention Δ^{ϕ} associated to a candidate explanation ϕ. We show that Definition 2.6 is equivalent to the minimal fixpoint of a monotone query; in particular, it follows that the minimal valid intervention is well defined and unique. We also discuss convergence properties of this recursive query.

3.1 Rules for Intervention

Given a database D, consider the following recursive program $\mathbf{P}$, computing k sets $\Delta = (\Delta_1, \Delta_2, \cdots, \Delta_k)$:

$$\boxed{\begin{array}{lll} \Delta_i = R_i - \Pi_{A_i}\sigma_{\neg\phi}[R_1 \bowtie \cdots \bowtie R_k] & \forall i \in [k] & \text{(i)} \\ \Delta_i = R_i - \Pi_{A_i}[(R_1 - \Delta_1) \bowtie \cdots \bowtie (R_k - \Delta_k)] & \forall i \in [k] & \text{(ii)} \\ \Delta_i = R_i \ltimes_{\texttt{pk=fk}} \Delta_j & & \text{(iii)} \\ \qquad (\forall \text{ back-and-forth } R_j.\texttt{fk} \hookrightarrow R_i.\texttt{pk}) & & \end{array}}$$

The body of each rule in $\mathbf{P}$ is a relational algebra expression; multiple rules with the same head predicate Δ_i are interpreted as a union. The join operators in Rules (i) and (ii) are on all foreign key constraints: that is, both compute the universal relation. We explain the program next. Rule (i) computes Δ_i that represent a minimum set of tuples to be deleted from each R_i to ensure that no tuples in the universal relation satisfy ϕ. In general, this Δ_i is a strict subset of $\Pi_{A_i}\sigma_{\phi}[R_1 \bowtie \cdots \bowtie R_k]$; in our running example for the candidate explanation $\phi : [\texttt{Author.name} = \texttt{JG} \wedge \texttt{Publication.year} = \texttt{2001}]$, we do not want to delete the tuple r_1 with $[\texttt{Author.name} = \texttt{JG}]$ from the `Author` relation since he is a coauthor of another publication that does not satisfy the predicate ϕ (tuple t_2 with $\texttt{year} = 2011$). Rule (ii) performs a semijoin reduction on $R_1 - \Delta_1, \ldots, R_k - \Delta_k$; in particular this rule also enforces the cascade semantics for every foreign key dependency. There is one instance of Rule (iii) for every back-and-forth foreign key, which simply enforces the backwards cascade semantics for that foreign key. Recall that the *semi-join* operation is defined as $R \ltimes S = \Pi_{Attr(R)}(R \bowtie S)$; Rule (iii) simply removes from R_i (by inserting into Δ_i) all tuples t_i for which there exists some $t_j \in \Delta_j$ with $t_j.\texttt{fk} = t_i.\texttt{pk}$.

Proposition 3.1. *The program* $\mathbf{P}$ *is monotone in* $\Delta_1, \cdots, \Delta_k$*, but is* not *monotone in the input database* $R_1, \ldots, R_k$ *(and therefore is not equivalent to any datalog program).*

To see that $\mathbf{P}$ is monotone in $\Delta_1, \cdots, \Delta_k$, note that in Rule (ii) all Δ_i's occur under two set difference operators, hence all three rules are monotone in Δ. Therefore, $\mathbf{P}$ has a unique minimal fixpoint, which can be computed in the normal fashion, by starting with $\Delta^0 = (\varnothing, \ldots, \varnothing)$, then computing iteratively $\Delta^0 \subseteq \Delta^1 \subseteq \Delta^2 \subseteq \ldots$ until we reach a fixpoint $\Delta^{\ell} = \Delta^{\ell+1}$; we denote $\mathbf{P}(D) = \Delta^{\ell}$ the minimal fixpoint. Interestingly, the program is *not* monotone in the input database; in Example 2.10, Δ^{ϕ} decreased when D increased. Nevertheless,

Proposition 3.2. *The program* $\mathbf{P}$ *can be expressed in datalog*$^{\neg}$ *by a straightforward rewriting.*

Below, $\mathbf{x_i}$ denote the set of variables used in R_i, all $\mathbf{x_i}$ use the same variable for the same attribute, and $\mathbf{x} = \cup_i \mathbf{x_i}$:

$$S_i(\mathbf{x}_i) \quad :- \quad R_1(\mathbf{x_1}), \cdots, R_k(\mathbf{x_k}), \neg\phi(\mathbf{x}) \qquad \forall i \in [k]$$

[7] For simple aggregate (SPJA) queries this is not a limitation (the dangling tuples do not contribute to the query answer). But for more complex nested aggregate queries this constraint is needed to have a unique minimal intervention.

[8] This is somewhat surprising, since, in contrast, D_{ϕ}, is a monotone function in the database.

$$
\begin{array}{llll}
\Delta_i(\mathbf{x}_i) & :- & R_i(\mathbf{x}_i), \neg S_i(\mathbf{x}_i) \qquad \forall i \in [k] & \text{(Rule (i))} \\
T_i(\mathbf{x}_i) & :- & R_1(\mathbf{x_1}), \neg\Delta_1(\mathbf{x_1}), \cdots, R_k(\mathbf{x_k}), \neg\Delta_k(\mathbf{x_k}) & \\
 & & & \forall i \in [k] \\
\Delta_i(\mathbf{x}_i) & :- & R_i(\mathbf{x}_i), \neg T_i(\mathbf{x}_i) \qquad \forall i \in [k] & \text{(Rule (ii))} \\
\Delta_i(\mathbf{x}_i) & :- & R_i(\mathbf{x}_i), \Delta_j(\mathbf{x}_j) \quad \text{for } R_j.\mathtt{fk} \hookrightarrow R_i.\mathtt{pk} & \text{(Rule (iii))}
\end{array}
$$

Rule (i) is not recursive and is applied only in the first iteration. We will refer to the tuples collected after the first iteration $\Delta^1 = \{\Delta_1^1, \cdots, \Delta_k^1\}$ as the *seed tuples*. The following theorem shows that P computes the (unique) intervention Δ^ϕ for a given ϕ.

Theorem 3.3. *The unique minimal fixpoint $\mathbf{P}(D)$ is equal to the intervention Δ^ϕ associated to an explanation ϕ (Definition 2.6).*

Proof. $U(D - \Delta^1) = (R_1 - \Delta_1^1) \bowtie \cdots \bowtie (R_1 - \Delta_1^1)$ is exactly the subset of $U(D)$ that does not satisfy ϕ, where Δ^1 denotes the seed tuples collected in the first iteration by Rule (i). Hence any minimal $\Delta^\phi = (\Delta_1, \cdots, \Delta_k)$ satisfying the third condition of Definition 2.6 satisfies that

$$U(D - \Delta^\phi) \subseteq U(D - \Delta^1) \tag{4}$$

Rules (ii) and (iii) ensure the first and second conditions of Definition 2.6 that Δ^ϕ is closed and $D - \Delta^\phi$ is semijoin-reduced. Hence it suffices to show that any such valid $\Delta^\phi \supseteq \Delta^1$. Then it immediately follows that a minimal Δ^ϕ is the unique least minimal fixpoint of the program $\mathbf{P}(D)$.

