GDPView: An Interactive Interface that Unifies Variants of Deletion Propagation (DP)

Yihan Luo
Northeastern University, USA
luo.yih@northeastern.edu

Neha Makhija
University of Massachusetts Amherst, USA
nehamakhija@umass.edu

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

ABSTRACT

After a query has been run, analysts often want to ask reverse questions: which input tuples must change to make selected output tuples disappear, preserve others, and limit side-effects?

GDPView is an interactive workbench for this kind of reverse reasoning, focused on deletion propagation (DP) for conjunctive queries. Using four user-facing primitives (Delete, Preserve, Minimize, and Maximize), attendees can express and compare classical tasks such as resilience, deletion propagation with source or view side-effects, aggregated deletion propagation, smallest witness problems, and new mixed objectives within a single unified interface. Attendees can explore curated challenge scenarios or create and edit example scenarios. These examples are designed for fast, hands-on exploration in a live demo setting. The system also includes an optional explanation panel that exposes the underlying provenance and ILP formulation.

PVLDB Reference Format:
Yihan Luo, Neha Makhija, and Wolfgang Gatterbauer. GDPView: An Interactive Interface that Unifies Variants of Deletion Propagation (DP). PVLDB, 19(12): 4670 - 4673, 2026.
doi:10.14778/3827998.3828093

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/northeastern-datalab/gdp-demo.

1 INTRODUCTION

Deletion Propagation (DP) is a classic problem in database theory: *If we want to delete certain tuples from a view, which tuples in the base database should be deleted to achieve this goal optimally?* In other words, deletions specified at the view level must be *propagated back* to the input. Early work on view updates provides the broader context for this line of research [3]. Over time, different notions of optimality and additional side constraints have led to a range of DP variants, often studied in isolation and accompanied by separate algorithmic and complexity results [2, 5, 8–10]. These variants arise naturally in data-management tasks involving analysis, debugging, and explainability.

Our recent PVLDB 2025 paper [11] unifies all previously studied variants of DP within a single framework, called Generalized Deletion Propagation (GDP), and also introduces a broader family of new, well-motivated problems. The central insight is that these variants can all be viewed as instances of a single family of combinatorial optimization problems where constraints specify the desired effect on the view together with any additional user requirements, while an optimization objective defines what counts as an optimal solution. This unification can be expressed through just 4 primitive abstractions: Delete, Preserve, Minimize, and Maximize. A key technical result is that our algorithm is *coarse-grained instance-optimal*: it runs in polynomial time on all currently known tractable query classes for the full range of individual DP problems that were previously addressed only by separate, specialized algorithms [2, 5, 8–10].

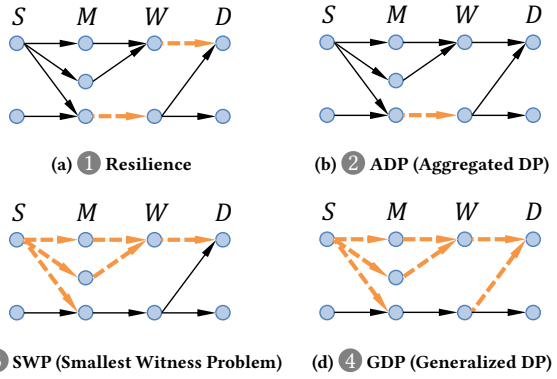

Figure 1: Example 1: A truck network in which nodes represent locations (Suppliers S, Manufacturers M, Warehouses W, and Distributors D), and the 10 directed edges represent truck routes. Orange dashed edges indicate the routes removed in optimal solutions to 4 variants of Deletion Propagation (DP), labeled 1–4.

We present GDPView, an interactive workbench for exploring and applying deletion propagation variants that is built on the unified GDP formulation of our PVLDB 2025 paper. Using the 4 primitives, users can define views, specify desired effects on query outputs, impose deletion and preservation constraints, and choose optimization objectives; the system then computes the corresponding optimal source-side deletions. Thus, a single interface supports many named DP problems that were previously studied separately. We illustrate this next through an example.

