EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

TLM - Transformation Lifecycle Management

Melanie Herschel
Universität Tübingen
Germany

LRI & INRIA Saclay, France
March 28, 2011

What is Transformation Lifecycle Management?

Transformation Management

- Transformation exists over period of time.
- Makes transformation a first-class entity to be managed.
- Insert, update, and delete transformations over their lifecycle.

Transformation Lifecycle

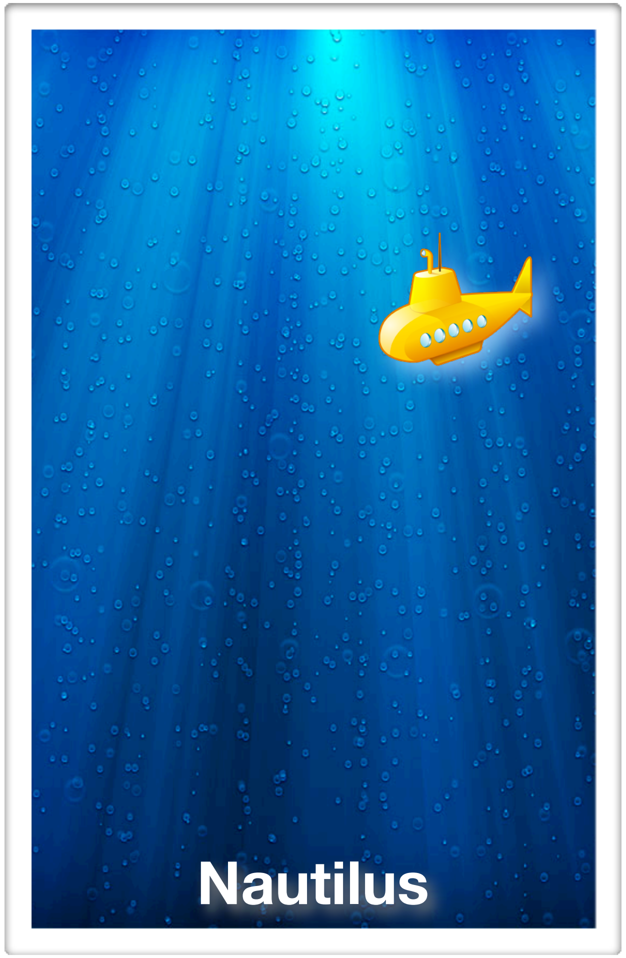
- Transformation is developed
  - Potentially requires debugging, transformation changes, and testing until it works as expected.
  - Transformation management
- Transformation needs to be changed
  - Again, analyze, fix, and test.
  - Change cycles occur over time.
- Transformation retired.

Nautilus

Why Transformation Lifecycle Management?

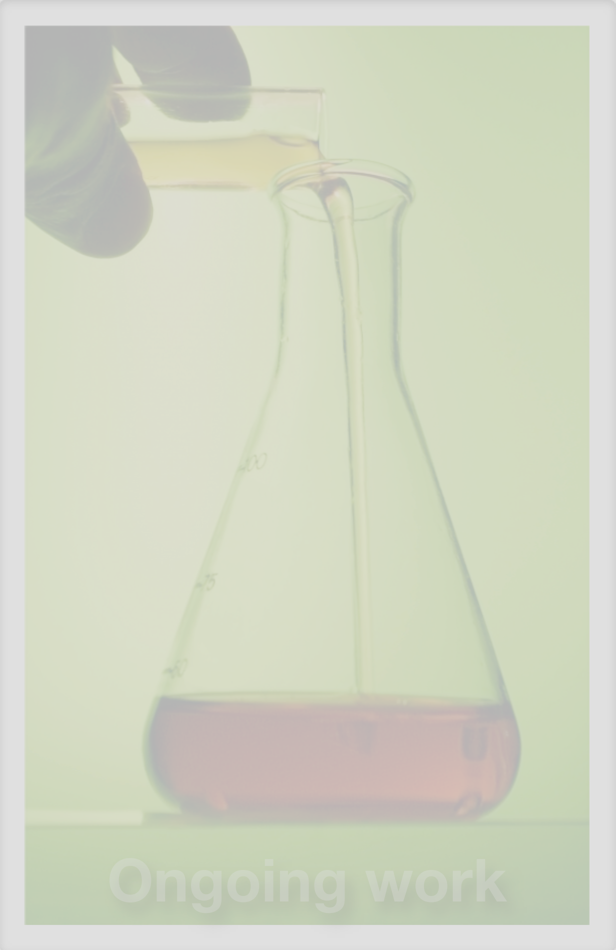
- Manage, share, or document a transformation throughout its entire lifecycle.
- Tool-supported help for developing and evolving transformations.
- Faster development or reaction to requirement changes.
- Makes transformation development easier for non experts (e.g., Web 2.0 content contributors)

Agenda

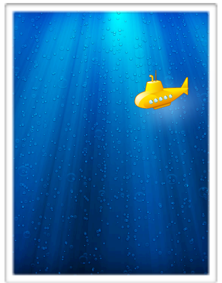


Nautilus

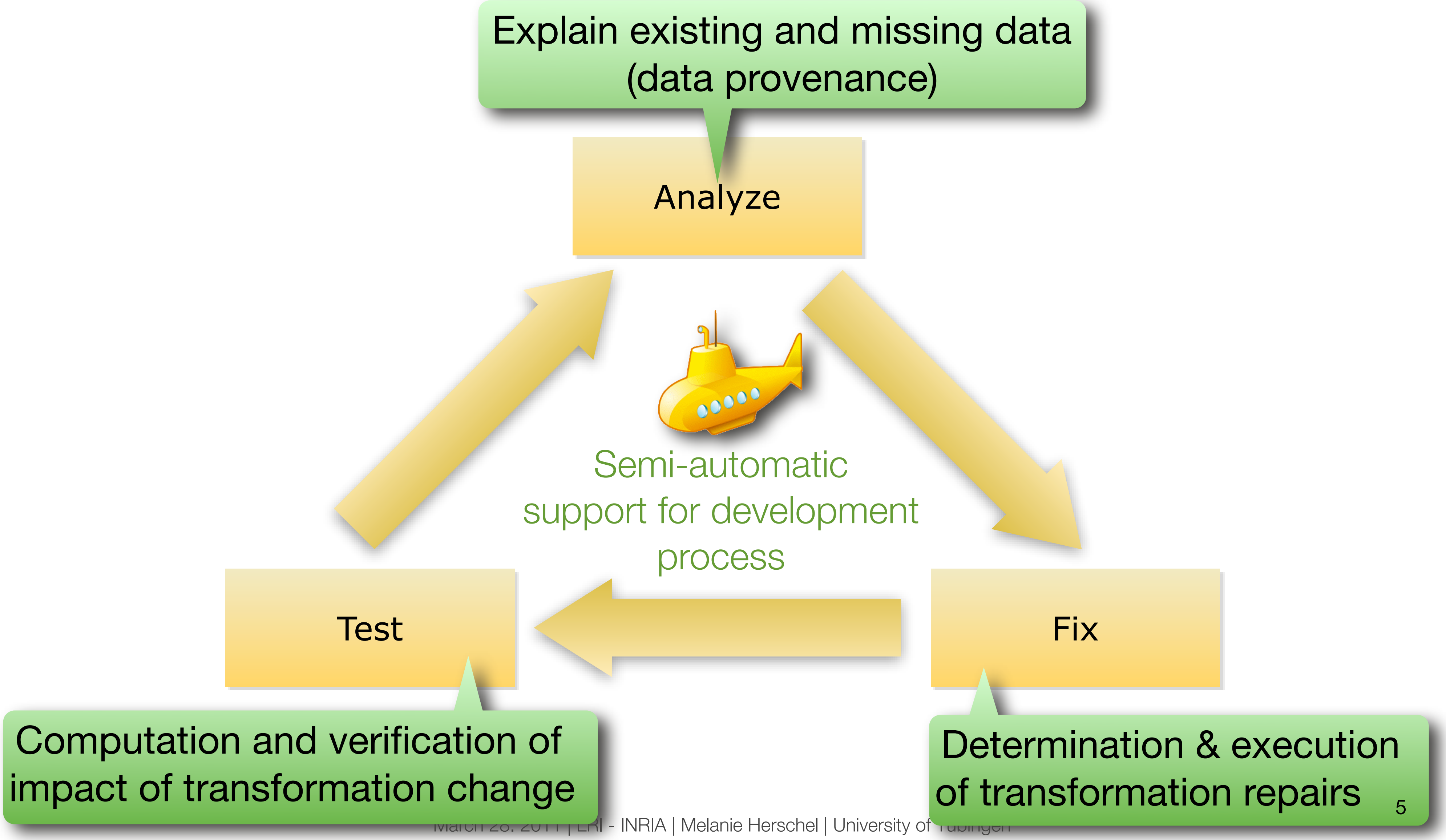
Artemis Algorithm



Ongoing work



Manual vs. Semi-Automatic Process



Explain existing and missing data (data provenance)

Analyze

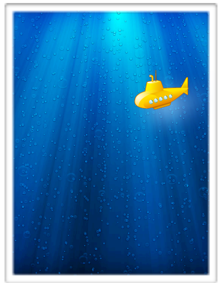
Semi-automatic support for development process

