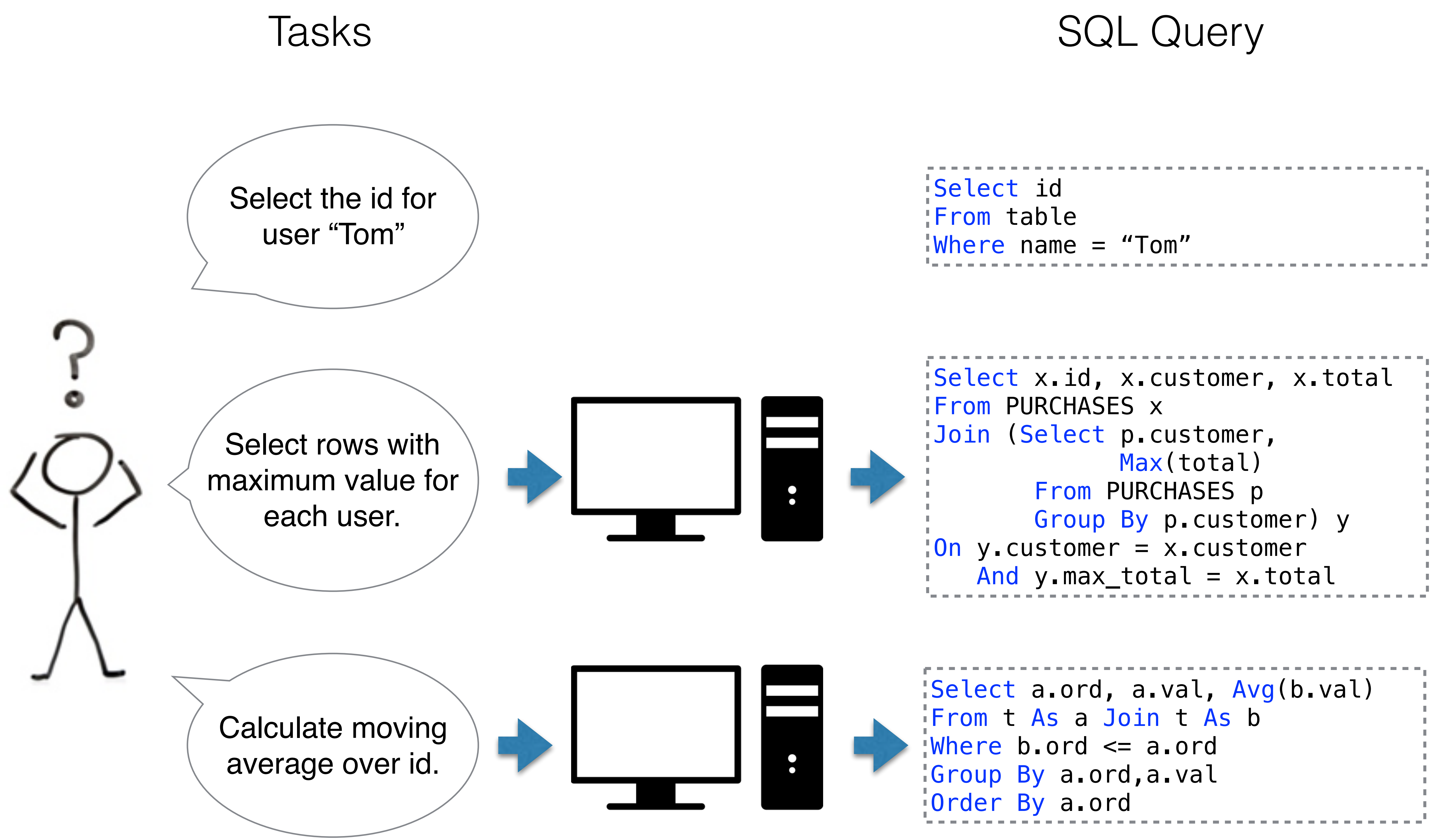
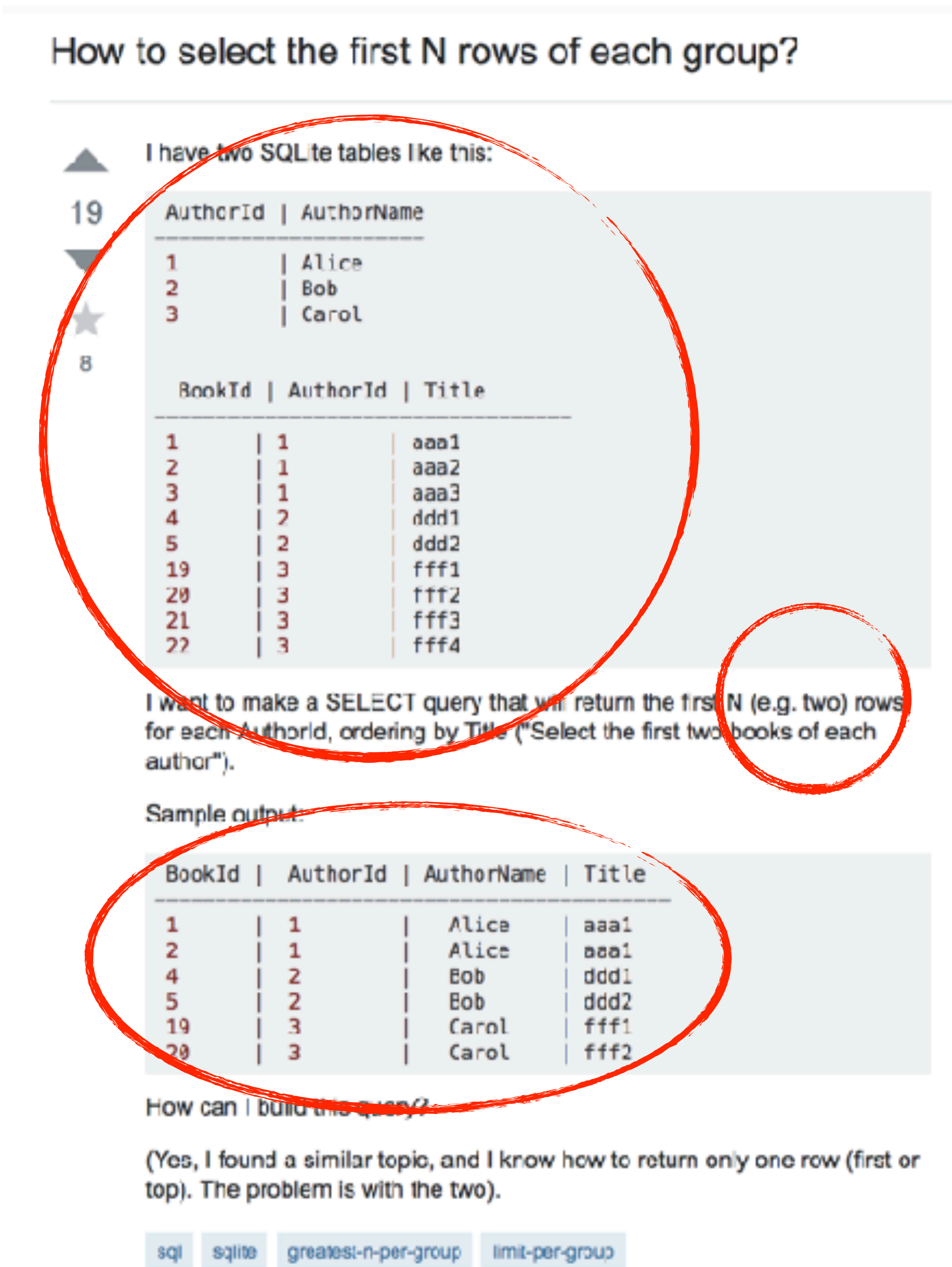
Synthesizing Highly Expressive SQL Queries From Input-Output Examples

http://scythe.cs.washington.edu

Chenglong Wang, Alvin Cheung, Ras Bodík
University of Washington



Problem: Advanced SQL operators make SQL powerful but hard to master.



How to select the first N rows of each group?

