DATALAB
@Northeastern

A unified and practical approach to Generalized Deletion Propagation

Wolfgang Gatterbauer, joint work with **Neha Makhija**

https://northeastern-datalab.github.io/unified-reverse-data-management/

Logic and Algorithms in Database Theory and AI Reunion
Jan 15, 2024

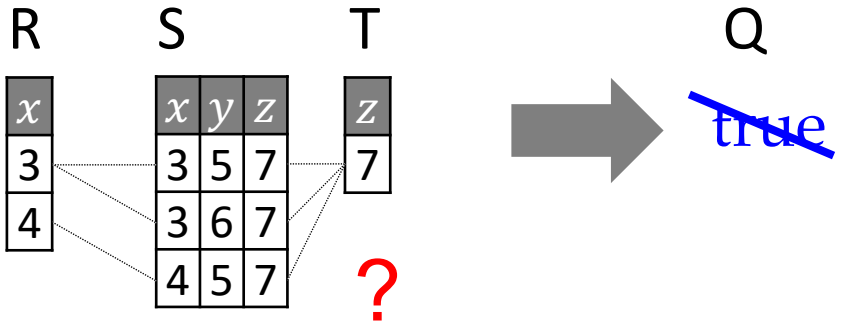
https://simons.berkeley.edu/workshops/logic-algorithms-database-theory-ai-reunion

Example Reverse Data Management Problems

Q:-R(x),S(x,y,z),T(z)

1. Resilience

Delete min number of tuples to make Q false

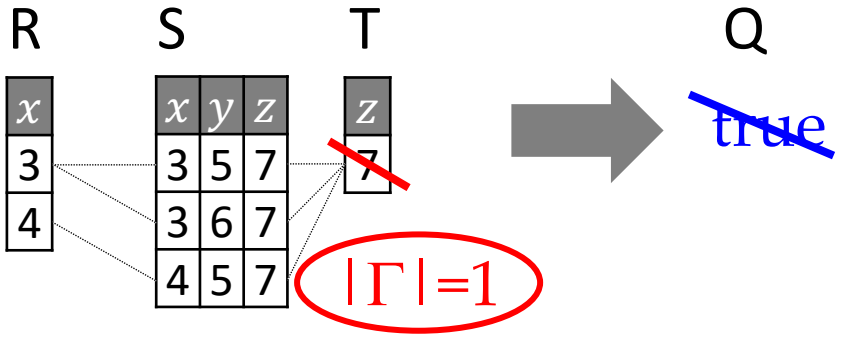


Example Reverse Data Management Problems

Q:-R(x),S(x,y,z),T(z)

1. Resilience

Delete min number of tuples to make Q false



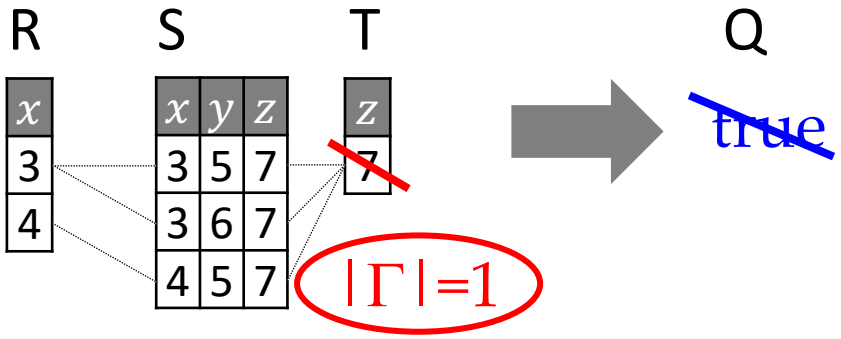
Freire, Gatterbauer, Immerman, Meliou. The complexity of resilience and responsibility for self-join-free conjunctive queries, VLDB 2015. https://doi.org/10.14778/2850583.2850592
Makhija, Gatterbauer. A Unified Approach for Resilience and Causal Responsibility with Integer Linear Programming (ILP) and LP Relaxations, SIGMOD 2024. https://doi.org/10.1145/3626715

Example Reverse Data Management Problems

Q:-R(x),S(x,y,z),T(z)

1. Resilience

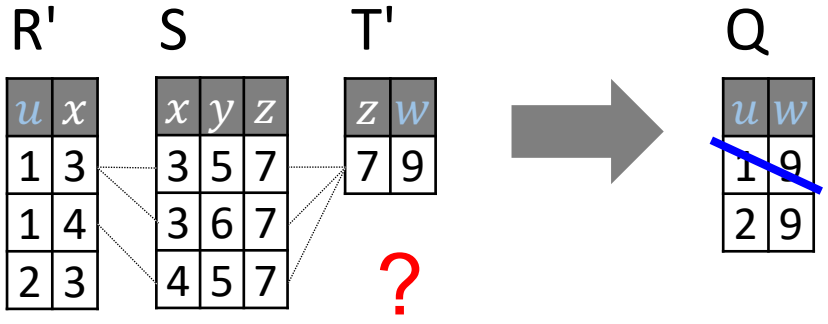
Delete min number of tuples to make Q false



Q(u,w):-R'(u,x),S(x,y,z),T'(z,w)

2. Source side-effects

Delete min number of tuples to delete an output tuples



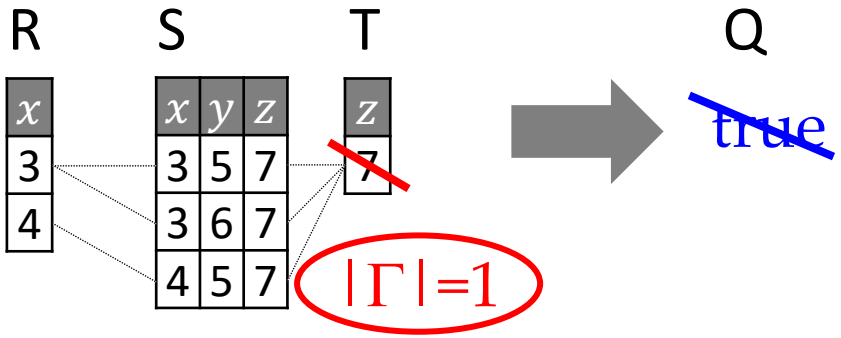
Dayal, Bernstein. On the correct translation of update operations on relational views, TODS 1982, https://doi.org/10.1145/319732.319740
Buneman, Khanna, Tan. On propagation of deletions and annotations through views, PODS 2002, https://doi.org/10.1145/543613.543633

Example Reverse Data Management Problems

Q:-R(x),S(x,y,z),T(z)

1. Resilience

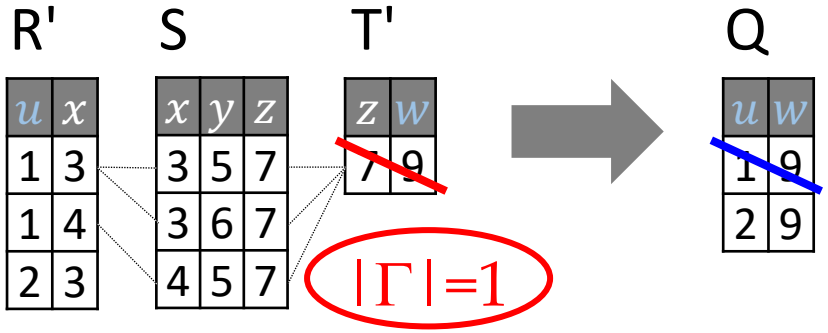
Delete min number of tuples to make Q false



Q(u,w):-R'(u,x),S(x,y,z),T'(z,w)

2. Source side-effects

Delete min number of tuples to delete an output tuples



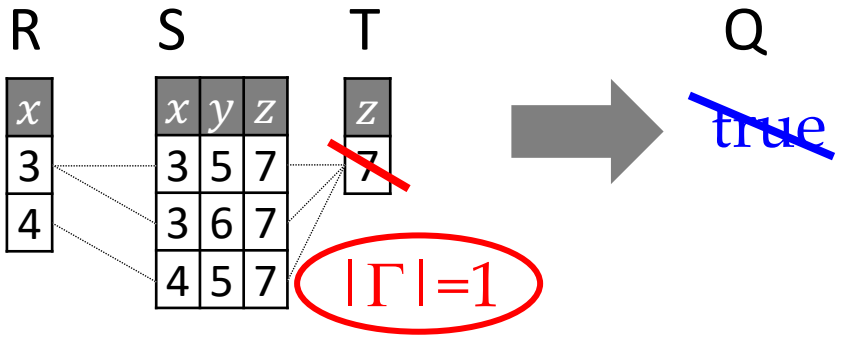
Dayal, Bernstein. On the correct translation of update operations on relational views, TODS 1982, https://doi.org/10.1145/319732.319740
Buneman, Khanna, Tan. On propagation of deletions and annotations through views, PODS 2002, https://doi.org/10.1145/543613.543633
Freire, Gatterbauer, Immerman, Meliou. The complexity of resilience and responsibility for self-join-free conjunctive queries, VLDB 2015. https://doi.org/10.14778/2850583.2850592

Example Reverse Data Management Problems

Q:-R(x),S(x,y,z),T(z)

1. Resilience

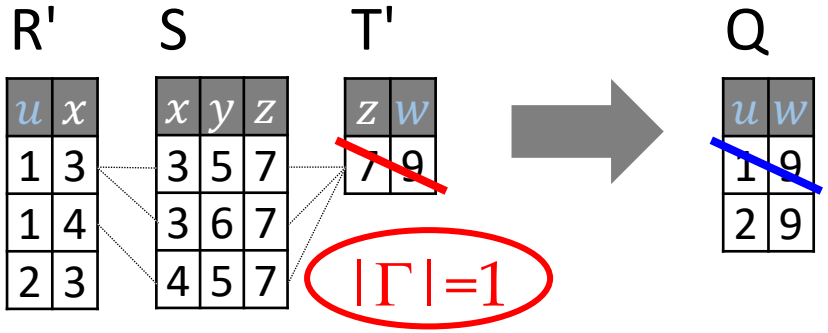
Delete min number of tuples to make Q false



Q(u,w):-R'(u,x),S(x,y,z),T'(z,w)

2. Source side-effects

Delete min number of tuples to delete an output tuples



3. Aggregated deletion propagation

Delete min number of tuples to delete $\geq k$ *output tuples*

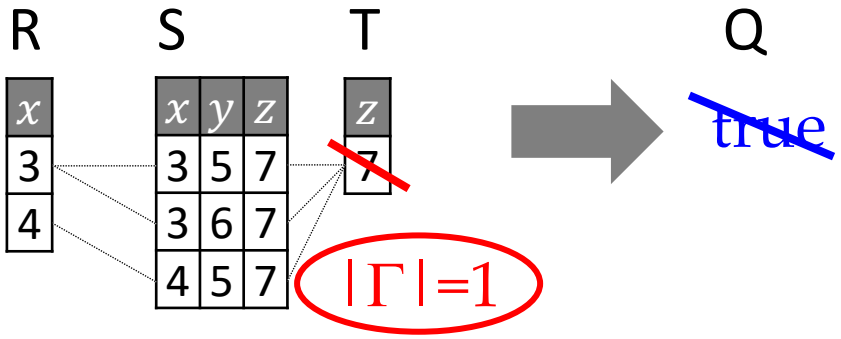
Hu, Sun, Patwa, Panigrahi, Roy. Aggregated Deletion Propagation for Counting Conjunctive Query Answers. PVLDB 2020. https://doi.org/10.14778/3425879.3425892

Example Reverse Data Management Problems

Q:-R(x),S(x,y,z),T(z)

1. Resilience

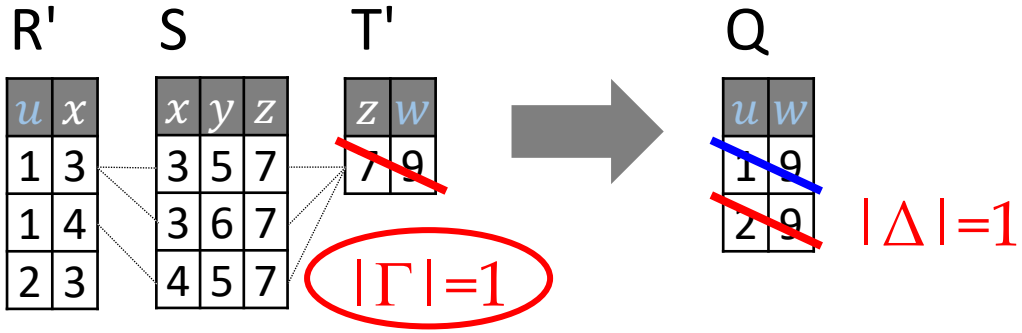
Delete min number of tuples to make Q false



Q(u,w):-R'(u,x),S(x,y,z),T'(z,w)

2/3. (Aggregated) Source side-effects

Delete min number of tuples to delete ≥k output tuples



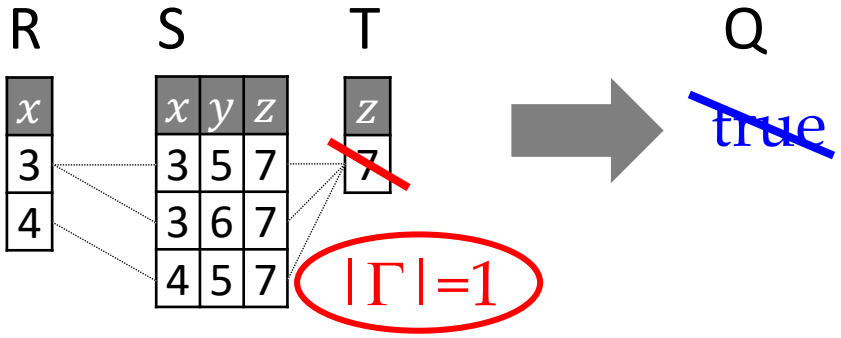
Hu, Sun, Patwa, Panigrahi, Roy. Aggregated Deletion Propagation for Counting Conjunctive Query Answers. PVLDB 2020. https://doi.org/10.14778/3425879.3425892

Example Reverse Data Management Problems

Q:-R(x),S(x,y,z),T(z)

1. Resilience

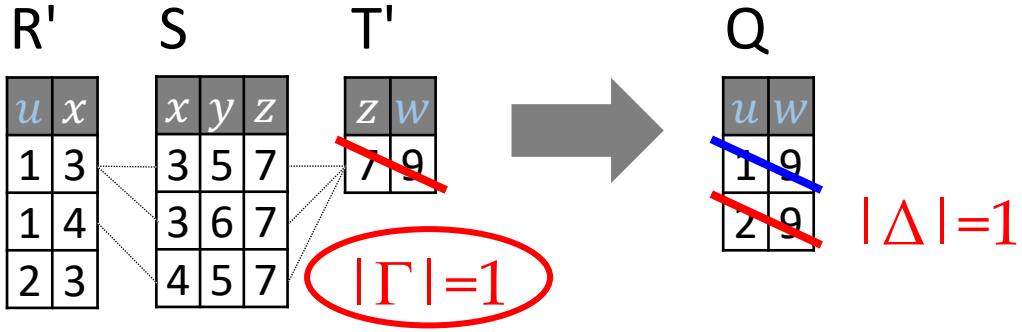
Delete min number of tuples to make Q false



Q(u,w):-R'(u,x),S(x,y,z),T'(z,w)

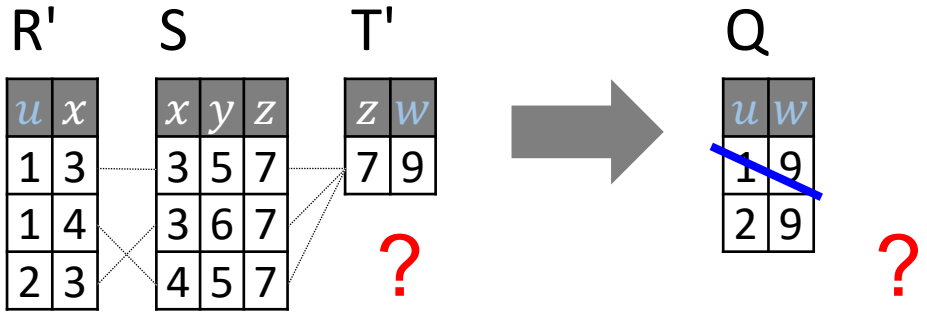
2/3. (Aggregated) Source side-effects

Delete min number of tuples to delete ≥k output tuples



4. View side-effects (in deletion propagation)

Delete tuples in order to delete an output tuple, while minimizing the other output tuples deleted



