SQL 3: Nesting in Where and Having

Yufei Tao

Department of Computer Science and Engineering
Chinese University of Hong Kong
SQL offers several constructs beyond relational algebra to allow users to write more powerful queries. In the previous lecture, we have studied a collection of constructs for statistical analysis. This lecture will introduce another collection for enhancing the functionality of WHERE and HAVING.

We will first discuss WHERE and then HAVING.
Recall that the where clause contains conditions of the form $A \ op \ v$
where $A$ is an attribute/value, $op$ is an arithmetic operator (e.g., $<$), and $v$ is another attribute/value.

Next, we will learn new conditions where

- $v$ is an SQL statement
- $op$ is an operator that compares a value to the result of $v$. 
Membership Test

\[(A_1, \ldots, A_n) \text{ in ([an SQL statement])}\]

where each \(A_i\) is an attribute/value. \((A_1, \ldots, A_n)\) must obey the schema of the table \(T\) returned by the SQL statement.

The expression returns:

- true, if tuple \((A_1, \ldots, A_n)\) appears in \(T\).
- false, otherwise.

The bracket of \("(A_1, \ldots, A_n)"\) can be omitted if \(n = 1\).
select pid from PROF
where dept in (select dept from PROF where sal >= 10000)

Result:

<table>
<thead>
<tr>
<th>pid</th>
<th>name</th>
<th>dept</th>
<th>rank</th>
<th>sal</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>Adam</td>
<td>CS</td>
<td>asst</td>
<td>6000</td>
</tr>
<tr>
<td>p2</td>
<td>Bob</td>
<td>EE</td>
<td>asso</td>
<td>8000</td>
</tr>
<tr>
<td>p3</td>
<td>Calvin</td>
<td>CS</td>
<td>full</td>
<td>10000</td>
</tr>
<tr>
<td>p4</td>
<td>Dorothy</td>
<td>EE</td>
<td>asst</td>
<td>5000</td>
</tr>
<tr>
<td>p5</td>
<td>Emily</td>
<td>EE</td>
<td>asso</td>
<td>8500</td>
</tr>
<tr>
<td>p6</td>
<td>Frank</td>
<td>CS</td>
<td>full</td>
<td>9000</td>
</tr>
</tbody>
</table>
select pid from PROF
where (dept, rank) in (select dept, rank from PROF where sal >= 10000)

Result:

<table>
<thead>
<tr>
<th>pid</th>
<th>name</th>
<th>dept</th>
<th>rank</th>
<th>sal</th>
</tr>
</thead>
<tbody>
<tr>
<td>p3</td>
<td>Calvin</td>
<td>CS</td>
<td>full</td>
<td>10000</td>
</tr>
<tr>
<td>p6</td>
<td>Frank</td>
<td>CS</td>
<td>full</td>
<td>9000</td>
</tr>
</tbody>
</table>
select pid from PROF
where (dept, sal) in (select dept, rank from PROF where sal >= 10000)

Error! (dept, sal) does not match the schema of the table returned by the nested SQL statement.
Membership Test

Not in

\((A_1, \ldots, A_n) \text{ not in } ([\text{an SQL statement}])\)

where each \(A_i\) is an attribute/value. \((A_1, \ldots, A_n)\) must obey the schema of the table \(T\) returned by the SQL statement.

The expression returns:

- true, if tuple \((A_1, \ldots, A_n)\) does not appear in \(T\).
- false, otherwise.

The bracket of “\((A_1, \ldots, A_n)\)” can be omitted if \(n = 1\).
select pid from PROF
where (dept, rank) not in
(select dept, rank from PROF where sal >= 10000)

Result:
Some

\[ A > \text{some ([an SQL statement])} \]

where \( A \) is an attribute/value, and must obey the schema of the table \( T \) returned by the SQL statement.

The expression returns:

- true, if \( A \) is greater than at least one tuple in \( T \).
- false, otherwise.
select pid from PROF
where sal > some (select sal from PROF where dept = 'CS')

Result:

<table>
<thead>
<tr>
<th>pid</th>
</tr>
</thead>
<tbody>
<tr>
<td>p2</td>
</tr>
<tr>
<td>p3</td>
</tr>
<tr>
<td>p5</td>
</tr>
<tr>
<td>p6</td>
</tr>
<tr>
<td>All</td>
</tr>
<tr>
<td>-----</td>
</tr>
</tbody>
</table>

\[ A > \text{all} \left([\text{an SQL statement}]\right) \]

where \( A \) is an attribute/value, and must obey the schema of the table \( T \) returned by the SQL statement.

The expression returns:

- true, if \( A \) is greater than all tuples in \( T \).
- false, otherwise.
select pid from PROF
where sal > all (select sal from PROF where dept = 'EE')

Result:

<table>
<thead>
<tr>
<th>pid</th>
<th>name</th>
<th>dept</th>
<th>rank</th>
<th>sal</th>
</tr>
</thead>
<tbody>
<tr>
<td>p3</td>
<td>Calvin</td>
<td>CS</td>
<td>full</td>
<td>10000</td>
</tr>
<tr>
<td>p6</td>
<td>Frank</td>
<td>CS</td>
<td>full</td>
<td>9000</td>
</tr>
</tbody>
</table>
The operator $>$ in “$>$ some (all)” can be replaced with $<$, $<=$, $=$, $<>$ and $>=$.