19

8

I have two SQLite tables like this:

```
AuthorId | AuthorName
----------------------
1        | Alice
2        | Bob
3        | Carol

 BookId | AuthorId | Title
----------------------------------
1       | 1        | aaa1
2       | 1        | aaa2
3       | 1        | aaa3
4       | 2        | ddd1
5       | 2        | ddd2
19      | 3        | fff1
20      | 3        | fff2
21      | 3        | fff3
22      | 3        | fff4
```

I want to make a SELECT query that will return the first N (e.g. two) rows for each AuthorId, ordering by Title ("Select the first two books of each author").

Sample output:

```
BookId |  AuthorId | AuthorName | Title
------------------------------------------
1      |  1        | Alice      | aaa1
2      |  1        | Alice      | aaa1
4      |  2        | Bob        | ddd1
5      |  2        | Bob        | ddd2
19     |  3        | Carol      | fff1
20     |  3        | Carol      | fff2
```

How can I build this query?

(Yes, I found a similar topic, and I know how to return only one row (first or top). The problem is with the two).

sql sqlite greatest-n-per-group limit-per-group

Synthesize queries from …?

Input Example

AuthorId	AuthorName
1	Alice
2	Bob
3	Carol

BookId	AuthorId	Title
1	1	aaa1
2	1	aaa2
3	1	aaa3
4	2	ddd1
5	2	ddd2
19	3	fff1
20	3	fff2
21	3	fff3
22	3	fff4

Output Example

BookId	AuthorId	AuthorName	Title
1	1	Alice	aaa1
2	1	Alice	aaa2
4	2	Bob	ddd1
5	2	Bob	ddd2
19	3	Carol	fff1
20	3	Carol	fff2

Constants

{2}

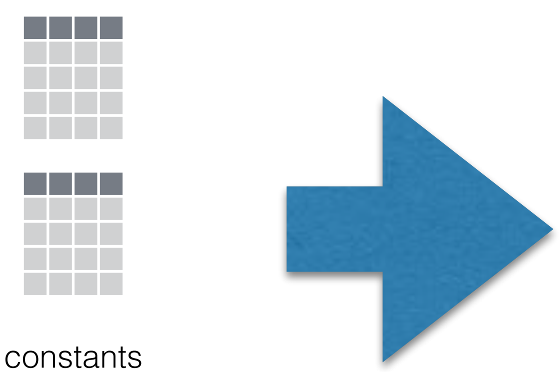
Aggregation Functions (Optional)

{ Count, Max, Min, Sum, Avg … }

Key: The synthesizer takes inputs that users can provide online.

Talk Outline

- Motivation & Problem Definition
- Synthesis Algorithm



constants

aggregation functions

```sql
Select b.BookId, a.AuthorId,
       a.AuthorName, b.Title
From Author a Join
     Book b
     On a.AuthorId = b.AuthorId
Where (Select count(*)
       From book b2
       Where b2.bookId <= b.BookId
       And b2.AuthorId = b.AuthorId
      ) <= 2;
```

- Evaluation on Stack Overflow Posts

Running Example

Task: Collect the max vals below `50` for all `oid` groups in `T2` and join them with `T1`.

T1

id	date	uid
1	12/25	1
2	11/21	3
4	12/24	2

T2

oid	val
1	30
1	10
1	10
2	50
2	10

Out

oid	date	uid	oid	MaxVal
1	12/25	1	1	30
4	12/24	2	2	10

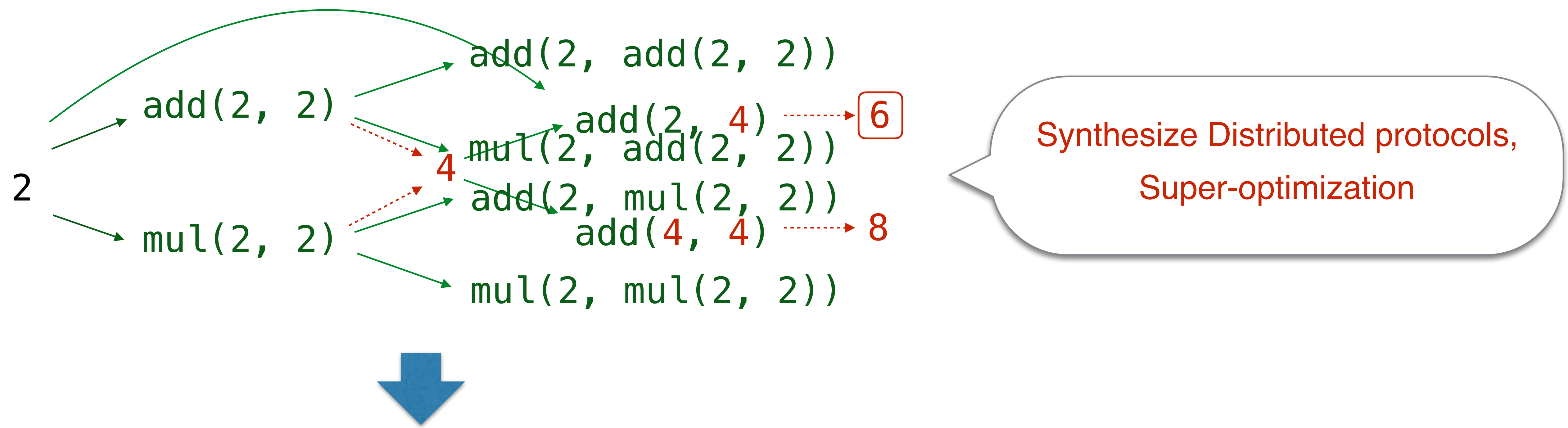
```
Select *
From (Select oid, Max(val)
      From  T2
      Where val < 50
      Group By oid) T3
Join T1
On T3.oid = T1.uid
```

Constants = { 50 }

AggrFunc = { Max, Min }

Basic Algorithm: Enumerative Search

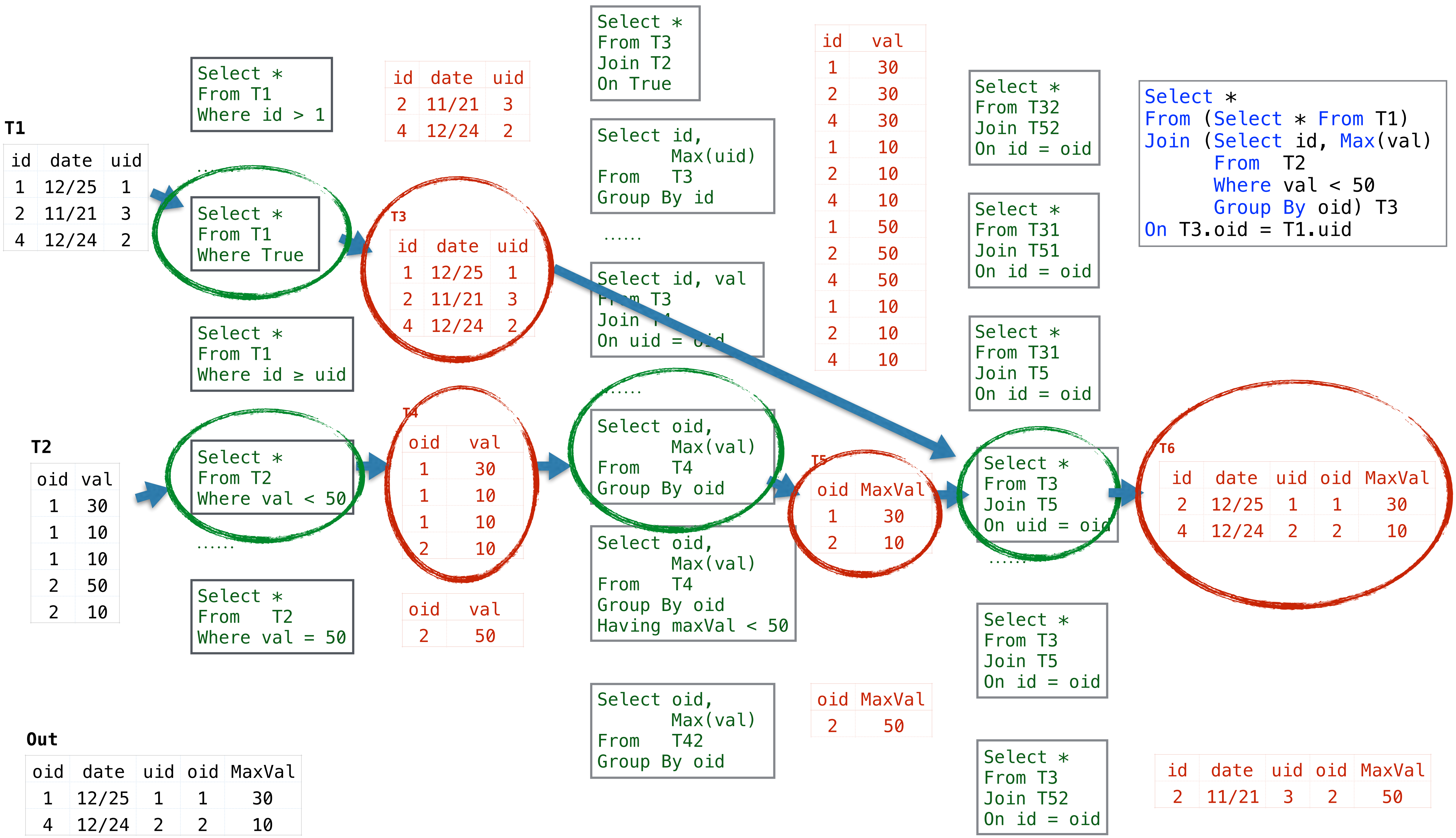
Input: 2 **Output:** 6 **Operators:** `add, mul`