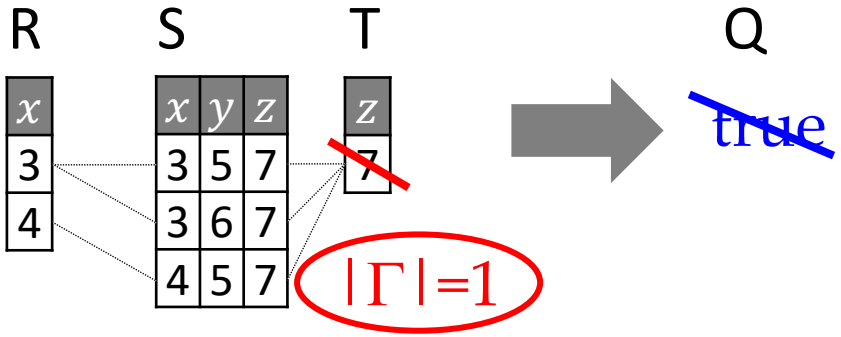
Dayal, Bernstein. On the correct translation of update operations on relational views, TODS 1982, https://doi.org/10.1145/319732.319740
Buneman, Khanna, Tan. On propagation of deletions and annotations through views, PODS 2002, https://doi.org/10.1145/543613.543633
Kimelfeld, Vondrak, Williams. Maximizing conjunctive views in deletion propagation, PODS 2011, https://doi.org/10.1145/1989284.1989308

Example Reverse Data Management Problems

Q:-R(x),S(x,y,z),T(z)

1. Resilience

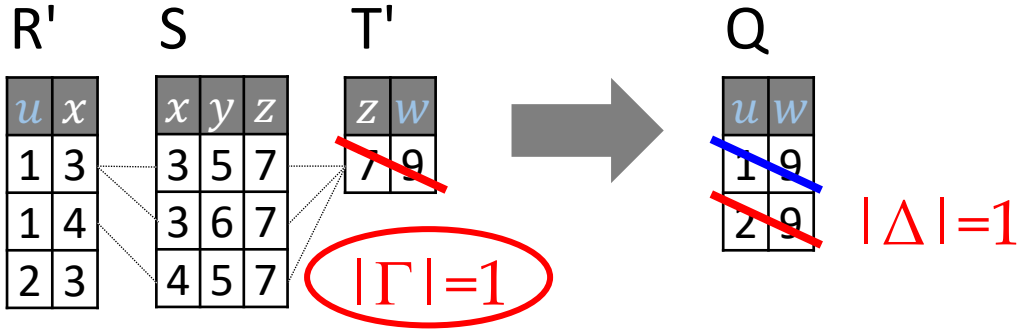
Delete min number of tuples to make Q false



Q(u,w):-R'(u,x),S(x,y,z),T'(z,w)

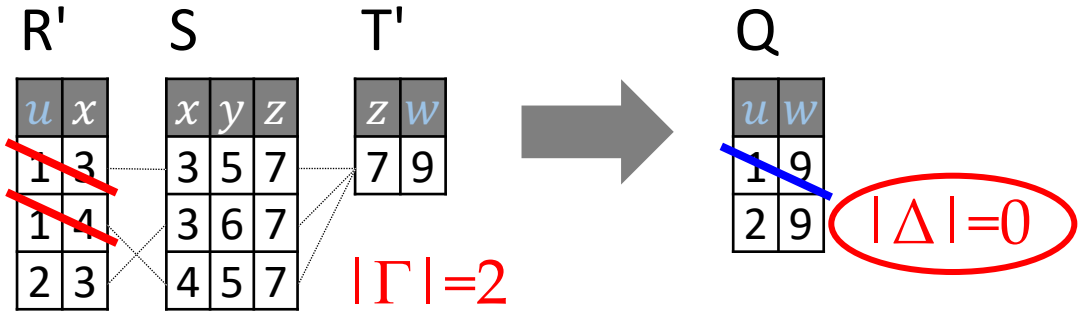
2/3. (Aggregated) Source side-effects

Delete min number of tuples to delete ≥k output tuples



4. View side-effects (in deletion propagation)

Delete tuples in order to delete an output tuple, while minimizing the other output tuples deleted



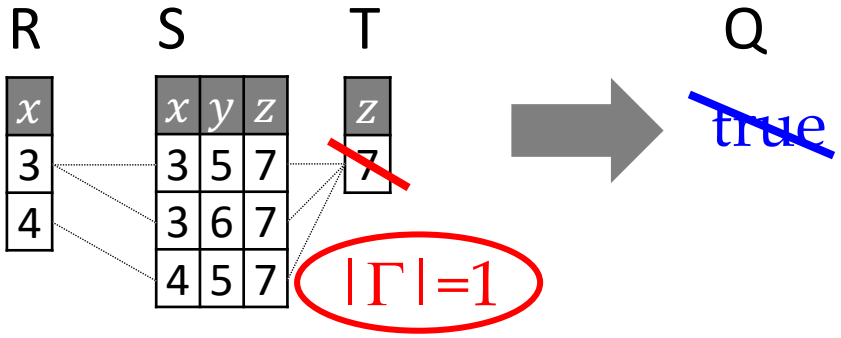
Dayal, Bernstein. On the correct translation of update operations on relational views, TODS 1982, https://doi.org/10.1145/319732.319740
Buneman, Khanna, Tan. On propagation of deletions and annotations through views, PODS 2002, https://doi.org/10.1145/543613.543633
Kimelfeld, Vondrak, Williams. Maximizing conjunctive views in deletion propagation, PODS 2011, https://doi.org/10.1145/1989284.1989308

Example Reverse Data Management Problems

Q:-R(x),S(x,y,z),T(z)

1. Resilience

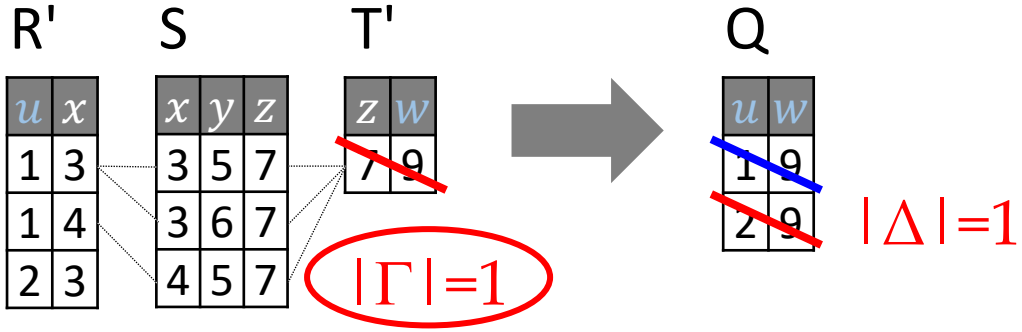
Delete min number of tuples to make Q false



Q(u,w):-R'(u,x),S(x,y,z),T'(z,w)

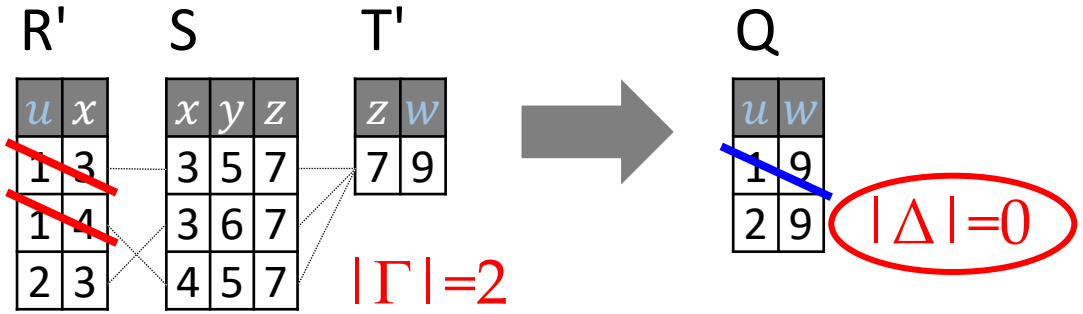
2/3. (Aggregated) Source side-effects

Delete min number of tuples to delete ≥k output tuples



4/5. (Aggregated) View side-effects

Delete tuples in order to delete ≥k output tuple, while minimizing the other output tuples deleted

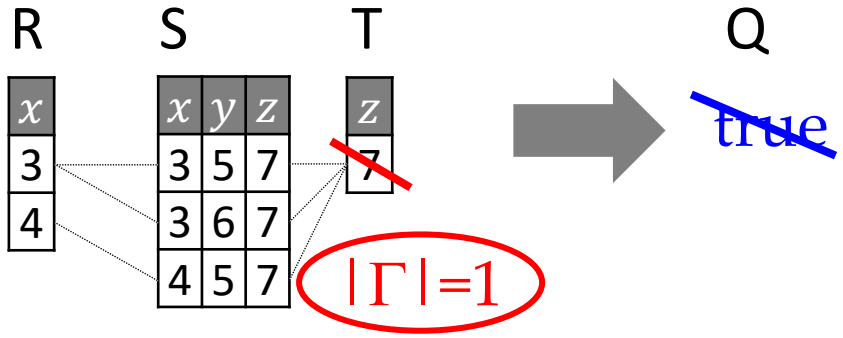


Example Reverse Data Management Problems

Q:-R(x),S(x,y,z),T(z)

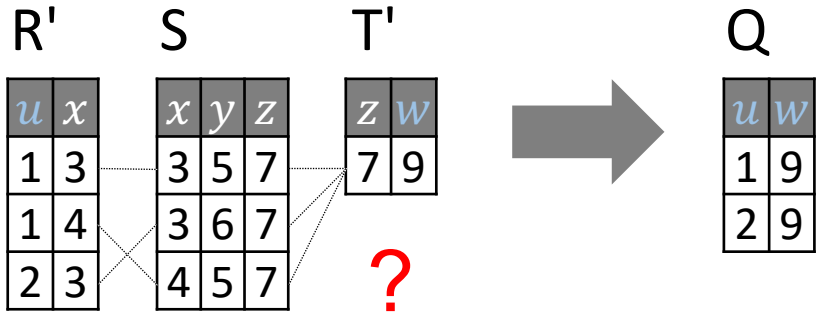
1. Resilience

Delete min number of tuples to make Q false



5. Smallest witness problem

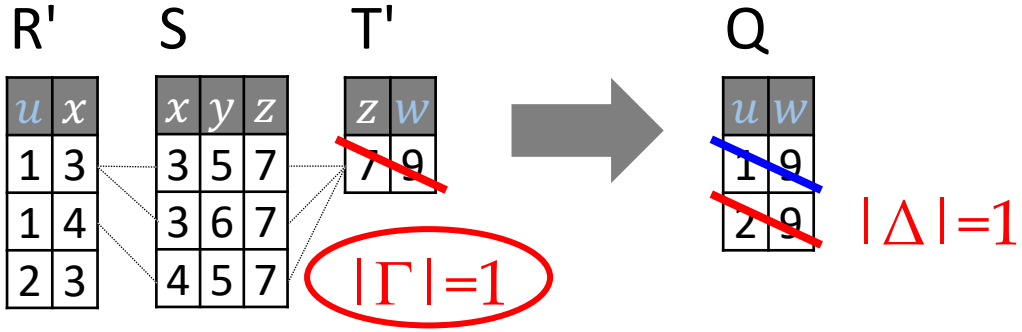
Delete max num. of tuples while keeping all output tuples



Q(u,w):-R'(u,x),S(x,y,z),T'(z,w)

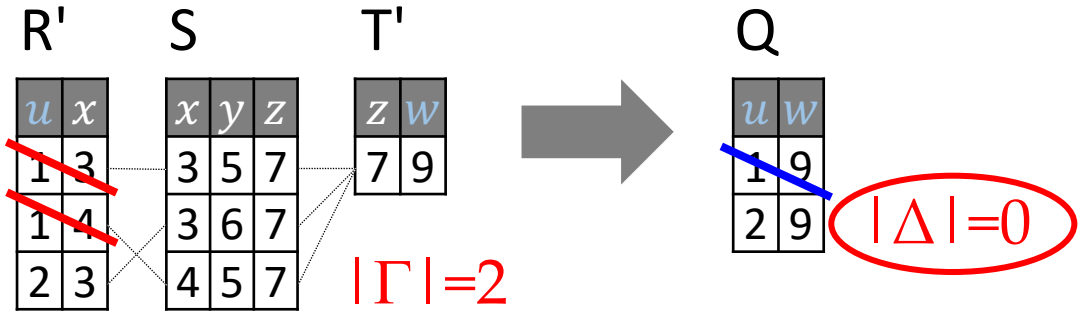
2/3. (Aggregated) Source side-effects

Delete min number of tuples to delete ≥k output tuples



4/5. (Aggregated) View side-effects

Delete tuples in order to delete ≥k output tuple, while minimizing the other output tuples deleted



Miao, Roy, Yang. Explaining wrong queries using small examples. SIGMOD, 2019. https://doi.org/10.1145/3299869.3319866

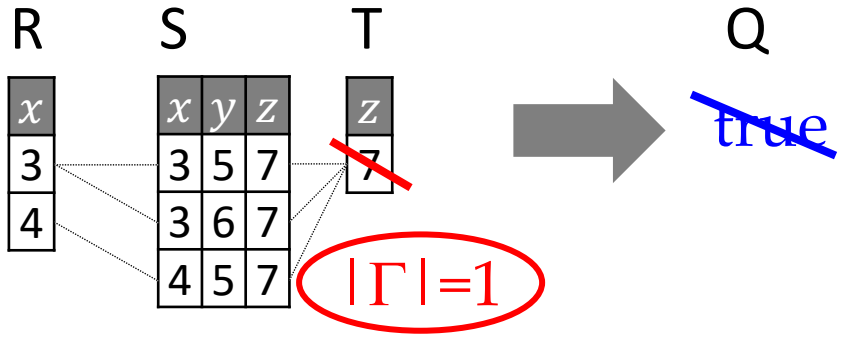
Hu, Sintos. Finding Smallest Witnesses for Conjunctive Queries, ICDT 2024. https://doi.org/10.4230/LIPIcs.ICDT.2024.24

Example Reverse Data Management Problems

Q:-R(x),S(x,y,z),T(z)

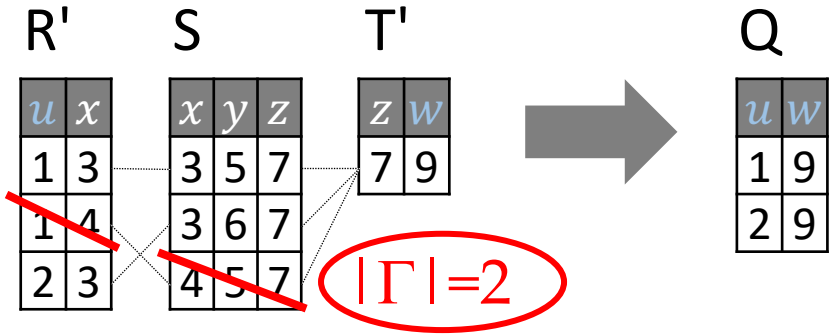
1. Resilience

Delete min number of tuples to make Q false



5. Smallest witness problem

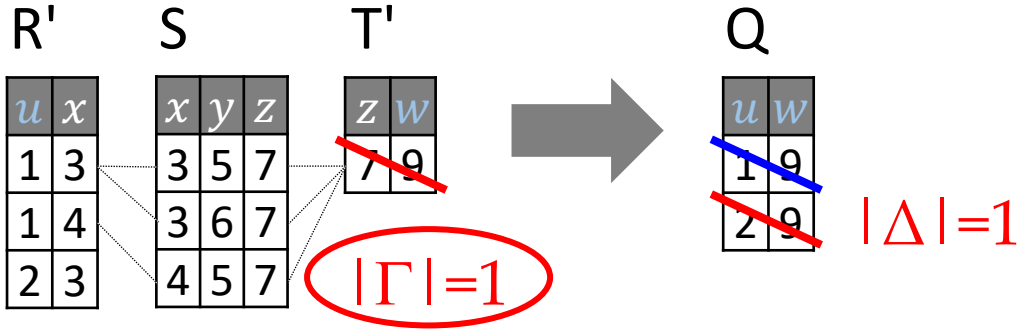
Delete max num. of tuples while keeping all output tuples



Q(u,w):-R'(u,x),S(x,y,z),T'(z,w)

