Queries for Similarity Analytics

Vlasta Dohnal

Data Analytics

- a process of analyzing data and presenting results to users
  - to make informed decisions
- tools and applications (e.g. Data Warehouse)
  - to collect data
  - to prepare it for storage and analysis
  - *to develop and run queries*
  - to create reports and dashboards
  - to visualize data
- evolved from decision support systems
- business analytics / (advanced) data analytics
  - prescriptive analytics

Outline

- Motivation, examples
- Similarity operators
  - Similarity Selection / Join / Set / Group By
- Operator evaluation
  - Consistent and efficient
- Extensions to similarity group by
- Conclusion

- Credits
  - Silva, Y.N., Aref, W.G., Larson, PA., Pearson S.S., Ali M.H.: *Similarity queries: their conceptual evaluation, transformations, and processing.* VLDB Journal, vol. 22, pp 395–420, 2013.
  - Tang, Mingjie, Tahboub, Ruby Y., Aref, Walid G., Atallah, Mikhail J., Malluhi, Qutaibah M., Ouzzani, Mourad, Silva, Yasin N. *Similarity Group-By operators for multi-dimensional relational data.* ICDE, pp 1448-1449, 2016.
  - Al Marri, W. J., Malluhi, Q., Ouzzani, M., Tang, M., Aref, W. G. *The similarity-aware relational database set operators.* Information Systems, vol. 59, pp. 79-93, Elsevier, 2015.

Motivation & Examples

- Data processing
  - Searching that allows some "fuzzy" comparison between data objects
- Database systems
  - Make them similarity-aware

- Example queries
  - Find the closest three suppliers from my location
    - → k-nearest neighbor query
  - Find the cheapest gas station within 20km
    - → range query with order by/limit

Motivation & Examples

- More examples
  - Find the closest three suppliers for every customer within 100 miles from our Chicago headquarters
    - → range query on customers and kNN-join with suppliers
  - Considering the customers that are located within 200 miles from our Chicago headquarters, cluster the customers around certain locations of interest, and report the size of each cluster
    - → range query on customers and group by around some points and compute aggregates
  - For every customer, identify its closest 3 suppliers and for each such supplier, identify its closest 2 potential new suppliers
    - → kNN-join with suppliers and kNN-join with pot. suppliers

Similarity Query Language

- SQL extended with similarity operators
- Implemented in a relational DBMS
  - Extended query grammar
  - Extended query optimizer
    - Requires statistics about similarity operators
  - Implemented similarity operators
- Applied on TPC-H benchmark
  - E.g. Retrieve customers with similar buying power and account balance

Similarity Operators

- Assume
  - a dataset *X*, i.e. a set of data objects (*o, p, q,..., x, y*),
  - a distance function $d(o_1,o_2)$, and
  - descriptors (attributes/properties) of objects: o.A, o.B, …
- S. Selection
  - k-nearest neighbor query (kNN)
  - range query
  - combination (kNN(q,r))
- S. Join
  - ε-join
  - kNN-join
  - self joins
- S. Intersection, Union, Difference
- S. Group By
  - Around selected points
  - Unsupervised grouping
- → More operators used within one query

Similarity Selection (SS)

- Well-known similarity queries on a dataset

$$\sigma_{\Theta_S(x.A,c.A)}(X) = \{x | \Theta_S(x.A, c.A), x \in X\}$$

- ε-selection
  - $\Theta_{\varepsilon=\boldsymbol{r}}(x.A, q.A) \coloneqq (d(x.A, q.A) \leq \boldsymbol{r})$
  - Alt. $\Theta_{\varepsilon,q.A}(x.A)$
- kNN selection
  - $\Theta_{kNN=\boldsymbol{k}}(x.A, q.A) \coloneqq \text{true if } x \in \boldsymbol{k}NN(q.A)$
  - Alt. $\Theta_{kNN,q.A}(x.A)$
  - If there are more objects at the distance of k^{th} neighbor, all are reported.