Fix

Test

Computation and verification of impact of transformation change

Determination & execution of transformation repairs

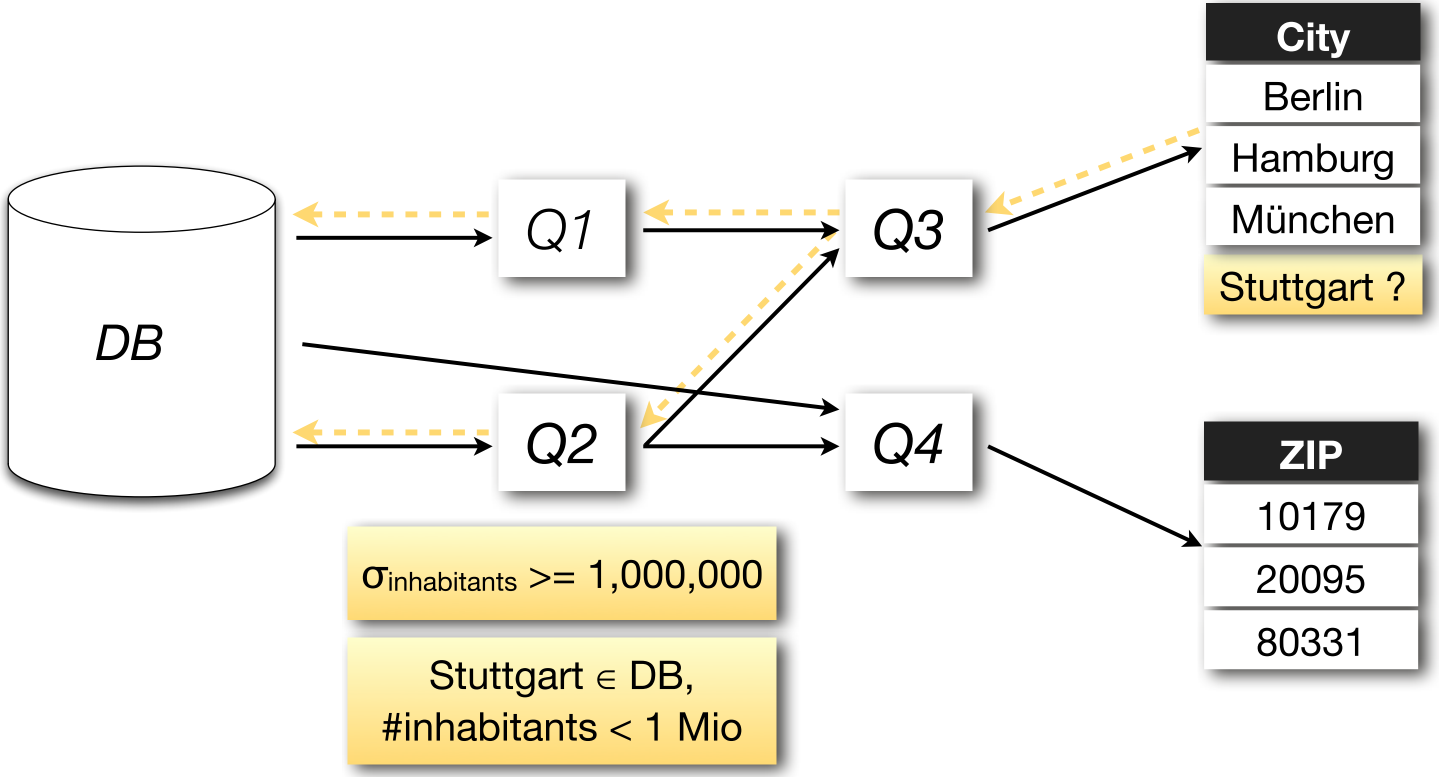


Sample Workflow

Analyze

Fix

Test

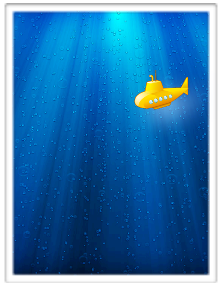


City
Berlin
Hamburg
München
Stuttgart ?

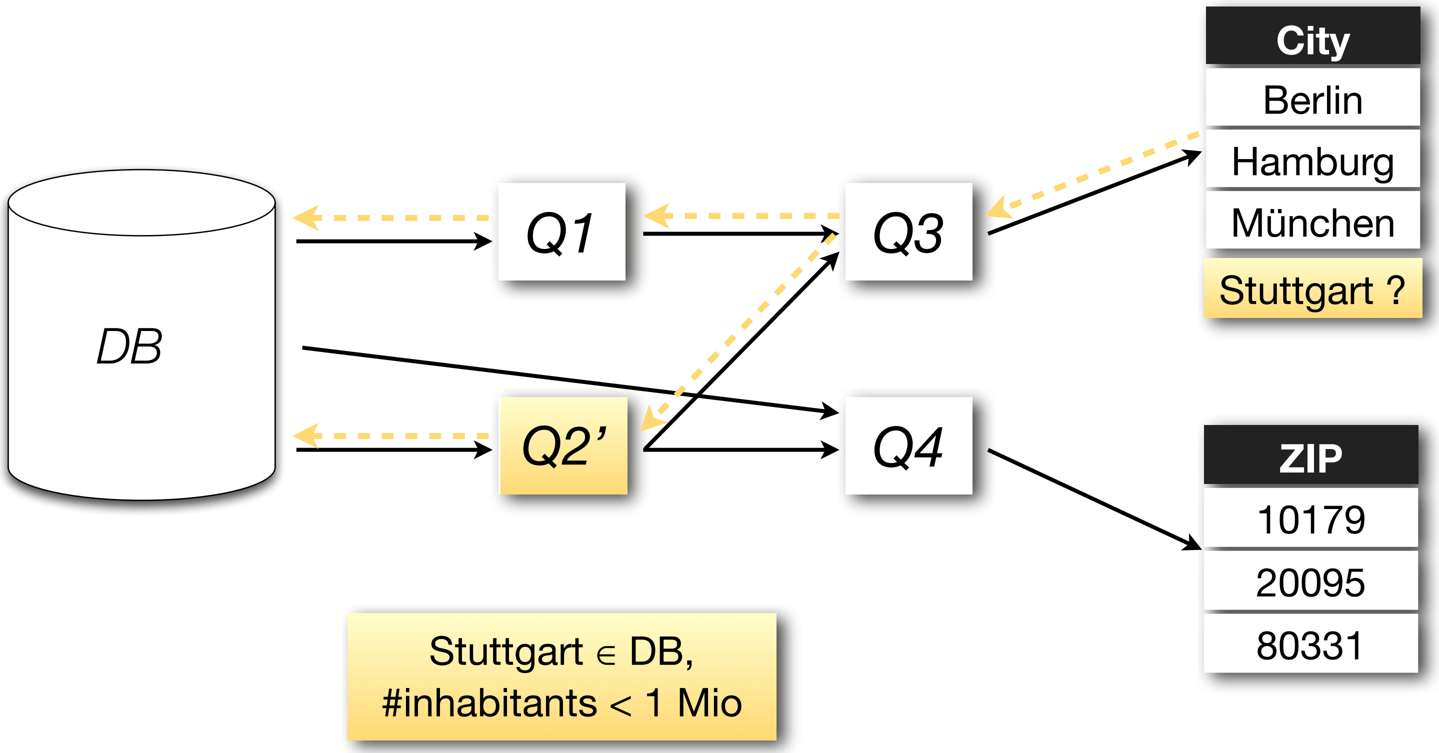
ZIP
10179
20095
80331

$\sigma_{\text{inhabitants}}$ >= 1,000,000

Stuttgart ∊ DB,
#inhabitants < 1 Mio



Sample Workflow



City
Berlin
Hamburg
München
Stuttgart ?