The semantics in each case agrees with the literal meaning in English. For example, “$<$ some” means “smaller than some element (in a table)”.

SQL 3: Nesting in Where and Having
Next, we will learn two more useful conditions of the form:

\[ \text{op ([an SQL statement])} \]
Emptiness Test

### Exists

exists ([an SQL statement])

The expression returns:
- true, if the SQL statement returns a table with at least one tuple.
- false, otherwise.

not exists ([an SQL statement])

The expression returns:
- true, if the SQL statement returns an empty table.
- false, otherwise.
select pid from PROF as P
where not exists
(select sal from PROF
where sal > P.sal)

Result:

<table>
<thead>
<tr>
<th>pid</th>
<th>name</th>
<th>dept</th>
<th>rank</th>
<th>sal</th>
</tr>
</thead>
<tbody>
<tr>
<td>p3</td>
<td>Calvin</td>
<td>CS</td>
<td>full</td>
<td>10000</td>
</tr>
</tbody>
</table>

Note
Observe how \( P \) is used in the nested query.
select distinct pid from T1 as P
where not exists ( 
  (select cid from T2)
  minus
  (select cid from T1 where pid = P.pid))

Result:

\[
\begin{array}{c|c}
\text{pid} & \text{cid} \\
\hline
\text{p1} & \text{c1} \\
\text{p1} & \text{c2} \\
\text{p1} & \text{c3} \\
\text{p2} & \text{c2} \\
\text{p2} & \text{c3} \\
\text{p3} & \text{c1} \\
\text{p4} & \text{c1} \\
\text{p4} & \text{c2} \\
\text{p4} & \text{c3} \\
\end{array}
\]

Note that this is another way to do division in SQL.
Next we extend the above discussion to HAVING.
select \( A_1, \ldots, A_t, \text{agg}_1(B_1), \ldots, \text{agg}_m(B_m) \)
from \( T_1, \ldots, T_n \)
where \( P \)
group by \( C_1, \ldots, C_g \)
having \( H \)

where

- \( C_1, \ldots, C_g \) are called group-by attributes.
- \( H \) is a group predicate.
The group predicate can contain conditions of the form:

\[ \text{agg}(A) \ op \ (\text{SQL statement}) \]

where

- \text{agg} is an aggregate function
- \text{op} can be
  - in, not in
  - \text{< some/all}
  - \text{\leq some/all}
  - \text{= some/all}
  - \text{<> some/all}
  - \text{> some/all}
  - \text{\geq some/all}

- The nested SQL statement must return a table of a single numeric column.
<table>
<thead>
<tr>
<th>pid</th>
<th>name</th>
<th>dept</th>
<th>rank</th>
<th>sal</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>Adam</td>
<td>CS</td>
<td>asst</td>
<td>6000</td>
</tr>
<tr>
<td>p2</td>
<td>Bob</td>
<td>EE</td>
<td>asso</td>
<td>8000</td>
</tr>
<tr>
<td>p3</td>
<td>Calvin</td>
<td>CS</td>
<td>full</td>
<td>10000</td>
</tr>
<tr>
<td>p4</td>
<td>Dorothy</td>
<td>EE</td>
<td>asst</td>
<td>5000</td>
</tr>
<tr>
<td>p5</td>
<td>Emily</td>
<td>EE</td>
<td>asso</td>
<td>8500</td>
</tr>
<tr>
<td>p6</td>
<td>Frank</td>
<td>CS</td>
<td>full</td>
<td>9000</td>
</tr>
</tbody>
</table>

```sql
select dept from PROF
group by dept
having avg(sal) >= some (select avg(sal) from PROF)
```

**Result:**

<table>
<thead>
<tr>
<th>dept</th>
</tr>
</thead>
<tbody>
<tr>
<td>cs</td>
</tr>
</tbody>
</table>
### PROF

<table>
<thead>
<tr>
<th>pid</th>
<th>name</th>
<th>dept</th>
<th>rank</th>
<th>sal</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>Adam</td>
<td>CS</td>
<td>asst</td>
<td>6000</td>
</tr>
<tr>
<td>p2</td>
<td>Bob</td>
<td>EE</td>
<td>asso</td>
<td>8000</td>
</tr>
<tr>
<td>p3</td>
<td>Calvin</td>
<td>CS</td>
<td>full</td>
<td>10000</td>
</tr>
<tr>
<td>p4</td>
<td>Dorothy</td>
<td>EE</td>
<td>asst</td>
<td>5000</td>
</tr>
<tr>
<td>p5</td>
<td>Emily</td>
<td>EE</td>
<td>asso</td>
<td>8500</td>
</tr>
<tr>
<td>p6</td>
<td>Frank</td>
<td>CS</td>
<td>full</td>
<td>9000</td>
</tr>
</tbody>
</table>

```sql
select dept from PROF
  group by dept
  having avg(sal) >= some (select sal from PROF)

Result:

<table>
<thead>
<tr>
<th>dept</th>
</tr>
</thead>
<tbody>
<tr>
<td>cs</td>
</tr>
<tr>
<td>ee</td>
</tr>
</tbody>
</table>
```