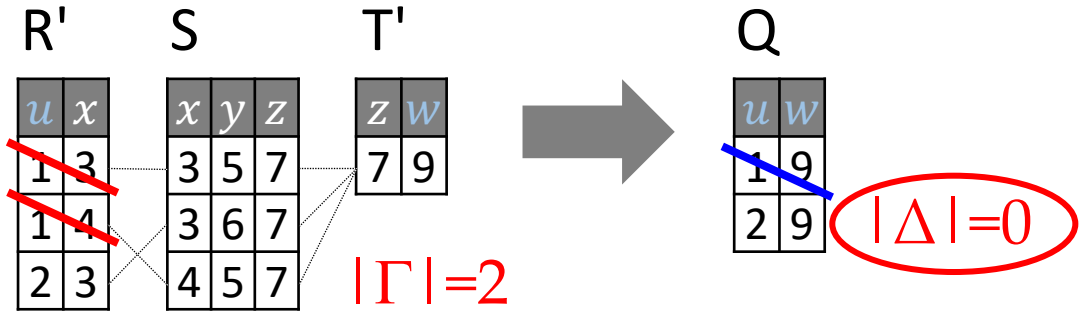
2/3. (Aggregated) Source side-effects

Delete min number of tuples to delete ≥k output tuples



4/5. (Aggregated) View side-effects

Delete tuples in order to delete ≥k output tuple, while minimizing the other output tuples deleted



Miao, Roy, Yang. Explaining wrong queries using small examples. SIGMOD, 2019. https://doi.org/10.1145/3299869.3319866

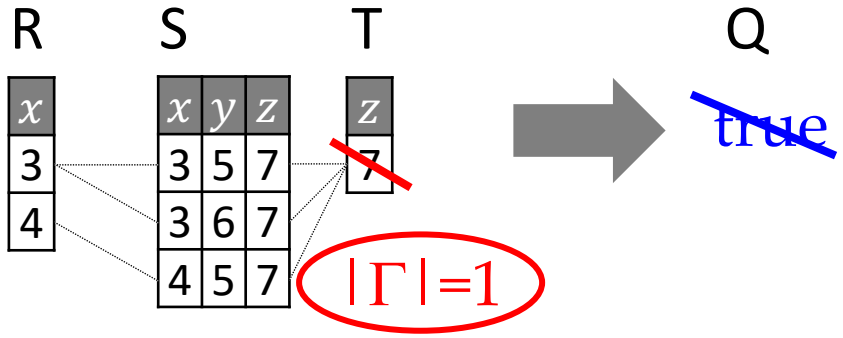
Hu, Sintos. Finding Smallest Witnesses for Conjunctive Queries, ICDT 2024. https://doi.org/10.4230/LIPIcs.ICDT.2024.24

Example Reverse Data Management Problems

Q:-R(x),S(x,y,z),T(z)

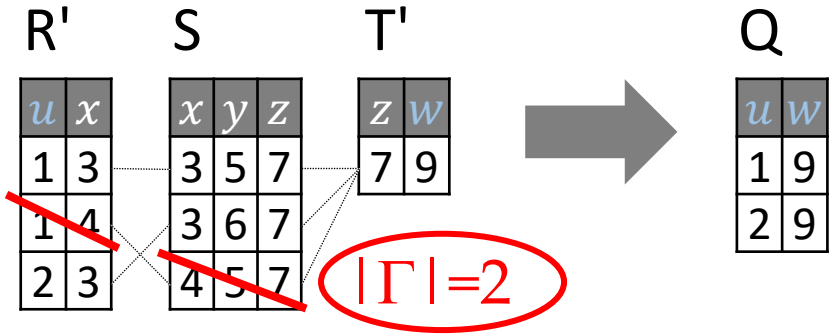
1. Resilience

Delete min number of tuples to make Q false



6./7 (Aggregated) Smallest witness problem

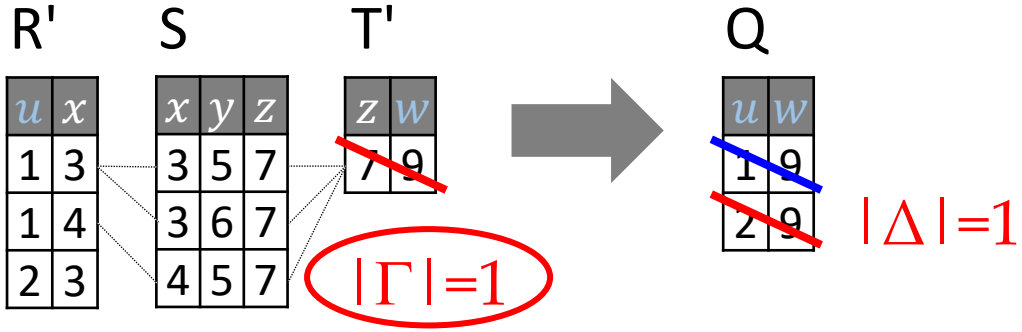
Delete max num. of tuples while keeping ≥k output tuples



Q(u,w):-R'(u,x),S(x,y,z),T'(z,w)

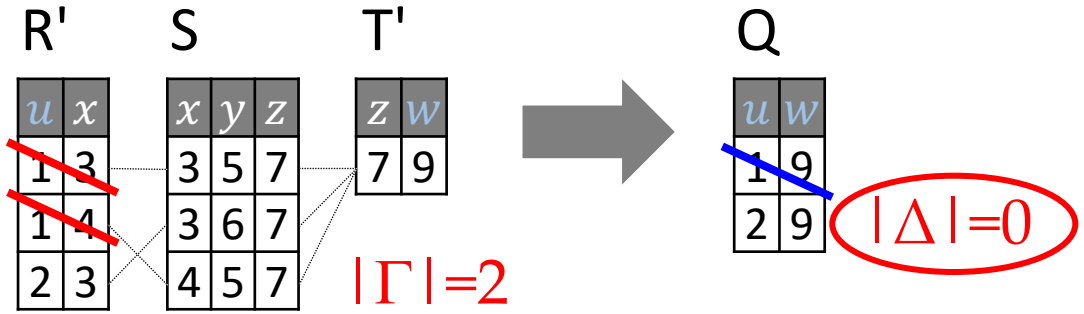
2/3. (Aggregated) Source side-effects

Delete min number of tuples to delete ≥k output tuples



4/5. (Aggregated) View side-effects

Delete tuples in order to delete ≥k output tuple, while minimizing the other output tuples deleted



Example Reverse Data Management Problems

Q:-R(x),S(x,y,z),T(z)

Q(u,w):-R'(u,x),S(x,y,z),T'(z,w)

1. Resilience

Delete min number of tuples to make Q false

2/3. (Aggregated) Source side-effects

Delete min number of tuples to delete ≥k output tuples

6./7 (Aggregated) Smallest witness problem

Delete max num. of tuples while keeping ≥k output tuples

4/5. (Aggregated) View side-effects

Delete tuples in order to delete ≥k output tuple, while minimizing the other output tuples deleted

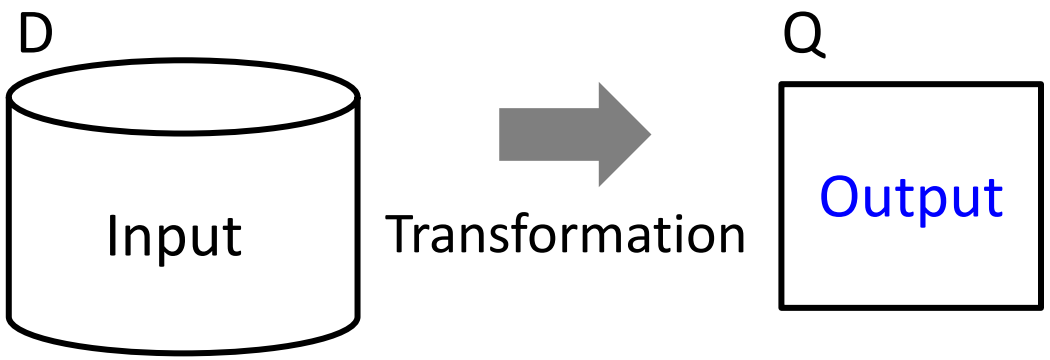
Our topic today: Generalized Deletion Propagation

1. Unify and generalize these problems as instances of "generalized deletion propagation"
   - also allows new problem variants
2. Propose one algorithm to solve all these problems
   - including difficult cases, such as self-joins, or bag semantics
3. The algorithm terminates in PTIME for all known PTIME cases
   - including all known PTIME cases for self-join free queries for the problems of: resilience [VLDB'15], aggregated deletion propagation [VLDB'20], view-side effects [PODS'11], smallest witness problem [ICDT'24]

Reverse Data Management for Conjunctive Queries

QUERY EVALUATION: (Talks today in the morning)

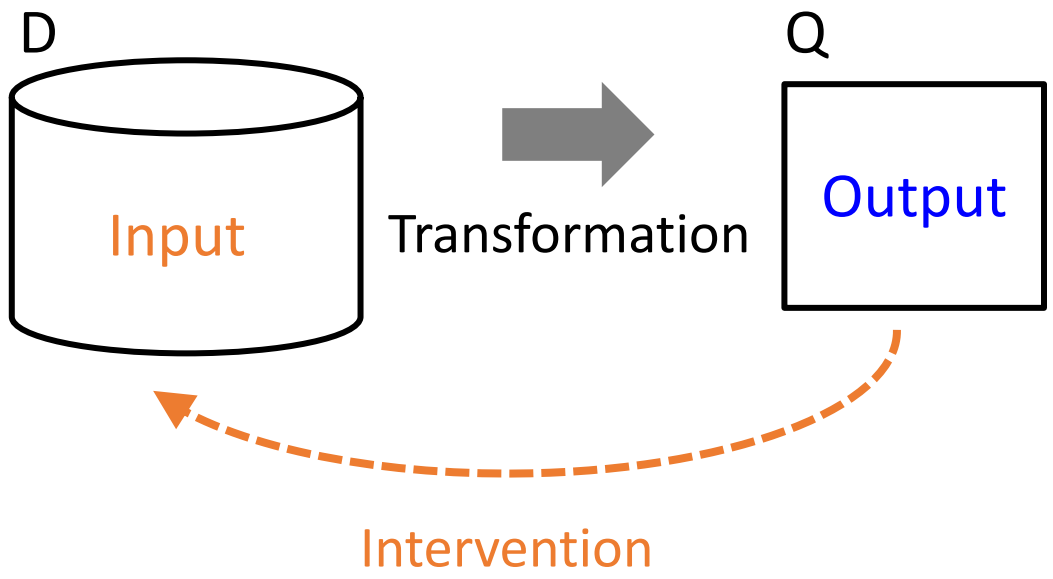
A transformation of the input to the output



Reverse Data Management

QUERY EVALUATION:

A transformation of the input to the output



REVERSE DATA MANAGEMENT:

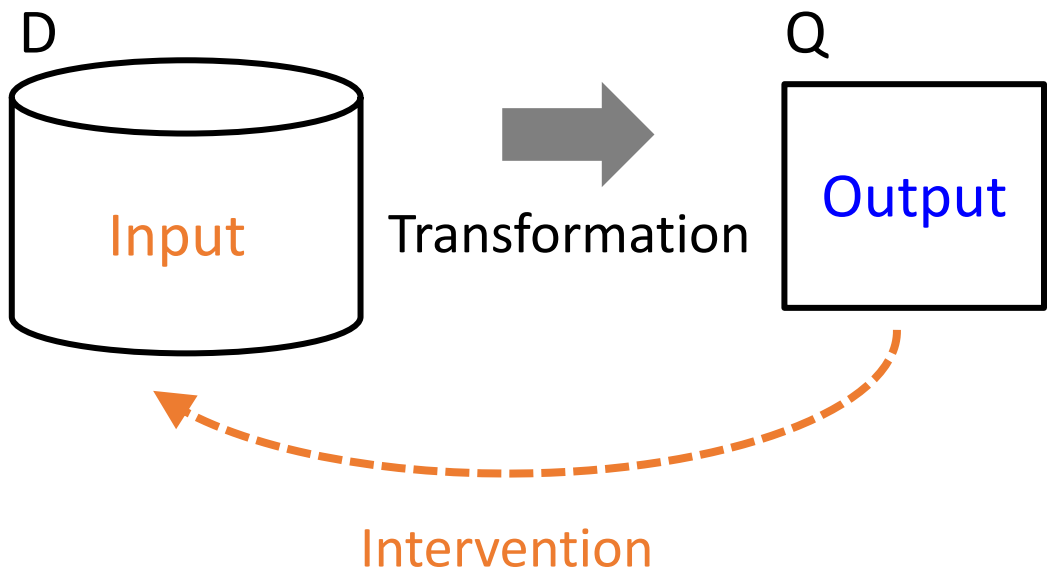
What are the required changes to the input,
in order to achieve a desired output?

Meliou, Gatterbauer, Suciu. Reverse Data Management. VLDB 2011 Vision track. https://doi.org/10.14778/3402755.3402803

Reverse Data Management for Conjunctive Queries

QUERY EVALUATION:

A transformation of the input to the output



REVERSE DATA MANAGEMENT:

What are the required changes to the input, in order to achieve a desired output?

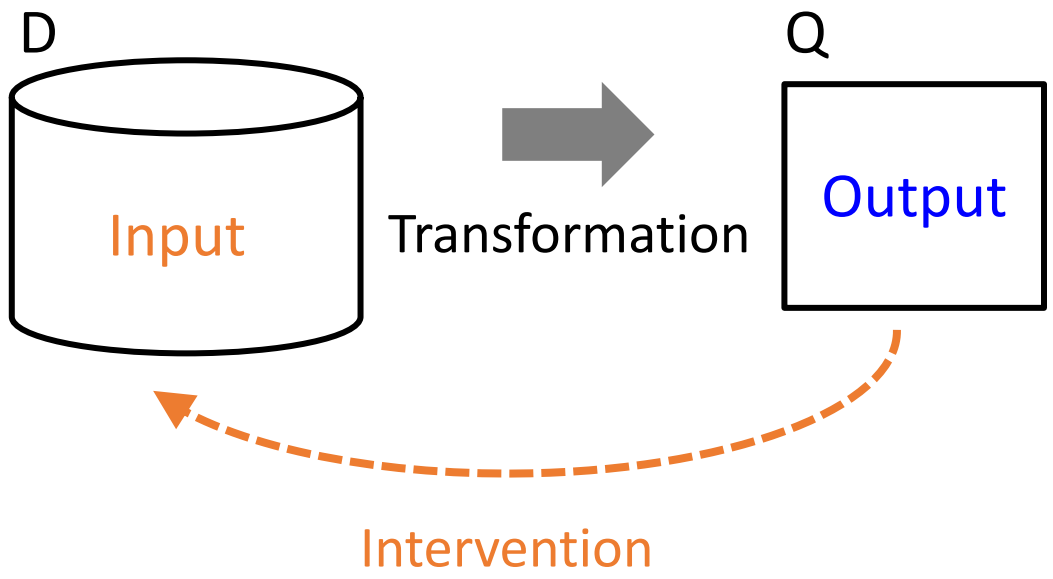
1. Contrastive (CXp): subset-minimal set of features (tuples) that are sufficient for changing a prediction (from true to false).

Meliou, Gatterbauer, Suciu. Reverse Data Management. VLDB 2011 Vision track. https://doi.org/10.14778/3402755.3402803
Marques-Silva. Logic-Based Explainability in Machine Learning. Reasoning Web. Causality, Explanations and Declarative Knowledge. 2023. https://doi.org/10.1007/978-3-031-31414-8_2

Reverse Data Management for Conjunctive Queries

QUERY EVALUATION:

A transformation of the input to the output



REVERSE DATA MANAGEMENT:

What are the required changes to the input, in order to achieve a desired output?

2. Abductive (AXp): a subset-minimal set of features (tuples) that are sufficient for ensuring a certain prediction (i.e. fixing these tuples guarantees the output)

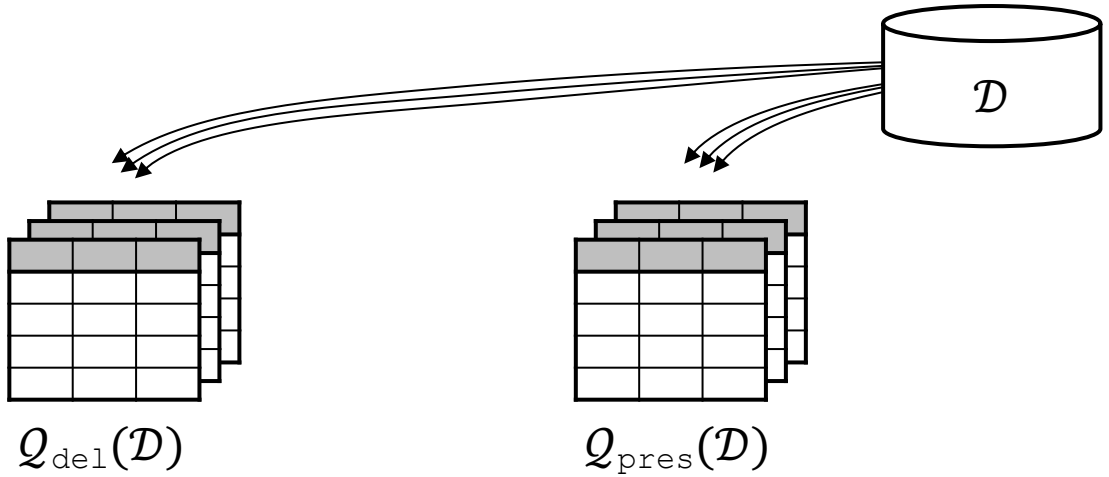
= maximum deletions of tuples

1. Contrastive (CXp): subset-minimal set of features (tuples) that are sufficient for changing a prediction (from true to false).