Suppose $\Delta^\phi \not\supseteq \Delta^1$. Assume, without loss of generality, that $\Delta_1 \not\supseteq \Delta_1^1$. Then, there exists $y \in \Delta_1^1 - \Delta_1$ Then, $y \in \Delta_1^1 \Rightarrow y \notin R_1 - \Delta_1^1$, which implies that

$$\nexists \quad t \in U(D - \Delta^1) \text{ such that } \Pi_{A_i} t = y \tag{5}$$

Also, $y \notin \Delta_1 \Rightarrow y \in R_1 - \Delta_1$. Since $D - \Delta^\phi$ is semijoin-reduced,

$$\exists \quad t \in U(D - \Delta^\phi) \text{ such that } \Pi_{A_i} t = y \tag{6}$$

(5) and (6) together contradict (4). Hence $\Delta^\phi \supseteq \Delta^1$ and the theorem holds. □

3.2 Convergence Properties of the Rule Set

Since the program $\mathbf{P}$ is monotone, the polynomial data complexity is immediate:

Proposition 3.4. *The Rules (i)-(iii) converge in $\leq n$ iterations where $n = \sum_{i=1}^k |R_i|$.*

Further, if there are no back-and-forth foreign keys (only standard cascade delete holds), we have a much faster convergence:

Proposition 3.5. *If the input database has no back-and-forth foreign keys, then Rules (i)-(iii) in program $\mathbf{P}$ converge in only two steps,* i.e., *$\Delta_i^2 = \Delta_i^3$ for all $i \in [k]$.*

Proof sketch. After iteration 1, the seed tuples (if any) are computed applying Rule (i), so possibly some $\Delta_i^1 \supsetneq \Delta_i^0$. After iteration 2 (if needed), relations are again semijoin-reduced by applying Rule (ii), so possibly some $\Delta_i^2 \supsetneq \Delta_i^1$. Rule (iii) cannot be applied since there are no back-and-forth foreign keys. By Rule (ii), all $R_i - \Delta_i^2$ are semijoin-reduced relations. Therefore, $R_i - \Delta_i^2 = \Pi_{A_i}[(R_1 - \Delta_1^2) \bowtie \cdots \bowtie (R_k - \Delta_k^2)]$; *i.e.*, $\Delta_i^3 = R_i - (R_i - \Delta_i^2) = \Delta_i^2$. □

Table $\mathtt{R_1}$

<u>a</u>	
a_1	r_1
a_2	r_2

Table $\mathtt{R_3}$

<u>c</u>	a	b	
c_1	a_1	b_0	s_{1a}
c_2	a_1	b_1	s_{1b}
c_3	a_2	b_1	s_{2a}
c_4	a_2	b_2	s_{2b}

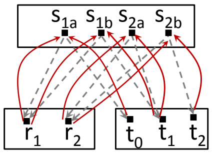
Table $\mathtt{R_2}$

<u>b</u>	
b_0	t_0
b_1	t_1
b_2	t_2

(a) Individual relations

$\mathtt{R_1.a}$	$\mathtt{R_3.c}$	$\mathtt{R_3.a}$	$\mathtt{R_3.b}$	$\mathtt{R_2.b}$
a_1	c_1	a_1	b_0	b_0
a_1	c_2	a_1	b_1	b_1
a_2	c_3	a_2	b_1	b_1
a_2	c_4	a_2	b_2	b_2

(b) Universal relation



(c) Data causal graph

Figure 5: Example where program $\mathbf{P}$ needs $n - 1$ iterations. For general $n = 2p + 1$, R_1, R_2, R_3 will have tuples $(r_1, \cdots, r_p)$, $(t_0, \cdots, t_p)$, and $(s_{1a}, s_{1b}, \cdots, s_{pa}, s_{pb})$ respectively, and the rules converge in $2p$ steps.

For no back-and-forth foreign keys, the following corollary also holds. This will be useful to efficiently compute the degree of explanations in the next section.

Corollary 3.6. *If there are no back-and-forth foreign keys, then*

$$(R_1 - \Delta_1) \bowtie \cdots \bowtie (R_k - \Delta_k) = \sigma_{\neg\phi}(R_1 \bowtie \cdots \bowtie R_k)$$

Proof Sketch. By Proposition 3.5, $\Delta = \Delta^2$. After the first iteration, by Rule (i), $U(D - \Delta^1) = \sigma_{\neg\phi}(R_1 \bowtie \cdots \bowtie R_k)$. Since $\Delta^2 \supseteq \Delta^1$, $U(D - \Delta^1) \supseteq U(D - \Delta^2)$. We claim that $U(D - \Delta^1) = U(D - \Delta^2)$, which will prove the lemma.

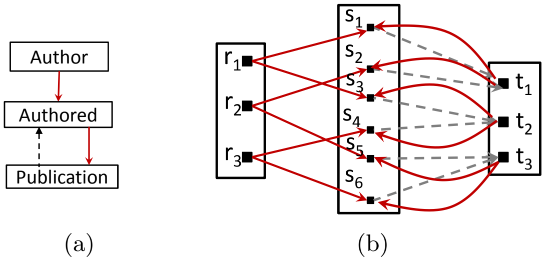
Suppose not. Then there exists $t \in U(D-\Delta^1) - U(D-\Delta^2)$. Since $t \notin U(D - \Delta^2)$, there exists $i \in [k]$ such that $\Pi_{A_i} t \notin R_i - \Delta_i^2$. Then $\Pi_{A_i} t \in \Delta_i^2$. Hence $\Pi_{A_i} t \in R_i - \Pi_{A_i} U(D - \Delta^1)$. Hence $\Pi_{A_i} t \notin \Pi_{A_i} U(D - \Delta^1)$, which implies that $t \notin U(D - \Delta^1)$. This is a contradiction since we assumed that $t \in U(D - \Delta^1) - U(D - \Delta^2)$. □

On the other hand, the following example shows that, if we have a very simple schema with three relations $R_1(\underline{a})$, $R_2(\underline{b})$, $R_3(\underline{c}, a, b)$, and two back-and-forth foreign keys $R_3.a \hookrightarrow R_1.a$ and $R_3.b \hookrightarrow R_2.b$, then Rules (i)-(iii) take $n - 1$ iterations to converge, which also shows that the upper bound in Proposition 3.4 is essentially tight.

Example 3.7. *We illustrate for $n = 9$. The three relations and the initial universal relation are shown in Figure 5, where every tuple is assigned an identifier for easy reference. Our candidate explanation is $\phi : (\mathtt{c} = c_1)$. Initially all $\Delta_i^0 = \emptyset$, for $i \in [1, 3]$. In iteration 1, only Δ_3^1 changes to $\{s_{1a}\}$. In iteration 2, $\Delta_1^2 = \{r_1\}, \Delta_2^2 = \{t_0\}$ by Rule (iii), $\Delta_3^2 = \Delta_3^1$. In iteration 3, $\Delta_1^3 = \Delta_1^2, \Delta_2^3 = \Delta_2^2$, but $\Delta_3^3 = \Delta_3^2 \cup \{s_{1b}\}$ by Rule (ii). In iteration 4, only Δ_2^4 changes: $\Delta_2^4 = \Delta_2^3 \cup \{t_1\}$ by Rule (iii). The same process is repeated through iterations 4 to 8, and the tuples s_{2a}, r_2, s_{2b}, t_2 are acquired in their respective Δ_is.*

3.3 Avoiding Recursion in Special Cases

In general we need a recursive query to compute Δ^ϕ (Example 3.7), while Proposition 3.5 shows that in a very simple case recursion can be avoided completely. Here we generalize Proposition 3.5 and show that for a class of acyclic causal



(a) (b)

Figure 6: (a) Schema causal graph, (b) Data causal graph

graphs recursion can be avoided and be replaced by an iterative query with a bounded number of steps[9].