Similarity Join (SJ)

- Extends regular join by identifying similar pairs instead of equal ones

$$X \bowtie_{\Theta_S(x.A,y.B)} Y = \{\langle x, y\rangle | \Theta_S(x.A, y.B), x \in X, y \in Y\}$$

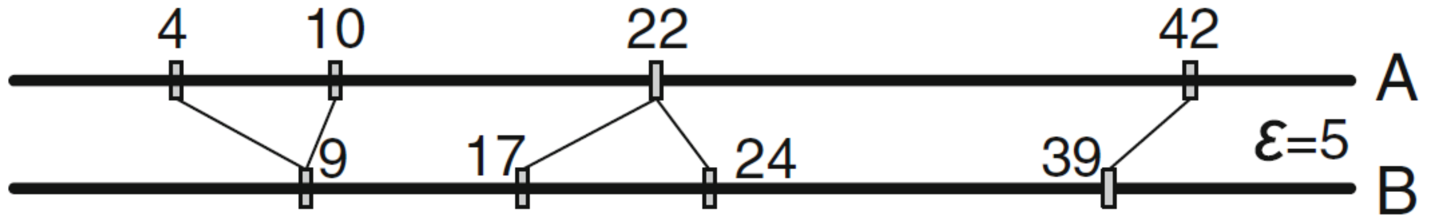
$\Theta_S(x.A, y.B)$ - similarity predicate

$$\sigma_{\Theta_S(x.A,y.A)}(X \times Y)$$

- Variants:
  - Range distance join (ε-join)
  - k-nearest neighbor join (kNN-join)
  - k-distance join (kD-join)
  - Join around (Join-Around)
  - Wide-join (by Traina group)

Similarity Join (SJ) – Variants

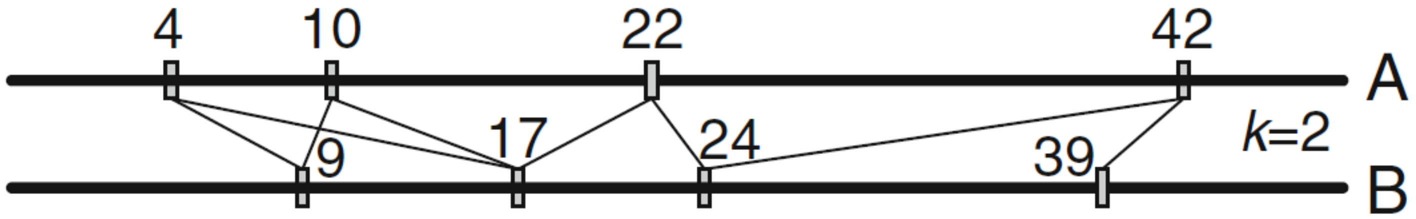
- Range distance join (ε-join)
  - $\Theta_{\varepsilon}(x.A, y.B) \coloneqq (d(x.A, y.B) \leq \varepsilon)$



(a) ε-Join: SELECT … FROM A, B WHERE A.a **WITHIN ε OF** B.b

Similarity Join (SJ) – Variants

- k-nearest neighbor join (kNN-join)
  - $\Theta_{kNN}(x.A, y.B) \coloneqq \text{true if } y.B \in kNN(x.A) \text{ on } Y.B$

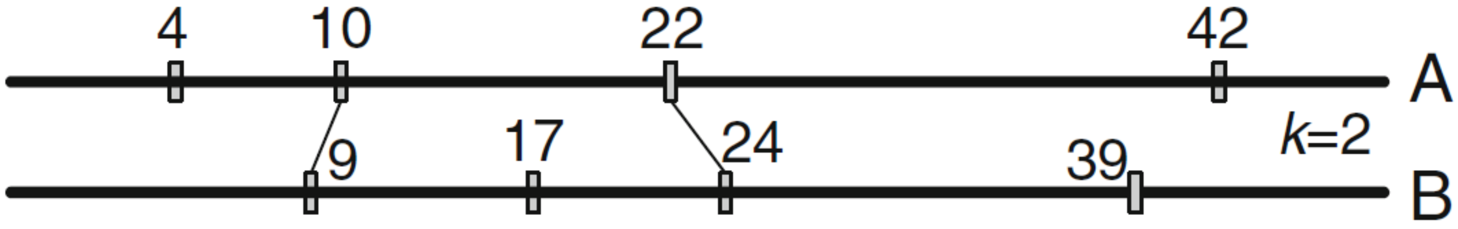


(b) kNN-Join: SELECT ... FROM A, B
WHERE B.b ***k* NEAREST_NEIGHBOR_OF** A.a

- Note for kNN:
  - If there are more objects at the distance of k^th neighbor, all are reported.