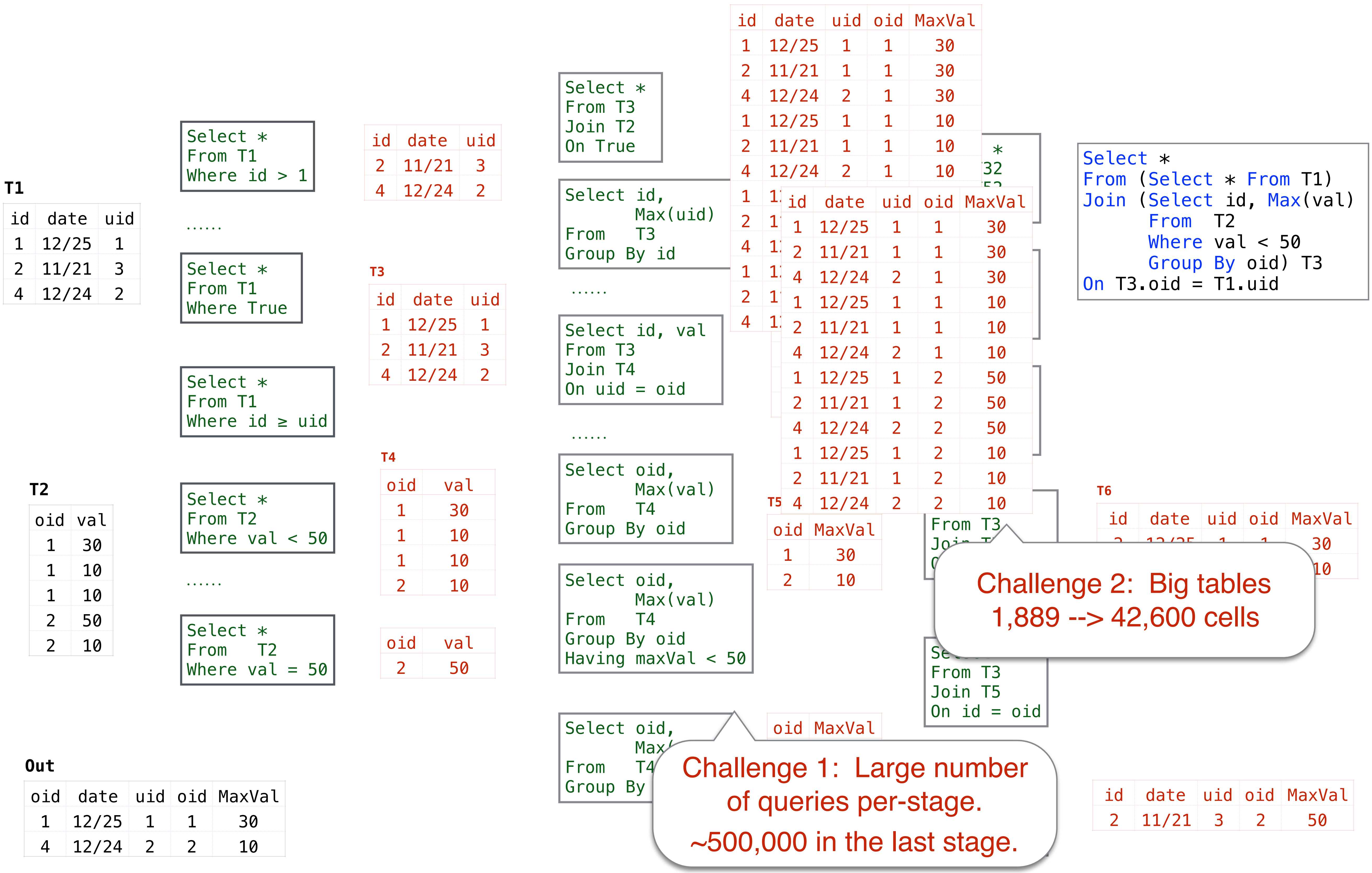

```
add(2, add(2, 2)) = 6
add(2, mul(2, 2)) = 6
```

Key: Compressing the search space by memoizing values.

Input: T1, T2 **Output:** T_{out} **Operators:** Select, Join, Aggr



Input: T1, T2 **Output:** T_{out} **Operators:** Select, Join, Aggr



Problem: Value-based compression is inefficient & ineffective.

Insight: Decompose Search Process

T1

id	date	uid
1	12/25	1
2	11/21	3
4	12/24	2

T2

oid	val
1	30
1	10
1	10
2	50
2	10

Tout

oid	date	uid	oid	MaxVal
2	12/25	1	1	30
4	12/24	2	2	10

Search SQL queries

```
Select *
From (Select *
      From T1
      Where True)
Join (Select id,
             Max(val)
      From  T2
      Where val < 50
      Group By oid
      Having True) T3
On T3.oid = T1.uid
```

Insight: Decompose Search Process With Abstract Queries

Queries whose predicates are holes.

T1

id	date	uid
1	12/25	1
2	11/21	3
4	12/24	2

T2

oid	val
1	30
1	10
1	10
2	50
2	10

Search abstract SQL queries

```
Select *
From    (Select *
         From T1
         Where □)
Join    (Select id,
                Max(val)
         From  T2
         Where □
         Group By oid
         Having □) T3
On      □
```

Instantiate Abstract Queries

```
Select *
From (Select *
      From T1
      Where True)
Join (Select id,
             Max(val)
      From  T2
      Where val < 50
      Group By oid
      Having True) T3
On T3.oid = T1.uid
```

Tout

oid	date	uid	oid	MaxVal
2	12/25	1	1	30
4	12/24	2	2	10

1. Prune query families

If a skeleton cannot be instantiated to return output, prune all queries with the skeleton

2. Speed up predicate synthesis

Insight: Decompose Search Process With Abstract Queries

T1

id	date	uid
1	12/25	1
2	11/21	3
4	12/24	2

T2

oid	val
1	30
1	10
1	10
2	50
2	10

Search abstract SQL queries

```
Select *
From   (Select *
        From T1
        Where □)
Join   (Select id,
               Max(val)
        From  T2
        Where □
        Group By oid
        Having □) T3
On      □
```

Instantiate Abstract Queries

```
Select *
From (Select *
      From T1
      Where True)
Join (Select id,
             Max(val)
      From  T2
      Where val < 50
      Group By oid
      Having True) T3
On T3.oid = T1.uid
```

Tout

oid	date	uid	oid	MaxVal
2	12/25	1	1	30
4	12/24	2	2	10

1. Prune query families

If a skeleton cannot be instantiated to return output, prune all queries with the skeleton

How?