Definition 3.8. *Fix a database schema. The* schema causal graph, *G, has one node for each relation $R_1, \cdots, R_k$; for every standard foreign key $R_j.\texttt{fk} \rightarrow R_i.\texttt{pk}$ there is an edge from R_i to R_j; for every back-and-forth foreign key $R_j.\texttt{fk} \hookrightarrow R_i.\texttt{pk}$ there is an edge from R_i to R_j, and a dotted edge from R_j to R_i.*

Let D be a database. The data causal graph, *G_D, has one node for each tuple in D and the following edges:*

1. *There is a directed solid edge from $t_i \in R_i$ to $t_j \in R_j$ if $\forall\ u \in U(D)$, $\Pi_{A_j} u = t_j \ \Rightarrow \Pi_{A_i} u = t_i$.*
2. *There is a directed dotted edge from $t_j \in R_j$ to $t_i \in R_i$ if there is a dotted edge (R_j, R_i) in G and $t_j.\texttt{fk} = t_i.\texttt{pk}$.*

Intuitively, both graphs depict the standard and back-and-forth causal relations through solid and dotted edges, at the schema and at the data level respectively. However, at the data level, the definition is more involved because it also needs to account for semi-join reduction. First, note that for every standard foreign key $R_j.\texttt{fk} \rightarrow R_i.\texttt{pk}$ and two tuples t_i, t_j s.t. $t_i.\texttt{pk} = t_j.\texttt{fk}$ there is a solid edge $t_i \rightarrow t_j$, because every universal tuple u for which $\Pi_{A_j} = t_j$ must have $t_i = \Pi_{A_i}(u)$. However, suppose that t_j is the only tuple in R_j with the property $t_j.\texttt{fk} = t_i.\texttt{pk}$: then if we delete t_j we end up deleting t_i too when doing the semi-join reduction. The definition ensures that in the latter case we also have a solid edge $t_j \rightarrow t_i$.

In figures we show the relations R_i as *supernodes* containing data nodes; also if there is both a solid edge and a dotted edge between two nodes, we omit the solid edge. The schema causal graph and the data causal graph of our running example (Example 2.2) is shown in Figure 6, whereas Figure 5c shows the data causal graph for Example 3.7. Recall that a path in a graph is called *simple* if every node is visited at most once.

Definition 3.9. *A* causal path *P is a simple, directed path in G_D. The* causal-length *of P is the number of directed dotted edges in P.*

For instance, in Figure 6, $P = r_1 \rightarrow s_1 \dashrightarrow t_1 \rightarrow s_2$ is a causal path from r_1, of causal length 1. Further, P is *maximal.*

Recall that Δ^1 denotes the set of the seed tuples, defined by Rule (i). The following proposition gives a tighter convergence property of program $\mathbf{P}$ in terms of the maximum causal length of the causal paths from the seed tuples.

Proposition 3.10. *The program $\mathbf{P}$ converges in $\leq 2q + 2$ steps, where q is the maximum causal-length over all causal paths P starting at a seed tuple.*

We get Proposition 3.5 as a corollary of Proposition 3.10 when there are no back-and-forth foreign keys (*i.e.*, no dotted edges in G_D, or $q = 0$). It can also be verified that there is a causal path in Example 3.7 of length $q = |R_3| = (n-1)/2$ (n is odd), and therefore it takes $n - 1 \leq 2q + 2$ steps to converge.

Proof. The proof is an immediate extension of the proof of Proposition 3.5. It suffices to note that Rule (ii) in isolation can fire at most once, because after one application all relations $R_i - \Delta_i$ are semi-join reduced. Therefore, progress is made only as long as Rule (iii) can discover new facts, and this happens at most q times. The factor 2 accounts for possible alternation between the rules (ii) and (iii), while the additive term +2 is for applying Rule (i) in the first iteration, and for the final semi-join reduction in the last iteration. □

Proposition 3.10 tightens Proposition 3.4. As we saw in Example 3.7, if some relation R_j has two back-and-forth foreign keys $R_j.\texttt{fk} \hookrightarrow R_i.\texttt{pk}$ and $R_j.\texttt{fk} \hookrightarrow R_p.\texttt{pk}$, then recursion is required. However, we use the proposition to show that, in an acyclic schema, if no relation R_j has two back-and-forth foreign keys, then recursion can be avoided (in this case the maximum number of back-and-forth foreign keys $\leq k$).

Proposition 3.11. *Suppose the schema causal graph is simple[10] and acyclic. Furthermore, suppose that every relation R_j has at most one back-and-forth foreign key. Then the program $\mathbf{P}$ converges in $2s + 2$ steps where s = the total number of back-and-forth foreign keys in the schema. As a consequence, Δ^ϕ can be computed by a non-recursive query.*

Proof. We show that the causal length of any causal path is bounded by s. Then the proposition follows from Proposition 3.10. In particular, we show that for any causal path P and any relation R_i, there can be at most one tuple $t \in R_i$ on P with a dotted outgoing edge on P.

Suppose not. Then there is a relation R_i, a database D, a causal path $P = u_0 \rightarrow \ldots t \dashrightarrow v \rightarrow \ldots t' \dashrightarrow v' \rightarrow \ldots u_\ell$ in G_D, where t, t' are two distinct tuples in R_i (note that a causal path is simple and the schema causal graph is acyclic), and both outgoing edges (t, v) and (t', v') are dotted edges. Note that v, v' must belong to the same relation R_j since there is at most one outgoing dotted edge from R_i. Further, $v \neq v'$ since the causal path is simple.

By Definition 3.8, if there is a directed path from $x \in R_i$ to $y \in R_j$ in G_D, then there is a tuple $u \in U(D)$ such that $\Pi_{A_i} u = x$ and $\Pi_{A_j} u = y$. Then the causal path $P_{v \rightsquigarrow t'}$, which is a sub-path of P from v to t', indicates that there is a tuple $u \in U(D)$ with $\Pi_{A_j} u = v$ and $\Pi_{A_i} u = t'$. This is a contradiction since there is a back-and-forth foreign key $R_j.\texttt{fk} \hookrightarrow R_i.\texttt{pk}$, and therefore $t \in R_i$ can only join with $v \in R_j$ whereas $v \neq v'$. Hence the proposition follows. □

As an application, consider our running example in Figure 3, whose schema has the two foreign keys in Eq.(2). Only one is a back-and-forth foreign key, therefore the program $\mathbf{P}$ terminates in at most 4 iterations. Concretely, it can be unfolded into four steps: Rule (i) computes the seeds, Rule (ii) performs semi-join reduction, Rule (iii) does backwards cascade, and finally Rule (ii) performs the final semi-join reduction.