= minimum deletions of tuples

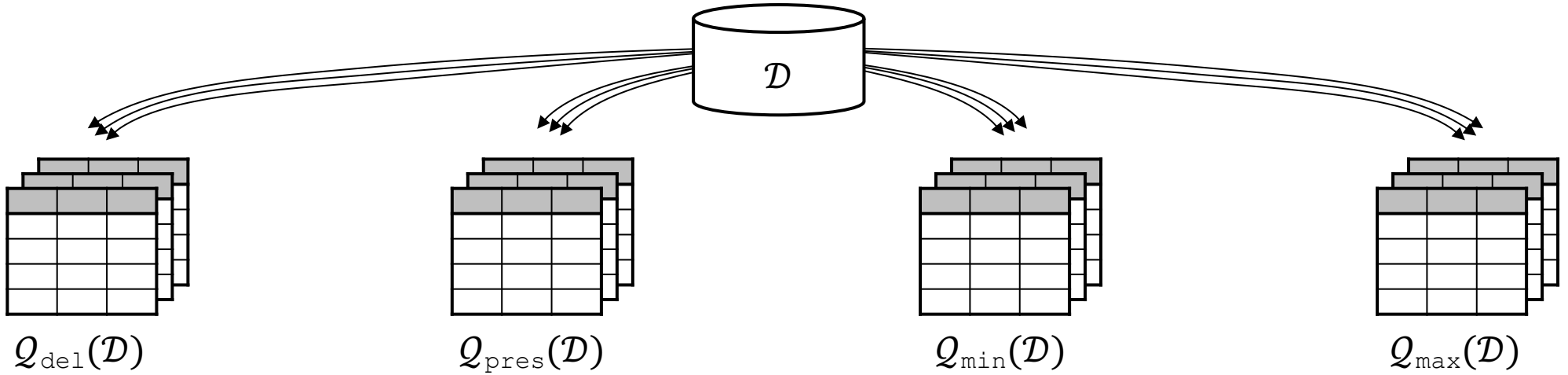
Meliou, Gatterbauer, Suciu. Reverse Data Management. VLDB 2011 Vision track. https://doi.org/10.14778/3402755.3402803
Marques-Silva. Logic-Based Explainability in Machine Learning. Reasoning Web. Causality, Explanations and Declarative Knowledge. 2023. https://doi.org/10.1007/978-3-031-31414-8_2

Generalized Deletion Propagation



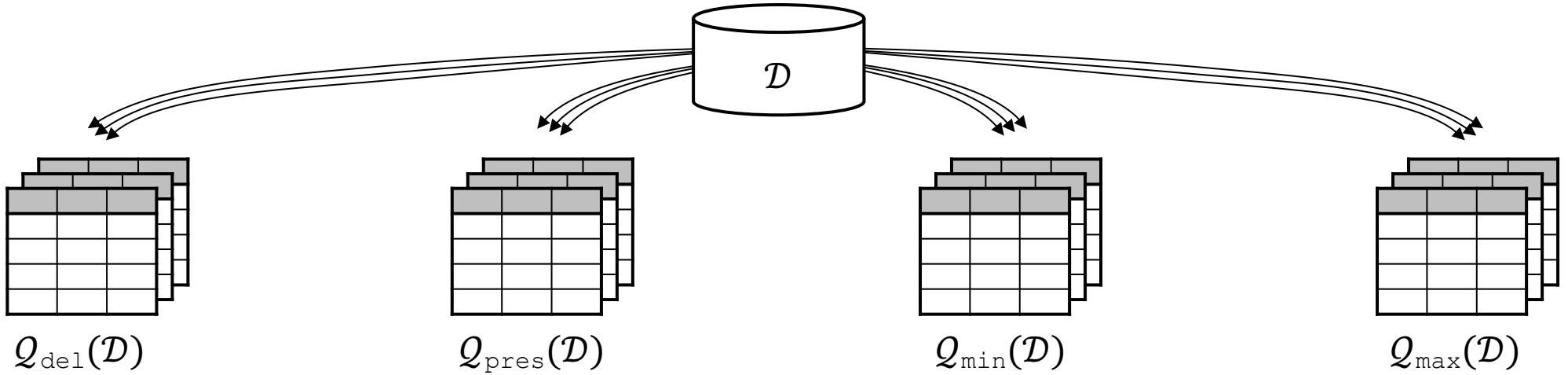
Propagate Requirements (hard constraints)	
Delete	Preserve
$\geq \mathrm{k}^{i}_{\mathbf{del}}$ tuples in $\mathcal{Q}^{i}_{\texttt{del}}(\mathcal{D})$	$\geq \mathrm{k}^{i}_{\mathbf{pres}}$ tuples in $Q^{i}_{\texttt{pres}}(D)$

Generalized Deletion Propagation



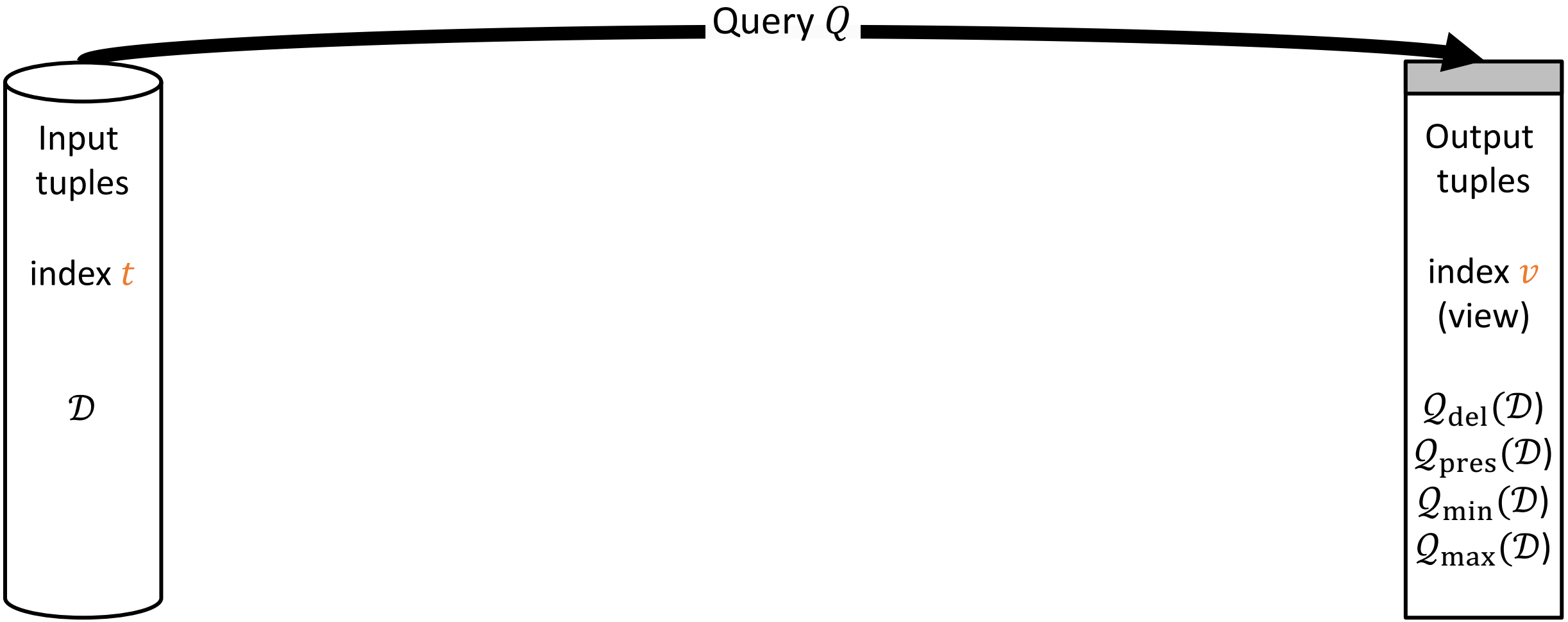
Propagate Requirements (hard constraints)		Optimize Side-Effect (soft constraints)	
Delete	Preserve	Minimize	Maximize
$\geq \mathrm{k}^i_{\mathbf{del}}$ tuples in $\mathcal{Q}^i_{\mathtt{del}}(\mathcal{D})$	$\geq \mathrm{k}^i_{\mathbf{pres}}$ tuples in $Q^i_{\mathtt{pres}}(D)$	Deletions in $\mathcal{Q}_{\mathsf{min}}$	Deletions in $\mathcal{Q}_{\mathsf{max}}$

Generalized Deletion Propagation



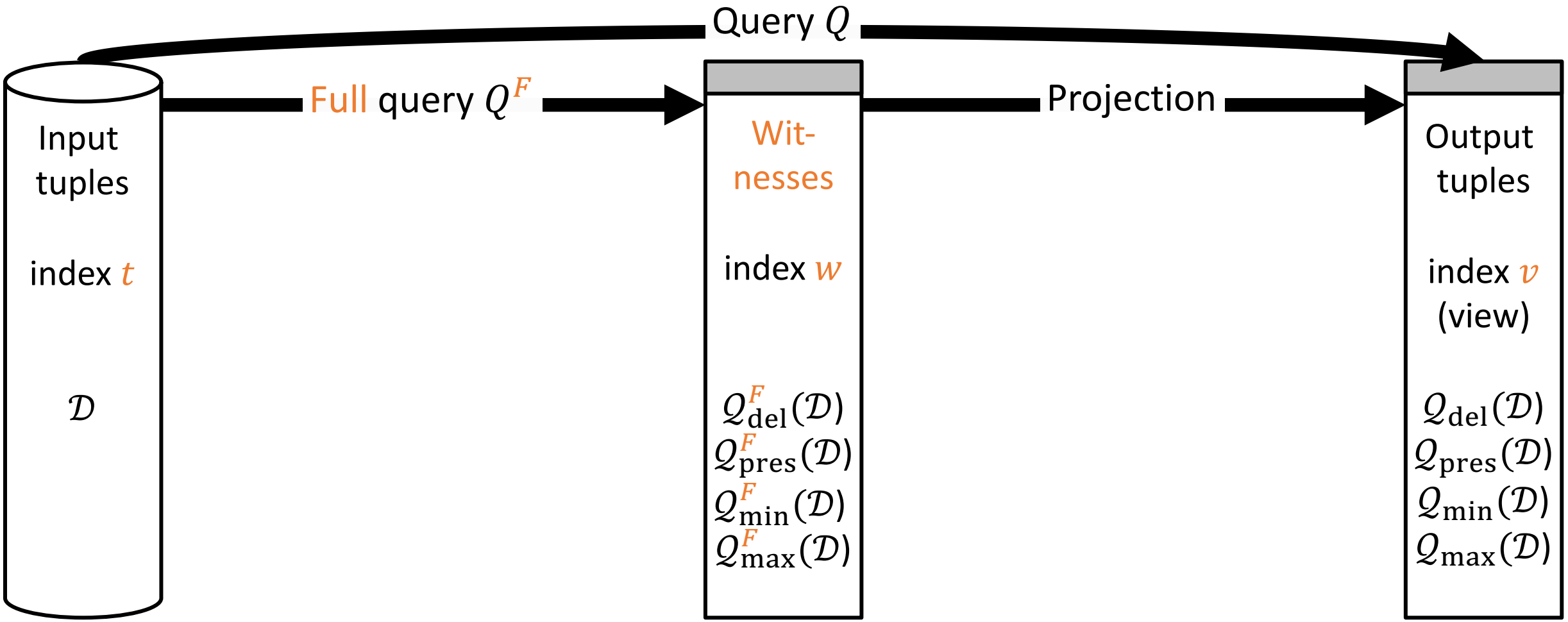
	Propagate Requirements (hard constraints)		Optimize Side-Effect (soft constraints)	
	Delete $\geq \mathrm{k}^{i}_{\mathbf{del}}$ tuples in $\mathcal{Q}^{i}_{\mathtt{del}}(\mathcal{D})$	Preserve $\geq \mathrm{k}^{i}_{\mathbf{pres}}$ tuples in $Q^{i}_{\mathtt{pres}}(D)$	Minimize Deletions in $\mathcal{Q}_{\mathtt{min}}$	Maximize Deletions in $\mathcal{Q}_{\mathtt{max}}$
DP-SS	All tuples in $\mathrm{Q}_{\mathtt{del}}(\mathcal{D})$	–	Deletions in $\mathcal{Q}_{\mathtt{min}} = \{R \in \mathcal{D}\}$	–
ADP-SS	$\geq k$ tuples in $\mathrm{Q}_{\mathtt{del}}(\mathcal{D})$	–	Deletions in $\mathcal{Q}_{\mathtt{min}} = \{R \in \mathcal{D}\}$	–
DP-VS	All tuples in $\mathrm{Q}_{\mathtt{del}}(\mathcal{D})$	–	Deletions in $\mathcal{Q}_{\mathtt{min}}(\mathcal{D}) \supseteq \mathcal{Q}_{\mathtt{del}}(\mathcal{D})$	–
SWP	–	All tuples in $\mathcal{Q}_{\mathtt{pres}}(\mathcal{D})$	–	Deletions in $\mathcal{Q}_{\mathtt{max}} = \{R \in \mathcal{D}\}$

