### Our Contributions

1. We introduce the Distributed Similarity Grouping (DSG) operator to efficiently identify similarity groups in big data sets.
2. DSG supports the identification of similarity groups where all the elements of a group are within a given threshold (\( \varepsilon \)) from each other.
3. DSG guarantees that each group is generated only once.
4. DSG can be used with any metric and supports many standard non-similarity-based grouping operators.

### The Problem

- Simple grouping operations are fast but are limited to similarity-based grouping. More sophisticated grouping techniques capture complex groups but often at a steep increase in execution time.
- Previous work introduced the Similarity Grouping (SG) operator which aims to have fast execution times and capture complex groups. SG, however, was proposed for single node relational database systems.

### Overall Algorithm

- Each input record is assigned to the partition whose pivot is closest to it.
- DSG and K-Means are expected to have the same output.
- DSG guarantees that each group is generated only once.

### References


### Experimental Results

#### Increasing Dataset Size

**Eps=100, Pivots: 40P**

<table>
<thead>
<tr>
<th>Part1</th>
<th>In partition Part1:</th>
<th>Then</th>
<th>Group in</th>
<th>Then</th>
<th>In partition Part1:</th>
<th>Then</th>
<th>In partition Part1:</th>
<th>Then</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group A in C and D</td>
<td>Ignore</td>
<td>Group in C and D</td>
<td>Ignore</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Group in B and D</td>
<td>Ignore</td>
<td>Group in B and D</td>
<td>Ignore</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Increasing Dataset Size and Cluster Size

**Eps=100, Pivots: 40P**

<table>
<thead>
<tr>
<th>Part1</th>
<th>In partition Part1:</th>
<th>Then</th>
<th>Group in</th>
<th>Then</th>
<th>In partition Part1:</th>
<th>Then</th>
<th>In partition Part1:</th>
<th>Then</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group A in C and D</td>
<td>Ignore</td>
<td>Group in C and D</td>
<td>Ignore</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Group in B and D</td>
<td>Ignore</td>
<td>Group in B and D</td>
<td>Ignore</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Increasing Dimensionality

**Eps=100, Pivots: 200P**

<table>
<thead>
<tr>
<th>Part1</th>
<th>In partition Part1:</th>
<th>Then</th>
<th>Group in</th>
<th>Then</th>
<th>In partition Part1:</th>
<th>Then</th>
<th>In partition Part1:</th>
<th>Then</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group A in C and D</td>
<td>Ignore</td>
<td>Group in C and D</td>
<td>Ignore</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Group in B and D</td>
<td>Ignore</td>
<td>Group in B and D</td>
<td>Ignore</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Increasing Number of Pivots and Memory Threshold

**Spark: Eps=100, 1000P**

<table>
<thead>
<tr>
<th>Part1</th>
<th>In partition Part1:</th>
<th>Then</th>
<th>Group in</th>
<th>Then</th>
<th>In partition Part1:</th>
<th>Then</th>
<th>In partition Part1:</th>
<th>Then</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group A in C and D</td>
<td>Ignore</td>
<td>Group in C and D</td>
<td>Ignore</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Group in B and D</td>
<td>Ignore</td>
<td>Group in B and D</td>
<td>Ignore</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>