[9] Note that (i) our general framework does not require the causal graph to be acyclic, and (ii) causal graphs can have cycles even if the schema is acyclic, as is the case with our running example.

[10] There is at most one foreign key between any two relations.

4. FINDING TOP EXPLANATIONS

Given a user question (Q, dir), the system needs to iterate over all candidate explanations $\phi_1, \phi_2, \ldots$, compute their degrees, and rank them accordingly (either by aggravation μ_{aggrv} or by intervention μ_{interv}). To compute μ_{interv}, we have shown in Section 3 that each intervention $\Delta^{\phi_1}, \Delta^{\phi_2}, \ldots$, can be computed in polynomial time in the size of the database; however, a naive approach that computes sequentially $\mu_{interv}(\Delta^{\phi_1}), \mu_{interv}(\Delta^{\phi_2}), \ldots$ is too inefficient to be used interactively. In this section we show that, under certain conditions, the top-K explanations can be found efficiently using existing DBMS systems that support *data cube* operations. Throughout this section we only consider candidate explanations where the predicate is an equality operator.

The data cube operation is supported in most of the commercial database management systems (SQLServer, DB2, Oracle). It can find the value of an aggregate query for all combinations of values of the attributes specified in the Group By clause, where the 'dont-care' attributes take `null` value. We illustrate the cube operation below:

Example 4.1. *The following query computes a cube over the data in Figure 3.*

```
select x.name, z.year, count(*)
from Author x, Authored y,
           Publication z
where x.id = y.id
   and y.pubid = z.pubid
group by x.name, z.year
with cube
```

name	year	
JG	*2001*	*1*
JG	*2011*	*1*
RR	*2001*	*2*
CM	*2001*	*1*
CM	*2011*	*1*
JG	*null*	*2*
RR	*null*	*2*
CM	*null*	*2*
null	*2001*	*4*
null	*2011*	*2*
null	*null*	*6*

Each row in the cube corresponds to an explanation: e.g., *the row* $(null, 2001, 4)$ *corresponds to* $\texttt{z.year} = 2001$*, while the row* $(CM, 2011, 1)$ *to* $[\texttt{x.name} = \texttt{CM} \wedge \texttt{z.year} = 2001]$.

4.1 Degrees of Explanation using Data-Cube

We start by showing how to compute μ_{aggr} using data-cube. Recall from Eq.(1) that Q is an arithmetic expression on several aggregate queries $Q = E(q_1, \ldots, q_m)$. $U = U(D) = R_1 \bowtie \ldots \bowtie R_k$ is the universal relation, and D_ϕ is the database whose relations are $\Pi_{A_1}(\sigma_\phi(U)), \ldots, \Pi_{A_k}(\sigma_\phi(U))$. Recall: $\mu_{aggr}(\phi) = Q(D_\phi) = E[q_1(D_\phi), \cdots, q_m(D_\phi)]$.

For each aggregate query q_j compute a data-cube where each row contains the values $q_j(\sigma_\phi(U)) = q_j(D_\phi)$ for some candidate explanation ϕ. Next, join these m cubes, and compute the expression E on the values $q_1(D_\phi), \ldots, q_m(D_\phi)$. The join is a *full outer join* so that we do not lose explanations ϕ which do not appear in some of these m cubes; in those cubes they are considered to have zero value. Finally, return the explanations with the top K values of E.

Next, we show how to compute μ_{interv} using data-cube, under certain restrictions. For presentation purposes we assume that the degree of explanation (Definition 2.7) is given by $\mu_{interv}(\phi) = Q(D - \Delta^\phi)$; the case when the sign is negative is similar and omitted.

Definition 4.2. *An aggregate query q is* intervention-additive *if for all input databases D and candidate explanation ϕ, we have $q(D - \Delta^\phi) = q(D) - q(D_\phi)$, where Δ^ϕ is the intervention defined in Definition 2.6.*

A numerical query $Q = E(q_1, \cdots, q_m)$ is intervention-additive *if all $q_j, j \in [m]$ are intervention-additive.*

When the numerical query Q is intervention-additive, we can compute $\mu_{interv}(\phi)$ for all ϕ using data-cube operations. For each query q_j, start by computing a data-cube where each row contains $q_j(\sigma_\phi(U)) = q_j(D_\phi)$ for some candidate explanations ϕ; derive $q_j(D - \Delta^\phi) = q_j(D) - q_j(D_\phi)$. Next, we join these m cubes, and in each row compute the expression E on the values $q_1(D - \Delta^\phi), \ldots, q_m(D - \Delta^\phi)$. Finally, we return the explanations with the top-K values of E.

In general, note that, $U(D - \Delta^\phi) \subseteq U(D - \Delta^1) = U_{\neg\phi} = U - U_\phi$ (see the proof of Theorem 3.3), where Δ^1 is the set computed by the first iteration of program $\mathbf{P}$ by Rule (i). So, in general, $q(D - \Delta^\phi) = q(D) - q(D_\phi)$ does not hold. We give two sufficient conditions for the query to be intervention-additive.

Count$(*)$. If q is a `count(*)` aggregate over the universal relation $R_1 \bowtie \ldots \bowtie R_k$ where there are no back-and-forth foreign keys, then it is intervention-additive. This follows from Corollary 3.6 which shows that $U(D - \Delta^\phi) = U - U_\phi$, and due to the additive property of `count(*)` queries.

Count(distinct R$_i$.pk). If q is a `count(distinct` $\texttt{R}_i\texttt{.pk}$`)` aggregate over the universal relation $R_1 \bowtie \ldots \bowtie R_k$, and there exists some back-and-forth foreign key $R_j.\texttt{fk} \hookrightarrow R_i.\texttt{pk}$ such that every row in $U(D)$ contains a unique tuple from R_j, then q_j is intervention-additive. For instance, our running DBLP example with the foreign keys given in Eq.(2), a query with an aggregate of the form `count(distinct pubid)` is intervention-additive for any explanation ϕ, since every row of $U(D)$ contains a unique entry from the `Authored` table, although $U(D - \Delta^\phi)$ may be a strict subset of $U - U_\phi$.[11]

In the presence of back-and-forth foreign keys the intervention-additive property does not hold in general (*e.g.*, `count(*)` queries on our running example are not intervention-additive). However, we can sometimes transform the database into an equivalent database (meaning that it has the same causal paths) where each back-and-forth foreign key is replaced with several standard foreign keys. We illustrate this on our running example in Figure 3; a formal argument is deferred to the full version of this paper. Assume that every paper in the database has at most 3 authors. Then we make three copies of the `Author` and `Authored` tables: $\texttt{Author}_\texttt{i}(\underline{\texttt{id}_\texttt{i}}, \texttt{name}_\texttt{i}, \texttt{inst}_\texttt{i}, \texttt{dom}_\texttt{i})$ and $\texttt{Authored}_\texttt{i}(\underline{\texttt{kad}_\texttt{i}}, \texttt{id}_\texttt{i}, \texttt{pubid})$, where kad_i is the primary key of $\texttt{Authored}_i$. Then we convert `Publication` to

$$\texttt{Publication}'(\texttt{kad}_1, \texttt{kad}_2, \texttt{kad}_3, \underline{\texttt{pubid}}, \texttt{year}, \texttt{venue}).$$