Similarity Join (SJ) – Variants

- k-distance join (kD-join)
  - $\Theta_{kD}(x.A, y.B) :=$ true if $\langle x.A, y.B\rangle \in$ overall k closest pairs in $X.A \times Y.B$

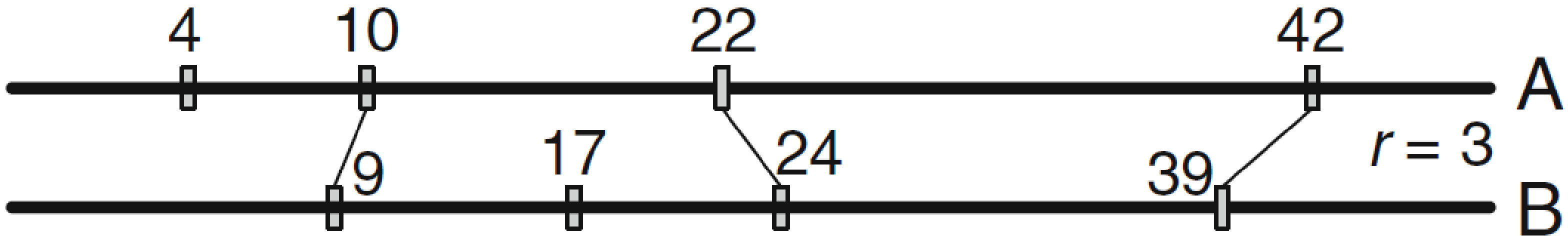


(c) kD-Join: SELECT ... FROM A, B
WHERE A.a ***k* TOP_CLOSEST_PAIRS** B.b

- Note for kD:
  - If there are more pairs with the k^th distance, all are reported.

Similarity Join (SJ) – Variants

- Join around (Join-Around)
  - $\Theta_{A,MD=2r}(x.A, y.B) \equiv \Theta_{1NN,2r}(x.A, y.B) \coloneqq$ true if $y.B \in 1NN(x.A, r)$ on $Y.B$
    - i.e. $y.B$ is the closest neighbor of $x.A$ and $d(x.A, y.B) \leq r$



(d) Join-Around: SELECT … FROM A, B
WHERE A.a **AROUND** B.b **[MAX_DIAMETER *2r*]**

- Note for $1NN, 2r$:
  - If there are more objects at the closest distance, all are reported.