Evaluating Abstract Queries with Over-Approximation

Inductively defined over abstract SQL operators

T1

id	date	uid
1	12/25	1
2	11/21	3
4	12/24	2

```
Select id, date
From T1
Where □
```

⇒

Summary

id	date
1	12/25
2	11/21
4	12/24

⊇

```
Select id, date
From T1
Where id <= 2
```

⇒

id	date	uid
1	12/25	1
2	11/21	3

T2

oid	MaxVal
1	30
2	10

```
T1 Join T2
On □
```

⇒

Summary2

id	date	uid	oid	MaxVal
1	12/25	1	1	30
2	11/21	1	1	30
4	12/24	2	1	30
1	12/25	1	1	10
2	11/21	1	1	10
4	12/24	2	1	10

Key: Evaluating abstract queries into over-approximations of concrete query results.

oid	date	uid	oid	MaxVal
1	12/25	1	1	30
4	12/24	2	2	10

Pruning with Abstract Queries

Input: T1, T2, **Output:** T_{out}, **Operators:** Select, Aggr, Join

```
Select *
From   (Select *
        From T1
        Where □)
Join   (Select id,
               Max(val)
        From  T2
        Where □
        Group By oid
        Having □) T3
On      □
```

T1

id	date	uid
1	12/25	1
2	11/21	3
4	12/24	2

T2

oid	val
1	30
1	10
1	10
2	50
2	10

```
Select *
From   T1
Where  □
```

```
Select *
From   T2
Where  □
```

T3

id	date	uid
1	12/25	1
2	11/21	3
4	12/24	2

T4

oid	uid
1	30
1	10
1	10
2	50
2	10

```
Select id,
   Max(date)
From T3
Group By id
Having □
```

```
Select id,
   Max(uid)
From T3
Group By id
Having □
```

```
Select oid,
   Max(val)
From   T4
Group By oid
Having □
```

id	date
1	12/25
2	11/21
4	12/24

id	uid
1	1
2	3
4	2

T5

oid	MaxVal
1	30
1	10
2	50
2	10

```
Select *
From T1
Join T3
On   □
```

```
Select *
From T2
Join T4
On   □
```

```
Select *
From T3
Join T4
On   □
```

```
Select *
From T3
Join T5
On   □
```

T6

id	date	uid	oid	MaxVal
1	12/25	1	1	30
2	11/21	3	1	30
4	12/24	2	1	30
1	12/25	1	1	10
2	11/21	3	1	10
4	12/24	2	1	10
1	12/25	1	2	50
2	11/21	3	2	50
4	12/24	2	2	50
1	12/25	1	2	10
2	11/21	3	2	10
4	12/24	2	2	10

✓

$\cup|$

oid	date	uid	oid	MaxVal
1	12/25	1	1	30
4	12/24	2	2	10

Out

oid	date	uid	oid	MaxVal
1	12/25	1	1	30
4	12/24	2	2	10

On average, number of tables generated is **7×** less v.s. concrete case.

Search with Abstract Queries

T1

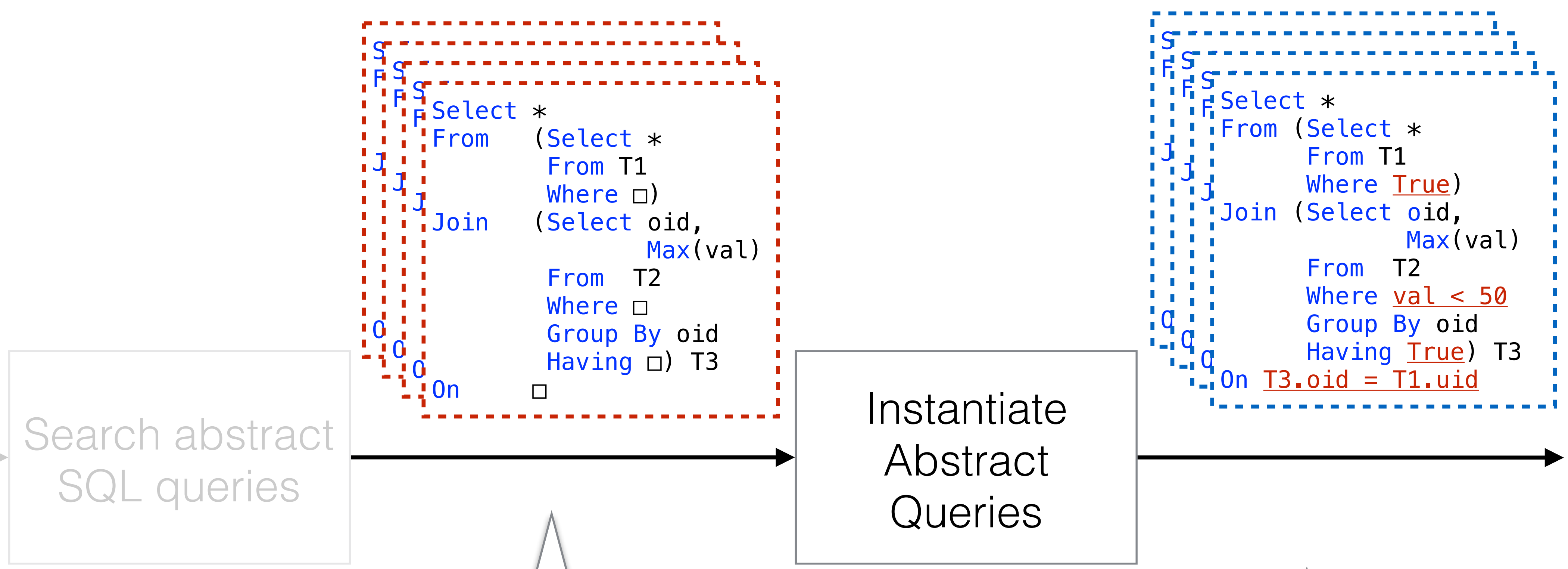
id	date	uid
1	12/25	1
2	11/21	3
4	12/24	2

T2

oid	val
1	30
1	10
1	10
2	50
2	10

Tout

oid	date	uid	oid	MaxVal
2	12/25	1	1	30
4	12/24	2	2	10



1. Prune query families

In average over 90% of queries skeletons are pruned.

2. Speed up predicate synthesis

Predicate Search Space

```
Select *
From   (Select *
        From T1
        Where □)
Join   (Select oid, Max(val)
        From  T2
        Where □
        Group By oid
        Having □) T3
On     □
```

```
True,
False,
uid < id,
……
uid < id And uid > val
```

~1000

```
True,
False
val < 50,
val > id,
……
```

~500

```
True,
False,
T1.uid = T3.oid,
T1.id = T3.oid,……
```

~1000

```
True,
False,
MaxVal < 50,
……
```

~500

Challenge: Large number of predicate combinations to search.

Enumerative Predicate Synthesis

```
Select *
From   (Select *
        From T1
        Where □) As T3
Join   (Select id, Max(val)
        From  T2
        Where □
        Group By oid
        Having □) T5
On     □
```

T1

id	date	uid
1	12/25	1
2	11/21	3
4	12/24	2

```
Select *
From   T1
Where  True
```
```
Where  id<uid
```
```
Where  id=uid
```

T3

id	date	uid
1	12/25	1
2	11/21	3
4	12/24	2

```
Select *
From   T1
Where  □
```

```
Select *
From   T2
Where  □
```

```
Select oid,
       MAX(val)
From   T4
Group By oid
Having □
```

```
Select *
From T3
Join T5
On   □
```

T6

oid	date	uid	oid	MaxVal
1	12/25	1	1	30
4	12/24	2	2	10

T2

oid	val
1	30
1	10
1	10
2	50
2	10

```
Select *
From   T2
Where  val < 50
```
```
Where  val ≤ 50
```
```
Where  oid < val
```
```
Where  True
```

T4

oid	val
1	30
1	10
1	10
2	10

```
Select oid,
       MAX(val)
From   T4
Group By oid
Having True
```
```
Having oid < mVal
```
```
Having mVal < 50
```

T5

oid	val
1	30
2	10

```
Select *
From T3
Join T5
On uid = oid
```
```
On id < oid
```
```
On True
```

Out

oid	date	uid	oid	MaxVal
1	12/25	1	1	30
4	12/24	2	2	10

Enumerative Predicate Synthesis

```
Select *
From   (Select *
        From T1
        Where □) As T3
Join   (Select id, Max(val)
        From  T2
        Where □
        Group By oid
        Having □) T5
On     □
```