Example 1 (Supply Chain Management). *Consider a supply-chain company that ships goods by truck from Suppliers (S), via Manufacturers (M) and Warehouses (W), to Distributors (D). These routes form the partite transportation network shown in Fig. 1: nodes denote locations, and edges denote truck routes. The same network can*

Proceedings of the VLDB Endowment, Vol. 19, No. 12 ISSN 2150-8097.
doi:10.14778/3827998.3828093

	Delete	Preserve	Minimize deletions	Maximize deletions
DP with Source side-effects (DP-SS) [2, 5]	ALL in output view		in input views	
DP with View side-effects (DP-VS) [10]	ALL in output view		in output view	
Aggregated DP (ADP) [9]	ANY-k in output view		in input views	
Smallest Witness Problem (SWP) [8]		ALL in output view		in input views
Generalized DP (GDP) [11]	ANY-k in arbitrary views	ANY-k in arbitrary views	in arbitrary views	in arbitrary views

Table 1: A summary of Deletion Propagation (DP) variants and the constraints modeled by them. Generalized Deletion Propagation is a recently posed variant that captures prior DP variants as special cases. It is a generalization in multiple senses as it: 1) allows constraints on a view different from the original input and output, 2) allows each constraint to be enforced over multiple views, 3) allows a combination of hard constraints (Delete, Preserve) and soft constraints (Minimize, Maximize).

also be represented as a simple chain query with binary relations corresponding to the edges. To assess the robustness and efficiency of its supply chain, the company may ask the following questions, each of which is a variant of deletion propagation (DP):

❶ What is the minimum number of routes whose removal would prevent all distributors from receiving goods from any supplier? *This is an instance of the* ***Resilience Problem*** *[5].*[1] *The resilience is 2 in our example since removing the two highlighted truck routes in Fig. 1a disconnects all suppliers from all distributors.*

❷ What is the minimum number of routes whose removal would prevent at least 50% of the distributors from receiving goods from any supplier? *This is an instance of* ***Aggregated Deletion Propagation (ADP)*** *[9]. Deleting the highlighted truck route shown in Fig. 1b is optimal, since it causes 1 of the 2 distributors (50%) to become unreachable.*

❸ What is the maximum number of routes that can be removed while still allowing all distributors to receive goods from some supplier? *This is an instance of the* ***Smallest Witness Problem (SWP)*** *[8]. An optimal solution deletes the 6 highlighted routes in Fig. 1c, which still preserves at least one Supplier-to-Distributor path for every Distributor.*

❹ What is the maximum number of routes that can be removed while still allowing at least 50% of the distributors to receive goods from some supplier? *This problem can be viewed as an aggregated preservation variant. Although it has not been studied explicitly under that name, it is covered by the framework of* ***Generalized Deletion Propagation (GDP)*** *[11]. An optimal solution deletes the 7 routes highlighted in Fig. 1d, which preserves reachability for 1 of the 2 distributors (50%).*

Contributions. We present GDPView, a hands-on demo system that lets VLDB attendees experiment with deletion propagation (DP) through an interactive interface. GDPView implements our framework of Generalized Deletion Propagation (GDP) [11], which unifies all previously studied deletion propagation variants [2, 3, 5, 8–10] and also supports new, practically motivated problem formulations. Rather than viewing DP only as an abstract optimization problem, users can interact with it directly: they can upload data, define views, specify deletion and preservation constraints, and inspect the resulting optimal solutions.

[1]Note that resilience can be expressed as a special case of DP with source side-effects (DP-SS) (as shown in Table 1) over a boolean (existential) query, where the goal is to make the boolean query false, and minimize deletions in the input.