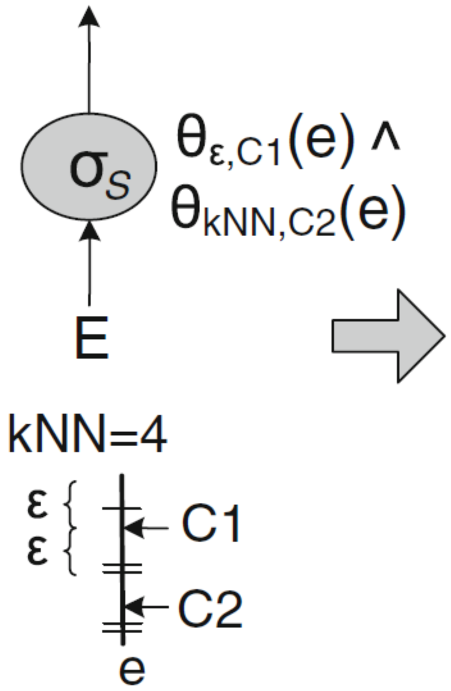
Combining Operators

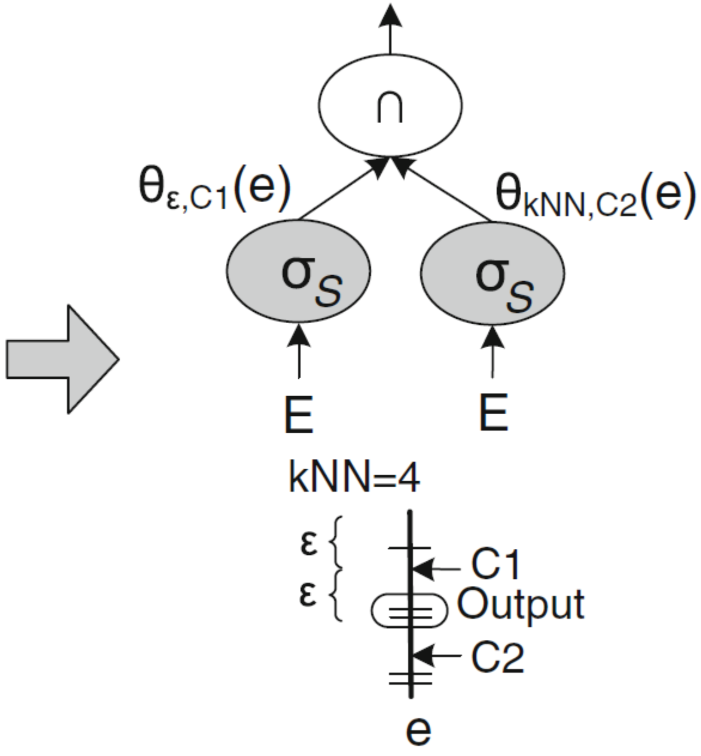
- Using the relational algebra style…

- Multiple predicates
  - Different selection predicates
    - $\sigma_{\Theta_{\varepsilon}(x.A,q1.A) \wedge \Theta_{kNN}(x.A,q2.A)}(X)$
- Multiple operators
  - $\sigma_{\Theta_{\varepsilon}(x.A,q1.A)}\left(\sigma_{\Theta_{kNN}(x.A,q2.A)}(X)\right)$
- Equivalence of operators
  - Similarity join vs. similarity selection
    - $X \bowtie_{\Theta_{\varepsilon}(x.A,y.B)} Y \equiv \sigma_{\Theta_{\varepsilon}(x.A,y.B)}(X \times Y)$

Combining Operators: Order Matters

- Query with C1, C2 q. objects: $\sigma_{\Theta_{\varepsilon,C1}(e)\wedge\Theta_{kNN,C2}(e)}(E)$