T1

id	date	uid
1	12/25	1
2	11/21	3
4	12/24	2

```
Select *
From   T1
Where  True
Where  id<uid
Where  id=uid
```

T3

id	date	uid
1	12/25	1
2	11/21	3
4	12/24	2

```
Select *
From   T1
Where  □
```

```
Select *
From   T2
Where  □
```

```
Select oid,
       MAX(val)
From   T4
Group By oid
Having □
```

```
Select *
From T3
Join T5
On   □
```

T2

oid	val
1	30
1	10
1	10
2	50
2	10

```
Select *
From   T2
Where  val < 50
Where  val ≤ 50
Where  oid < val
Where  True
```

T4

oid	val
1	30
1	10
1	10
2	10
2	10

```
Select oid,
       MAX(val)
From   T4
Group By oid
Having True
Having oid < mVal
Having mVal < 50
```

T5

oid	val
1	30
2	10
2	10
2	10
2	10
2	10

```
Select *
From T3
Join T5
On uid = oid
On id < oid
On True
```

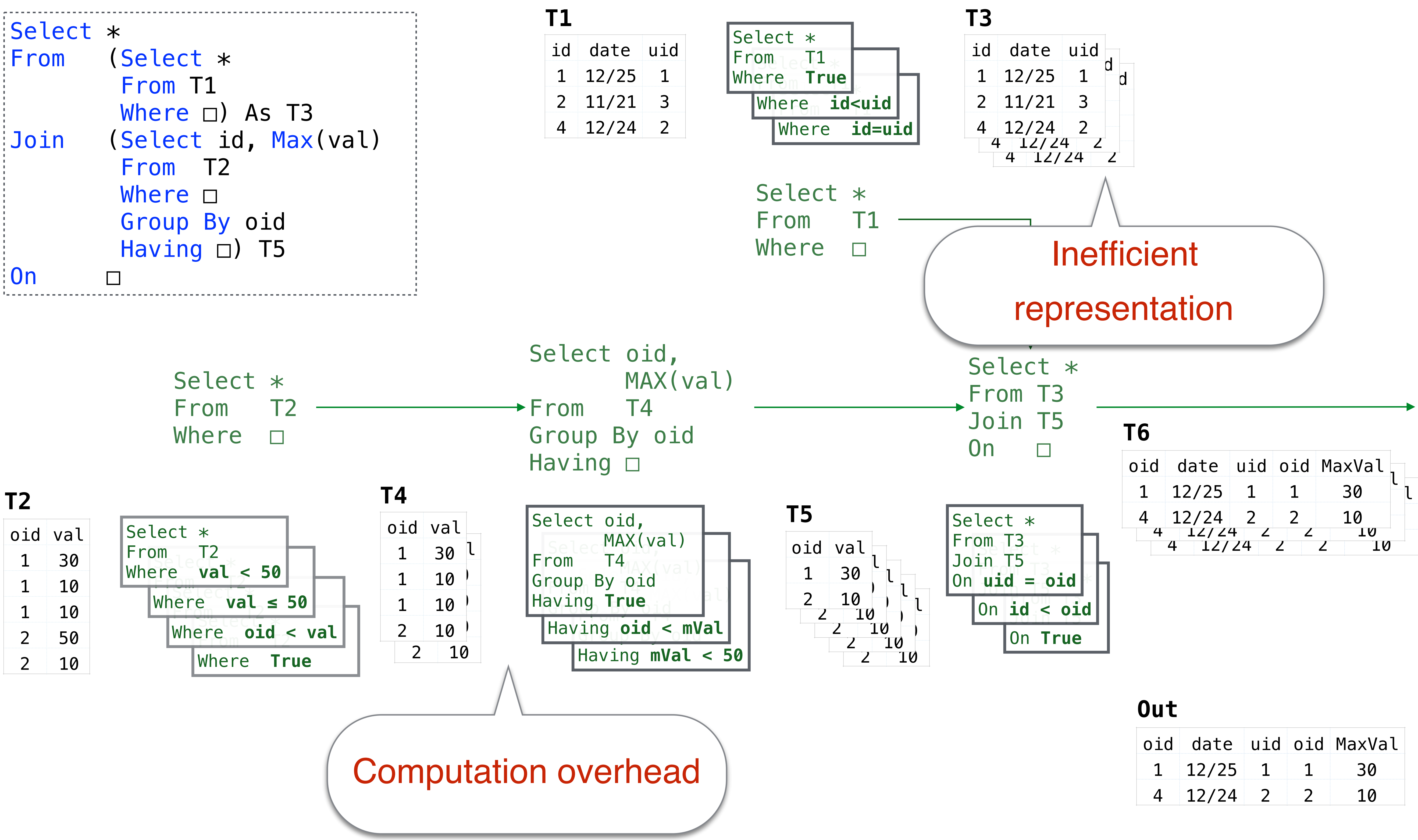
T6

oid	date	uid	oid	MaxVal
1	12/25	1	1	30
4	12/24	2	2	10
4	12/24	2	2	10
4	12/24	2	2	10

Out

oid	date	uid	oid	MaxVal
1	12/25	1	1	30
4	12/24	2	2	10

Enumerative Predicate Synthesis

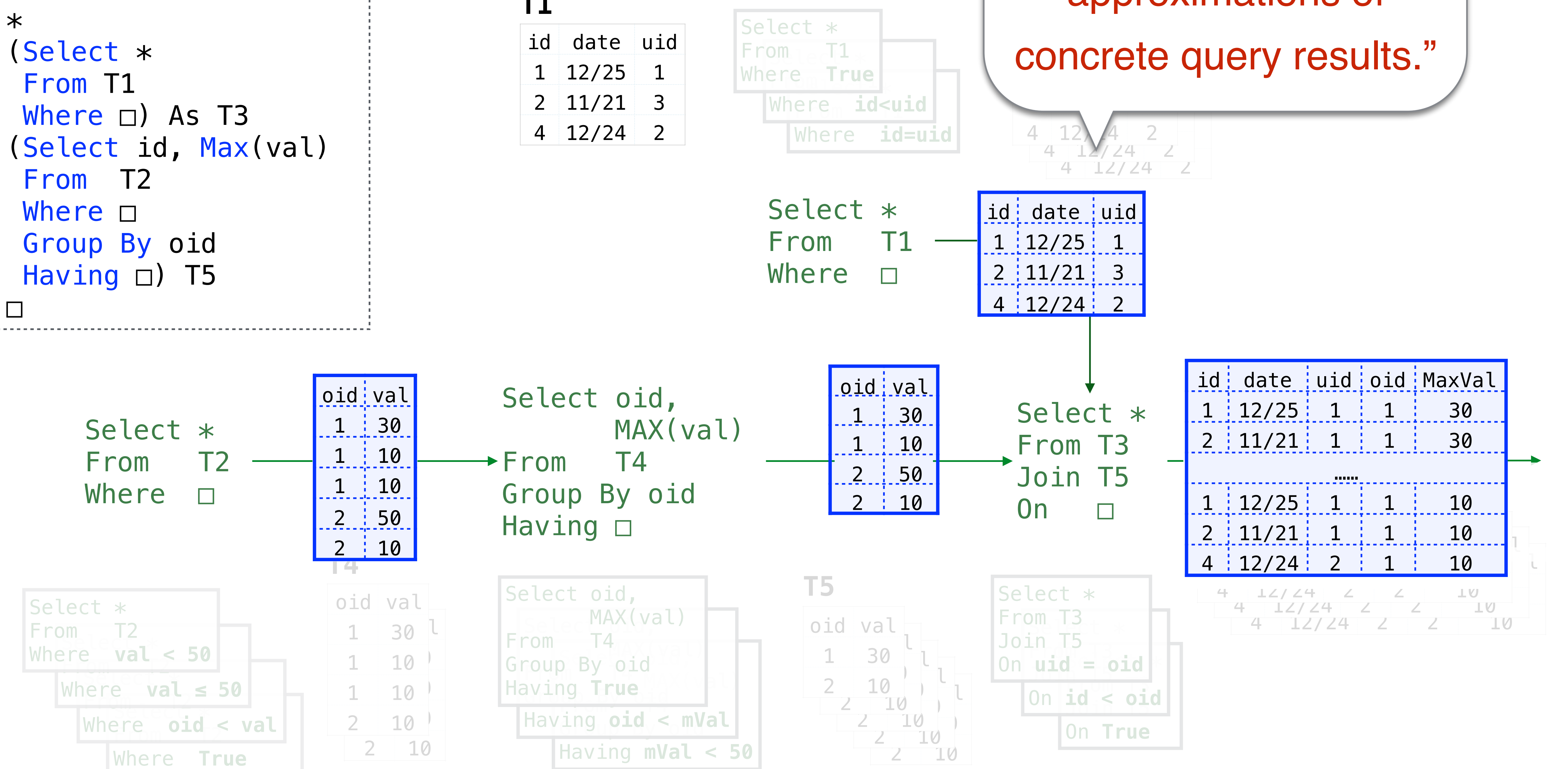


Enumerative Predicate Syn

```
Select *
From   (Select *
        From T1
        Where □) As T3
Join   (Select id, Max(val)
        From  T2
        Where □
        Group By oid
        Having □) T5
On     □
```