2 GENERALIZED DELETION PROPAGATION (GDP) AND RELATED WORK

Generalized Deletion Propagation (GDP) [11] is a recent framework that encapsulates prior known deletion propagation variants as special cases, including: deletion propagation with source [2, 5] and view side-effects [10], aggregated deletion propagation [9], and the smallest witness problem [8] (as summarized in Table 1). We rephrase the definition of GDP in a manner that is more suitable for a system demo.[2]

Definition 2.1 (Generalized Deletion Propagation [11] (reformulated, informal)). Given views V defined by conjunctive queries over a database D, hard and soft constraints C over those views, and an optimization objective, GDP asks for a set T of input tuples whose deletion from D satisfies C. The hard constraints in C can be deletion or preservation constraints on a certain number of tuples in the views, while the soft constraints require minimization or maximization of the number of tuples deleted from a view.

Deletion Propagation problems can in turn be seen as subclasses of Reverse Data Management [12], which asks the more general question of *"Assume we want to achieve a certain property in an output view, what interventions should we perform on the database to accomplish this goal in an optimal manner?"* Reverse data management problems encapsulate many types of analytical reasoning tasks, such as hypothetical reasoning [4], what-if queries [14], and how-to queries [6]. A related demo system is Tiresias [13], which solves how-to problems, a type of reverse data management problem using Mixed Integer Programming (MIP). Tiresias supports how-to queries involving broader changes to input values, whereas GDPView focuses specifically on tuple deletions and propagation constraints, providing runtime guarantees shown in [11].

3 THE GDPView INTERFACE

The goal of the GDPView interface is to provide a user-friendly platform for exploring and experimenting with different deletion propagation scenarios. The demo aims to show that a wide variety of deletion propagation problems can be expressed with just 4 simple primitive actions (Delete, Preserve, Minimize, and Maximize) on views defined by conjunctive queries.

Figure 2 provides an overview of the GDPView interface. Window 1 shows how users define their data, views, and constraints,

[2]The original formal definition defines four different sets of views (one for each of the four primitives), while in the demo we simply look at a single set of views and allow the user to apply any of the four primitives to any view.