Propagation Constraints in the Language of ILPs



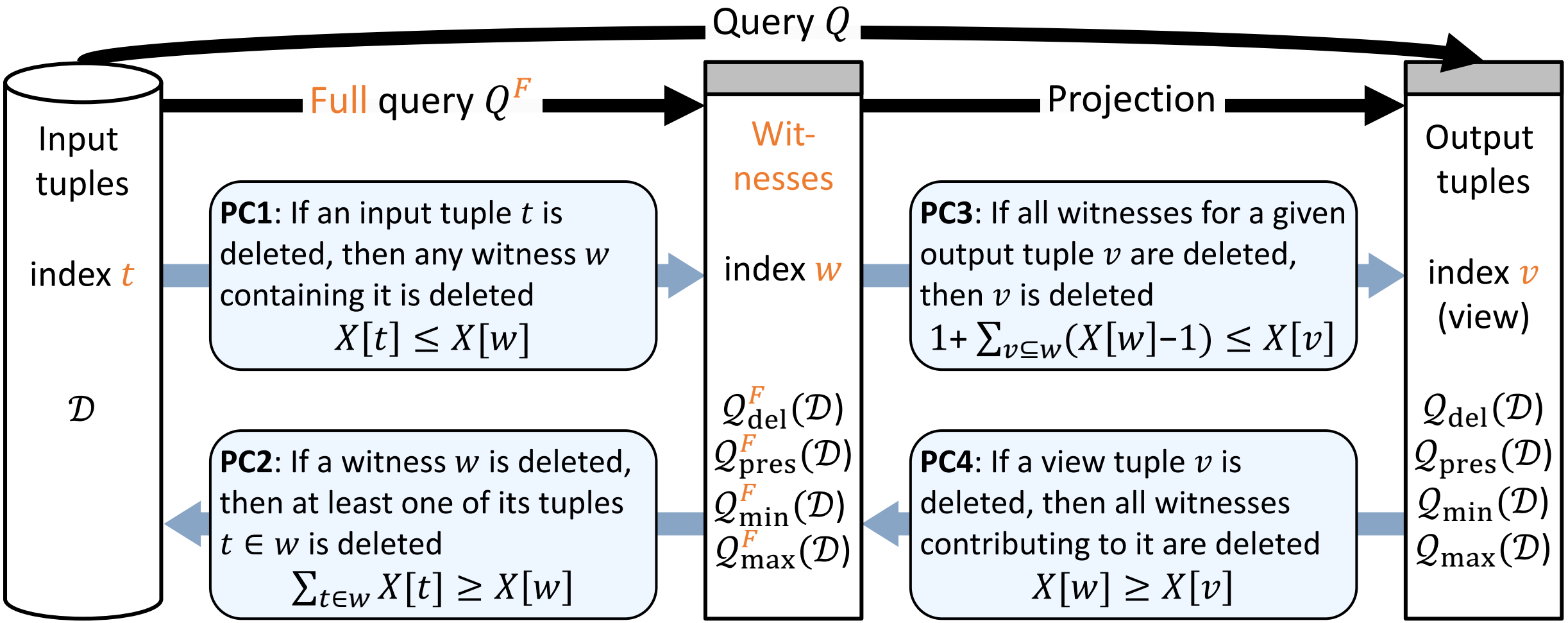
$X[t] = 1$ *means that* t *is deleted*

Propagation Constraints in the Language of ILPs



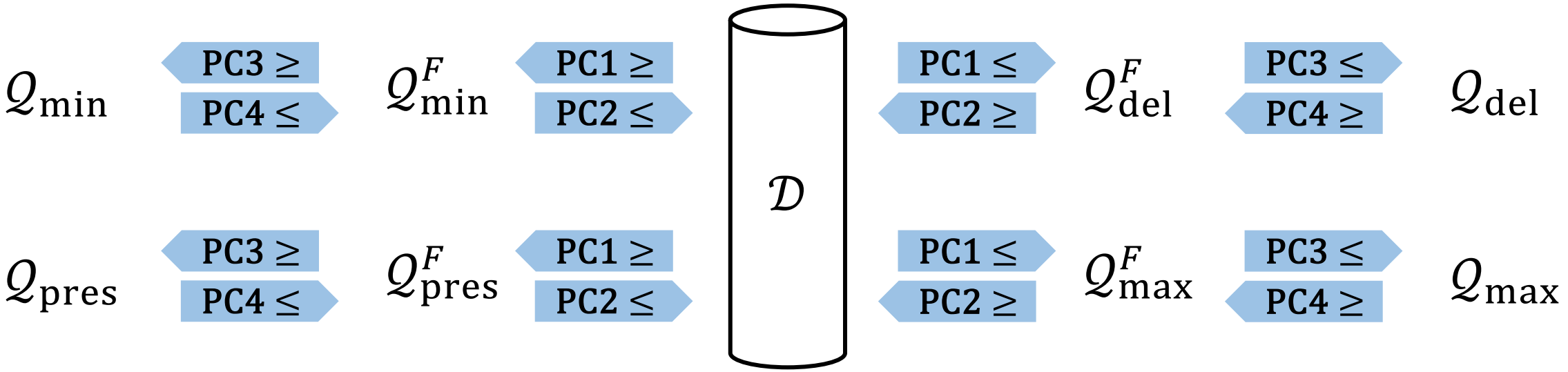
$X[t] = 1$ *means that* t *is deleted*

Propagation Constraints in the Language of ILPs

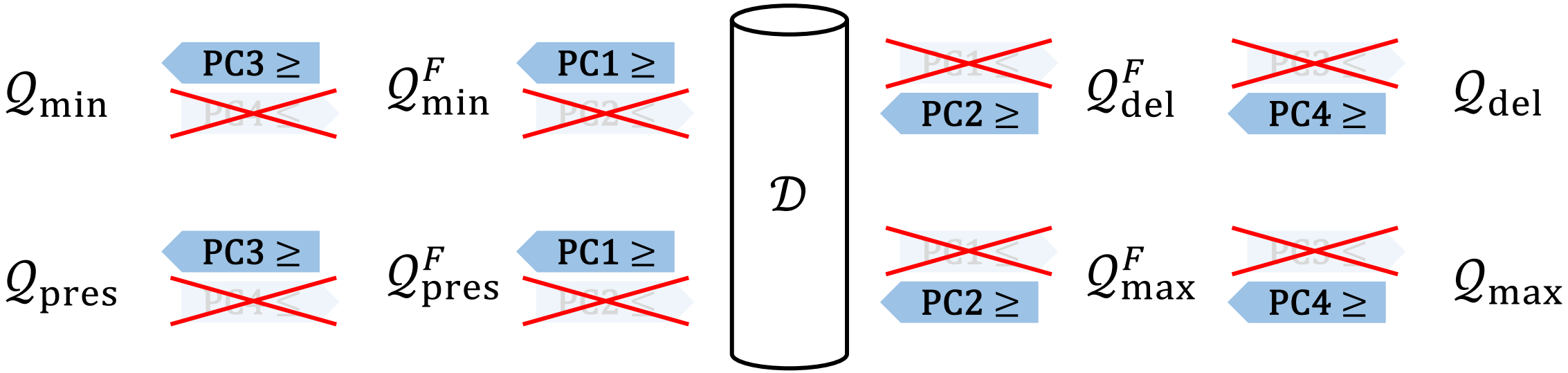


$X[t] = 1$ means that t is deleted

Propagation constraints: Naive ILP (1/3)



Propagation constraints: Wildcard ILP (2/3)

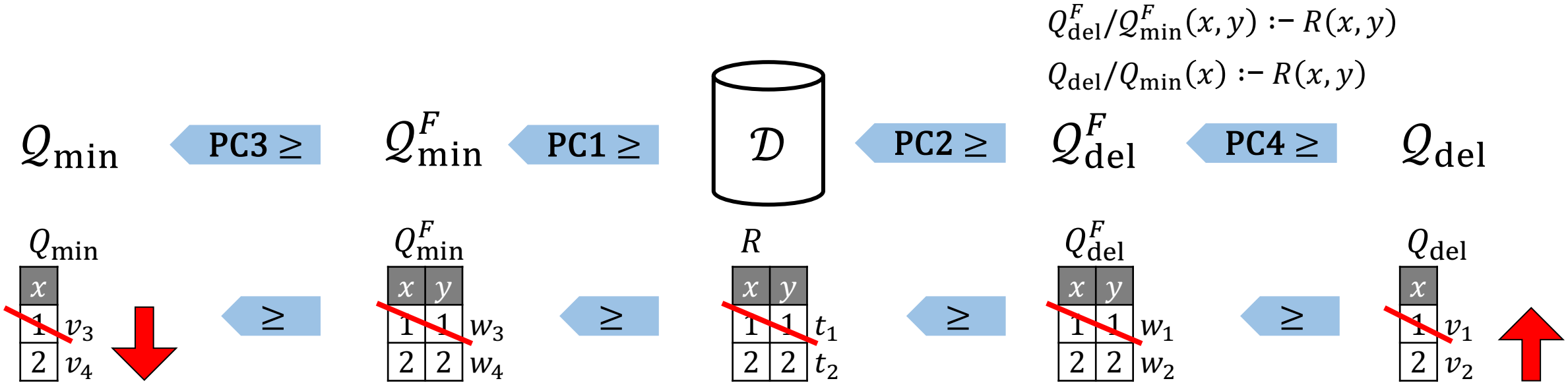


Certain constraints don't need to be explicitly enforced for a given type of view.

Removing the unnecessary constraints *increases* the solution space (from the point of view of the ILP formulation) but keeps all the optimal solutions (projected on the actual interventions on $\mathcal{D}$ that we care about).

Propagation constraints: Wildcard ILP (2/3)

DETAILS

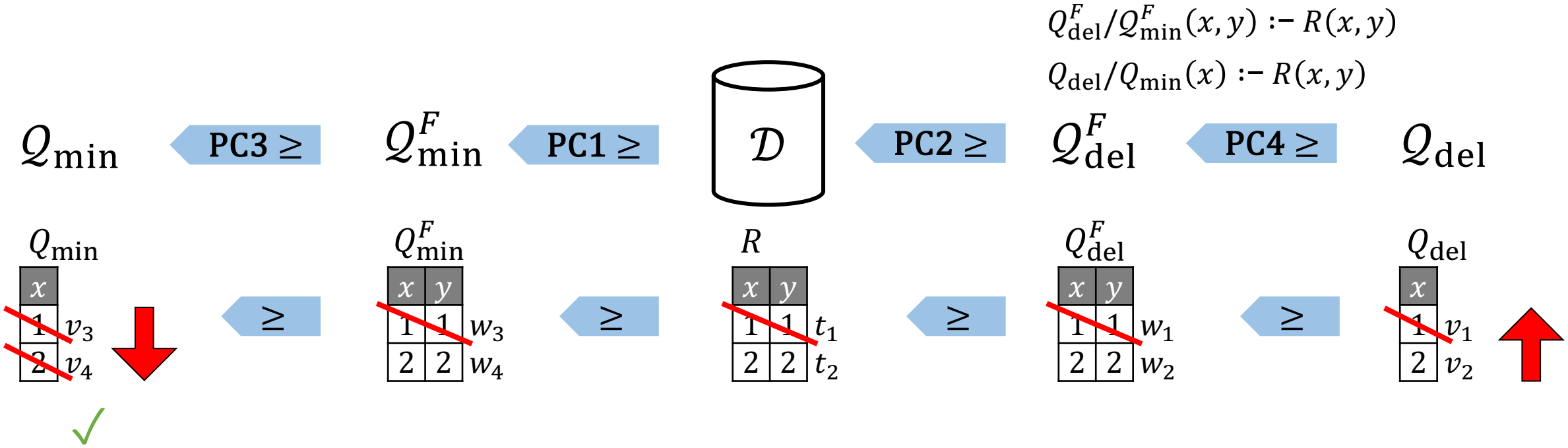


Soft constraint: we want to minimize the number of deletions from Q_{min}

User constraint: we are required to delete a given number of tuples from Q_{del}

Propagation constraints: Wildcard ILP (2/3)

DETAILS



Soft constraint: we want to minimize the number of deletions from Q_{min}

The variables are an upper bound and preservations (X=0) need to get propagated from left to right.

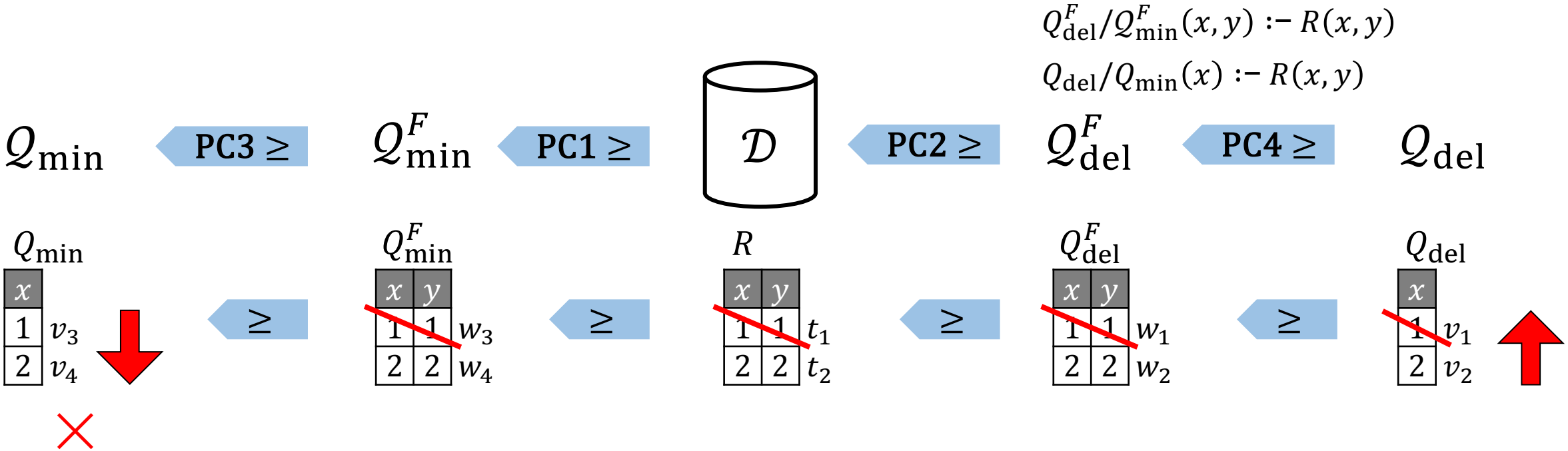
We can set more variables than necessary to 1, but that just penalizes our solution.

User constraint: we are required to delete a given number of tuples from Q_{del}

The variables are a lower bound and deletions (X=1) need to get propagated from right to left.

Propagation constraints: Wildcard ILP (2/3)

DETAILS



Soft constraint: we want to minimize the number of deletions from Q_{min}

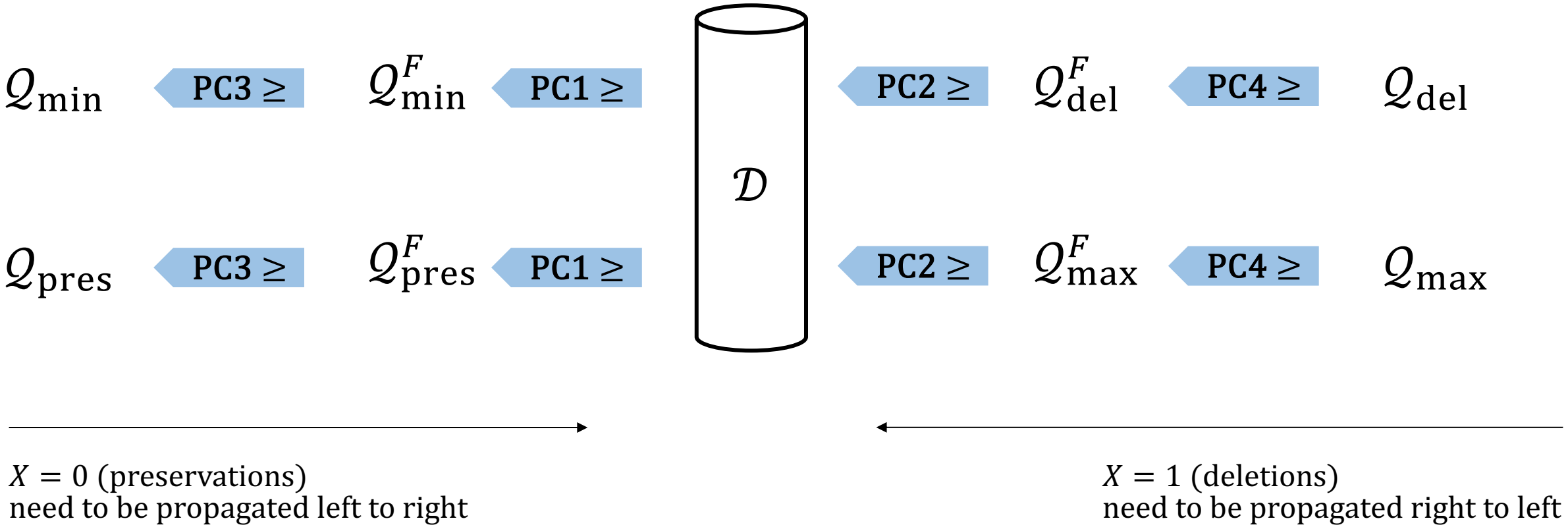
The variables are an upper bound and preservations (X=0) need to get propagated from left to right.

We can set more variables than necessary to 1, but that just penalizes our solution.

User constraint: we are required to delete a given number of tuples from Q_{del}

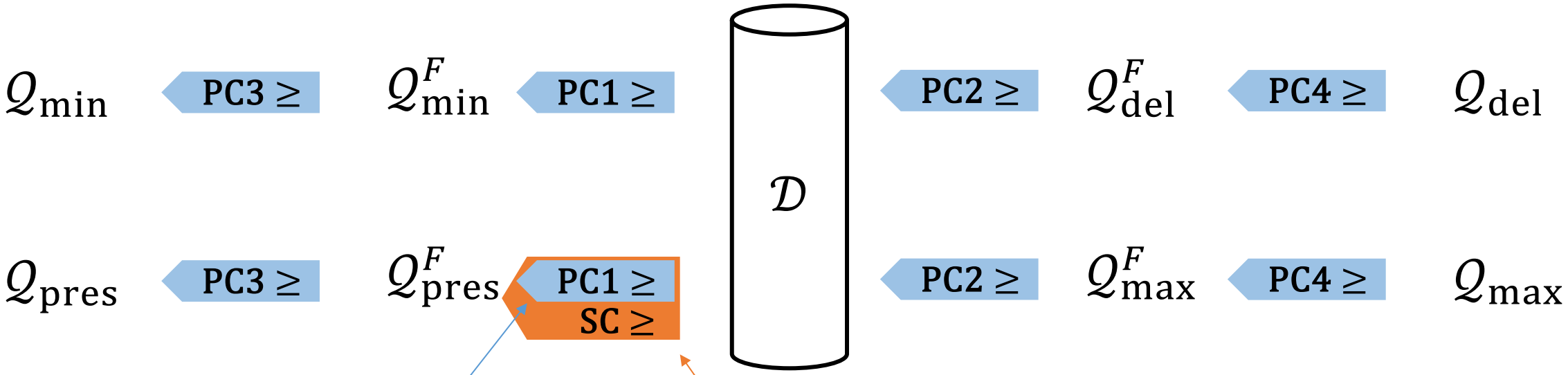
The variables are a lower bound and deletions (X=1) need to get propagated from right to left.

