SimDB: A Similarity-aware Database System

Yasin N. Silva and Walid G. Aref
Purdue University

**Similarities Everywhere**

- Many applications scenarios need the support of queries that capture and take advantage of similarities in the data.
- Data cleaning
- Multimedia and video applications
- Marketing analysis
- Shift from systems that focus on exact semantics of data and queries to systems that focus on approximate and imprecise semantics.
- Similarity-aware query processing in DB
- Integration of IR and DB operations
- Uncertain or probabilistic databases

**Our proposal: Similarity-aware DB Operators**

- We propose to extend the standard database operators to exploit similarities in the data
- The main goal of the proposed similarity-aware operators is to generate more meaningful and useful answers than those of their regular counterparts while maintaining:
  - Low running time
  - Good scalability properties
  - Efficient integration with the query processing engine

**Similarity-aware Operators**

**Similarity Group-by (SGB)**

Generic Definition:

\[(G_1, S_1) \ldots (G_n, S_n, J_1(\mathbf{A}_1), \ldots, J_m(\mathbf{A}_m)) (R)\]

**SGB Instances:**

- Unrestricted SGB
- Similarity Group Awarded
- SGB with Definition

**Similarity Join (SJ)**

Generic Definition:

\[A \mathcal{J}_\mathcal{J}_i B = \{(a, b) \mid \mathcal{J}_i(a, b), a \in A, b \in B\}\]

**SJ Instances:**

- Range Distance Join
- Instance-Join
- Join-Around

**Optimization**

- Core Equivalence Rules for similarity-aware operators
- Equivalence Among Similarity Operators
- Eager and Lazy Aggregation Transformations with SJ and SGB
- Extended techniques to use Materialized views to answer similarity-aware queries

**Example of Core Equivalence Rules**

- Eager and Lazy Aggregation Transformations from Join-Around to GB

**Implementation (PostgreSQL)**

- The Parser
  - Extended the grammar rules and parse tree structure
    - The Plan/Optimizer
      - Each SGB node processes 1 SGA/JP or more GAS/JPs
      - SGB nodes may use their input plan tree

**The Parser - SGB**

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

**The Executor - SGB**

- Full sweep approach
- Efficient storage and retrieval of data

**Performance Evaluation (TPC-H)**

- Performance of SGB while increasing dataset size
- Comparison of SGB and SJ with queries that obtain the same answer using regular operators
- Effectiveness of pushing selection under SJ
- Effectiveness of Associativity transformation for SJ
- Performance of complex TPC aggregation queries