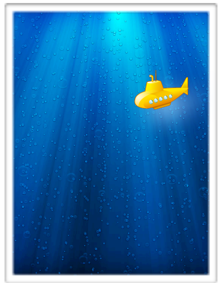
ZIP
10179
20095
80331

Stuttgart ∊ DB,
#inhabitants < 1 Mio

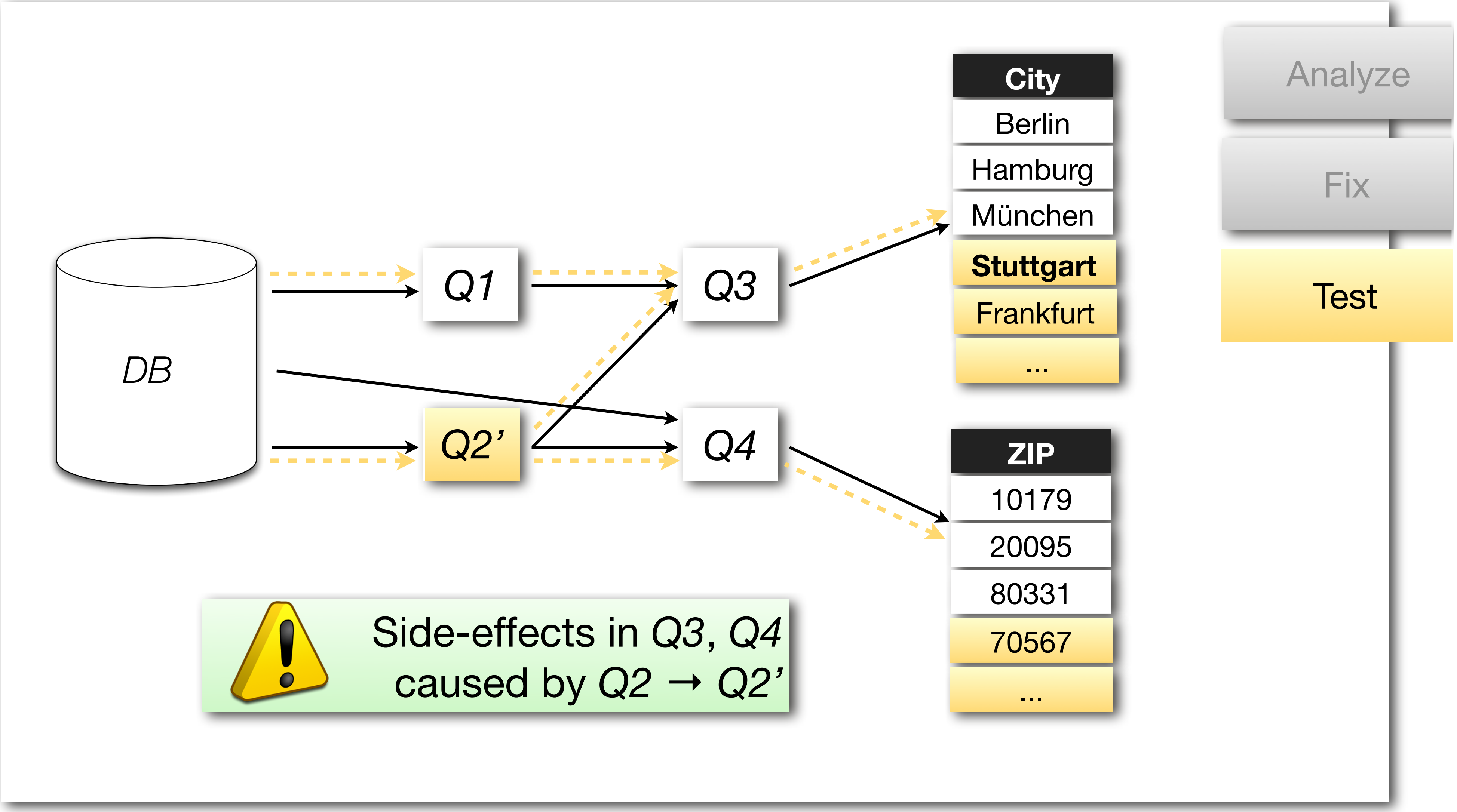
Analyze

Fix

Test



Sample Workflow

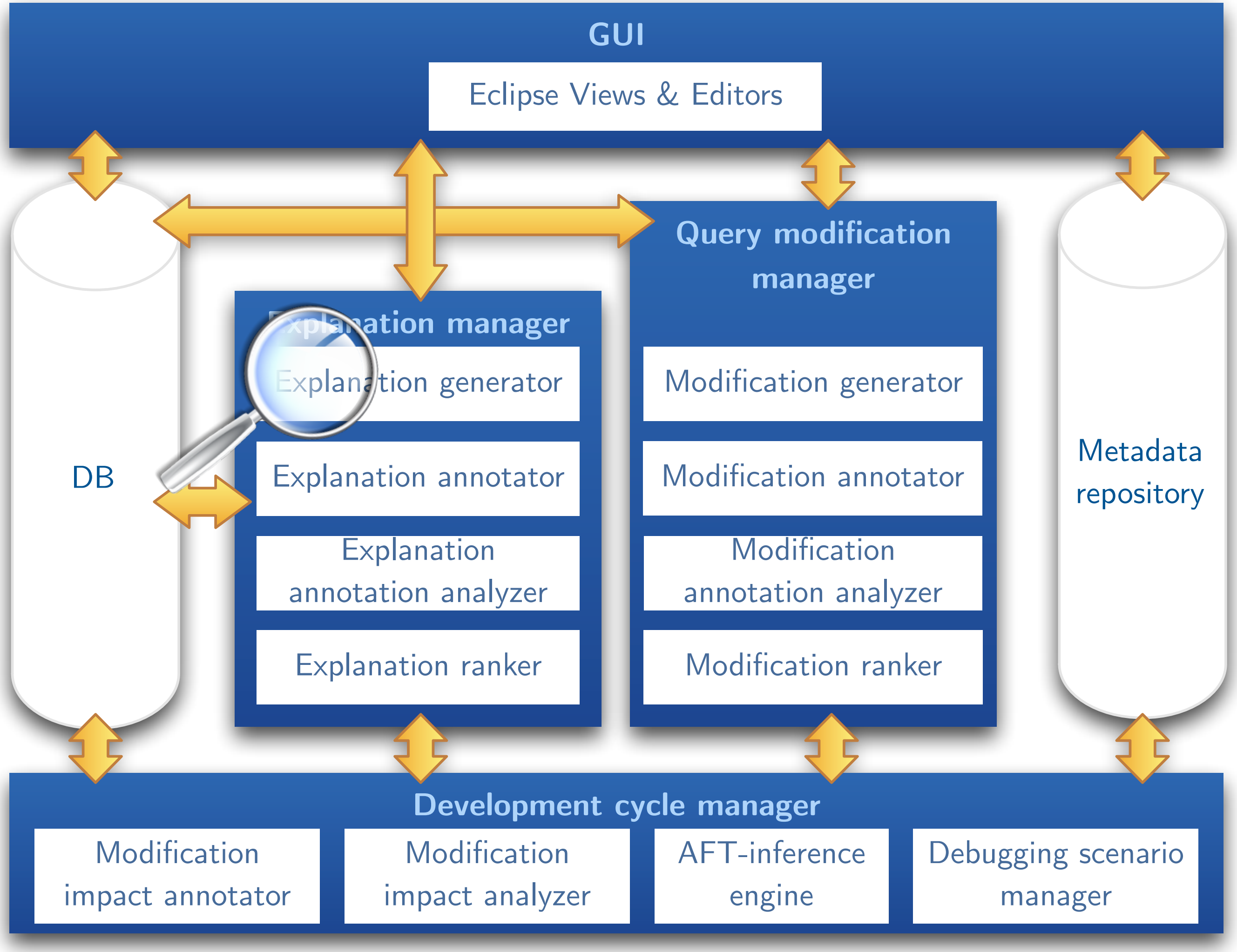


City
Berlin
Hamburg
München
Stuttgart
Frankfurt
...

ZIP
10179
20095
80331
70567
...

Side-effects in *Q3*, *Q4* caused by *Q2* → *Q2’*

Nautilus Architecture


GUI

Eclipse Views & Editors

DB

Explanation manager

Explanation generator

Explanation annotator

Explanation annotation analyzer

Explanation ranker

Query modification manager

Modification generator

Modification annotator

Modification annotation analyzer

Modification ranker

Metadata repository

Development cycle manager

Modification impact annotator

Modification impact analyzer

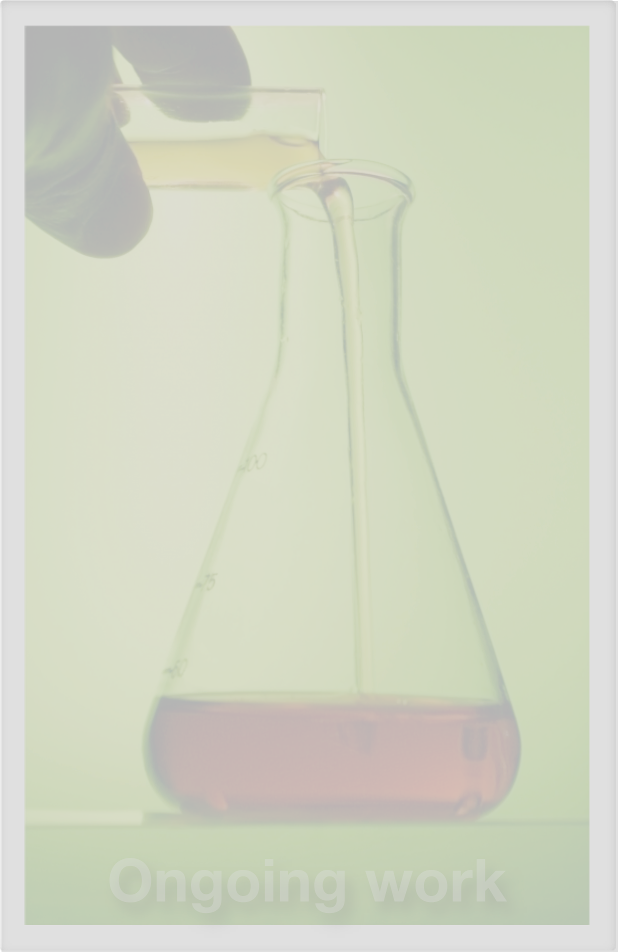
AFT-inference engine

Debugging scenario manager

Agenda

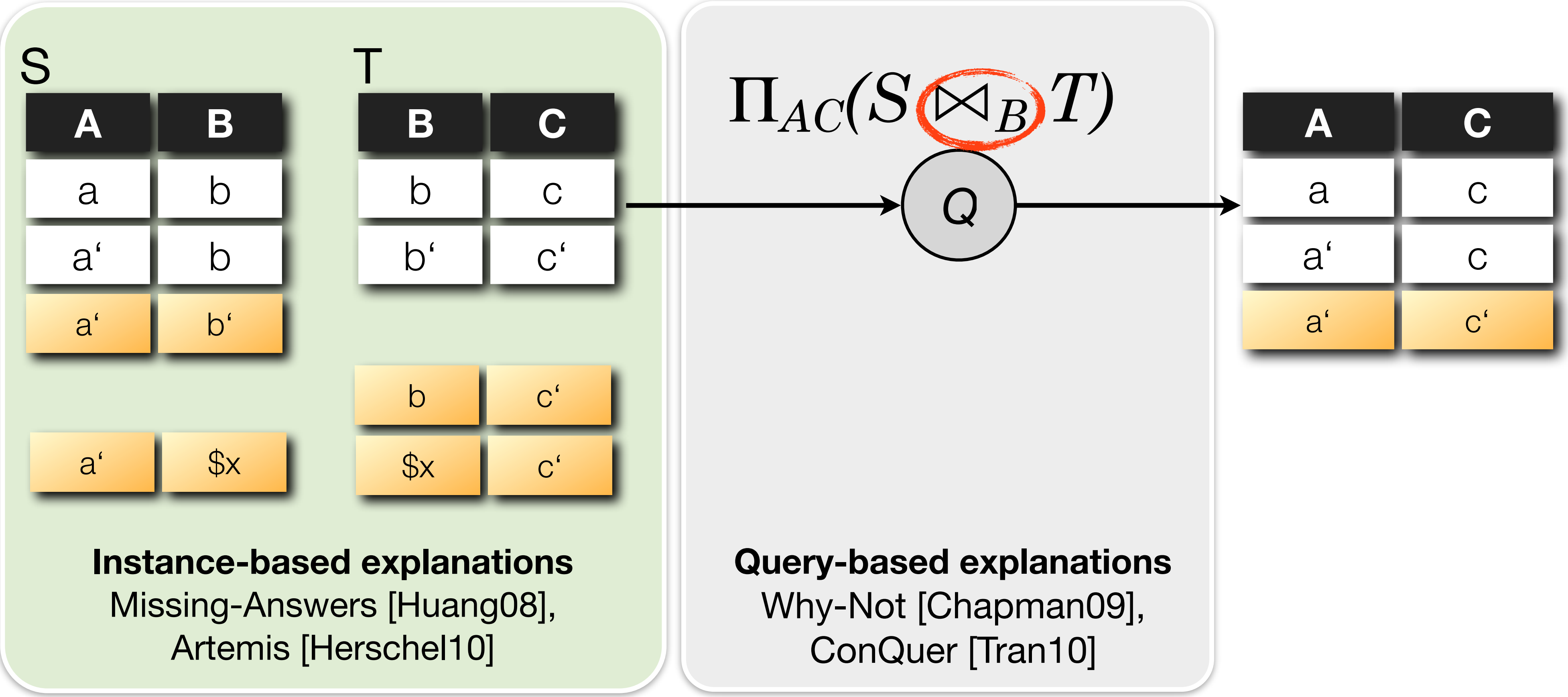
Nautilus

Artemis Algorithm


Ongoing work

Data Provenance of Missing Data

Why is some data not in the result of a query Q?