[11] To see this, consider the subset U_ϕ of U which are removed by Rule (i) to find the initial set Δ^1. Since every entry of `Authored` appears in exactly one row of U, $[\Pi_{Attr(\texttt{Authored})} U_\phi = \Delta^1_{\texttt{Authored}}]$ and $[\Pi_{Attr(\texttt{Authored})} U_{\neg\phi} = \texttt{Authored} - \Delta^1_{\texttt{Authored}}]$. By backwards cascade, Rule (iii) will include entries in $\Delta^2_{\texttt{Publication}}$ that tuples in $\Delta^1_{\texttt{Authored}}$ refer to. Then Rule (ii) will further include tuples in $\Delta^3_{\texttt{Authored}}$ (if any) that join with $\Delta^2_{\texttt{Publication}}$. As a result, $U(D - \Delta^\phi)$ will not have any publication from $\Delta^2_{\texttt{Publication}}$, and will have all the publications from $\texttt{Publication} - \Delta^2_{\texttt{Publication}}$. The `count(distinct pubid)` query q on $D_\phi \equiv U_\phi$ will exactly count the publications from $\Delta^2_{\texttt{Publication}}$. The `count(distinct pubid)` query q on $(D - \Delta^\phi)$ counts the publications in $\texttt{Publication} - \Delta^2_{\texttt{Publication}}$, which is exactly the same as $q(D) - q(D_\phi)$.

Then, the foreign keys given in Eq.(2) are replaced by the following standard foreign keys:

- $\texttt{Authored}_\texttt{i}.\texttt{id}_\texttt{i} \rightarrow \texttt{Author}_\texttt{i}.\texttt{id}_\texttt{i}$, $i \in [3]$
- $\texttt{Publication.kad}_\texttt{i} \rightarrow \texttt{Authored}_\texttt{i}.\texttt{kad}_\texttt{i}$,$i \in [3]$

The entries `Author` and `Authored` tables are replicated in their three instances, and a dummy value is added to every table to allow for < 3 authors for a publication. The (at most) three authors are assigned arbitrarily to $kad_1, \cdots, kad_3$, for < 3 authors, they take the dummy value. It is not hard to see that now there will be exactly one entry per distinct `pubid` in the universal table. Then `count(*)` is run on the universal table (the predicate on the `Author` table changes to a disjunction of the condition on three authors), and the results are grouped by a post-processing step.

4.2 Algorithm

We now describe briefly our algorithm that computes degrees of all explanations (both μ_{interv} and μ_{aggr}). The input is an intervention-additive numerical query $Q = E(q_1, \cdots, q_m)$ (Definition 4.2), and a subset of attributes A' on which explanations are being thought; the subset A' helps both in focusing the search and improving performance and can be set by the user.

Algorithm 1 Computes $\mu_{interv}(\phi)$ and $\mu_{aggr}(\phi)$ for all ϕ.

```
Let $u_1, \cdots, u_m$ be the values of all aggregate queries $q_1, \cdots, q_m$ on the original database $D$.
Let $C_1, \cdots, C_m$ be the data cubes for $q_1, \cdots, q_m$; store the aggregate values as attributes $v_1, \cdots, v_m$ in $C_1, \cdots, C_m$ respectively (i.e., for any row $\phi$, $v_j(\phi) = q_j(D_\phi)$).
Let $M$ = the full outer join on the cubes $C_1, \cdots, C_m$.
Add a column in $M$ for $\mu_{interv}$; set it as $\mu_{interv}(\phi) = sign \ \times E(u_1 - v_1(\phi), \cdots, u_m - v_m(\phi))$ for each row $\phi$ ($sign = +1$ if $dir = low$, $sign = -1$ if $dir = high$).
Add a column in $M$ for $\mu_{aggr}$; set it as $\mu_{aggr}(\phi) = sign \times E(v_1(\phi), \cdots, v_m(\phi))$ for each row $\phi$ ($sign = +1$ if $dir = high$, $sign = -1$ if $dir = low$).
```

Optimizations. We employed several optimizations. The individual cubes $C_1, \cdots, C_m$ are materialized before they were joined to compute M. As illustrated in Example 4.1, the rows in the cubes will have `null` values for 'dont-care' attributes, which prevents us from using equi-join, because if both attribute A and B are null, then the equality $A = B$ fails. A naive solution will be to check for all combinations of null or not null, as in (`isnull A and isnull B`) `or` (`A = B`). To use equi-join, we replaced all `null` values in all the m cubes with a dummy value before performing the join.

4.3 Finding Minimal and Top Explanations

Once the table M is materialized by Algorithm 1, we can run top-K SQL queries to find explanations with high degrees (μ_{interv} or μ_{aggr}). However, blindly looking at the top-K outputs may result in some redundant answers. Consider Example 4.1, and assume that $Q = q_1$. Here all the explanations will have the same value for both μ_{interv}, μ_{aggr}. Although $\phi_1 = [\texttt{name} = RR, \texttt{inst} = \texttt{null}]$ and $\phi_2 = [\texttt{name} = \texttt{null}, \texttt{inst} = MS]$ denote two "minimal" explanations both having the same value for μ_{interv}, the explanation $\phi_3 = [\texttt{name} = RR, \texttt{inst} = MS]$ is dominated by both ϕ_1, ϕ_2 and therefore is redundant. Formally, we call an explanation ϕ *minimal* if there are no other explanation ϕ' such that $\mu_{interv}(\phi) \leq \mu_{interv}(\phi')$, and the $(attribute, value)$ pairs with $value \neq \texttt{null}$ for ϕ' is a subset of such pairs for ϕ (similarly for μ_{aggr}). We ignore the trivial explanation where all attributes take `null` value.

We implemented two strategies for finding minimal top-K explanations[12], which we empirically evaluated.

1. **Minimal-self join:** while outputting the top-K explanations, use a self-join on M to find the minimal explanations that are not dominated by others (the actual query is simple and is omitted).
2. **Minimal-append:** A dummy value is chosen such that it is > all valid values. Then a Top-1 query is run for K steps, that outputs explanations in decreasing order of μ_{interv} (or μ_{aggr}) which will give preference to shorter explanations having the same degree. To eliminate the non-minimal explanations, all explanations $\phi_1, \cdots, \phi_{i-1}$ output in the first $i - 1$ steps are appended to the Top-1 query in the i-th iteration in the WHERE clause as $(\neg\phi_1) \wedge \cdots \wedge (\neg\phi_{i-1})$.

5. EXPERIMENTS

We present our experimental results in this section. We describe qualitative and performance evaluation on the natality dataset in Section 5.1, and on the DBLP dataset in Section 5.2. The prototype of our system was built in Java with JDK 6.0 over Microsoft SQLServer 2012. All experiments were run locally on a 64-bit Windows 7 machine with Intel(R) Core^{TM} i7-2600 processor (4 GB RAM, 3.40 GHz).

