Exploiting Similarity-aware Grouping in Decision Support Systems

ICDS Indiana Center for Database Systems

Yasin N. Silva, Muhammad U. Arshad, and Walid G. Aref
Purdue University

PURDUE UNIVERSITY

Similarity Group-by (SGB)

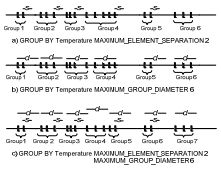
- SGB extends the standard grouping operator to group similar or approximate values
- The main goal of SGB is to generate more meaningful and useful similarity-based groupings than those of the regular group-by while maintaining:
  - Low running time
  - Good scalability properties
  - Efficient integration with the query processing engine

SGB: Three Instances

Unsupervised SGB

```
SELECT select_expr, ...
FROM table_references WHERE
    where_condition
GROUP BY col_name
    [MAXIMUM_ELEMENT_SEPARATION s]
    [MAXIMUM_GROUP_DIAMETER d], ...
```

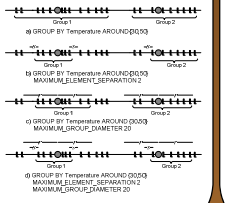
- No extra data provided to guide the process
- Clauses control group size and group compactness



Supervised Similarity Group Around

```
SELECT select_expr, ...
FROM table_references WHERE
    where_condition
GROUP BY col_name AROUND central-points
    [MAXIMUM_GROUP_DIAMETER 2r]
    [MAXIMUM_ELEMENT_SEPARATION s], ...
```

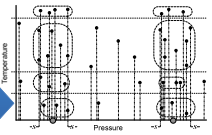
- Groups tuples *around* a set of guiding points
- Each tuple is assigned to the group of its closest central point.
- Clauses control group size and group compactness



Supervised Similarity Group with Delimiters

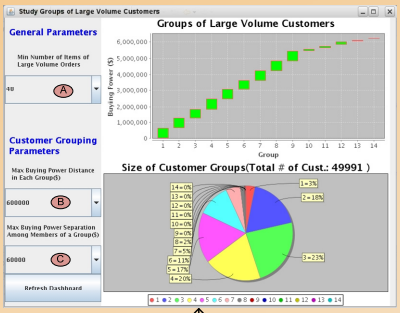
```
SELECT Avg(Temperature), Avg(Pressure)
FROM SensorsReadings
GROUP BY
Temperature DELIMITED BY
        (SELECT Value FROM Thresholds),
Pressure AROUND (30,50)
        MAXIMUM_ELEMENT_SEPARATION 3
```

- Forms groups based on a set of *delimiting* points
- Several similarity grouping strategies in the same SQL statement
- Each grouping attribute can use a different strategy



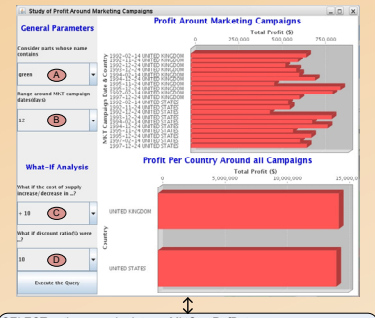
Exploiting SGB in Decision Support System Dashboards

Studying groups of large-volume customers with similar buying power



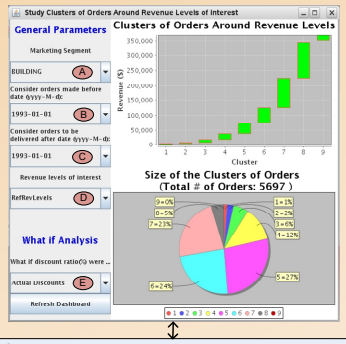

```
SELECT TotalBuy as TotalBuyLevelRef, min(TotalBuy),
       max(TotalBuy), count(TotalBuy), avg(TotalBuy)
FROM (SELECT c_name,c_custkey,sum(l_extendedprice) as TotalBuy
      FROM C, O, L WHERE c_custkey = o_custkey
        and o_orderkey = l_orderkey and
        o_orderkey IN (SELECT l_orderkey FROM L
                       GROUP BY l_orderkey
                       HAVING sum(l_quantity) > A)
      GROUP BY c_name,c_custkey)
GROUP BY TotalBuy MAXIMUM_GROUP_DIAMETER B
                  MAXIMUM_ELEMENT_SEPARATION C
```