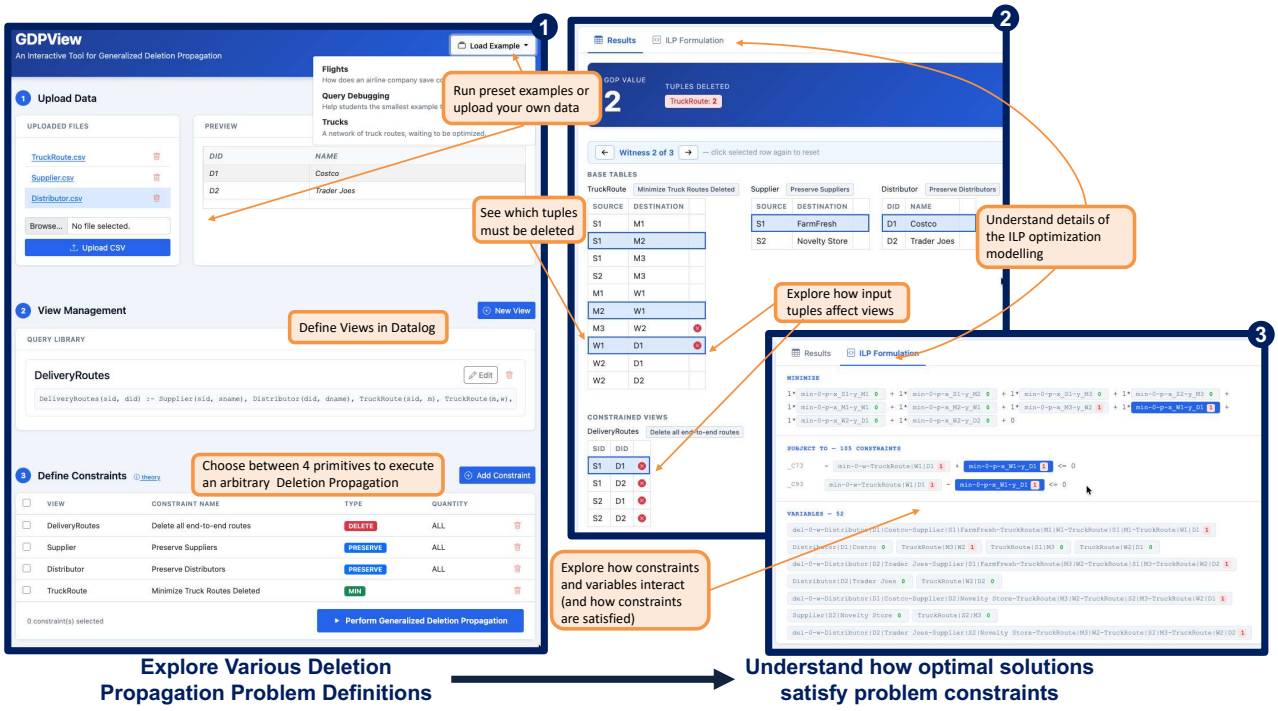

Figure 2: Overview of the GDPView interface, highlighting key interactions between the user and the system.

while Window 2 and Window 3 allow users to view the results of the deletion propagation and the underlying ILP formulation, respectively. Figure 3 illustrates the workflow of the system, covering various possible interactions that a user can have with the system, including defining the data, views, and constraints, running the optimization, and understanding the results. We highlight the functionality of the system by walking through 3 user interaction scenarios (Sections 3.1 to 3.3) that can be explored, using the truck network from Example 1 as a running example.

3.1 User Interaction 1: Preset Examples

For ease of use, the system provides several preset examples from different domains that users can load to explore different deletion propagation scenarios. A preset example includes the data, view definitions, and constraints for a specific deletion propagation scenario, allowing users to quickly see the system in action without having to define a problem from scratch.

For example, in line with Example 1, users can load the truck network preset example. They can preview the toy data uploaded as tables, and can also observe the query that defines the DeliveryRoutes view (as shown in Window 1 of Fig. 2). They can next read the constraints that are defined on the views (and/or constraints on the base tables). In the preset example, the constraints imposed are to DELETE ALL rows in the DeliveryRoutes view (i.e., all supplier-Distributor routes must be removed), while it must PRESERVE ALL rows in the Supplier and Distributor base tables (i.e., no supplier or Distributor can be removed), and MINIMIZE the number of deletions in the TruckRoutes base table (i.e., minimize the number of routes removed). This corresponds exactly to scenario 1 in Fig. 1 and the DP-SS problem in Table 1.[3] The visitor clicks *Run* and GDPView identifies the 2 routes whose removal disconnects the entire network by marking them with a cross (visible in Window 2 of Fig. 2). The results tab also uses a brushing and linking interaction to show how each output tuple is affected by the input - when the user clicks on an output row, all the corresponding input rows that contribute to it are highlighted, and the user can cycle through each witness (unique Supplier-to-Distributor route) that leads to the specific Supplier-Distributor pair being in the query result. Thus, users can see that in order to remove an output tuple from the view, each of the witnesses for that output tuple must be removed, and at least one of the corresponding input tuples must be deleted.

3.2 User Interaction 2: Exploring DP Variants

The user is not limited to simply looking at the results of the preset example, but can also upload their own data, define queries and modify constraints to explore how the results change. Data can be uploaded in CSV format per table, and views can be defined using Datalog queries in the view management panel. Constraints can be defined on the views and base tables using the constraint management panel. For example, they can change the DELETE constraint on the DeliveryRoutes view from ALL to ANY 1 (equivalently in this example, ANY 50%) of all the Supplier-Distributor pairs, which corresponds to the ADP problem in Table 1 (scenario 2 in Fig. 1). A single change to the constraints transforms the problem from DP-SS to ADP, and GDPView identifies a different solution: now only 1

[3] If modeled as a DP-SS problem instead of a GDP problem, the two preservation constraints here would be modeled by marking the Supplier and Distributor tables as *exogenous*, which simply means no tuples from them can be deleted. We can capture such additional terminology with the four basic primitives (in this case, with PRESERVE).