S

A	B
a	b
a‘	b
a‘	b‘

a‘	$x

T

B	C
b	c
b‘	c‘

b	c‘
$x	c‘

$\Pi_{AC}(S \bowtie_B T)$

A	C
a	c
a‘	c
a‘	c‘

Instance-based explanations
Missing-Answers [Huang08],
Artemis [Herschel10]

Query-based explanations
Why-Not [Chapman09],
ConQuer [Tran10]

Artemis Algorithm

- Explains a **set** of missing tuples over a **set** of queries.
- Queries may involve selection, projection, join, union, aggregation, and grouping (SPJUA).
- Considers **side-effects**.
- Guarantees on completeness and **correctness using a constraint solver**.

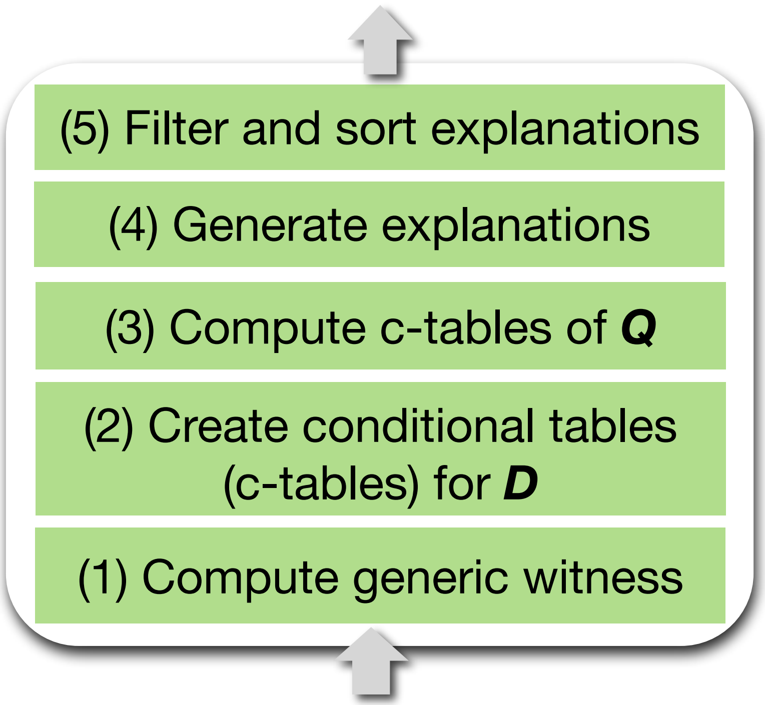
The Artemis Algorithm

For SPJU Queries

S		T	
a‘	b‘	b’	c’

Explanation 1

Insert (a‘, b‘) into *S* s.t. it joins with existing tuple (b‘, c‘) in *T*



Set of explanations ***X***

(5) Filter and sort explanations

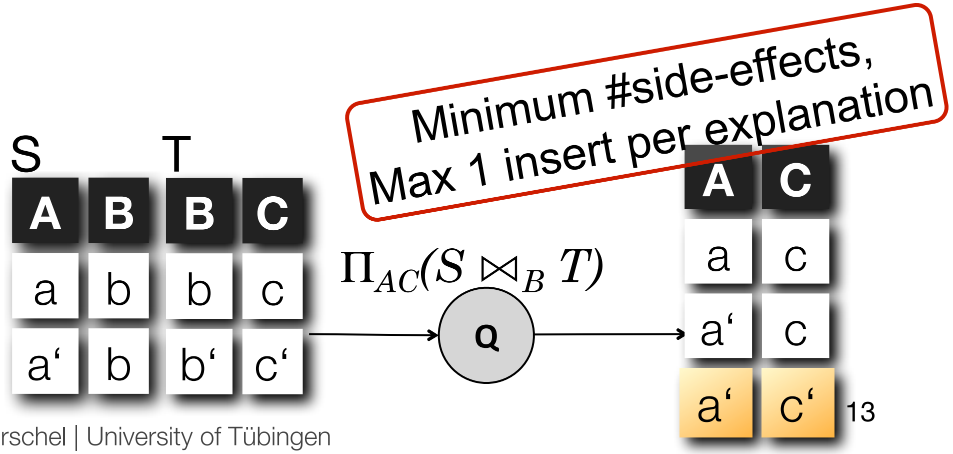
(4) Generate explanations

(3) Compute c-tables of ***Q***

(2) Create conditional tables (c-tables) for ***D***

(1) Compute generic witness

1) Source database ***D***
2) A set of SPJU queries ***Q***
3) A set of missing tuples ***E***
4) Further constraints



Minimum #side-effects,
Max 1 insert per explanation

S

A	B
a	b
a‘	b

T

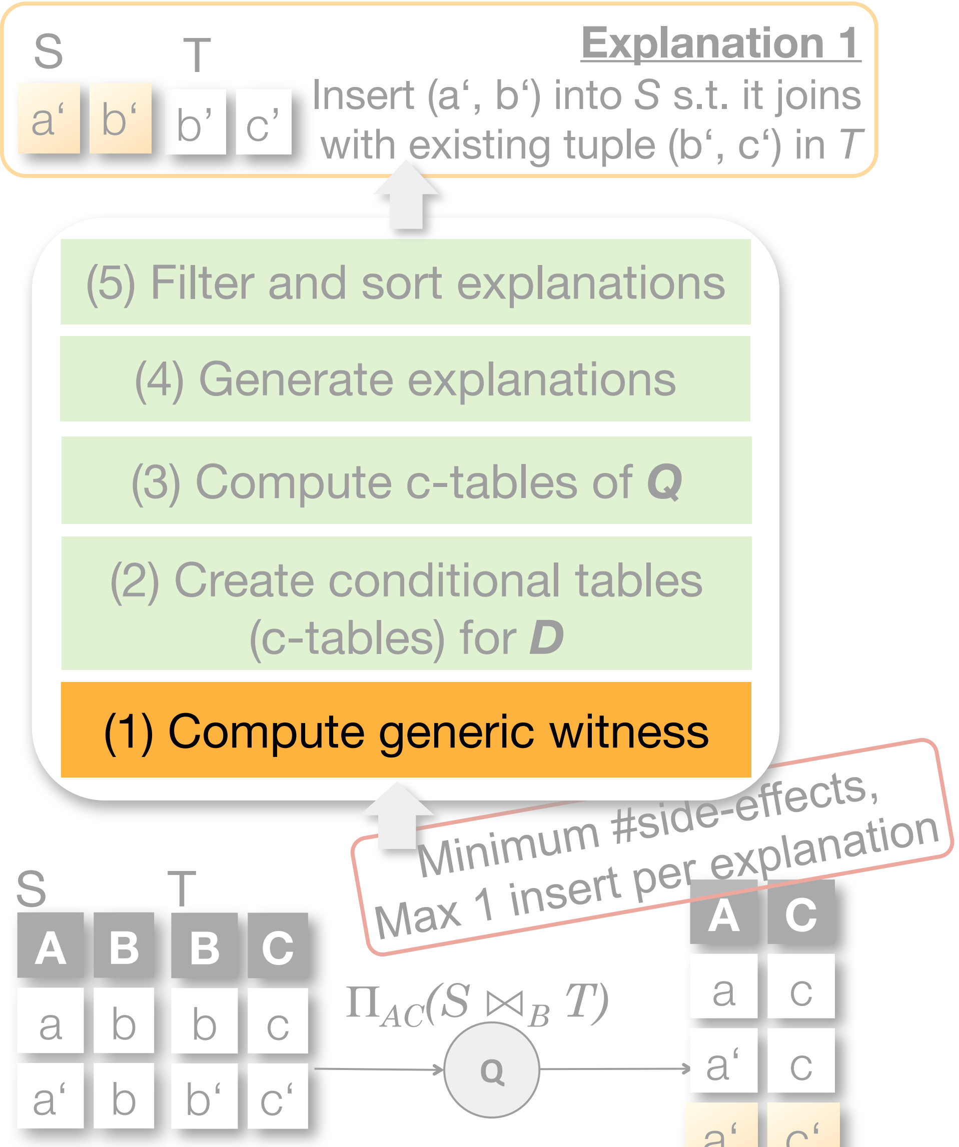
B	C
b	c
b‘	c‘

$\Pi_{AC}(S \bowtie_B T)$

A	C
a	c
a‘	c
a‘	c‘

The Artemis Algorithm

For SPJU Queries



Explanation 1

Insert (a‘, b‘) into *S* s.t. it joins with existing tuple (b‘, c‘) in *T*

(5) Filter and sort explanations

(4) Generate explanations

(3) Compute c-tables of ***Q***

(2) Create conditional tables (c-tables) for ***D***

(1) Compute generic witness

Minimum #side-effects, Max 1 insert per explanation

S

A	B
a	b
a‘	b

T

B	C
b	c
b‘	c‘

$\Pi_{AC}(S \bowtie_B T)$

A	C
a	c
a‘	c
a‘	c‘

Generic Witness:
R(a’, $x), S($x, c’)