Studying profit of a line of parts around marketing campaigns



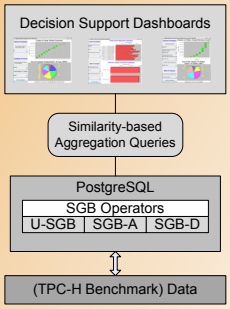

```
SELECT nation, o_orderdate as MktCmpRefDate,
       sum(amount) as sum_profit
FROM (SELECT n_name as nation, o_orderdate, l_extendedprice *
        (1 - l_discount) – ps_supplycost * l_quantity*(1+C) as amount
      FROM P, S, L, PS, O, N WHERE
        s_suppkey = l_suppkey and ps_suppkey = l_suppkey and
        ps_partkey = l_partkey and p_partkey = l_partkey and
        o_orderkey = l_orderkey and s_nationkey = n_nationkey
        and p_name like '%A%') as profit
GROUP BY nation, o_orderdate AROUND <MktCmpDates>
        MAXIMUM_GROUP_DIAMETER interval 'B'
ORDER BY nation
Note: l_discount is replaced by D in case D is not '(Actual Discounts)'
```

Studying groups of orders around revenue levels of interest




```
SELECT revenue as RevLevel, count(revenue),
       min(revenue), max(revenue), avg (revenue)
FROM  (SELECT l_orderkey, sum(l_extendedprice* (1-l_discount)) as
         revenue FROM C, O, L
         WHERE c_mktsegment = 'A' and c_custkey = o_custkey and
         l_orderkey = o_orderkey and o_orderdate < date 'B' and
         l_shipdate > date 'C'
       GROUP BY l_orderkey) as R1
GROUP BY revenue AROUND <D>
Note: l_discount should be replaces by E if E is not '(Actual Discounts)'
```

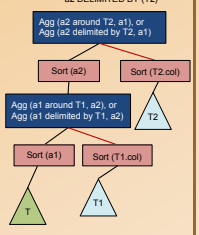
DSS Architecture



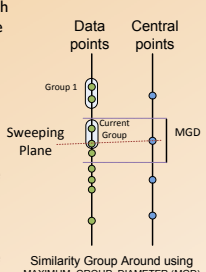
SGB Implementation (PostgreSQL)

- Parser
  - Extended the grammar rules
  - Extended the parse-tree and query-tree structures
- Planer/Optimizer
  - Made use of the RHS input plan tree of aggregation nodes to process the reference points
  - Each internal aggregation node processes 1 SGA and 1 or more GAs
  - SGAs can be ordered to reduce number of flowing tuples

1. SELECT ... FROM (T) GROUP BY a1 AROUND (T1), a2 AROUND (T2)
2. SELECT ... FROM (T) GROUP BY a1 DELIMITED BY (T1), a2 DELIMITED BY (T2)

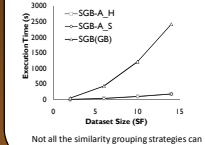


- The executor
  - Hash-based approach used to maintain the formed groups
  - Single plane sweep approach used to form the groups
  - The tuples to be grouped and the reference points are processed simultaneously
  - Data tuples and reference points are sorted before being processed by the aggregation node



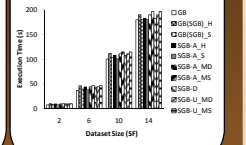
Performance Evaluation (TPC-H)

Performance of generating similarity groups with GB vs. SGB

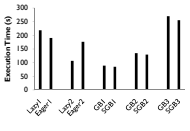


Not all the similarity grouping strategies can be obtained using only regular group-by

Performance while increasing dataset size



Performance of complex (TPC-H based) queries



Performance while increasing SGAs