5.1 Natality Dataset

The *natality dataset* contains all the births registered in the United States in 2010, and is a public dataset available from the National Center for Health Statistics (NCHS) through the Centers for Disease Control and Prevention (CDC), `http://www.cdc.gov/nchs/data_access/ftp_data.htm`. The dataset has 233 attributes, 4,007,106 anonymized entries, and 2.89GB size. The attributes are about the mother (age, race, marital status, education, smoking habit, disease information during pregnancy, etc), about the father, about the delivery process (in hospital or not, caesarian or not, time and type of prenatal care received during pregnancy), and about the health of the baby after birth (as *APGAR* score[18]). The APGAR score (`AP`) is an integer between 0 and 10; we define `AP` = `good` if `AP` $\in [7, 10]$, and `poor` if $\in [0, 6]$ [18]. Our goal is to "explain" the APGAR score (`AP`) in terms of the other attributes, and we report here two such experiments for qualitative and performance evaluation[13]:

(i) **APGAR Score vs. race (R) of the mother ($\mathbf{Q_{Race}}$)**: Figure 8 (details in Figure 7) shows that for R = *Asian*,

[12] Our definition of minimality prefers 'general' explanations with less conditions that a larger number of tuples satisfy. An alternative meaning of minimality can prefer 'specific' explanations with more conditions that a smaller number of tuples satisfy. Our techniques can support both definitions and the user will be able to specify her choice in our system.

[13] Our goal in these experiments was to efficiently visualize different conditions that characterize the predicates in the questions asked by the user, and not to establish any social, medical, or scientific claims.

AP	Race			
	White	Black	Am. Ind.	Asian
poor	49873	19600	1037	3058
good	3012046	613464	45580	242593

AP	Marital status	
	married	unmarried
poor	35961	35491
good	2322404	1591279

Figure 7: No. of tuples for Q_{Race} and $Q_{Marital}$

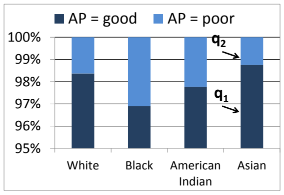


Figure 8: Plot for Q_{Race}

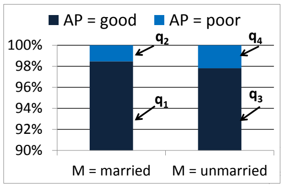


Figure 9: Plot for $Q_{Marital}$

the percentage of babies with AP = good is higher than the percentage of babies with AP = poor. For this observation, the user question is $(Q_{Race}, high)$, where $Q_{Race} = \frac{q_1}{q_2}$; q_1, q_2 are count(*) queries for [AP = good, R = *Asian*] and [AP = poor, R = *Asian*]. A more interesting question may be why this ratio (AP = good vs. poor) is higher for R = *Asian* than for R = *Black*, which can be easily captured as another question $(Q'_{Race}, high)$, $Q'_{Race} = (\frac{q_1}{q_2})/(\frac{q_3}{q_4})$, here q_3, q_4 are count(*) queries similar to q_1, q_2.

(ii) APGAR Score vs. marital status (M) of the mother ($\mathbf{Q_{Marital}}$): Figure 9 (with details in Figure 7) shows that the ratio of AP = good to AP = poor is higher for M = married, than for M = unmarried. For this observation, the user question is $(Q_{Marital}, high)$, where $Q_{Marital} = \frac{q_1}{q_2}/\frac{q_3}{q_4}$; q_1, q_2, q_3, q_4 respectively count the entries in the database that satisfy (i) M =married, AP = good, (ii) M =married, AP = poor, (iii) M =unmarried, AP = good, and (iv) M =unmarried, AP = poor.

5.1.1 Qualitative Evaluation

We restricted the search space of candidate explanations to five attributes (recoded in groups): (1) Age (A), (2) whether the mother used tobacco during pregnancy (T), (3) the month when prenatal care began (1st/2nd/3rd trimester or none) (PN), (4) level of education (Edu), (5) marital status of the mother (M) for Q_{Race} and race (R) for $Q_{Marital}$. We chose a threshold such that at least one of the aggregate queries q_j has value ≥ 1000. We also added a small threshold of 0.0001 to all counts to avoid any division by zero error in the calculation of the ratios.

$\mathbf{Q_{Race}}$

	explanation	$\mu_{\mathbf{interv}}$
1	M=married	-62.3
2	PN=1st trim.	-75.1
3	T=non smoking	-75.3
4	Edu>=16 yrs.	-75.6
5	A=30-34 yrs.	-75.8

$\mathbf{Q_{Marital}}$

	explanation	$\mu_{\mathbf{interv}}$
1	Edu> 16yrs	-1.381
2	A=30-34 yrs	-1.387
3	PN=1st. trim, T=non smoking	-1.396
4	A= 15-19 yrs	-1.397
5	PN=1st. trim	-1.405

Figure 10: Top-5 (minimal) explanations by intervention for Q_{Race} and $Q_{Marital}$.

Explanation by intervention for Q_{Race} and $Q_{Marital}$. The top-5 (minimal) explanations are shown in Figure 10. They are very general containing only one or two attribute, and therefore have high support (high counts for $q_1, q_2, \cdots$). By removing the intervention Δ, the value of $Q_{Race}(D - \Delta^{\phi})$ becomes smaller than the original value $Q_{Race}(D)$ = 79.3. Intuitively, the top-5 answers say that the query's output can be explained by the fact that Asian mothers are not too young, married, non-smoking, received higher education, and received early prenatal care; because, if they were removed from the population, the value of Q_{Race} becomes smaller. Similarly, the original value of $Q_{Marital}(D)$ was 1.46, and the top explanations lower this value either by decreasing $\frac{q_1}{q_2}$ more (removing older, educated, non-smoking mothers reduces this fraction), or by increasing $\frac{q_3}{q_4}$ more (removing younger mothers will affect the unmarried population more than the married population). We get similar observations for Q'_{Race}, the details are omitted due to lack of space.

Explanation by aggravation for Q_{Race} and $Q_{Marital}$. Figure 11 shows the top-3 explanations by aggravation. Similar to explanations by intervention, for Q_{Race}, mothers who are not too young, do not smoke, highly educated, married, and received early prenatal care appear in the top explanations. However, these explanations are more specific (conjunction of multiple attributes), and therefore their support (counts) are smaller (a few thousands). If the queries q_1, q_2 are restricted to these sub-populations, the value of Q_{Race} becomes much higher than the original value of 79.3. On the other hand, for $Q_{Marital}$, the two top explanations make the value of $\frac{q_1}{q_2}$ infinity (∞) since $q_2 = 0$.

5.1.2 Performance Evaluation

Our framework to output top explanations consists of two main steps: (i) compute the degrees of explanations for all candidate explanations and store them in a table M (ref. Section 4.2), and (ii) given M, compute the *minimal* top-k explanations (ref. Section 4.3). In this section we evaluate different approaches for these two steps varying the following parameters: (a) *the size of input data* – to vary input data size, we selected and stored $x\%$ entries from the natality table at random and varied x from 0.01 (only ~400 rows) to 50 (2M rows); (b) *the number of attributes used in explanation predicates* – the attributes are multi-valued (with 2-9 distinct values), more attributes lead to more candidate explanations; and (c) *complexity of the user question* – this is number of select-aggregate queries $q_1, q_2, \cdots$ used in the questions. We considered Q_{Race} and $Q_{Marital}$, which involve two (q_1, q_2) and four $(q_1, \cdots, q_4)$ such queries respectively. In addition, for (ii), we vary k to evaluate the approaches to output minimal explanations discussed in Section 4.3.