Propagation constraints: Wildcard ILP (2/3)



Removing the unnecessary constraints *increases* the solution space (from the point of view of the ILP formulation) but keeps all the optimal solutions (projected on the actual interventions on $\mathcal{D}$ that we care about).

Propagation constraints: Smoothened ILP (3/3)



PC1: If an input tuple t is deleted, then any witness w containing it is deleted

$$X[t] \leq X[w]$$

SC: If an input tuple t is not deleted, then at least one of its witnesses is preserved, for each view tuple v

$$X[t] \leq 1 + \sum_{\substack{w:\, t \in w \\ w \supseteq v}} (X[w]-1)$$

Example: Smoothening for Smallest Witness Problem

DETAILS

$Q_{\text{pres}}(x) :- R(x, y), S(x)$

Q_{pres}

x	
1	v_1

R

x	y	
1	1	t_1
1	2	t_2

S

x	
1	t_3

$Q^F_{\text{pres}}(x, y) :- R(x, y), S(x)$

x	y	
1	1	w_1
1	2	w_2

Optimal integral solution:

$f^* = -1$, $\mathbf{X}^*[t_1, t_2, t_3] = (1,0,0)$ or $(0,1,0)$

*To simplify, ILP formulation is slightly simplified from the actual formulation (PC2, PC3, PC4 are moved into the user constraint). See example 3 in paper for details.

Example: Smoothening for Smallest Witness Problem

DETAILS

$$Q_{\text{pres}}(x) :- R(x,y), S(x)$$

Q_{pres}

x	
1	v_1

R

x	y	
1	1	t_1
1	2	t_2

S

x	
1	t_3

$$Q^F_{\text{pres}}(x,y) :- R(x,y), S(x)$$

x	y	
1	1	w_1
1	2	w_2

Optimal integral solution:

$f^* = -1$, $\mathbf{X}^*[t_1, t_2, t_3] = (1,0,0)$ or $(0,1,0)$

*To simplify, ILP formulation is slightly simplified from the actual formulation (PC2, PC3, PC4 are moved into the user constraint). See example 3 in paper for details.

Example: Smoothening for Smallest Witness Problem

DETAILS

$Q_{\text{pres}}(x) :- R(x,y), S(x)$

Q_{pres}

x	
1	v_1

R

x	y	
1	1	t_1
1	2	t_2

S

x	
1	t_3

$Q^F_{\text{pres}}(x,y) :- R(x,y), S(x)$

x	y	
1	1	w_1
1	2	w_2

Naive ILP formulation

$X[t] = 1$ means that t is deleted

Minimization objective

$$f(\mathbf{X}) = -X[t_1] - X[t_2] - X[t_3]$$

Delete as many tuples as possible...

User constraint

$$X[w_1] + X[w_2] \leq 1$$

... but preserve at least one witness.

Propagation Constraints (PCs)*

$$X[t_1] \leq X[w_1]$$
$$X[t_2] \leq X[w_2]$$
$$X[t_3] \leq X[w_1]$$
$$X[t_3] \leq X[w_2]$$

If a tuple is deleted, then so is the corresponding witness

Optimal integral solution:

$f^* = -1$, $\mathbf{X}^*[t_1, t_2, t_3] = (1,0,0)$ or $(0,1,0)$

*To simplify, ILP formulation is slightly simplified from the actual formulation (PC2, PC3, PC4 are moved into the user constraint). See example 3 in paper for details.

Example: Smoothening for Smallest Witness Problem

DETAILS

$Q_{\text{pres}}(x) :- R(x,y), S(x)$

Q_{pres}

x	
1	v_1

R

x	y	
1	1	t_1
1	2	t_2

S

x	
1	t_3

$Q^F_{\text{pres}}(x,y) :- R(x,y), S(x)$

x	y	
1	1	w_1
1	2	w_2

Naive ILP formulation

$X[t] = 1$ *means that* t *is deleted*

Minimization objective

$$f(\mathbf{X}) = -X[t_1] - X[t_2] - X[t_3]$$
$$0.5 \qquad 0.5 \qquad 0.5$$

Delete as many tuples as possible...

User constraint

$$X[w_1] + X[w_2] \leq 1$$
$$0.5 \qquad 0.5$$

... but preserve at least one witness.

Propagation Constraints (PCs)*

$$0.5\ X[t_1] \leq X[w_1]\ 0.5$$
$$0.5\ X[t_2] \leq X[w_2]\ 0.5$$
$$0.5\ X[t_3] \leq X[w_1]\ 0.5$$
$$0.5\ X[t_3] \leq X[w_2]\ 0.5$$

If a tuple is deleted, then so is the corresponding witness

Optimal integral solution (ILP)

$f^* = -1, \mathbf{X}^*[t_1,t_2,t_3] = (1,0,0)$ or $(0,1,0)$

Optimal fractional solution (LP relaxation)

$f^* = -1.5, \mathbf{X}^*[t_1,t_2,t_3] = (0.5,0.5,0.5)$ ☹

Example: Smoothening for Smallest Witness Problem

DETAILS

$Q_{\text{pres}}(x) :- R(x,y), S(x)$

Q_{pres}

x	
1	v_1

R

x	y	
1	1	t_1
1	2	t_2

S

x	
1	t_3

$Q^F_{\text{pres}}(x,y) :- R(x,y), S(x)$

x	y	
1	1	w_1
1	2	w_2

Smoothened ILP formulation

$X[t] = 1$ *means that* t *is deleted*

Minimization objective

$$f(\mathbf{X}) = -X[t_1] - X[t_2] - X[t_3]$$ ← *Delete as many tuples as possible...*

User constraint

$$X[w_1] + X[w_2] \leq 1$$ ← *... but preserve at least one witness.*

Propagation Constraints (PCs)*

$$X[t_1] \leq X[w_1]$$
$$X[t_2] \leq X[w_2]$$
$$X[t_3] \leq X[w_1]$$
$$X[t_3] \leq X[w_2]$$

← *If a tuple is deleted, then so is the corresponding witness*

Smoothening constraint (SCs)

$$X[t_3] \leq X[w_1] + X[w_2] - 1$$

← *If a tuple is not deleted, then at least one of its witnesses is preserved*

$$0.5 \not\leq 0.5 + 0.5 - 1$$

← *The smoothening constraint eliminates the fractional solution with* $f^* = -1.5$

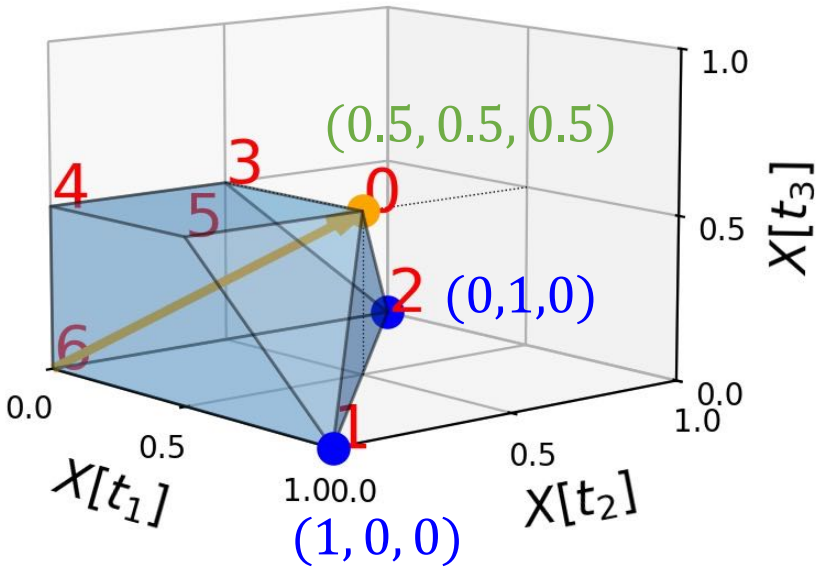
Optimal integral solution (ILP)
= optimal fractional solution (LP relaxation)

$$f^* = -1,\ \mathbf{X}^*[t_1, t_2, t_3] = (1,0,0) \text{ or } (0,1,0)$$ ☺

*ILP formulation is slightly simplified from the actual formulation (PC2, PC3, PC4 are moved into the user constraint). See example 3 in paper for details.

Example: Smoothening for Smallest Witness Problem

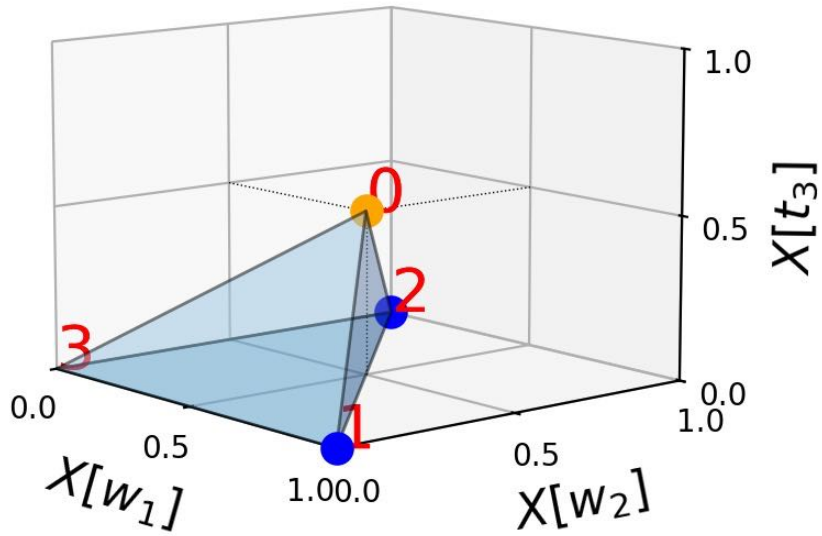
DETAILS



0: $\mathbf{X[t]} = (0.5, 0.5, 0.5)$

1: $\mathbf{X[t]} = (1, 0, 0)$

2: $\mathbf{X[t]} = (0, 1, 0)$



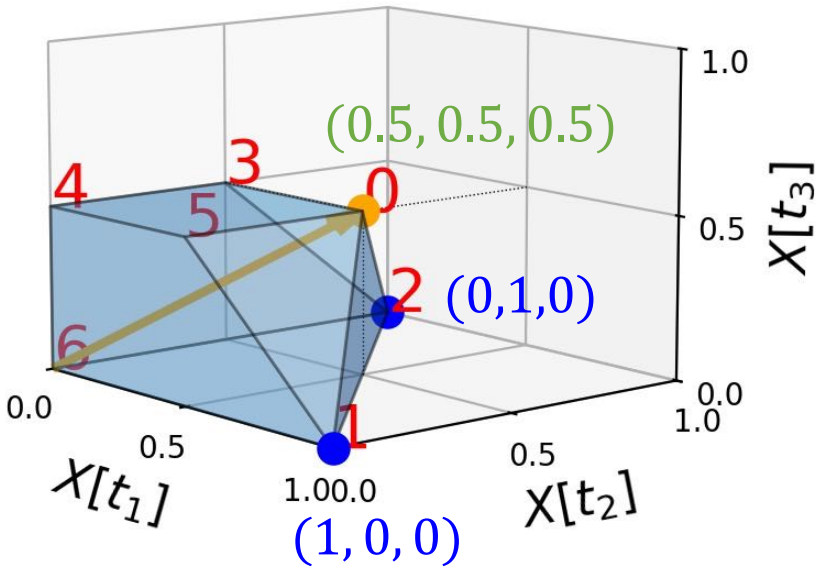
Smoothening constraint (SCs)

$$X[t_3] \leq X[w_1] + X[w_2] - 1$$

$$0.5 \not\leq 0.5 + 0.5 - 1$$

Example: Smoothening for Smallest Witness Problem

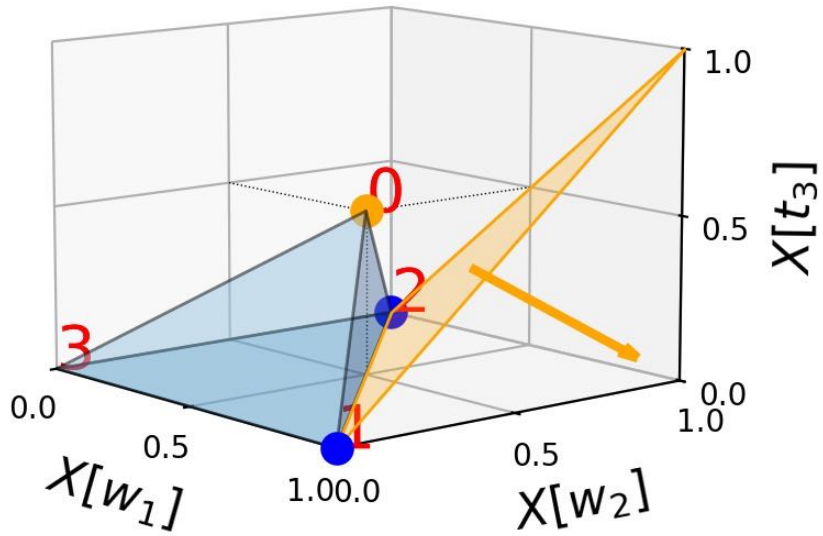
DETAILS



0: $\mathbf{X[t]} = (0.5, 0.5, 0.5)$

1: $\mathbf{X[t]} = (1, 0, 0)$

2: $\mathbf{X[t]} = (0, 1, 0)$

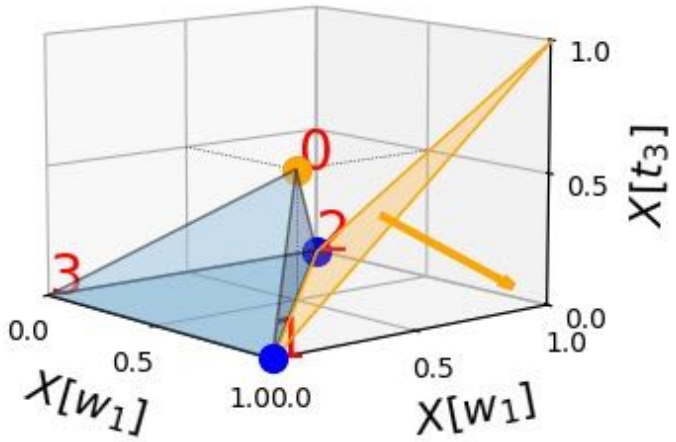


The smoothening constraint eliminates the fractional solution with $f^* = -1.5$

Smoothening constraint (SCs)