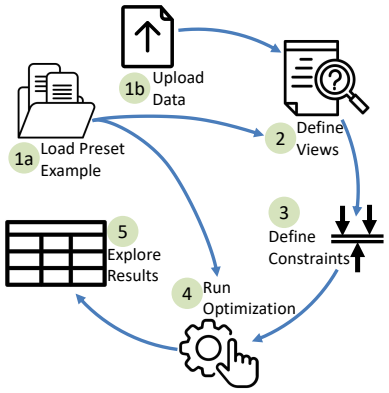

Figure 3: Workflow of GDPView, illustrating how a user can define their deletion propagation problem step by step.

route needs to be removed to disconnect at least 1 Distributor from all suppliers. Similarly, now adding a constraint to PRESERVE ANY 1 Supplier-Distributor pair and DELETE ANY 1 Supplier-Distributor pair transforms the problem into a novel problem not previously studied in isolation, yet expressible in GDPView with one additional constraint. This highlights how just the 4 primitive actions on views defined by conjunctive queries can express a wide variety of deletion propagation problems.

3.3 User Interaction 3: Understanding Optimization Results

A key aspect of the system is to not only provide the results of the deletion propagation, but also to allow users to understand how the results were computed and how different constraints and objective functions affect the results. The ILP overview (Window 3 of Fig. 2) allows users to see the underlying ILP formulation that is used to compute the results. Each ILP variable is labelled with the optimal value it takes in the solution, allowing users to see how the ILP solution corresponds to the deletion propagation solution, and how each constraint is satisfied (and how the objective is calculated). A brushing and linking interaction allows users to explore the constraints each variable is involved in, by highlighting all other occurrences of a variable when the user clicks on it. The interaction focuses the ILP tab on the constraints that variable is involved in, allowing a user to explore step by step how the ILP formulation captures the constraints defined by the user, and how the different constraints interact with each other to produce the final solution.

4 SYSTEM ARCHITECTURE

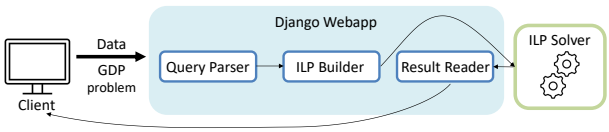

Figure 4: Section 4: System Architecture of GDPView.

A high-level overview of the system architecture is shown in Fig. 4. GDPView is a Django web application written in Python. The users share their data and specifications through the web interface (detailed in Section 3). We use a custom Datalog parser and provenance calculator to parse user queries, and compute query results along with its provenance. In a full system version, we would like to integrate techniques for computing provenance at scale [15] and extend to DP problems beyond conjunctive queries [1].

The ILP builder builds an ILP formulation based on the user-defined views and constraints in a manner outlined in [11]. The ILP solver used in the system is Gurobi [7] (with a free academic license), but the system can be extended to use other ILP solvers as well. We note that the GDP problem is NP-Hard in general, but can be solved in polynomial time for certain tractable classes of queries [11]. Our system uses the same techniques that guarantee polynomial time termination for the tractable classes of queries, and we refer the reader to [11] for experimental evidence of scalability of the system on larger datasets. The GDP optimization always runs on the server, returning the optimization objective and tuples to be deleted. However, since the task of exploring and understanding the problem is better suited to smaller examples, we enable ILP explanation features only for examples under a certain configured size to help users understand the solution and how it is derived.

Acknowledgements. This work was supported in part by the National Science Foundation (NSF) under award number IIS-1762268.