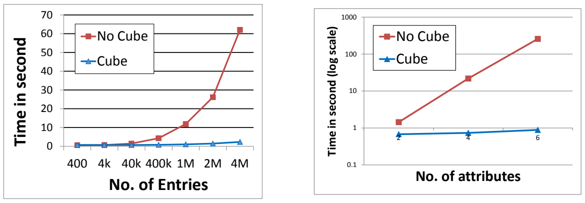


Figure 12: Benefits of data cube optimization for Q_{Race}: (a) Data size vs. time (with two attributes), (b) No. of attributes vs. time (with 1% entries)

Benefits of data cube optimization. Figure 12 compares the naive evaluation that iterates over all combinations

$\mathbf{Q}_{\mathbf{Race}}$

	explanation	$\mu_{\mathbf{aggr}}$
1	M=married, Edu≥16 yrs, PN=2nd trim.	174
2	M=married, A=40–44 yrs, Edu≥ 16 yrs, PN=1st trim.	173
3	Edu≥16 yrs, PN=2nd trim., T=non smoking	172

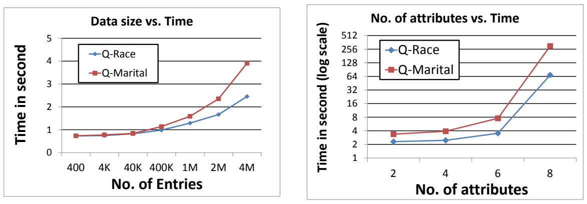
$\mathbf{Q}_{\mathbf{Marital}}$

	explanation	μ_{aggr}
1	A=15–19 yrs, Edu=9–11 yrs, PN=3rd trim.	$1.36\times10^4(\infty)$
2	A<15 yrs	$1.23\times10^4(\infty)$
3	Edu = 9–11 yrs, PN=no prenatal care, T=non smoking	3.2

Figure 11: Top-3 (minimal) explanations by aggravation for Q_{Race} and $Q_{Marital}$.

of attributes used in the explanations (referred to as 'No Cube'), v.s. Algorithm 1 (referred to as 'Cube'), for the query Q_{Race}. In the first graph we vary the input data size (the number of entries on the horizontal axis are approximate), whereas in the second graph we vary the number of attributes used in the explanations. Both the graphs show that the benefit from the data cube is dramatic. In the remaining experiments we only discuss the data cube implementation.

Size of the data vs. time. Figure 13a shows the time required to compute the degrees for all candidate explanations (*i.e.*, table M) for Q_{Race} and $Q_{Marital}$ as a function of the size of the input database. The same four attributes A, T, PN, BP were used as relevant attributes in the explanations for both the queries. As expected, the time taken by the queries increases with the database size. Recall that to compute table M, Algorithm 1 first computes the cubes $C_1, C_2, \cdots$ for $q_1, q_2, \cdots$, and then computes M by doing full outer joins on $C_1, C_2, \cdots$. Hence $Q_{Marital}$ requires more time, because compared to two queries q_1, q_2 in Q_{Race}, $Q_{Marital}$ has four such queries. However, the time taken by both queries is only < 4 seconds even for the entire dataset, making it suitable for an interactive setting.



(a) Data size vs. time (b) No. of attributes vs. time

Figure 13: Time to compute degrees for all explanations

Number of attributes in explanation predicates vs. time. Figure 13b shows the time to compute the degrees of explanations as a function of the number of attributes used in the explanations. Here we used the entire dataset (4M entries) and added four other attributes to A, T, BN, BP (mother's education, sex of the infant born, whether the mother had hypertension/diabetes). The time is shown in log-scale; as expected, the time increases rapidly with the number of attributes. The cube algorithm can run up to six attributes within a few seconds, afterward the time increases due to increase in the number of candidate explanations. For eight attributes there are $> 71K$ candidate explanations for Q_{Race} and $> 192K$ candidate explanations for $Q_{Marital}$ joining (resp. two and four) cubes with $33k - 78K$ rows. The cubes $C_1, C_2, \cdots$ and the table M that contains all explanations and their degrees are small (< 3.8 MB for Q_{Race} and < 11.32 MB for $Q_{Marital}$).

Outputting 'minimal' top-K explanations. Figure 14 shows the time taken to compute the top-K explanations by μ_{interv} for Q_{Race} on the entire natality dataset

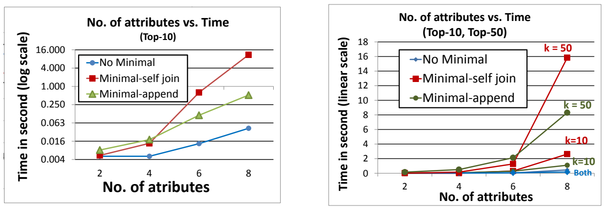


Figure 14: Time to compute *minimal* top-K explanations

where we again vary the number of attributes on the horizontal axis (since this is run on the stored table M, the behavior for $Q_{Marital}$ and μ_{aggr} is similar). The plots compare three top-K strategies, (i) **No Minimal:** simply output the top-K answers sorted by μ_{interv} by a SQL top-K query, (ii) **Minimal-self join**, and (iii) **Minimal-append**, as discussed in Section 4.3. Clearly, the *No Minimal* approach takes the least time, but can output redundant explanations (the explanation ranked 5 in Figure 10 is the 14th explanation if we do not enforce minimality). *Minimal-self join* performs slightly better than *Minimal-append* when the number of relevant attributes is small (since computing the self-join on M is less expensive than running a top-1 query K times). However, for larger value of the number of relevant attributes *Minimal-append* performs much better (takes < 1 sec for K = 10 even with eight attributes).

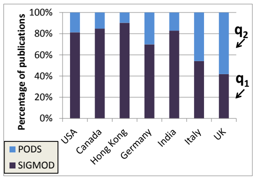
5.2 DBLP Dataset

We also experimented on a database obtained by integrating the popular DBLP data [17], from which we extracted the tables Author (1.095M entries), Authored (5.127M entries), and (conference) Publication (1.088M entries), with Geo-DBLP, a research project from another research group [4], which includes affiliation information (affiliation, city, and country) for every publication obtained by crawling ACM[14] Our goal was to validate our system's ability to explain queries over integrated datasets. We have already described one such query in Section 1 (Figures 1 and 2); here we discuss another experiment.

The graph Figure 15a shows the percentage of SIGMOD and PODS publications by country, between 2001 and 2011 (note that the country information is not available from all publications). As expected, there are fewer PODS papers than SIGMOD papers from most countries: however, surprisingly, more than 50% of the papers from the UK are in PODS. We used our system to search for explanations.