T1

id	date	uid
1	12/25	1
2	11/21	3
4	12/24	2



T2

oid	val
1	30
1	10
1	10
2	50
2	10

Out

oid	date	uid	oid	MaxVal
1	12/25	1	1	30
4	12/24	2	2	10

Enumerative Predicate Synthesis

```
Select *
From    (Select *
         From T1
         Where □) As T3
Join    (Select id, Max(val)
         From  T2
         Where □
         Group By oid
         Having □) T5
On      □
```

T1

id	date	uid
1	12/25	1
2	11/21	3
4	12/24	2

```
Select *
From   T1
Where  True
```
```
Where  id<uid
```
```
Where  id=uid
```

T3

id	date	uid
1	12/25	1
2	11/21	3
4	12/24	2

```
Select *
From   T1
Where  □
```

```
Select *
From   T2
Where  □
```

```
Select oid,
       MAX(val)
From   T4
Group By oid
Having □
```

```
Select *
From T3
Join T5
On   □
```

T6

oid	date	uid	oid	MaxVal
1	12/25	1	1	30
4	12/24	2	2	10

T2

oid	val
1	30
1	10
1	10
2	50
2	10

```
Select *
From   T2
Where  val < 50
```
```
Where  val ≤ 50
```
```
Where  oid < val
```
```
Where  True
```

T4

oid	val
1	30
1	10
1	10
2	10
2	10

```
Select oid,
       MAX(val)
From   T4
Group By oid
Having True
```
```
Having oid < mVal
```
```
Having mVal < 50
```

T5

oid	val
1	30
2	10

```
Select *
From T3
Join T5
On uid = oid
```
```
On id < oid
```
```
On True
```

Out

oid	date	uid	oid	MaxVal
1	12/25	1	1	30
4	12/24	2	2	10

Encoding Tables using Bit-vectors

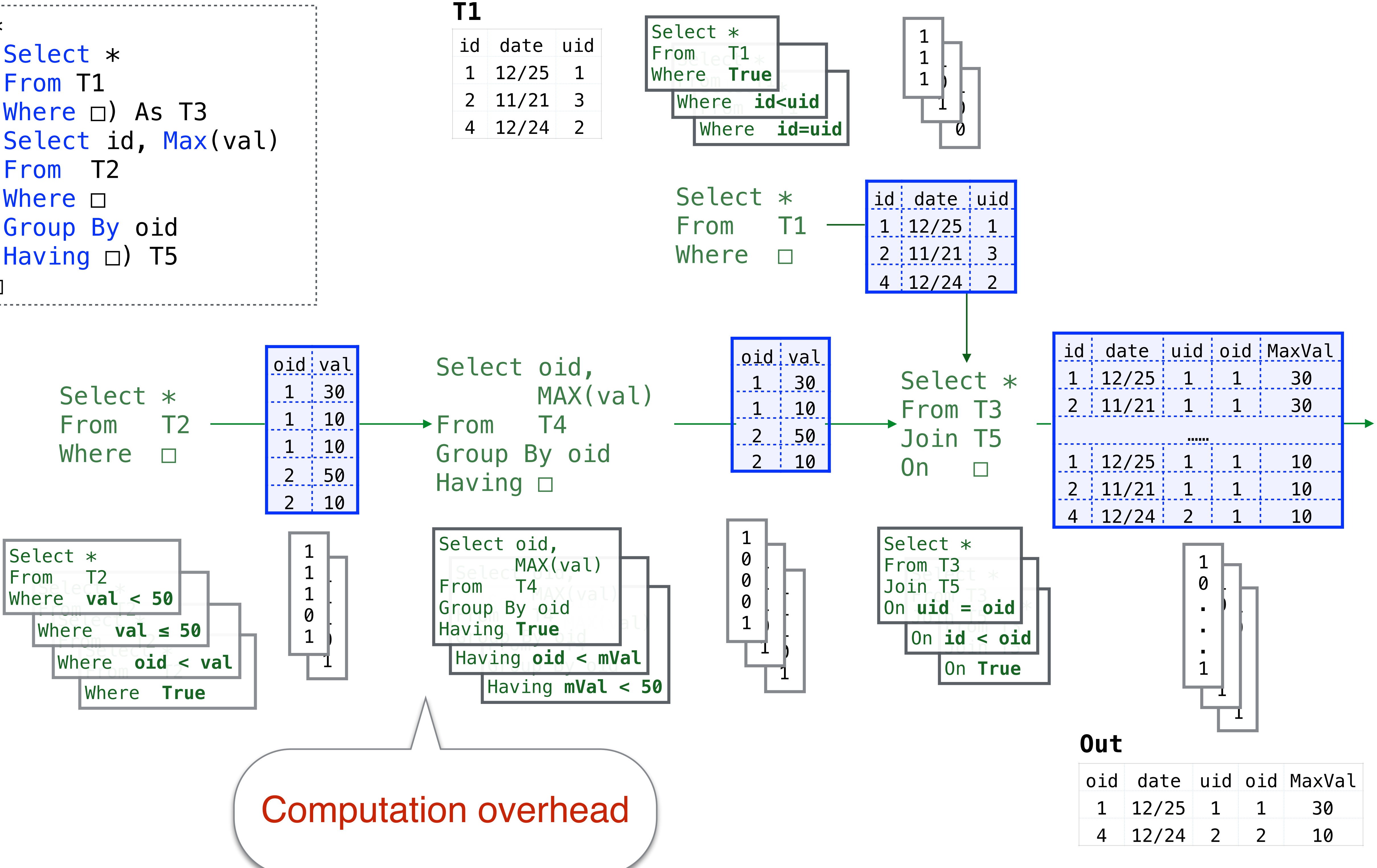
```
Select *
From   (Select *
        From T1
        Where □) As T3
Join   (Select id, Max(val)
        From  T2
        Where □
        Group By oid
        Having □) T5
On     □
```