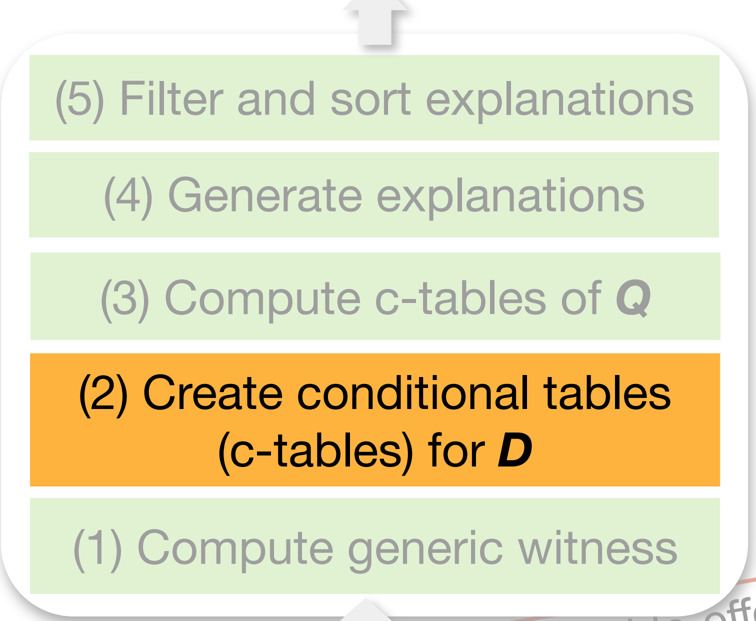
The Artemis Algorithm

For SPJU Queries

S		T	
a‘	b‘	b’	c’

Explanation 1

Insert (a‘, b‘) into *S* s.t. it joins with existing tuple (b‘, c‘) in *T*



(5) Filter and sort explanations

(4) Generate explanations

(3) Compute c-tables of ***Q***

(2) Create conditional tables (c-tables) for ***D***

(1) Compute generic witness

Minimum #side-effects, Max 1 insert per explanation

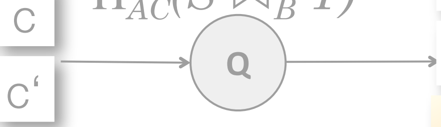
S

A	B
a	b
a‘	b

T

B	C
b	c
b‘	c‘

$\Pi_{AC}(S \bowtie_B T)$



A	C
a	c
a‘	c
a‘	c‘

S^C

A	B	con
a	b	TRUE
a‘	b	TRUE
a’	$x1	$x1 ≠ b

T^C

B	C	con
b	c	TRUE
b‘	c‘	TRUE
$x2	c’	$x2 ≠ b’

Generic Witness:
R(a’, $x), S($x, c’)

The Artemis Algorithm

For SPJU Queries

S		T	
a‘	b‘	b’	c’

Explanation 1

Insert (a‘, b‘) into *S* s.t. it joins with existing tuple (b‘, c‘) in *T*

(5) Filter and sort explanations

(4) Generate explanations

(3) Compute c-tables of *Q*

(2) Create conditional tables (c-tables) for ***D***

(1) Compute generic witness

S

A	B
a	b
a‘	b

T

B	C
b	c
b‘	c‘

$\Pi_{AC}(S \bowtie_B T)$

Q

A	C
a	c
a‘	c
a‘	c‘

Minimum #side-effects,
Max 1 insert per explanation

S^C

A	B	con
a	b	TRUE
a‘	b	TRUE
a’	\$x1	\$x1 ≠ b

T^C

B	C	con
b	c	TRUE
b‘	c‘	TRUE
\$x2	c’	\$x2 ≠ b’

$\Pi_{AC}(S \bowtie_B T)$

Q

A	C	con
a	c	TRUE
a‘	c	TRUE
a	c‘	\$x2 = b ∧ \$x2 ≠ b‘
a‘	c‘	\$x2 = b ∧ \$x2 ≠ b‘
a‘	c‘	\$x1 = b‘∧ \$x1 ≠ b
a‘	c‘	\$x1 = \$x2 ∧ \$x2 ≠ b‘ ∧ \$x1 ≠ b

The Artemis Algorithm

For SPJU Queries

S		T	
a‘	b‘	b’	c’

Explanation 1

Insert (a‘, b‘) into *S* s.t. it joins with existing tuple (b‘, c‘) in *T*

(5) Filter and sort explanations

(4) Generate explanations

(3) Compute c-tables of ***Q***

(2) Create conditional tables (c-tables) for ***D***

(1) Compute generic witness

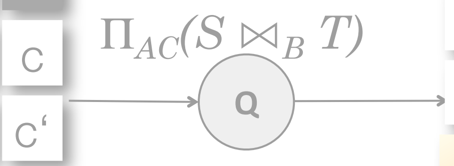
Minimum #side-effects, Max 1 insert per explanation

S

A	B
a	b
a‘	b

T

B	C
b	c
b‘	c‘



A	C
a	c
a‘	c
a‘	c‘

side-effect

matches of (a’,c’)

A	C	con
a	c	TRUE
a‘	c	TRUE
a	c‘	$\$x2 = b \wedge \$x2 \neq b‘$
a‘	c‘	$\$x2 = b \wedge \$x2 \neq b‘$
a‘	c‘	$\$x1 = b‘ \wedge \$x1 \neq b$
a‘	c‘	$\$x1 = \$x2 \wedge \$x2 \neq b‘ \wedge \$x1 \neq b$

The Artemis Algorithm

For SPJU Queries

Explanation 1

S

a‘	b‘

T

b’	c’

Insert (a‘, b‘) into *S* s.t. it joins with existing tuple (b‘, c‘) in *T*

(5) Filter and sort explanations

(4) Generate explanations

(3) Compute c-tables of ***Q***

(2) Create conditional tables (c-tables) for ***D***

(1) Compute generic witness

Minimum #side-effects, Max 1 insert per explanation

Constraint Satisfaction Problem

tuple (a’,c’) exists

AND

minimum number of side-effects

S

A	B
a	b
a‘	b

T

B	C
b	c
b‘	c‘

$\Pi_{AC}(S \bowtie_B T)$

Q

A	C
a	c
a‘	c
a‘	c‘

side-effect

matches of (a’,c’)

A	C	con
a	c	TRUE
a‘	c	TRUE
a	c‘	$\$x2 = b \wedge \$x2 \neq b‘$
a‘	c‘	$\$x2 = b \wedge \$x2 \neq b‘$
a‘	c‘	$\$x1 = b‘ \wedge \$x1 \neq b$
a‘	c‘	$\$x1 = \$x2 \wedge \$x2 \neq b‘ \wedge \$x1 \neq b$

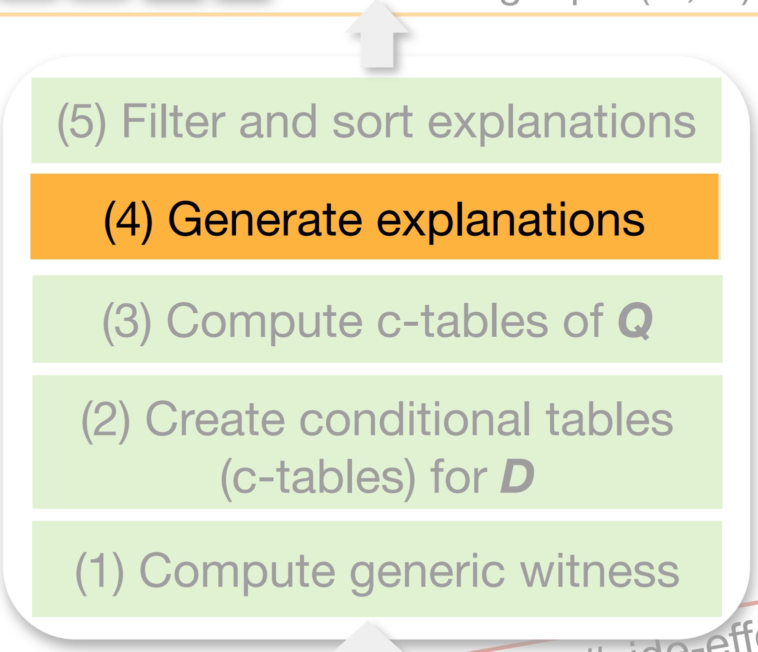
The Artemis Algorithm

For SPJU Queries

S		T	
a‘	b‘	b’	c’

Explanation 1

Insert (a‘, b‘) into *S* s.t. it joins with existing tuple (b‘, c‘) in *T*



(5) Filter and sort explanations
(4) Generate explanations
(3) Compute c-tables of ***Q***
(2) Create conditional tables (c-tables) for ***D***
(1) Compute generic witness

Constraint Satisfaction Problem

$\$x2 = b \wedge \$x2 \neq b'$

AND

$\$x2 = b \wedge \$x2 \neq b'$

$\$x1 = b' \wedge \$x1 \neq b$

$\$x1 = \$x2 \wedge \$x2 \neq b' \wedge \$x1 \neq b$

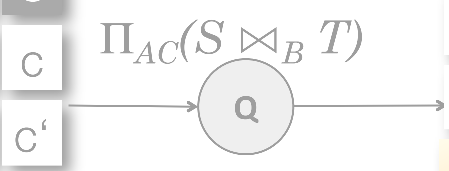
Minimum #side-effects,
Max 1 insert per explanation

S

A	B
a	b
a‘	b

T

B	C
b	c
b‘	c‘



A	C
a	c
a‘	c
a‘	c‘

side-effect

matches of (a’,c’)