$X[t_3] \leq X[w_1] + X[w_2] - 1$

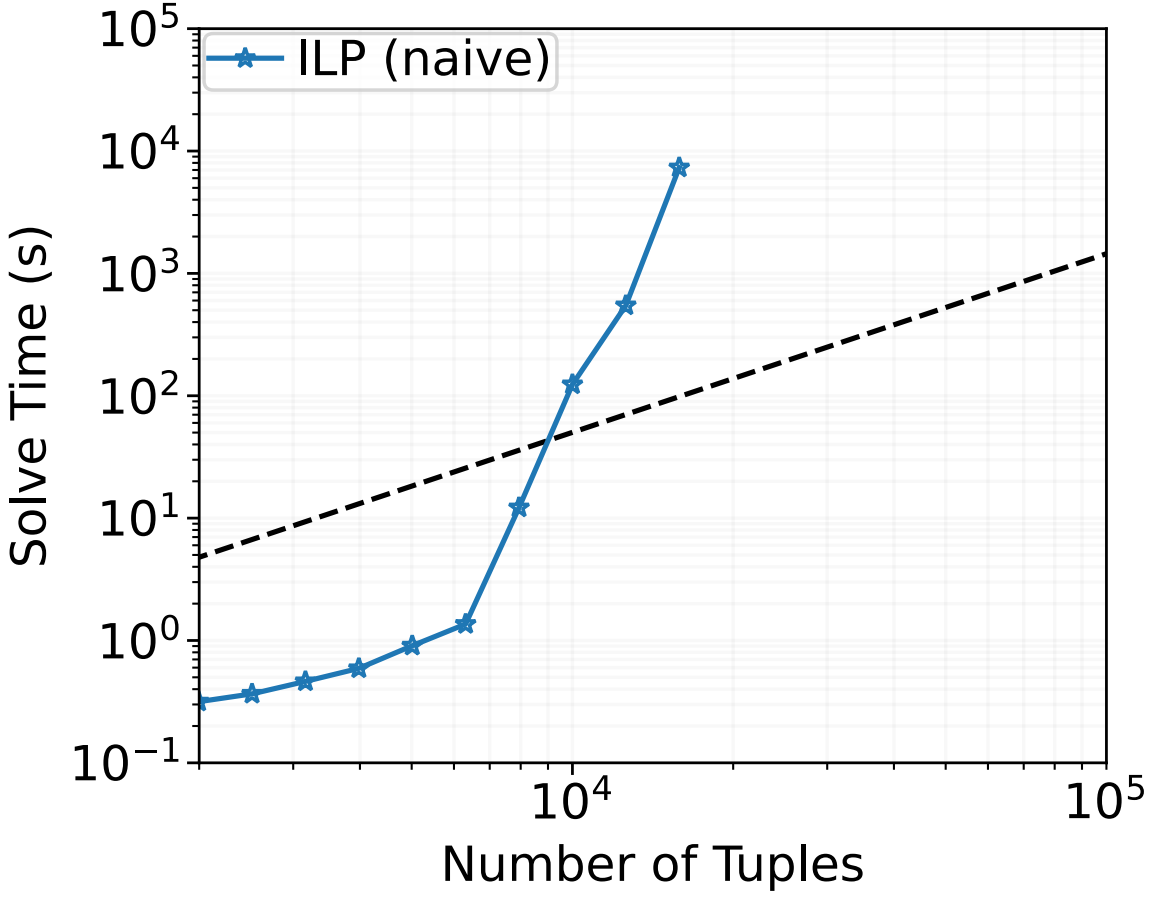
$0.5 \not\leq 0.5 + 0.5 - 1$



Scalability of Naive vs. Smoothened ILP for PTIME query

Finding an optimal solution for the Smallest Witness Problem

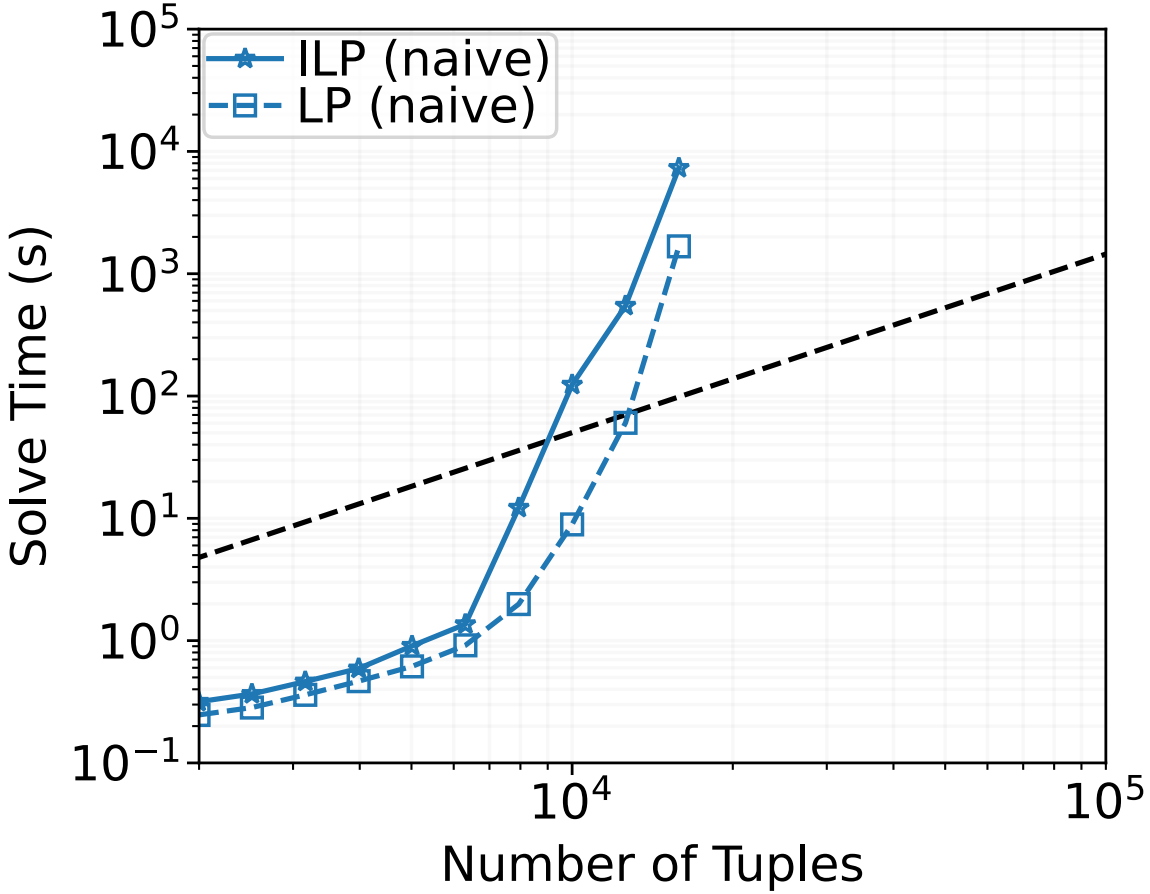
$Q_{5\text{star}}(x)\text{: } -R(x,a), S(x,b), T(x,c), U(x,d), V(x,e), A(x)$

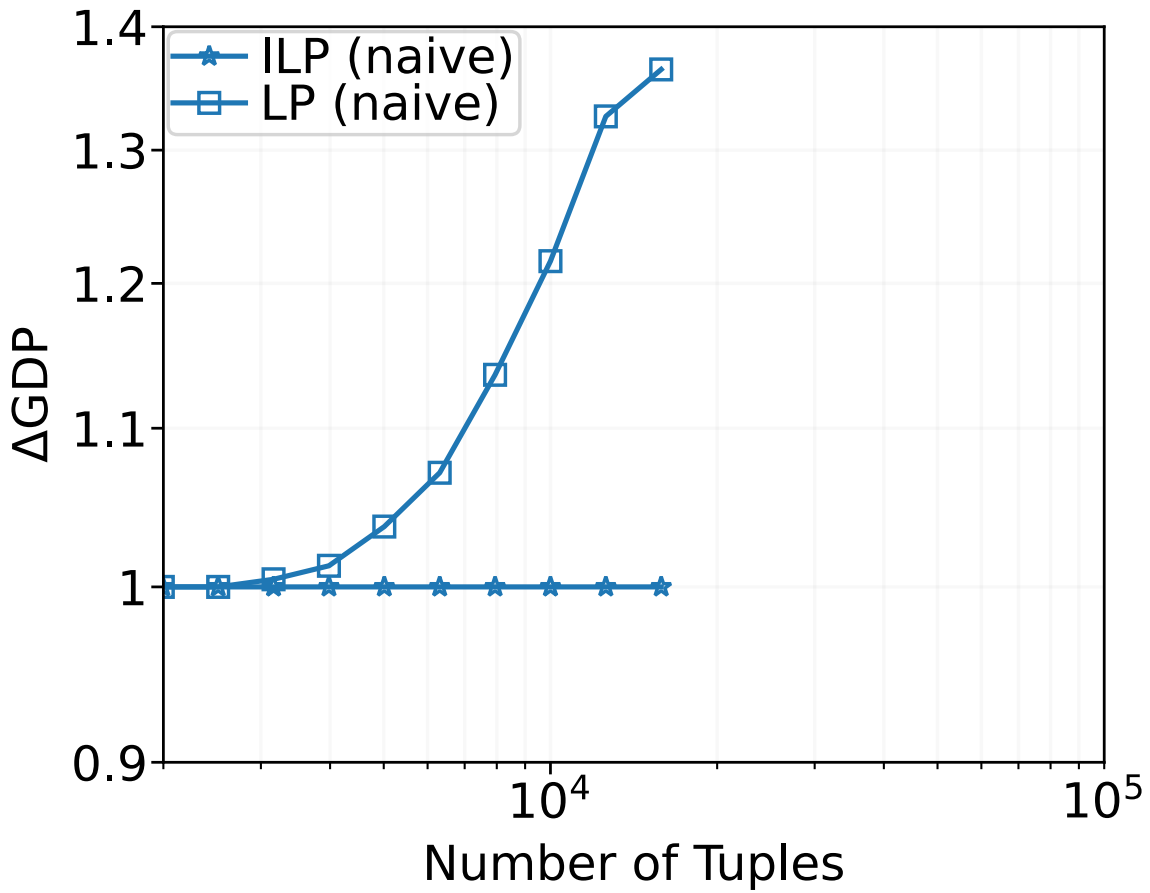


Scalability of Naive vs. Smoothened ILP for PTIME query

Finding an optimal solution for the Smallest Witness Problem

$Q_{5\text{star}}(x)$: $-R(x,a), S(x,b), T(x,c), U(x,d), V(x,e), A(x)$

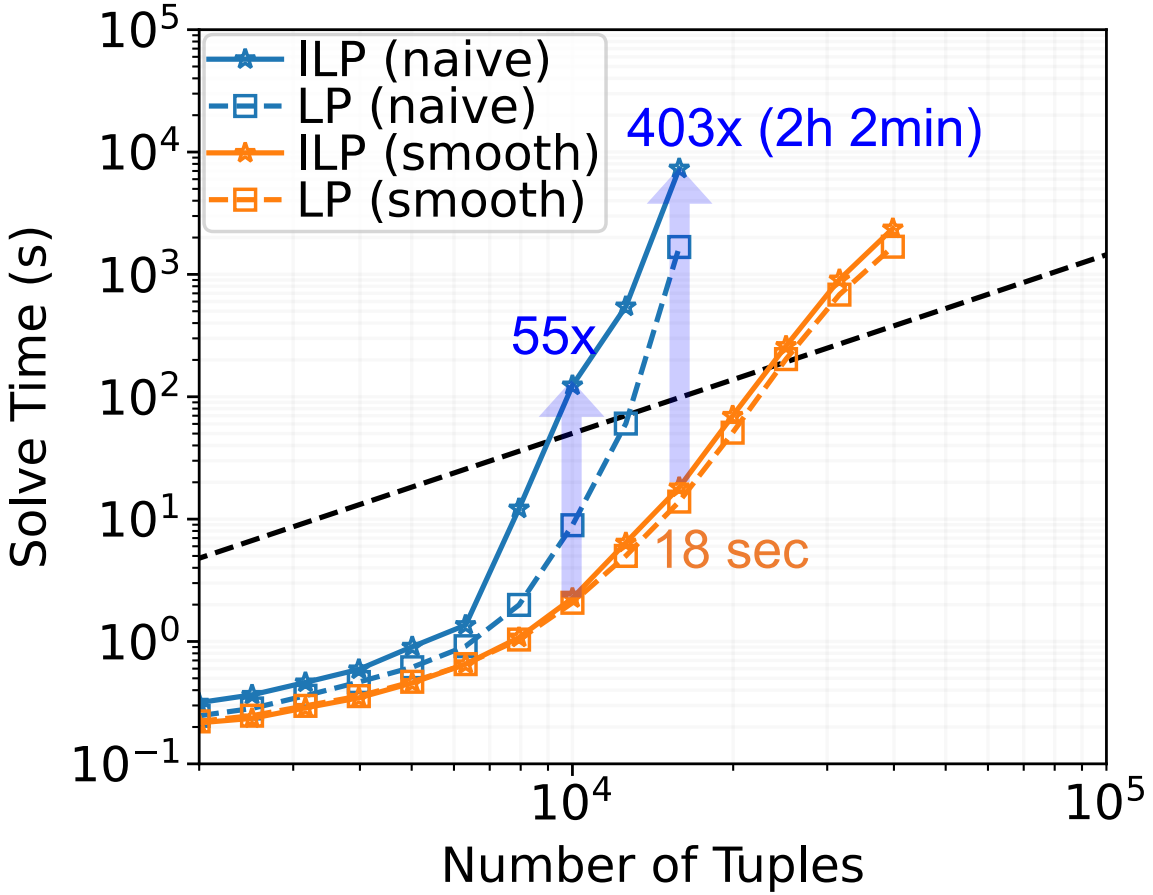


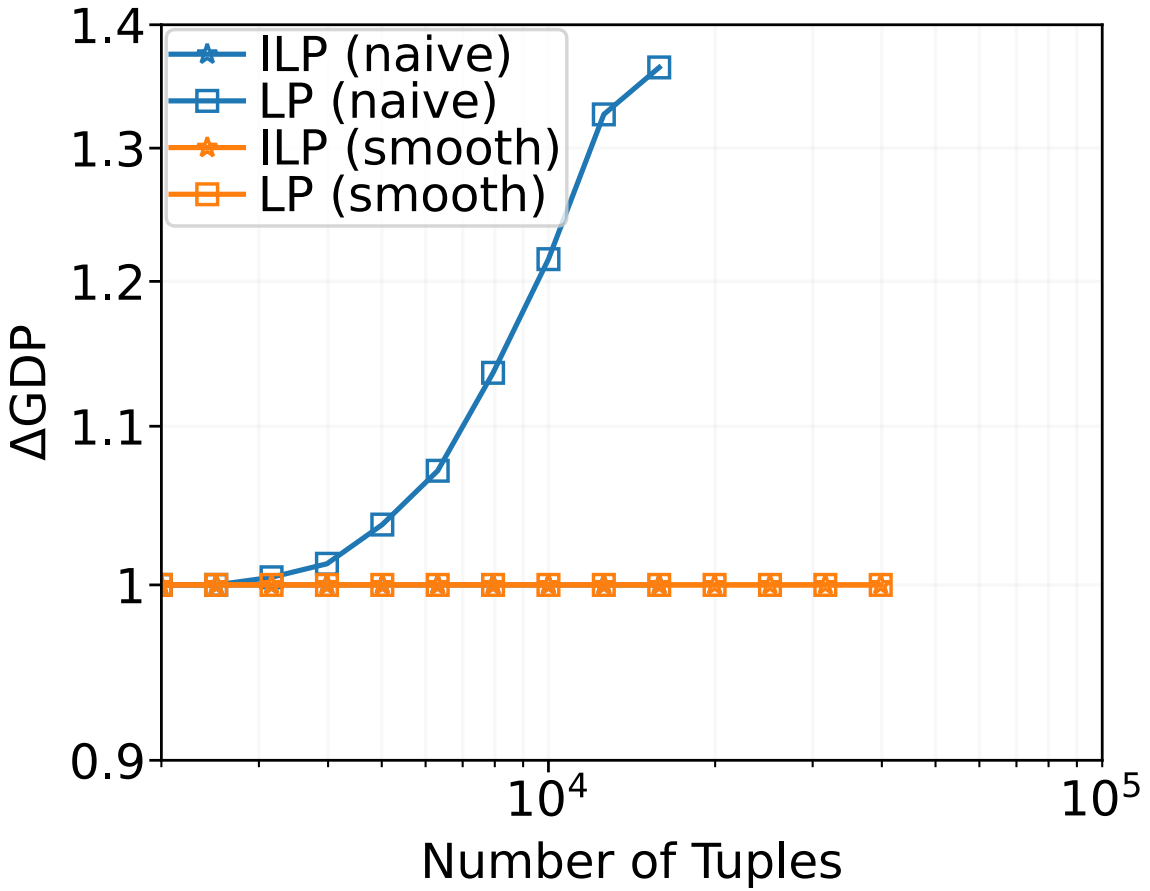


Scalability of Naive vs. Smoothened ILP for PTIME query

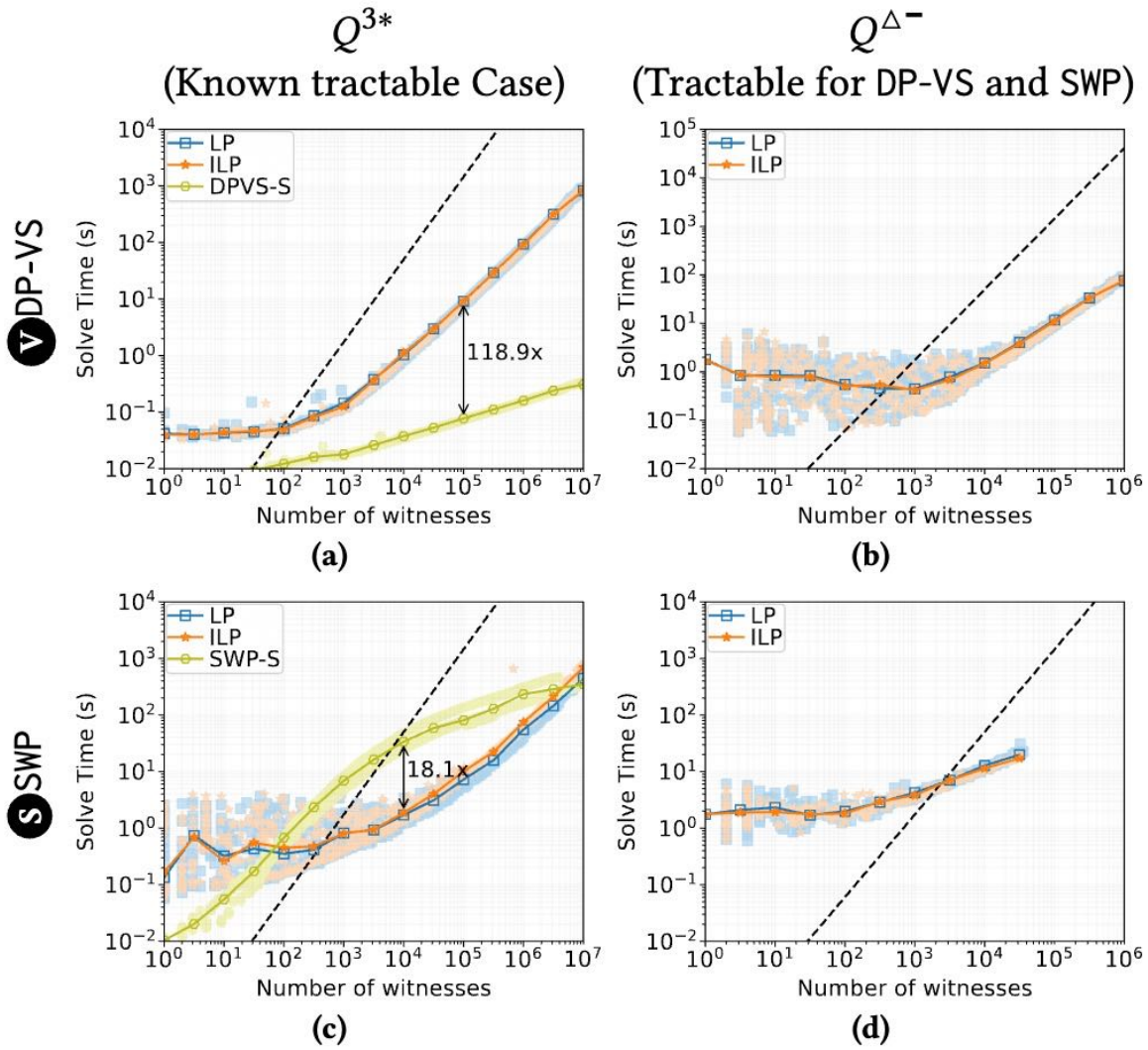
Finding an optimal solution for the Smallest Witness Problem