T1

id	date	uid
1	12/25	1
2	11/21	3
4	12/24	2

T2

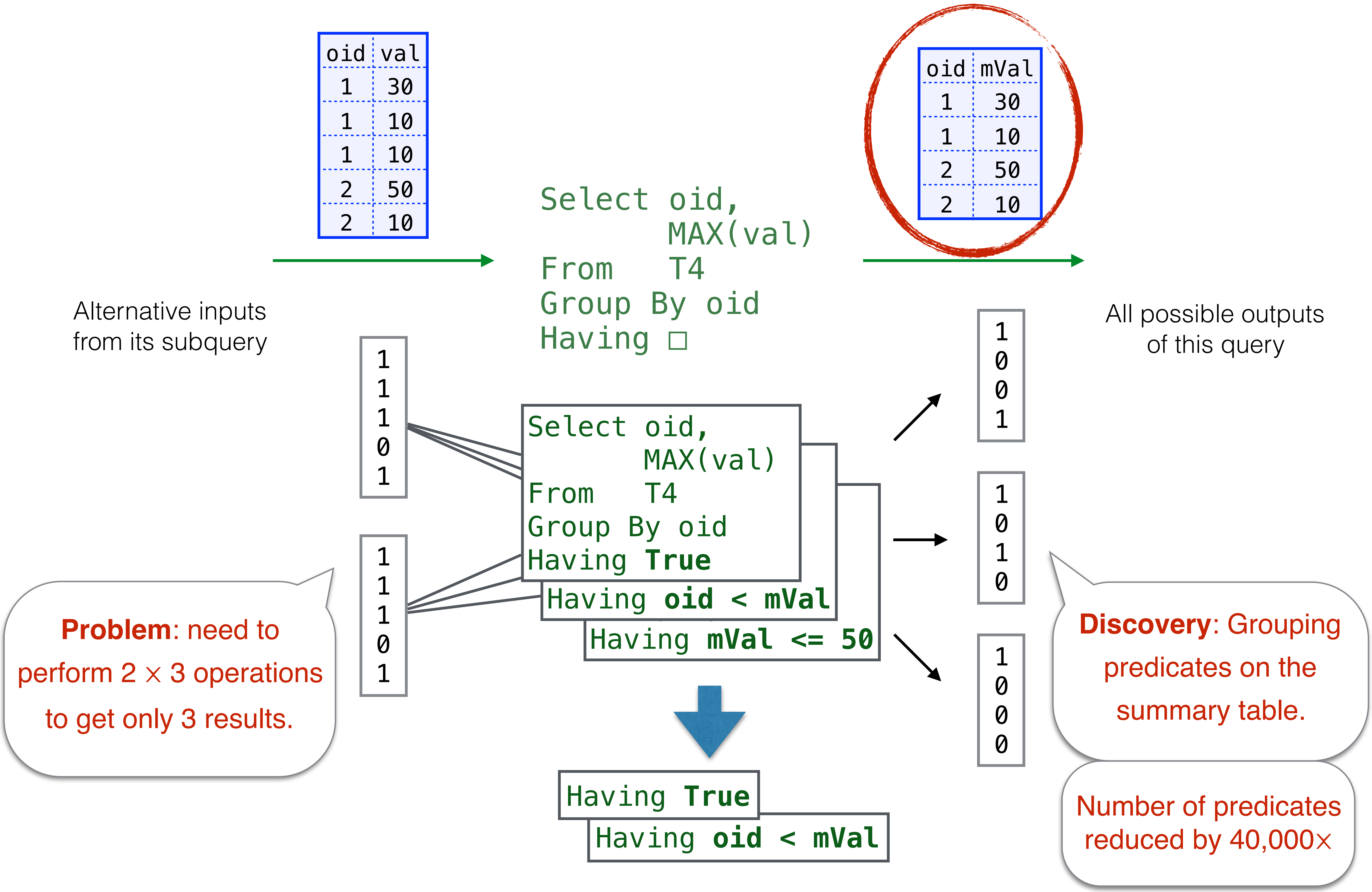
oid	val
1	30
1	10
1	10
2	50
2	10



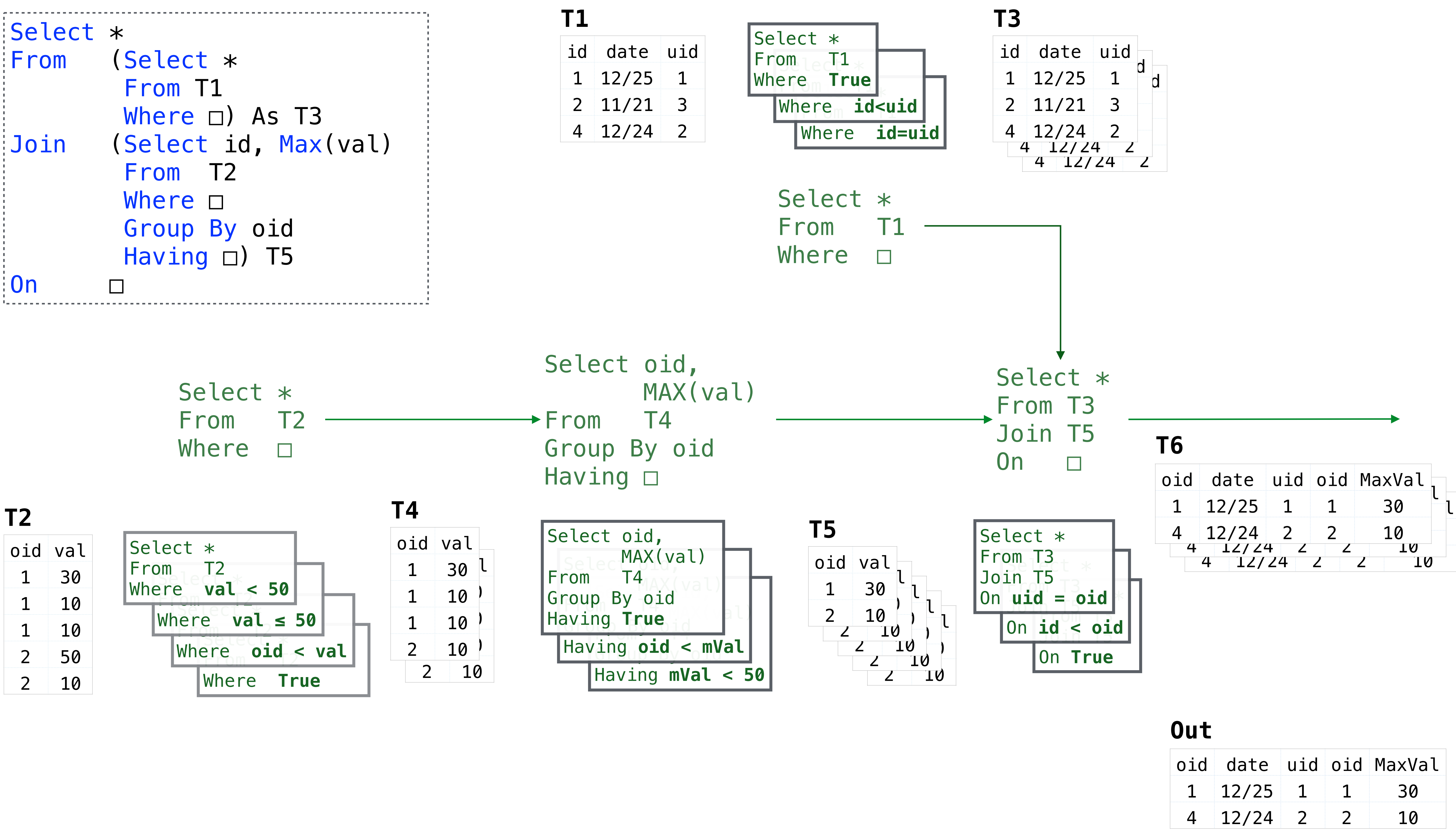
Out

oid	date	uid	oid	MaxVal
1	12/25	1	1	30
4	12/24	2	2	10

Optimize computation: Grouping Predicates



Enumerative Predicate Synthesis



Grouping Predicates + Bit-vector Representation

```
Select *
From    (Select *
         From T1
         Where □) As T3
Join    (Select id, Max(val)
         From  T2
         Where □
         Group By oid
         Having □) T5
On      □
```

T1

id	date	uid
1	12/25	1
2	11/21	3
4	12/24	2

```
Select *
From   T1
Where  True
```
```
Where  id<uid
```
```
Where  id=uid
```

1	.	.
1	)	.
1	1	)
		0

```
Select *
From   T1
Where  □
```

id	date	uid
1	12/25	1
2	11/21	3
4	12/24	2

T2

oid	val
1	30
1	10
1	10
2	50
2	10

```
Select *
From   T2
Where  val < 50
```
```
Where  True
```

```
Select *
From   T2
Where  □
```

oid	val
1	30
1	10
1	10
2	50
2	10

1	.
1	.
1	.
0	.
1	)
	1

```
Select oid,
       MAX(val)
From   T4
Group By oid
Having □
```

```
Select oid,
       MAX(val)
From   T4
Group By oid
Having True
```
```
Having mVal < 50
```

oid	val
1	30
1	10
2	50
2	10

1	.	.
0	.	.
0	.	.
0	)	.
1	1	)
		1

```
Select *
From T3
Join T5
On   □
```

```
Select *
From T3
Join T5
On uid = oid
```
```
On id < oid
```
```
On True
```

id	date	uid	oid	MaxVal
1	12/25	1	1	30
2	11/21	1	1	30
		……		
1	12/25	1	1	10
2	11/21	1	1	10
4	12/24	2	1	10

1	.	.
0	)	.
.	.	)
.	.	.
.	.	.
1	1	1

Out

oid	date	uid	oid	MaxVal
1	12/25	1	1	30
4	12/24	2	2	10

As a Programming-by-Example System

- **Synthesis process**
  - Iterating over the search depth for abstract queries
  - Instantiate abstract queries in the current depth and check results
- **Dealing with ambiguity**
  - Ranking programs by heuristic

    complexity, naturalness, constant coverage
  - Provide a new example / restrict aggregation functions.

