SimDB: A Similarity-aware Database System

Yasin N. Silva\textsuperscript{1} Ahmed M. Aly\textsuperscript{1} Walid G. Aref\textsuperscript{1} Per-Ake Larson\textsuperscript{2}  
\textsuperscript{1}Dept. of Computer Science \hspace{1em} \textsuperscript{2}Microsoft Research \newline Purdue University \hspace{1em} Redmond, WA, USA \newline \{ysilva, aaly, aref\}@cs.purdue.edu \hspace{1em} palarson@microsoft.com

ABSTRACT
The identification and processing of similarities in the data play a key role in multiple application scenarios. Several types of similarity-aware operations have been studied in the literature. However, in most of the previous work, similarity-aware operations are studied in isolation from other regular or similarity-aware operations. Furthermore, most of the previous research in the area considers a standalone implementation, i.e., without any integration with a database system. In this demonstration we present SimDB, a similarity-aware database management system. SimDB supports multiple similarity-aware operations as first-class database operators. We describe the architectural changes to implement the similarity-aware operators. In particular, we present the way conventional operators’ implementation machinery is extended to support similarity-aware operators. We also show how these operators interact with other similarity-aware and regular operators. In particular, we show the effectiveness of multiple equivalence rules that can be used to extend cost-based query optimization to the case of similarity-aware operations.

Categories and Subject Descriptors
H.2.4 [Systems]: Query processing.

General Terms

Keywords
Similarity-aware Query Processing and Optimization, Similarity Group-by, Similarity Join.

1. INTRODUCTION
Multiple application scenarios, e.g., marketing analysis, medical applications and data cleaning; can significantly benefit from the identification and processing of similarities in the data. Several techniques have been proposed to extend some data operations, e.g., selection and join, to process similarities in the data ([1], [2], [3], [4], [5], [6]). Unfortunately, in most of the previous work, similarity-aware operations are studied in isolation from other regular and similarity-aware operations. Furthermore, most of the previous research in the area considers a standalone implementation, i.e., without any integration with a database system.

In this demonstration we present SimDB, a similarity-aware database system. SimDB supports multiple similarity-aware operations as first-class physical database operators. The implementation of these operators at the database level has the following key advantages: (1) similarity-aware operators can be interleaved with other regular or similarity-aware operators and their results pipelined for further processing; (2) important optimization techniques, e.g., pushing certain filtering operators to lower levels of the execution plan, pre-aggregation, and the use of materialized views can be extended to the new operators; and (3) the implementation of these operators can reuse and extend other operators and structures, and use the cost-based query optimizer machinery to enhance execution time. SimDB currently supports multiple similarity grouping and similarity join operators. In this demonstration, we describe the architectural changes to implement the similarity-aware operators. In particular, we present the way the implementation machinery of conventional operators is extended to support similarity-aware operators. We also show practically how these operators interact with other similarity-aware and regular operators. In particular, we show experimentally the effectiveness of multiple equivalence rules that can be used to extend cost-based query optimization to similarity-aware operations. SimDB builds on the results of [7], [8], and [9].

The remaining part of the paper is organized as follows. Section 2 presents the similarity-aware operators supported in SimDB. Section 3 discusses the implementation of these operators and several optimization techniques. Section 4 presents the demonstration scenario and Section 5 the conclusions and future work paths.

2. SimDB’s SIMILARITY-AWARE OPERATORS
The current version of SimDB supports several types of similarity grouping and similarity join.

2.1 Similarity Grouping Operators
The generic definition of the similarity group-by (SGB) operator is as defined in [7]:

\[(G_{1}, S_1), \ldots, (G_n, S_n) \gamma F_1(A_1), \ldots, F_m(A_m) (R)\]

where \(R\) is a relation name, \(G_i\) is an attribute of \(R\) that is used to generate the groups, i.e., a similarity grouping attribute, \(S_i\) is a segmentation of the domain of \(G_i\) in non-overlapping segments, \(F_i\) is an aggregation function, and \(A_i\) is an attribute of \(R\).