A	C	con
a	c	TRUE
a‘	c	TRUE
a	c‘	$\$x2 = b \wedge \$x2 \neq b'$
a‘	c‘	$\$x2 = b \wedge \$x2 \neq b'$
a‘	c‘	$\$x1 = b' \wedge \$x1 \neq b$
a‘	c‘	$\$x1 = \$x2 \wedge \$x2 \neq b' \wedge \$x1 \neq b$

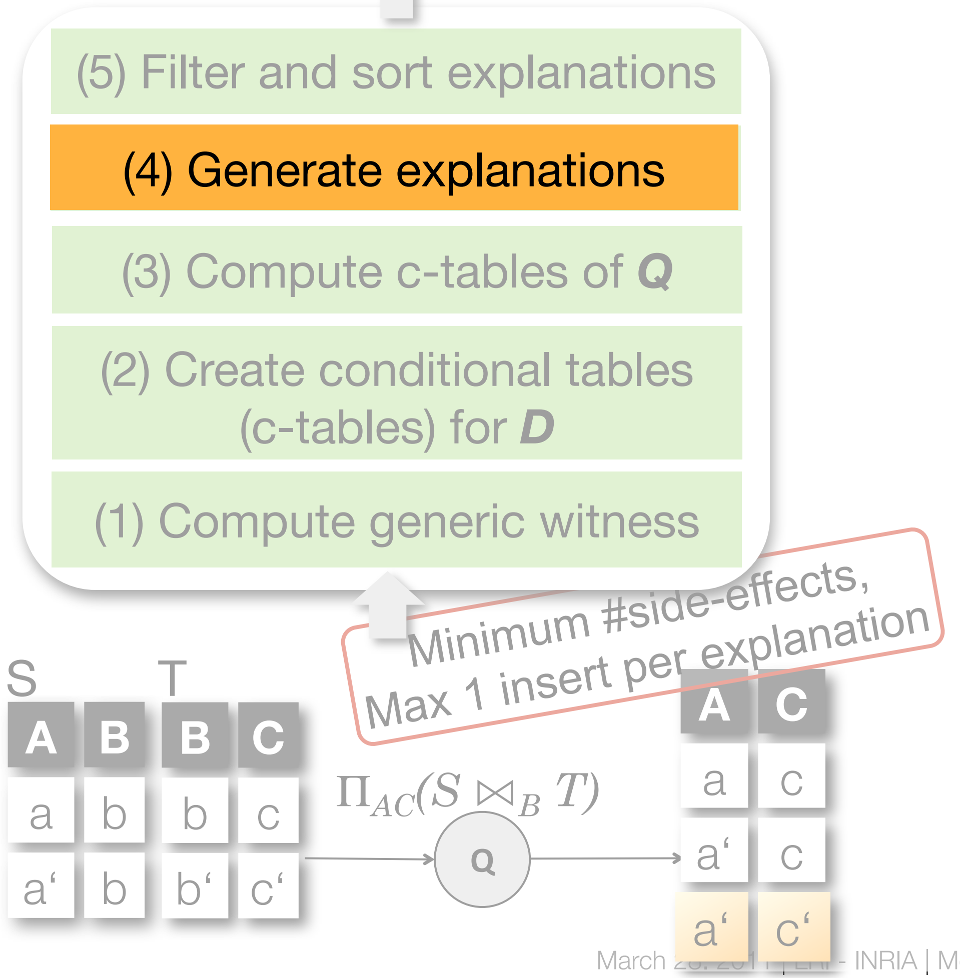
The Artemis Algorithm

For SPJU Queries

S T a‘ b‘ b’ c’

Explanation 1

Insert (a‘, b‘) into *S* s.t. it joins with existing tuple (b‘, c‘) in *T*



(5) Filter and sort explanations

(4) Generate explanations

(3) Compute c-tables of ***Q***

(2) Create conditional tables (c-tables) for ***D***

(1) Compute generic witness

Minimum #side-effects, Max 1 insert per explanation

S

A	B
a	b
a‘	b

T

B	C
b	c
b‘	c‘

$\Pi_{AC}(S \bowtie_B T)$

A	C
a	c
a‘	c
a‘	c‘

S T a‘ b‘ b’ c’

Explanation 1

Insert (a‘, b‘) into *S* s.t. it joins with existing tuple (b‘, c‘) in *T*

S T a‘ $x1 $x2 c’

Explanation 2

Insert (a’,$x1) into *S* and ($x2, c’) into T s.t. $x1 = $x2 and new tuples are no duplicates

Output of constraint solver for...

~~**1st match**: $x2 = b, 1 side-effect~~

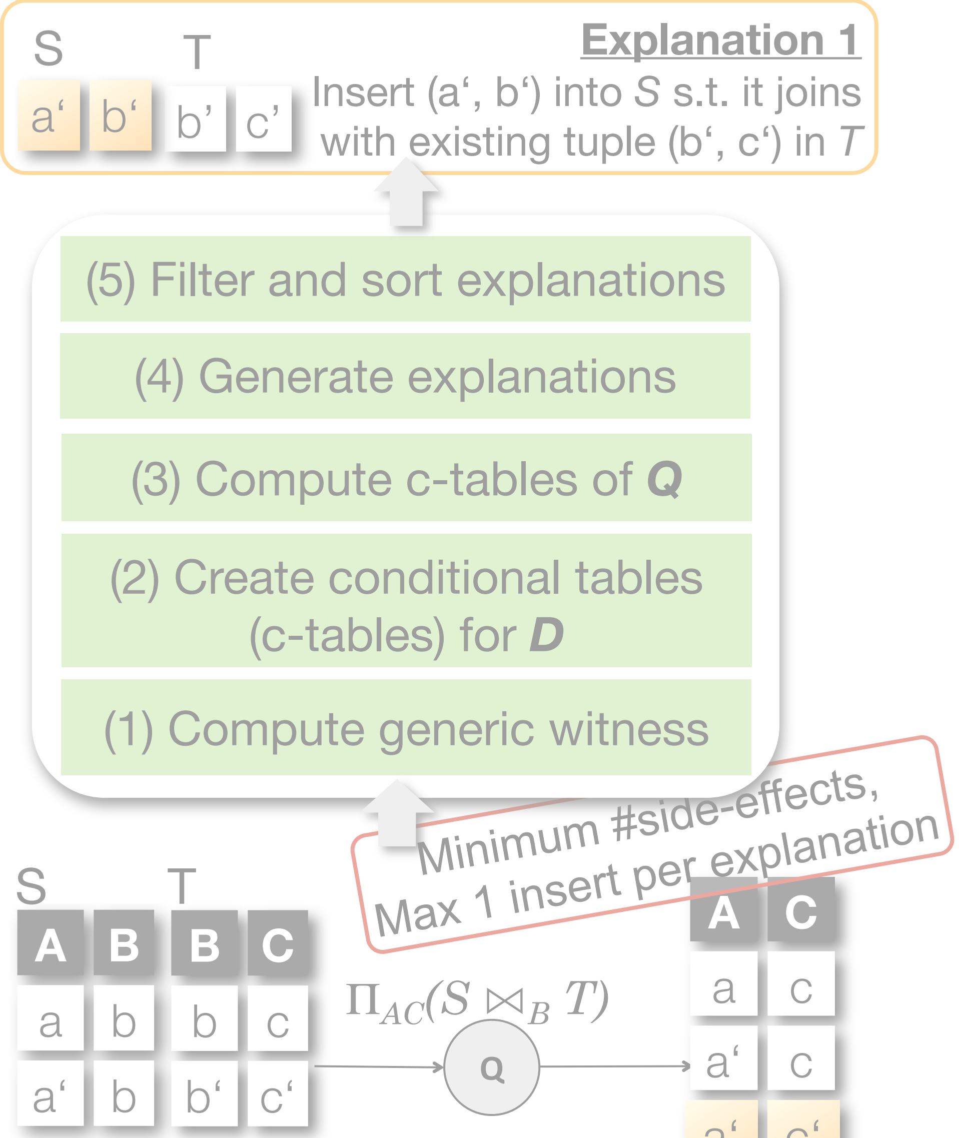
Too many side-effects

2nd match: $x1 = b’, 0 side-effects

3rd match: $x1 = $x2, $x1 ≠ b, $x2 ≠ b’, 0 side-effects

The Artemis Algorithm

For SPJU Queries



Explanation 1

S		T	
a‘	b‘	b’	c’

Insert (a‘, b‘) into *S* s.t. it joins with existing tuple (b‘, c‘) in *T*

(5) Filter and sort explanations

(4) Generate explanations

(3) Compute c-tables of ***Q***

(2) Create conditional tables (c-tables) for ***D***

(1) Compute generic witness

Minimum #side-effects,
Max 1 insert per explanation

S		T	
A	B	B	C
a	b	b	c
a‘	b	b‘	c‘

$\Pi_{AC}(S \bowtie_B T)$

A	C
a	c
a‘	c
a‘	c‘

Explanation 1

S		T	
a‘	b‘	b’	c’

Insert (a‘, b‘) into *S* s.t. it joins with existing tuple (b‘, c‘) in *T*

Explanation 2

S		T	
a‘	$x1	$x2	c’

Insert (a’,$x1) into *S* and ($x2, c’) into T s.t. $x1 = $x2 and new tuples are no duplicates

Output of constraint solver for...