Implementation — Scythe

http://scythe.cs.washington.edu

- Supported features:
  - Select, Join, Group By, Aggregation,
  - Subqueries, Outer Join, Exists, Union
- Unsupported
  - Arithmetics, Pivot, Window functions, Limit, Insert

Evaluation

- **Benchmarks from Stack Overflow:**
  - 57 used in development
  - 57 top-voted posts
  - 51 recent posts
- **Benchmarks from prior work:**
  [Zhang et al. ASE'13]
  - 23 textbook questions.
  - 5 forum posts.

In total 193 benchmarks.
Avg. Example Size: 34 cells

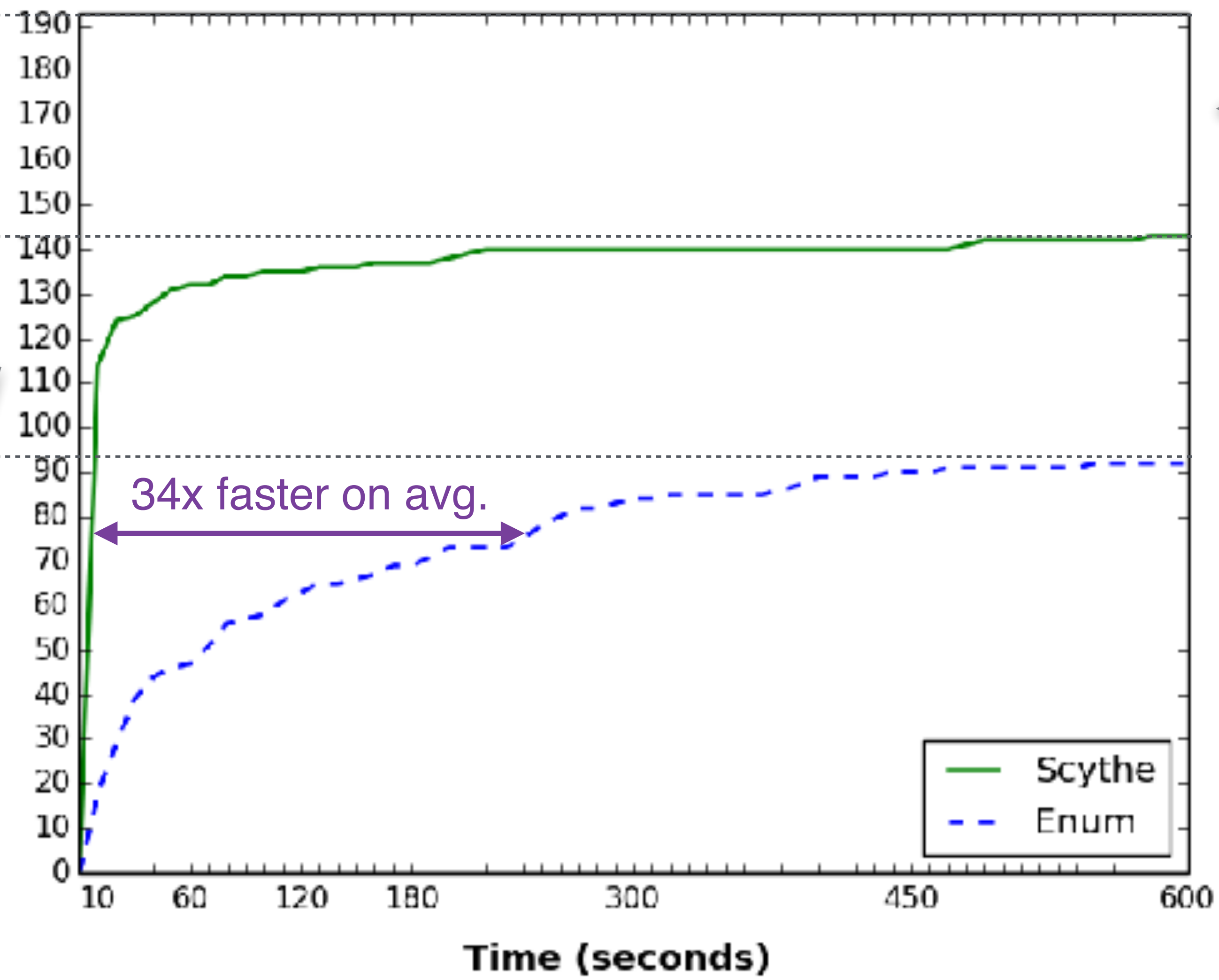
- **Algorithms**
  - Enumerative Search
    [Udupa et al. PLDI'13]
  - SqlSynthesizer
    (Decision tree algorithm)
    [Zhang et al. ASE'13]
  - Scythe
- **Evaluation Condition**
  - 4G memory, 600s timeout

Evaluation

Benchmark: 193

Scythe: 143

Enum: 92



59% can be answered within 10 seconds

Reasons for failures

34: missing features
15: timeout
1: failed to disambiguate

Comparing with SQLSynthesizer

Scythe:
18/28 in 120s
SQLSynthesizer:
15/28 in 120s

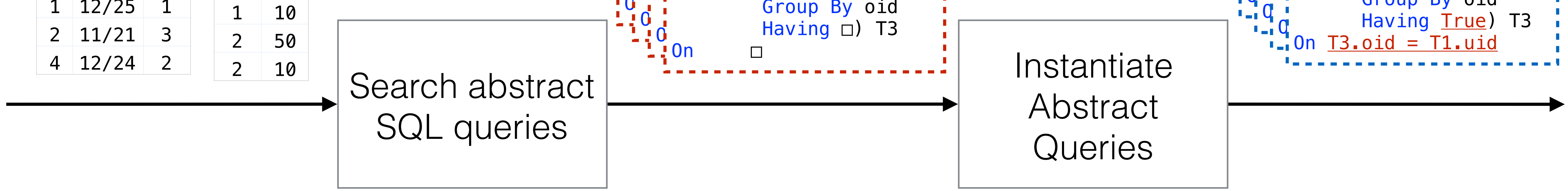
Some Related Work

- Enumerative search
  - Value-based Memoization [Udupa et al. PLDI'13]
- Search optimization with approximation
  - Synthesizing regex from examples [Lee et al. GPCE'16]
  - Monotonicity [Hu et al. PLDI'17]
- Synthesizing table manipulation programs
  - Pruning search space using partial programs [Feng et al. PLDI'17]

	Pruning Approach	Pruning Overhead	Pruning Power
Scythe	Over-approximation	Higher	Higher
Feng et al.	Constraint encoded properties	Lower	Lower

Benefit from value-based search space compression.

Algorithm: Decompose Search Process With Abstract Queries



T1

id	date	uid
1	12/25	1
2	11/21	3
4	12/24	2

T2

oid	val
1	30
1	10
1	10
2	50
2	10

```
Select *
From    (Select *
         From T1
         Where □)
Join    (Select id,
                 Max(val)
         From  T2
         Where □
         Group By oid
         Having □) T3
On      □
```

```
Select *
From (Select *
      From T1
      Where True)
Join (Select id,
             Max(val)
      From  T2
      Where val < 50
      Group By oid
      Having True) T3
On T3.oid = T1.uid
```

Tout

oid	date	uid	oid	MaxVal
2	12/25	1	1	30
4	12/24	2	2	10

Try demo on http://scythe.cs.washington.edu!