*Sim*DB: A Similarity-aware Database System*

ICDS
Indiana Center for Database Systems

Yasin N. Silva and Walid G. Aref
Purdue University

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

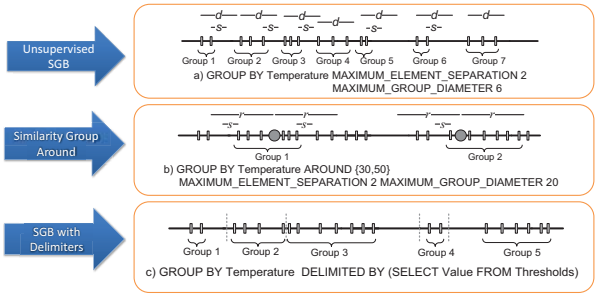
	Similarity Operator Implementation Approach			
	Integrated in DB Engine	Using Basic SQL Operators	Outside of DB	As Stored Procedures (SP)
Supported Operator Instances	All	Certain types may be unfeasible or require very complex queries	All	All
Implementation complexity	Can reuse and extend DB operators and structures	Queries use a complex mix of joins and aggregations	Requires the support of specialized structures, mechanisms to deal with very large data sets, etc.	Requires the support of specialized structures, spilling mechanisms, etc.
Composable with other DB operators (pipelining)	Yes	Yes (resulting queries can be highly complex)	No	No
Take advantage of DB cost based optimization	Yes (e.g., queries can be pre-aggregated, use MVs, be translated, etc.)	No directly	No	No

Similarity-aware Operators

Similarity Group-by (SGB)

Generic Definition: $(G_1,S_1),\ldots,(G_n,S_n)\gamma_{F_1(A_1),\ldots,F_m(A_m)}(R)$

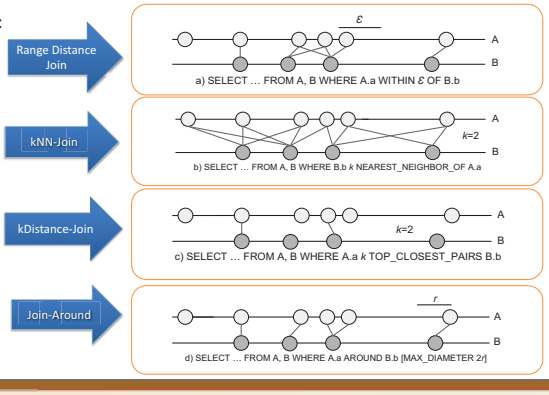
SGB Instances:



Similarity Join (SJ)

Generic Definition: $A \bowtie_{\theta_S} B = \{\langle a,b\rangle \mid \theta_S(a,b), a \in A, b \in B\}$

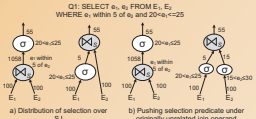
SJ Instances:



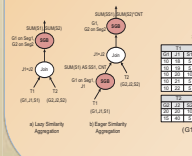
Optimization

- 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

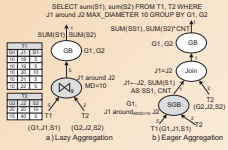
Example of Core Equivalence Rules



Eager and Lazy Aggr. Transf. for SGB



Pushing Similarity Predicate from Join-Around to GB

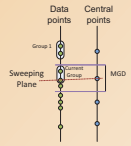


Implementation (PostgreSQL)

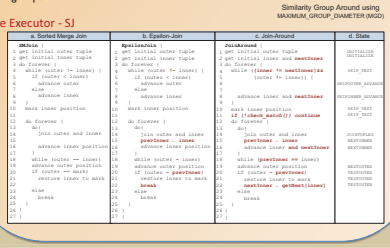
- The Parser
  - Extended the grammar rules and parse tree structure
- The Planer/Optimizer
  - Each SGB/SJ node processes 1 SGA/SJP and 1 or more GAs/JPs
  - SGB nodes make use of their inner input plan tree

The Executor - SGB

- 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
- Hash-based approach used to maintain the formed groups

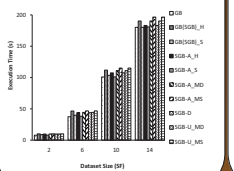


The Executor - SJ

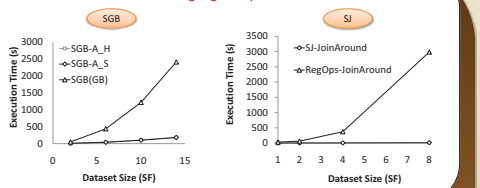


Performance Evaluation (TPC-H)

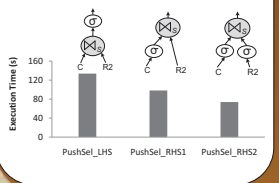
Performance of SGB while increasing dataset size



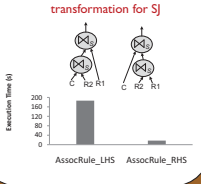
Comparison of SGB and SJ with queries that obtain the same answer using regular operators



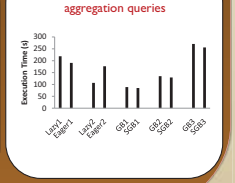
Effectiveness of pushing selection under SJ



Effectiveness of Associativity transformation for SJ



Performance of complex TPC aggregation queries



* This work was partially supported by NSF Grant IIS-0811954 and by NIH Grant NIGMS U24 GM077905 for the EcoliHub project.