$Q_{5\text{star}}(x)$: $-R(x,a), S(x,b), T(x,c), U(x,d), V(x,e), A(x)$





Comparison with specialized algorithms for PTIME cases



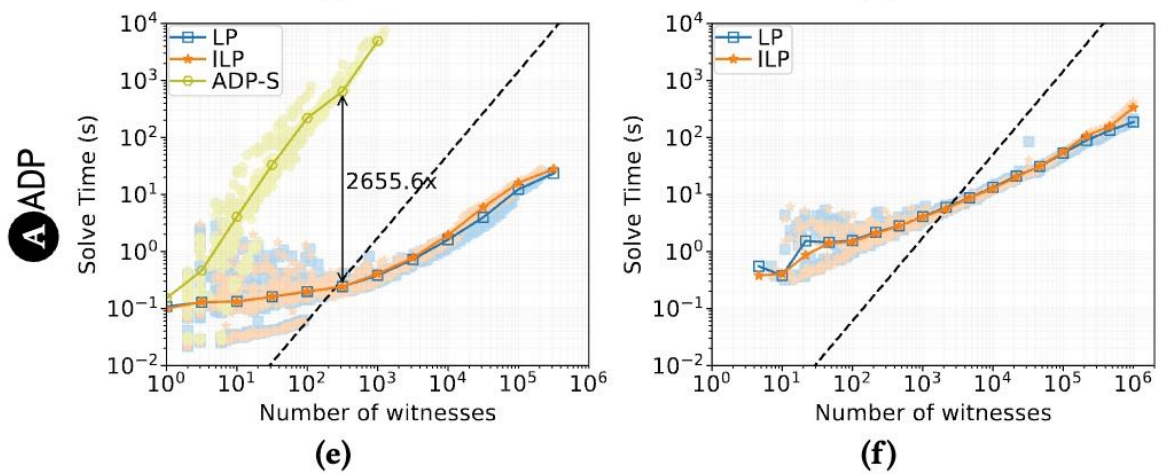


Figure 8: Performance of ILP[GDP] on a previously known (left column compared against prior specialized algorithms) and a newly discovered tractable query (right column with no prior known specialized algorithm) for three prior studied problems (V) DP-VS, (S) SWP, and (A) ADP. In all cases, ILP[GDP] scales well under the theoretical worst case complexity of ILPs and LPs. This is due to the nice algorithmic properties of ILP[GDP], which ensure that the optimal LP solution is integral for all known tractable queries.

A possible shift of focus of algorithm design in database theory?

Current focus: discrete algorithms

Identify tractable cases for a class of problems that can be solved with some dedicated discrete algorithm (like dynamic programming or greedy) or a reduction to flow

For the hard cases:

- prove hardness via some dedicated reduction from some NPC problem.
- optionally design a separate dedicated approximation algorithm

Partial solutions: Often, the algorithm (the dichotomy) does not extend to all types of queries like CQs with self-joins, or problems under bag semantics

Future: polyhedral algorithms

Design one "appropriate" ILP program to solve all problems

- "appropriate" here means that their natural LP relaxation has the same optimal objective for all PTIME cases ("LP=ILP"), which proves the ILP can be solved in PTIME.

All cases are covered (including the hard ones) ✓

- also approximation algorithms, just stop evaluation early, anytime algorithm comes for free ✓

Complete solutions: All problem types are covered ✓ (including self-joins or bag semantics)

A possible shift of focus of algorithm design in database theory?

Current focus: discrete algorithms

Identify tractable cases for a class of problems that can be solved with some dedicated discrete algorithm (like dynamic programming or greedy) or a reduction to flow

For the hard cases:

- prove hardness via some dedicated reduction from some NPC problem.
- optionally design a separate dedicated approximation algorithm

Partial solutions: Often, the algorithm (the dichotomy) does not extend to all types of queries like CQs with self-joins, or problems under bag semantics

Future: polyhedral algorithms

Design one "appropriate" ILP program to solve all problems

- "appropriate" here means that their natural LP relaxation has the same optimal objective for all PTIME cases ("LP=ILP"), which proves the ILP can be solved in PTIME.

All cases are covered (including the hard ones) ✓

- also approximation algorithms, just stop evaluation early, anytime algorithm comes for free ✓

Complete solutions: All problem types are covered ✓ (including self-joins or bag semantics)

An anonymous concern: "...the ILP constructed is not a simple mathematical object... Since the construction given is .... not a simple mathematical object, it is not clear to me how deep one can push this further by analyzing it."

A possible shift of focus of algorithm design in database theory?

Current focus: discrete algorithms

Theory

Identify tractable cases for a class of problems that can be solved with some dedicated discrete algorithm (like dynamic programming or greedy) or a reduction to flow

For the hard cases:

- prove hardness via some dedicated reduction from some NPC problem.
- optionally design a separate dedicated approximation algorithm

Partial solutions: Often, the algorithm (the dichotomy) does not extend to all types of queries like CQs with self-joins, or problems under bag semantics

Practical aspects: usually only some problem cases are solved, hard cases often not treated, the practical nature of the algorithms is not always clear

Future: polyhedral algorithms

Design one "appropriate" ILP program to solve all problems

- "appropriate" here means that their natural LP relaxation has the same optimal objective for all PTIME cases ("LP=ILP"), which proves the ILP can be solved in PTIME.

All cases are covered (including the hard ones) ✓

- also approximation algorithms, just stop evaluation early, anytime algorithm comes for free ✓

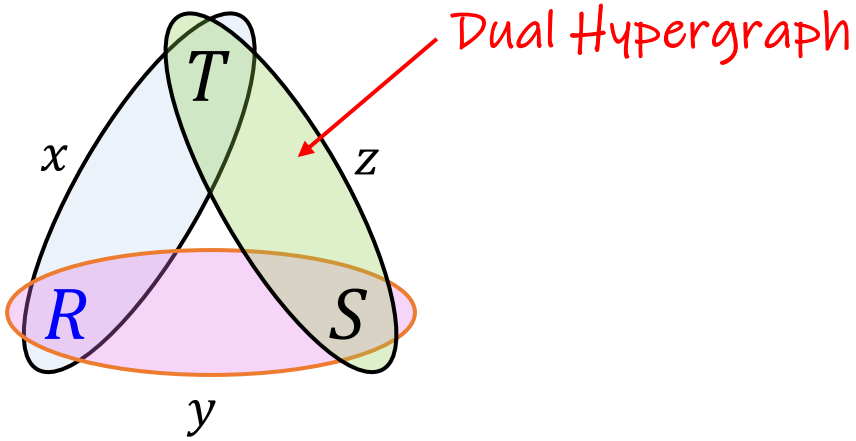
Complete solutions: All problem types are covered ✓ (including self-joins or bag semantics)

Practical aspects: it works from day one ✓

Example complicated landscape for resilience

Triangle query

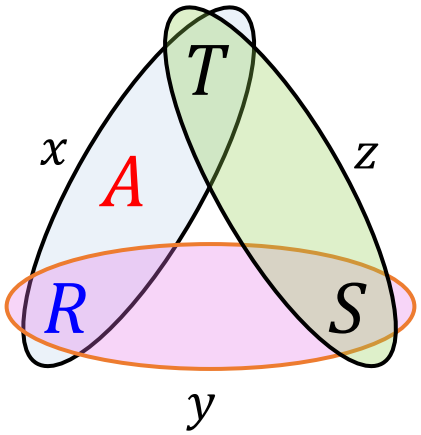
$Q^{\Delta}: -R(x,y), S(y,z), T(x,z)$



NPC

Triangle unary

$Q_A^{\Delta}: -R(x,y), S(y,z), T(x,z), A(x)$



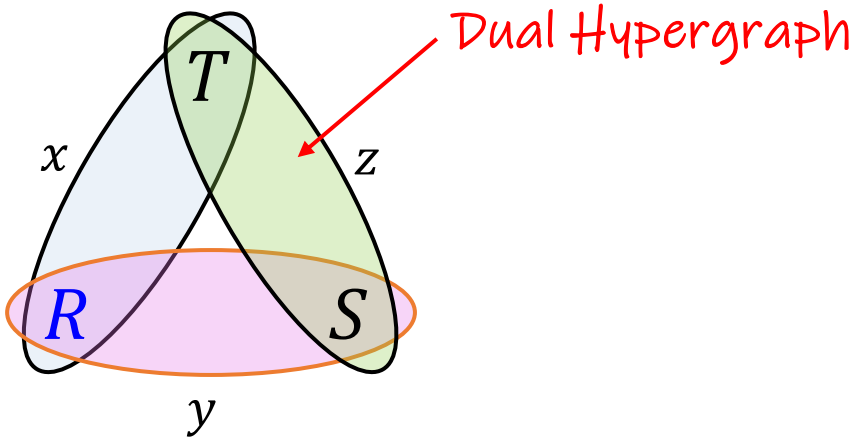
PTIME

Freire, Gatterbauer, Immerman, Meliou. The complexity of resilience and responsibility for self-join-free conjunctive queries, VLDB 2015. https://doi.org/10.14778/2850583.2850592

Example complicated landscape for resilience

Triangle query

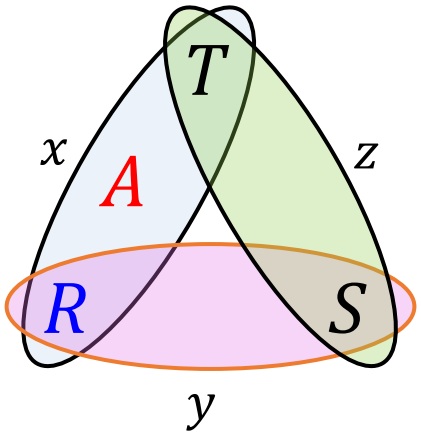
$Q^{\Delta}: -R(x,y), S(y,z), T(x,z)$



NPC

Triangle unary

$Q^{\Delta}_{A}: -R(x,y), S(y,z), T(x,z), A(x)$



PTIME

PTIME for FD $x \rightarrow y$

NPC under bag semantics

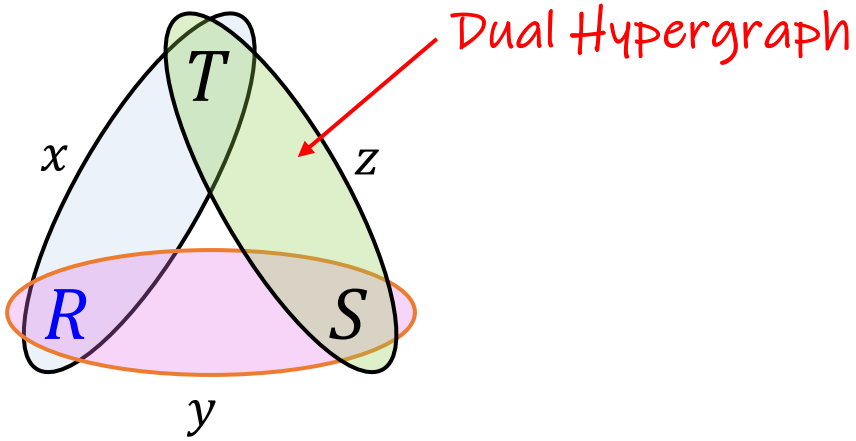
PTIME if provenance happens to be read-once

Freire, Gatterbauer, Immerman, Meliou. The complexity of resilience and responsibility for self-join-free conjunctive queries, VLDB 2015. https://doi.org/10.14778/2850583.2850592
Makhija, Gatterbauer. A Unified Approach for Resilience and Causal Responsibility with Integer Linear Programming (ILP) and LP Relaxations, SIGMOD 2024. https://doi.org/10.1145/3626715

Example complicated landscape for resilience

Triangle query

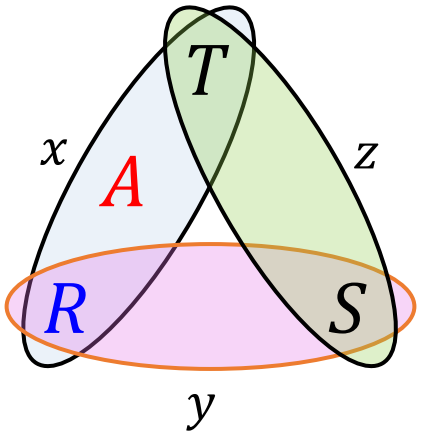
$Q^{\Delta}: -R(x,y), S(y,z), T(x,z)$



NPC

Triangle unary

$Q_A^{\Delta}: -R(x,y), S(y,z), T(x,z), A(x)$



PTIME

PTIME for FD $x \rightarrow y$

NPC under bag semantics

PTIME if provenance happens to be read-once

"Coarse-grained instance-optimal" algorithm

Freire, Gatterbauer, Immerman, Meliou. The complexity of resilience and responsibility for self-join-free conjunctive queries, VLDB 2015. https://doi.org/10.14778/2850583.2850592

Makhija, Gatterbauer. A Unified Approach for Resilience and Causal Responsibility with Integer Linear Programming (ILP) and LP Relaxations, SIGMOD 2024. https://doi.org/10.1145/3626715

Makhija, Gatterbauer: A Unified and Practical Approach for Generalized Deletion Propagation. https://arxiv.org/pdf/2411.17603

Take-aways Generalized Deletion Propagation

- Polyhedral theory solves many database theory problems "out of the box".
  - Shift in focus: instead of trying to find a dedicated PTIME algorithm for PTIME cases, start with a general formulation and prove it finishes in PTIME for PTIME cases.
  - There is some magic in getting the "right" formulation (ILP = LP for PTIME), we don't yet have "the" recipe
  - The proofs for LP=ILP go beyond standard optimization literature. Polyhedral theory alone does not help.
- The overall philosophy is way more general than reverse data management.
  - Makhija, Gatterbauer. *Minimally Factorizing the Provenance of Self-Join Free Conjunctive Queries*, PODS 2024. https://doi.org/10.1145/3651605
  - What about consistent query answering? And even more general logic optimization problems?
- More concretely open: Unifying deletion and insertion propagation ("change propagation"), basically positive and negative provenance / Why or Why not?
  - Meliou, Gatterbauer, Moore, Suciu. *Why so? or Why no? Functional causality for explaining query answers*. MUD 2010. https://arxiv.org/pdf/0912.5340
- Please talk to Neha ☺

Thank you ☺