REFERENCES

[1] Antoine Amarilli, Wolfgang Gatterbauer, Neha Makhija, and Mikaël Monet. 2025. Resilience for Regular Path Queries: Towards a Complexity Classification. *PACMMOD (PODS)* 3, 2, Article 108 (2025), 18 pages. doi:10.1145/3725245

[2] Peter Buneman, Sanjeev Khanna, and Wang-Chiew Tan. 2002. On Propagation of Deletions and Annotations Through Views. In *PODS*. 150–158. doi:10.1145/543613.543633

[3] Umeshwar Dayal and Philip A. Bernstein. 1982. On the Correct Translation of Update Operations on Relational Views. *ACM TODS* 7, 3 (1982), 381–416. doi:10.1145/319732.319740

[4] Daniel Deutch, Yuval Moskovitch, and Noam Rinetzky. 2019. Hypothetical Reasoning via Provenance Abstraction. In *SIGMOD*. 537–554. doi:10.1145/3299869.3300084

[5] Cibele Freire, Wolfgang Gatterbauer, Neil Immerman, and Alexandra Meliou. 2015. The Complexity of Resilience and Responsibility for Self-Join-Free Conjunctive Queries. *PVLDB* 9, 3 (2015), 180–191. http://www.vldb.org/pvldb/vol9/p180-freire.pdf

[6] Sainyam Galhotra, Amir Gilad, Sudeepa Roy, and Babak Salimi. 2022. HypeR: Hypothetical Reasoning With What-If and How-To Queries Using a Probabilistic Causal Approach. In *SIGMOD*. 1598–1611. doi:10.1145/3514221.3526149

[7] LLC Gurobi Optimization. 2022. Gurobi Optimizer Reference Manual. http://www.gurobi.com

[8] Xiao Hu and Stavros Sintos. 2024. Finding Smallest Witnesses for Conjunctive Queries. In *ICDT (LIPIcs, Vol. 290)*. 24:1–24:20. doi:10.4230/LIPIcs.ICDT.2024.24

[9] Xiao Hu, Shouzhuo Sun, Shweta Patwa, Debmalya Panigrahi, and Sudeepa Roy. 2020. Aggregated Deletion Propagation for Counting Conjunctive Query Answers. *PVLDB* 14, 2 (2020), 228–240. http://vldb.org/pvldb/vol14/p228-hu.pdf

[10] Benny Kimelfeld, Jan Vondrák, and Ryan Williams. 2012. Maximizing Conjunctive Views in Deletion Propagation. *ACM TODS* 37, 4, Article 24 (2012), 37 pages. doi:10.1145/2389241.2389243

[11] Neha Makhija and Wolfgang Gatterbauer. 2025. Is Integer Linear Programming All You Need for Deletion Propagation? A Unified and Practical Approach for Generalized Deletion Propagation. *PVLDB* 18, 8 (2025), 2667–2680. https://www.vldb.org/pvldb/vol18/p2667-makhija.pdf

[12] Alexandra Meliou, Wolfgang Gatterbauer, and Dan Suciu. 2011. Reverse Data Management. *PVLDB* 4, 12 (2011), 1490–1493. http://www.vldb.org/pvldb/vol4/p1490-meliou.pdf

[13] Alexandra Meliou and Dan Suciu. 2012. Tiresias: the database oracle for how-to queries. In *SIGMOD*. 337–348. doi:10.1145/2213836.2213875

[14] Haneen Mohammed, Alexander Yao, Charlie Summers, Hongbin Zhong, Gromit Yeuk-Yin Chan, Subrata Mitra, Lampros Flokas, and Eugene Wu. 2024. FaDE: More Than a Million What-Ifs Per Second. *PVLDB* 18, 4 (2024), 943–955. https://www.vldb.org/pvldb/vol18/p943-mohammed.pdf

[15] Pierre Senellart. 2018. Provenance and Probabilities in Relational Databases. *SIGMOD Rec.* 46, 4 (Feb. 2018), 5–15. doi:10.1145/3186549.3186551