~~**1st match**: $x2 = b, 1 side-effect~~

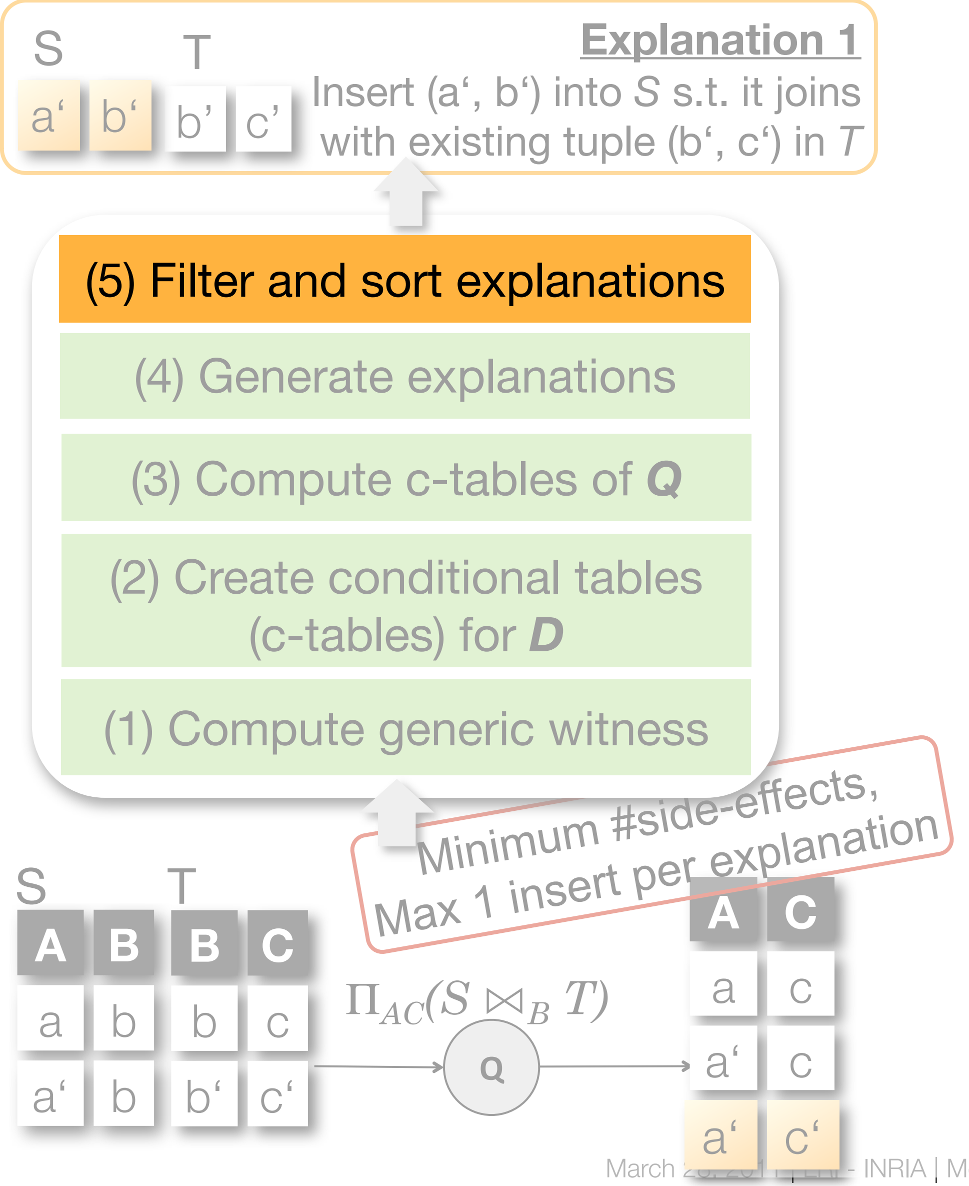
Too many side-effects

2nd match: $x1 = b’,
0 side-effects

3rd match: $x1 = $x2, $x1 ≠ b, $x2 ≠ b’,
0 side-effects

The Artemis Algorithm

For SPJU Queries



S
T
a‘
b‘
b’
c’
Explanation 1
Insert (a‘, b‘) into S s.t. it joins with existing tuple (b‘, c‘) in T
(5) Filter and sort explanations
(4) Generate explanations
(3) Compute c-tables of Q
(2) Create conditional tables (c-tables) for D
(1) Compute generic witness
Minimum #side-effects,
Max 1 insert per explanation
S
T
A
B
B
C
a
b
b
c
a‘
b
b‘
c‘
$\Pi_{AC}(S \bowtie_B T)$
Q
A
C
a
c
a‘
c
a‘
c‘

S		T	
a‘	b‘	b’	c’

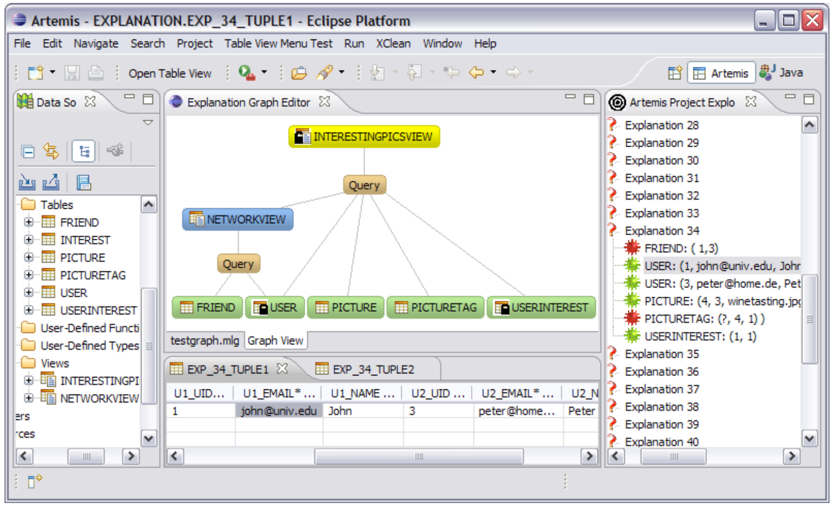
Explanation 1

Insert (a‘, b‘) into *S* s.t. it joins with existing tuple (b‘, c‘) in *T*

Experimental Setup

Implementation

- Eclipse Plugin [Herschel09].
- Artemis and Missing-Answers [Huang08]
- Minion used as constraint solver for Artemis.
- IBM DB2 9.5 used as RDBMS.



U1_UID...	U1_EMAIL* ...	U1_NAME ...	U2_UID ...	U2_EMAIL* ...	U2_N
1	john@univ.edu	John	3	peter@home...	Peter

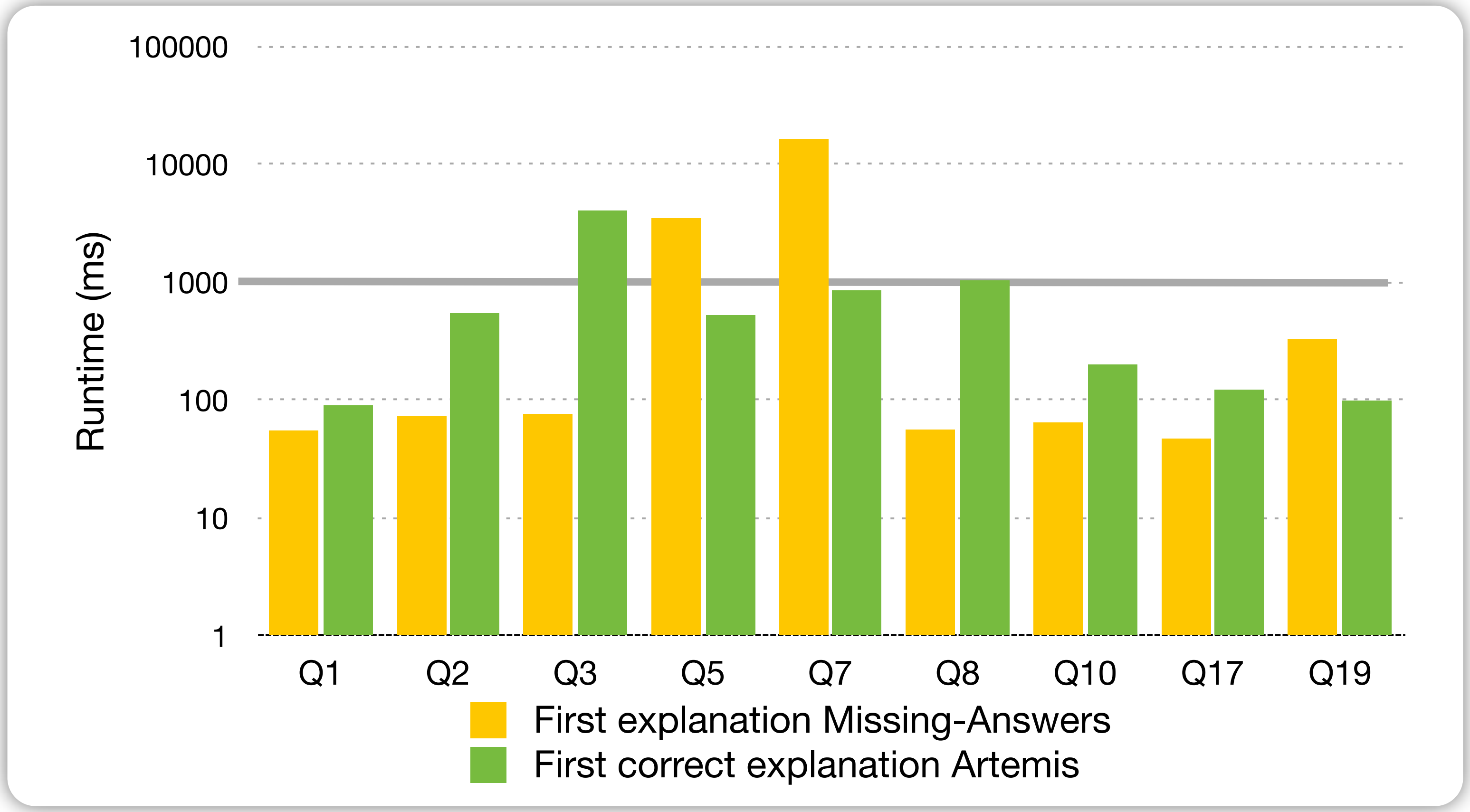
Explanation 28
Explanation 29
Explanation 30
Explanation 31
Explanation 32
Explanation 33
Explanation 34
FRIEND: (1,3)
USER: (1, john@univ.edu, Johr
USER: (3, peter@home.de, Pet
PICTURE: (4, 3, winetasting.jpg
PICTURETAG: (?, 4, 1))
USERINTEREST: (1, 1)
Explanation 35
Explanation 36
Explanation 37
Explanation 38
Explanation 39
Explanation 40

Datasets

- TPCH
  - 10 MB of data
  - 9 queries (adaptations of TPCH queries limited to supported types of queries)
  - No insertions on `Nation` and `Region`.

