Bringing SQL to the Masses
with Program Synthesis

Chenglong Wang, Alvin Cheung, Ras Bodik
University of Washington
End-User

Select rows with maximum value for each group.

Find rows containing duplicate values.

Calculate the running total for a table.

---

**SQL**

```sql
SELECT x.id, x.customer, x.total
FROM PURCHASES x
JOIN (SELECT p.customer,
        MAX(total)
    FROM PURCHASES p
    GROUP BY p.customer) y
ON y.customer = x.customer
   AND y.max_total = x.total

SELECT *
FROM Users a
WHERE EXISTS
    (SELECT *
     FROM Users b
     WHERE (a.name = b.name
            OR a.email = b.email)
            AND a.ID <> b.id)

SELECT a.ord, a.total,
       SUM(b.total)
FROM t AS a JOIN t AS b
WHERE b.ord <= a.ord
GROUP BY a.ord, a.total
ORDER BY a.ord
```
Observations

A lot of **common tasks** require using **complex SQL constructs**.

- **greatest-n-per-group**
- **running-total**
- **duplicates**

Many tasks can be concisely expressed with **input-output examples**.

Idea: summarize our observation on StackOverflow

Transition: these problems can be concisely expressed with examples, can we build some system that allows users to ask question using examples only?
Join two tables and return the rows containing the maximum `val` below 50 for each group.

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

Idea: introduce what we want to do: build a PBE system.
Transition: let's first see what is the traditional algorithm to build such system.
Synthesis Algorithm: Value-directed Search

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

- add(2, add(2, 2)) = 6
- add(2, mul(2, 2)) = 6
- mul(2, add(2, 2)) = 6
- mul(2, mul(2, 2)) = 8

FlashFill

SuperOptimizer

```
cmp r1, #0
add r2, r1, #7
movge r2, r1
```

```
asm r3, r1, #2
add r2, r1, r3, lsr #29
```
Enumerative Search on SQL

**Input:** T1, T2, **Output:** Tout, **Operators:** Select, Join, Aggr

**Challenge 1:**
Large number of queries

**Challenge 2:**
Large tables
Insight: Decomposition

Search in the space of SQL queries

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

Abstract Queries

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

Pro: Smaller space of programs.
Challenge: which ones to search for?
Search with Abstract Queries

**Input:** T1, T2, **Output:** Tout, **Operators:** abstract query operators

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

**How to evaluate abstract queries?**
Instantiate Abstract Queries

<table>
<thead>
<tr>
<th>T1</th>
<th>id</th>
<th>date</th>
<th>uid</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>1</td>
<td>12/25</td>
<td>1</td>
</tr>
<tr>
<td>T2</td>
<td>2</td>
<td>11/21</td>
<td>3</td>
</tr>
<tr>
<td>T2</td>
<td>4</td>
<td>12/24</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tout</th>
<th>oid</th>
<th>date</th>
<th>uid</th>
<th>oid</th>
<th>MaxVal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tout</td>
<td>1</td>
<td>12/25</td>
<td>1</td>
<td>1</td>
<td>30</td>
</tr>
<tr>
<td>Tout</td>
<td>1</td>
<td>12/25</td>
<td>2</td>
<td>2</td>
<td>10</td>
</tr>
</tbody>
</table>

Abstract Queries

\[
\text{Select} \quad * \\
\text{From} \quad (\text{Select} \quad * \\
\quad \text{From} \quad T1 \\
\quad \text{Where} \quad \square) \\
\text{Join} \quad (\text{Select} \quad id, \text{Max}(val) \\
\quad \text{From} \quad T2 \\
\quad \text{Where} \quad \square \\
\quad \text{Group By} \quad oid \\
\quad \text{Having} \quad \square) \quad T3 \\
\text{On} \quad \square
\]

Instantiate abstract queries

\[
\text{Select} \quad * \\
\text{From} \quad (\text{Select} \quad * \\
\quad \text{From} \quad T1 \\
\quad \text{Where} \quad \square) \\
\text{Join} \quad (\text{Select} \quad id, \text{Max}(val) \\
\quad \text{From} \quad T2 \\
\quad \text{Where} \quad val < 50 \\
\quad \text{Group By} \quad oid \\
\quad \text{Having} \quad \square) \quad T3 \\
\text{On} \quad T3.oid = T1.uid
\]
Instantiante Abstract Queries

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

T1.uid = T3.oid, 
T1.id = T3.oid,

True, False, uid < id,
...

val < 50, val == 50, val > id,
True, ...

val < 50 + False + T1.uid = T3.oid
False + val < 50 + False + T1.uid = T3.oid
True + val == 50 + False + T1.uid = T3.oid
True + val == 50 + MaxVal < 50 + T1.uid = T3.oid

......

A intuitive solution that does no scale.
Transition: can we use properties of the abstract query to optimize this?
Instantiate Abstract Queries
Generating Solutions

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

T1.uid = T3.oid
T1.uid = T3.oid And T1.id <> T2.id

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

......

Select *
From (Select * From T1 Where id <> uid)
Join (Select id, Max(val)
    From T2
    Where val <> 50
    Group By oid) T3
On T1.uid = T3.oid And T1.id <> T2.id
Ranking & Interaction

• Heuristically rank candidate queries.
  • Criteria: complexity, naturalness etc.

• When the result is not desirable:
  • Provide new input-output examples.
Evaluation

Benchmark: 193

Scythe: 143

Enum: 92

34: more features
15: run out of time
1: fail to disambiguate
Demo
Conclusion

• Goal: Helping end users to program SQL with input-output examples.

• Solution: An efficient two-phase synthesis algorithm.

• Evaluation: Able to solve 143/193 problems on StackOverflow.