SimDB supports several instances of the previous generic definition: Unsupervised Similarity Group-by (SGB-U), Supervised Similarity Group Around (SGB-A), and Supervised
The generic definition of the Similarity Join (SJ) operator is as defined in [8]:

$$A \Join_{\theta} B = \{ (a, b) | \theta_S(a, b), a \in A, b \in B \}$$

where $$\theta_S$$ represents the similarity join predicate. This predicate specifies the similarity-based conditions that the pairs $$<a, b>$$ need to satisfy to be in the SJ output. The SJ predicates for the similarity join operators supported in SimDB are as follows.

- **Range Distance Join (C-Join):** $$\theta_\epsilon : \text{dist}(a, b) \leq \epsilon$$
- **Join-Around (A-Join):** $$\theta_{A_{r}} : b \text{ is the closest neighbor of } a \text{ and } \text{dist}(a, b) \leq r$$

The $$\epsilon$$-join operator (e.g., Figure 2.a) is an extensively used type of SJ. The Join-Around (e.g., Figure 2.b) is a useful type of SJ in which every value of the first joined set is assigned to its closest value in the second set. Additionally, only the pairs separated by a distance of at most $$r$$ are part of the join output.

3. QUERY PROCESSING AND OPTIMIZATION IN SimDB

3.1 Query Processing in SimDB

SimDB extends PostgreSQL, an open source DBMS. The current implementation of similarity-aware operators in SimDB supports multiple independent numeric grouping attributes for SGB and multiple join predicates over numeric attributes for SJ. To add support for SGB and SJ in the parser, the raw-parsing grammar rules, e.g., the yacc rules in the case of PostgreSQL, are extended to recognize the syntax of the different new grouping approaches and join predicates. The parse-tree and query-tree data structures are extended to include the information about the type and parameters of the similarity-based operations. Figure 3 shows the changes in these data structures to support the SGB operators.

In the planning stage, when multiple similarity grouping attributes (SAGs) or SJ predicates are used, they are processed one at a time. Figure 4 gives the structure of the plan trees generated when two SAGs $$a_1$$ and $$a_2$$ are used. The bottom aggregation node applies similarity grouping on $$a_1$$ and regular aggregation on $$a_2$$. The output of this node is aggregated by the top aggregation node that applies similarity grouping on $$a_2$$ and regular aggregation on $$a_1$$. Note that supervised aggregation nodes make use of their

Figure 1. SimDB’s Similarity Group-by-operators

Figure 2. SimDB’s Similarity Join operators

SGB using Delimiters (SGB-D). SGB-U (e.g., Figure 1.a) enables grouping tuples based on desired group properties, e.g., size (MAXIMUM_GROUP_DIAMETER) and compactness (MAXIMUM_ELEMENT_SEPARATION). SGB-A (e.g., Figure 1.b) allows grouping around points of interest. SGB-D (e.g., Figure 1.c) enables segmenting the tuples based on given limiting values. These instances represent a middle ground between the regular group-by and clustering algorithms. They are intended to be much faster than regular clustering algorithms and generate results that capture similarities on the data not captured by the regular group-by. As evident from Figure 1, similarity group-by instances are able to identify successfully the naturally formed groups.

2.2 Similarity Join Operators

The similarity group-by operators in SimDB are as follows.

- **Group 1**
  - **Group 2**
  - **Group 3**
  - **Group 4**
  - **Group 5**
  - **Group 6**
  - **Group 7**

Figure 3. Modifications in the main query processing data structures to support SGB operators

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)
3. SELECT ... FROM (T) GROUP BY a1 MAX_ELMT_SEP_S1, a2 MAX_ELMT_SEP_S2

Figure 4. Path/Plan trees for SGB with multiple SAGs
SELECT … FROM T1, T2 WHERE T1.a AROUND T2.b AND T2.c AROUND T3.d

SELECT … FROM T1, T2, T3 WHERE T1.a AROUND T2.b AND T2.c AROUND T3.d AND T1.d BETWEEN E1 OF T2.b AND T2.c WITHIN E2 OF T3.d

Join-Around (a,b), or Epsilon-Join (c,d)
each query and experimentally demonstrate how the usage of equivalence rules, like the ones presented in section 3.2, allow the generation of better execution plans.

5. CONCLUSIONS AND FUTURE WORK

We present SimDB, a similarity-aware database system that supports multiple similarity-aware operators. We describe the way these operators have been implemented and how transformation rules are used to generate better execution plans. Plans for future work include the implementation of other similarity-aware operators and the integration of indexing techniques to support similarity-aware operations at the database level.

6. REFERENCES