TPC Transaction Processing Performance Council

Runtime to First Correct Explanation

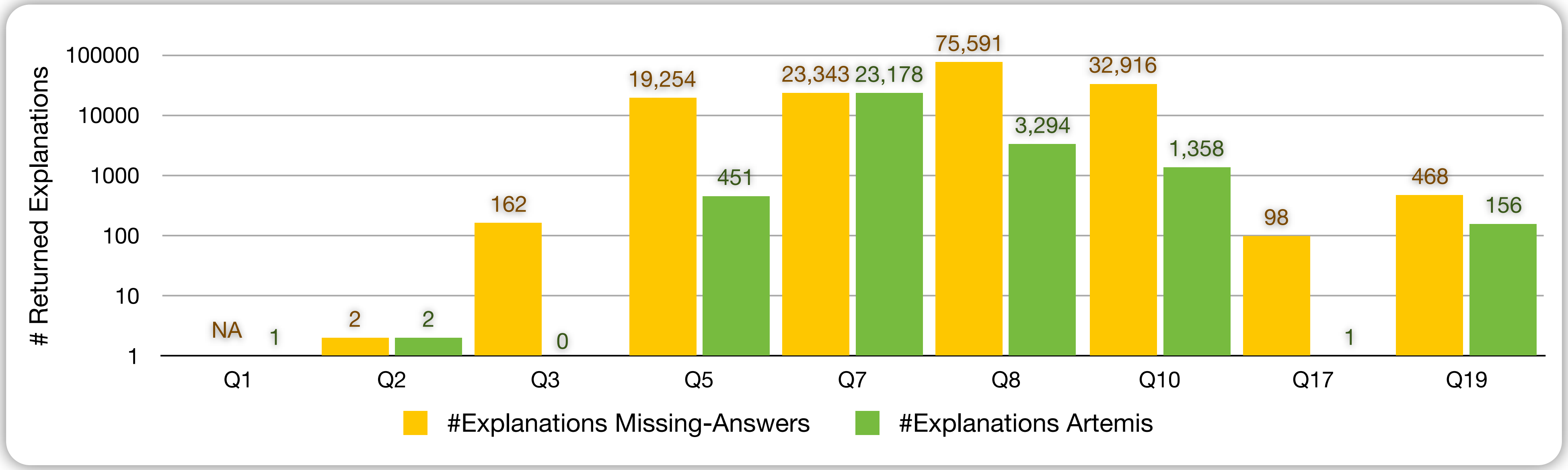


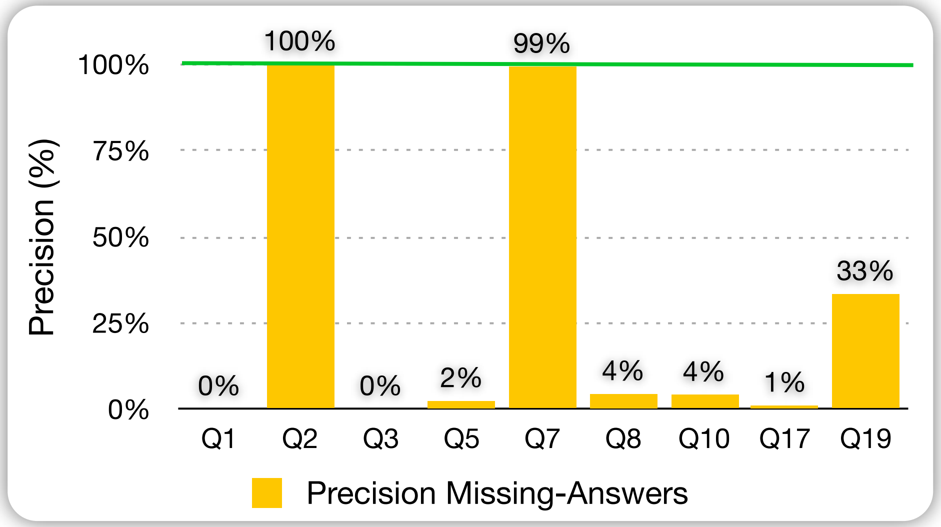
Runtime (ms)
100000
10000
1000
100
10
1
Q1
Q2
Q3
Q5
Q7
Q8
Q10
Q17
Q19
🟨 First explanation Missing-Answers
🟩 First correct explanation Artemis

Artemis takes less than a second to find first correct explanation in most cases.

Missing-Answers usually faster, but returned explanation can be wrong.

Effectiveness





Number of unsatisfiable explanations can be substantial when using Missing-Answers.

Constraint solver makes Artemis run slower, but effectiveness significantly improved.

Agenda

Nautilus

Artemis Algorithm

Ongoing work

The Conseil Algorithm

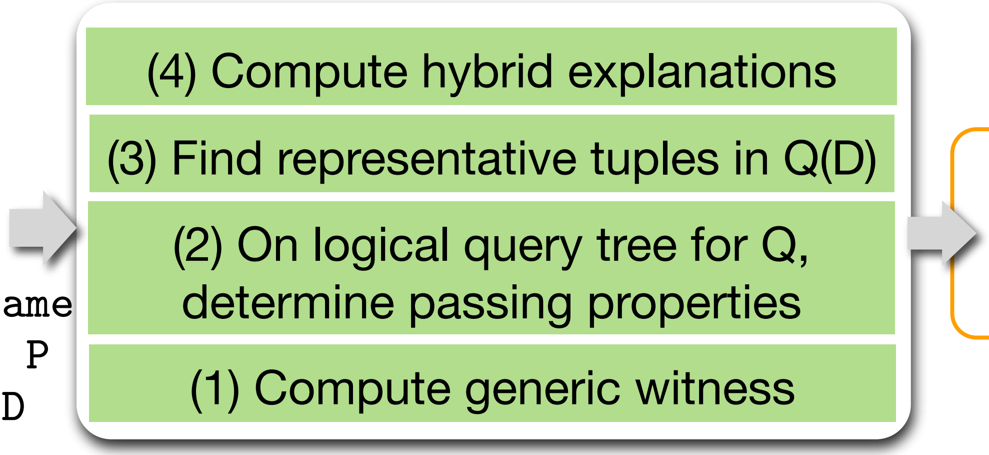
Instance-based + query-based = hybrid

- **Hybrid explanations** Combines advantages of instance-based and query-based explanations.

- Hybrid explanations generated by **Conseil** algorithm for non-monotonous queries.

ProdID	Name
P4	Yellow Submarine
P3	Green Bus
P1	Yellow Submarine

```
SELECT P.ProdID, P.ProdName
FROM Ratings R, Products P
WHERE R.ProdID = P.ProdID
AND P.Location = ‘DE’
GROUP BY P.ProdID, P.ProdName
HAVING MAX(R.Rating) <= 2
```



(4) Compute hybrid explanations

(3) Find representative tuples in Q(D)

(2) On logical query tree for Q, determine passing properties

(1) Compute generic witness

A hybrid exp.
(with D available)
Insert *Ratings*(P1, 1)
$\sigma_{P.Location\ =\ 'DE'}$ fails

The Conseil Algorithm

Instance-based + query-based = hybrid

1. **Generic witness computation:** extension to non-monotonous queries.

2. For each node in canonical query tree, determine if the nodes is **passing** (data is guaranteed to “survive” operator), **blocking** (the data is guaranteed not to make it through the operator), and **ambiguous** (all other cases).

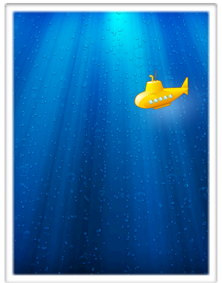
3. Find tuples in *Q(D)* similar to missing tuple *t* (used as positive examples).

4. Create passing canonical tree of similar tuple and **transform** blocking tree of t into representative passing tree (tree edit distance based). An **edit script** yields a hybrid explanation.

From Hybrid Explanations to Fixes

- Reuse tree with passing properties for missing tuple t from Conseil.
- Transform non-passing tree of t into a passing tree.
- Unlike for Conseil, we do not have a target tree, and tree transformations and costs are defined differently.
- Search problem determining the cheapest tree transformation, given a set of possible edit operations, that yields a passing tree.

Summary



Transformation Lifecycle Management with **Nautilus**

- **Analyse** query semantics using data provenance.
- **Fix** errors using automatically generated transformation modification suggestions.
- **Test** the modified transformation using information on modification impact.

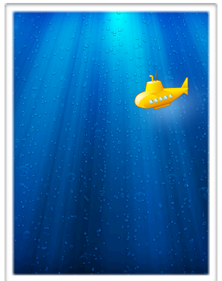
Explanation of missing-answers using **Artemis**

- Generates instance-based explanations for SPJUA queries
- Considers side-effects, correctness, and minimality of result

Explanation of missing-answers using **Conseil**

- Unifies instance- and query-based explanations into hybrid explanations.
- Applies on non-monotonous queries.
- Determines an explanation based on matching passing tree of missing-answer with passing tree of a (similar) existing tuple.

Outlook



Explanation of missing-answers using **Conseil**

- Proper analysis of complexity.
- Definition of cost function used to match passing trees.
- Evaluation in terms of efficiency and effectiveness, compared to other algorithms.

Extensions to **Nautilus**

- Development of an algorithm suggesting fixes, based on **Conseil**
- Support of debugging questions other than “Why-Not”
- Theoretical and practical aspects of the test phase.
- Build a real system and evaluate it.

Thank you for your attention.

http://nautilus-system.org