We express this as a user question (Q, low), where $Q = \frac{q_1}{q_2}$, and q_1, q_2 respectively count the number of distinct SIGMOD and PODS publications from UK between 2001 and 2011. Due to limited availability of the country information, we express publications from UK as $[\texttt{domain} =' uk' \vee \texttt{country} =' United\ Kingdom']$, where *domain* is extracted

[14] The affiliation in Geo-DBLP is not normalized, and may contain several formats like 'Univ.', 'U.' or 'University' for the same institute as well as joint affiliation using 'and'.



(a)

	explanation	μ
1	`name`=L. Libkin	1.04
2	`name`= G. Gottlob	0.81
3	`name`= F. Geerts	0.74
4	`name`= M. Benedikt	0.74
5	`city`= Edinburgh	0.73
6	`city`= Oxford	0.72
7	`inst`= Oxford Univ.	0.69

(b)

Figure 15: (a) Plot for percentage of SIGMOD and PODS publications in different countries between 2001 and 2011, (b) Top explanations by intervention μ_{interv}

from homepage urls in DBLP, and *country* is from Geo-DBLP. Therefore both q_1, q_2 involve equi-join of eight tables (three from DBLP and five from Geo-DBLP). The relevant attributes considered are *Author.name*, *Author$_G$.inst*, *Affiliation$_G$.city* (*Author* is from DBLP, whereas *Author$_G$ and Affiliation$_G$ are from Geo-DBLP*).

The top explanations by intervention are shown in Figure 15b which shows researchers from UK who published a lot more in PODS in 2001-2011 than in SIGMOD, and the universities in UK (researchers who published both in SIGMOD and PODS do not appear in the top explanations). The explanations [`city` = $Oxford$] has higher value of μ_{interv} than [`inst` = $Oxford\, Univ.$] due to the presence of [`inst` = *Semmle Ltd.*] in Oxford city, and different format of the university in the database ('Oxford', 'University of Oxford' etc.). The top explanations by aggravation are similar and therefore omitted. With three relevant attributes, the time taken to compute and materialize the table M (ref. Algorithm 1) is 2.176 seconds; although eight tables were joined in the cube operation, three having > $1M$ rows, the time taken is small possibly due to low selectivity of the queries q_1, q_2 and use of indexes on the primary keys of all the tables. The time taken to top-50 answers by *Minimal -self join* is < 4 ms (the joined table M has < 1000 entries).

6. DISCUSSIONS

We proposed a formal framework for finding explanations for database queries. Our approach is based on the notion of intervention from the causality literature, and exploits foreign key constraints in a novel way in order to describe causal paths in the data. We proved a key property, that every explanation has a unique minimal intervention, and gave a recursive query that can compute this intervention; we further showed that, under certain conditions, recursion can be avoided and the degrees of all candidate explanations can be efficiently computed using SQL's data cube operation. We evaluated our approach on two real datasets, showing that our techniques find interesting explanations and can find them efficiently.

Some extensions are required to expand the applicability of explanations, and we incorporate some of these in the system that we are currently developing. (i) **Beyond data cube:** The data cube optimization is highly effective when it applies, but when it fails to apply then naive iterative algorithm is too slow. We are currently investigating optimizations to the iterative algorithm that can be deployed for general queries. **(ii) Explanations with inequalities, and disjunctions:** In some applications meaningful explanations include inequalities (e.g. papers with `year` > 1977 ∧ `year` < 1982) and disjunctions (e.g. `author` = `Levy` ∨ `author` = `Halevy`) Conceptually, our framework can support such explanations, but adding them would increase the search space of explanations, making query evaluation and ranking more challenging. **(iii) Intervention vs. aggravation:** Unlike explanation by intervention, explanation by aggravation can always be computed using the data cube, without any restrictions. However, they completely ignore causal path, hence they are useful only in simple schemas that have few, or no foreign keys. In future work we consider a hybrid definition between intervention and aggravation that uses some, but not all causal path,a nd can always be evaluated by the data cube. (iv) **Beyond count queries** and `more complex user questions`**:** Complex questions are supported in our framework, and we are currently implementing them in our prototype. For example, *why is this sequence of bars increasing(decreasing)*, translates into *why is the slope of the linear regression of these datapoints positive(negative)?*, which can easily converted into a numerical query as in Eq.(1). A challenge, however, is designing a effective user interface that allows users to ask such complex questions.

7. REFERENCES

[1] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases.* Addison-Wesley, 1995.

[2] M. Das, S. Amer-Yahia, G. Das, and C. Yu. Mri: Meaningful interpretations of collaborative ratings. *PVLDB*, 4(11):1063–1074, 2011.

[3] D. Fabbri and K. LeFevre. Explanation-based auditing. *Proc. VLDB Endow.*, 5(1):1–12, Sept. 2011.

[4] F. Hadiji, K. Kersting, C. Bauckhage, and B. Ahmadi. Geodblp: Geo-tagging dblp for mining the sociology of computer science. *CoRR*, abs/1304.7984, 2013.

[5] B. Kanagal, J. Li, and A. Deshpande. Sensitivity analysis and explanations for robust query evaluation in probabilistic databases. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*, SIGMOD '11, pages 841–852, New York, NY, USA, 2011. ACM.

[6] N. Khoussainova, M. Balazinska, and D. Suciu. Perfxplain: debugging mapreduce job performance. *Proc. VLDB Endow.*, 5(7):598–609, Mar. 2012.

[7] A. Meliou, W. Gatterbauer, K. F. Moore, and D. Suciu. The complexity of causality and responsibility for query answers and non-answers. *PVLDB*, 4(1):34–45, 2010.

[8] A. Meliou, W. Gatterbauer, S. Nath, and D. Suciu. Tracing data errors with view-conditioned causality. In *SIGMOD Conference*, pages 505–516, 2011.

[9] M. Pacer, T. Lombrozo, T. Griffiths, J. Williams, and X. Chen. Evaluating computational models of explanation using human judgments. In *UAI*, 2013.

[10] J. Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988.

[11] J. Pearl. *Causality: models, reasoning, and inference.* Cambridge University Press, 2000.

[12] J. Pearl. An introduction to causal inference. *The International Journal of Biostatistics*, 6, 2010.

[13] S. Sarawagi and G. Sathe. i3: intelligent, interactive investigation of olap data cubes. In *SIGMOD*, 2000.

[14] G. Sathe and S. Sarawagi. Intelligent rollups in multidimensional olap data. In *VLDB*, 2001.

[15] C. Silverstein, S. Brin, R. Motwani, and J. Ullman. Scalable techniques for mining causal structures. *Data Min. Knowl. Discov.*, 4(2-3):163–192, July 2000.

[16] J. D. Ullman. *Principles of Database and Knowledge-Base Systems: Volume II: The New Technologies.* W. H. Freeman & Co., New York, NY, USA, 1990.

[17] `http://dblp.uni-trier.de/xml`.

[18] `http://en.wikipedia.org/wiki/Apgar_score`.

[19] E. Wu and S. Madden. Scorpion: Explaining away outliers in aggregate queries. *PVLDB*, 6(8), 2013.

[20] C. Yuan, H. Lim, and M. L. Littman. Most relevant explanation: computational complexity and approximation methods. *Ann. Math. Artif. Intell.*, 61(3):159–183, 2011.