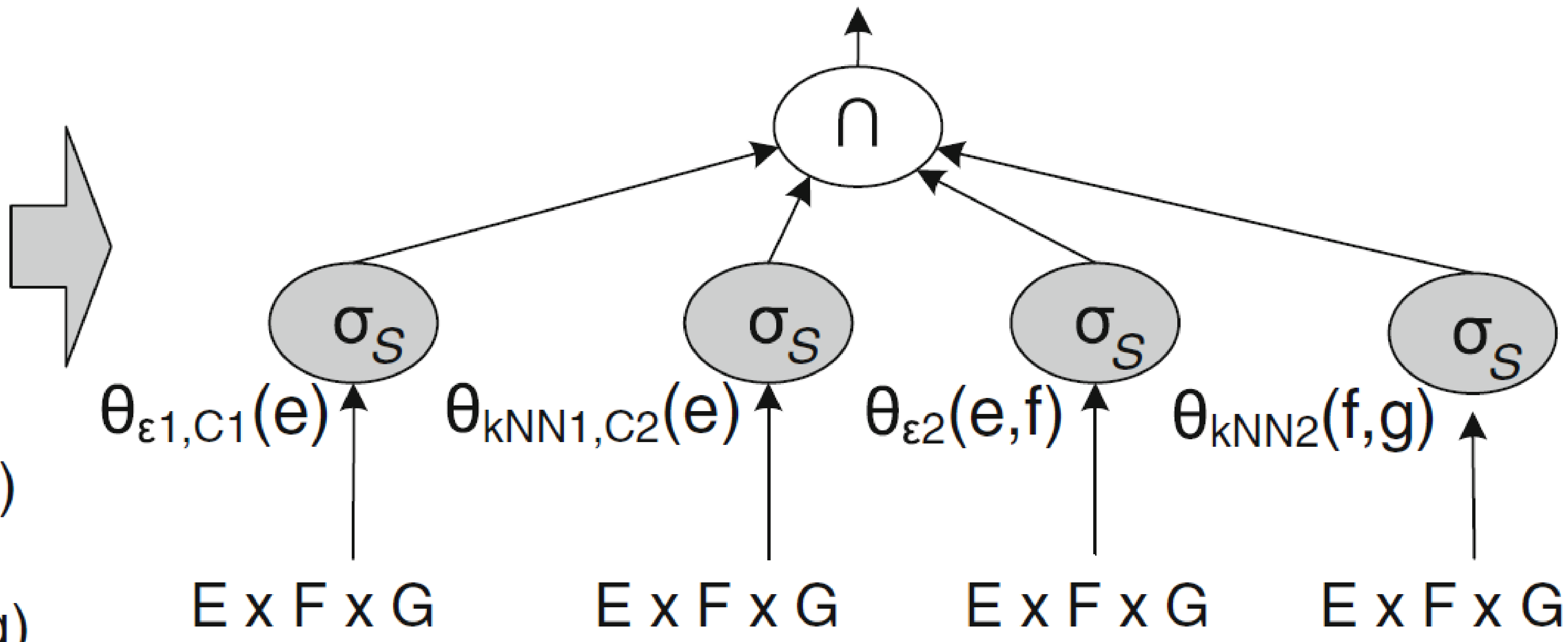


Conceptual Evaluation

Combining Operators: Conceptual Query Plan

- Combine sub-results with intersection → Consistent Evaluation

```
SELECT e, f, g
FROM E, F, G
WHERE
EpsSelPred(e)
AND
kNNSelPred(e)
AND
EpsJoinPred(e,f)
AND
kNNJoinPred(f,g)
```

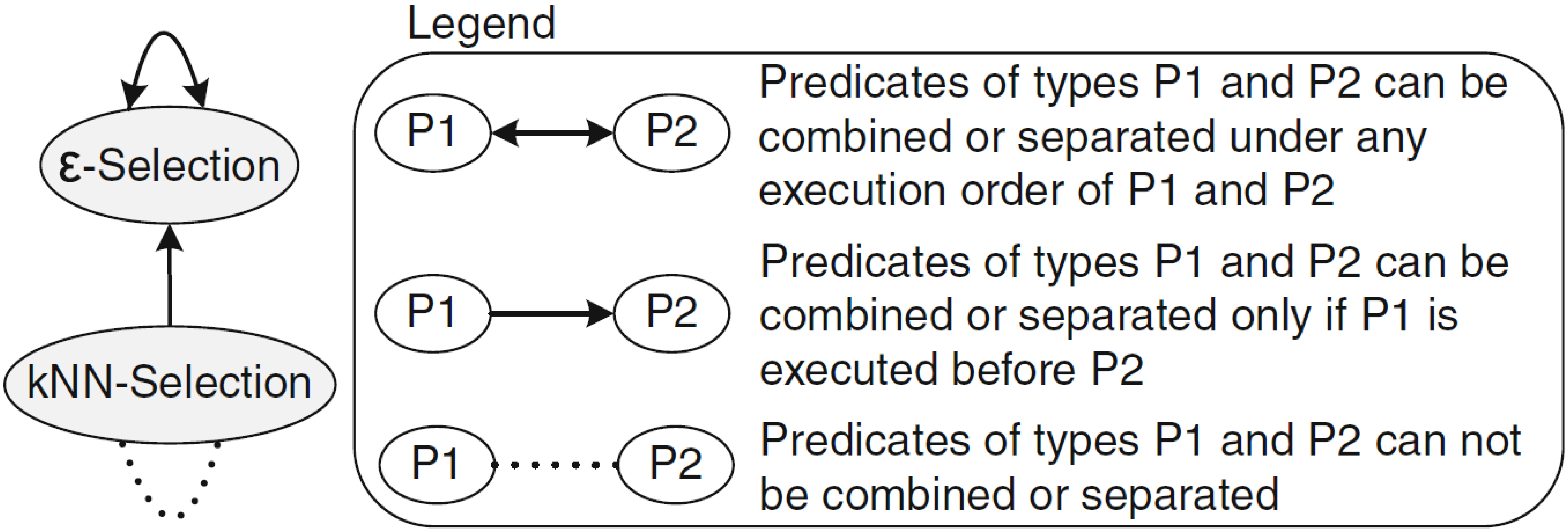


Optimizing Query Plan

- Query plan = a plan of executing individual operations to get query result
- Conceptual query plan is not optimal
  - Same data can be read multiple time
- Equivalence rules
  - Swapping operations in a plan to keep it equivalent to conceptual plan

  - Type of similarity predicates in operations define their order
    - kNN type has priority over range!

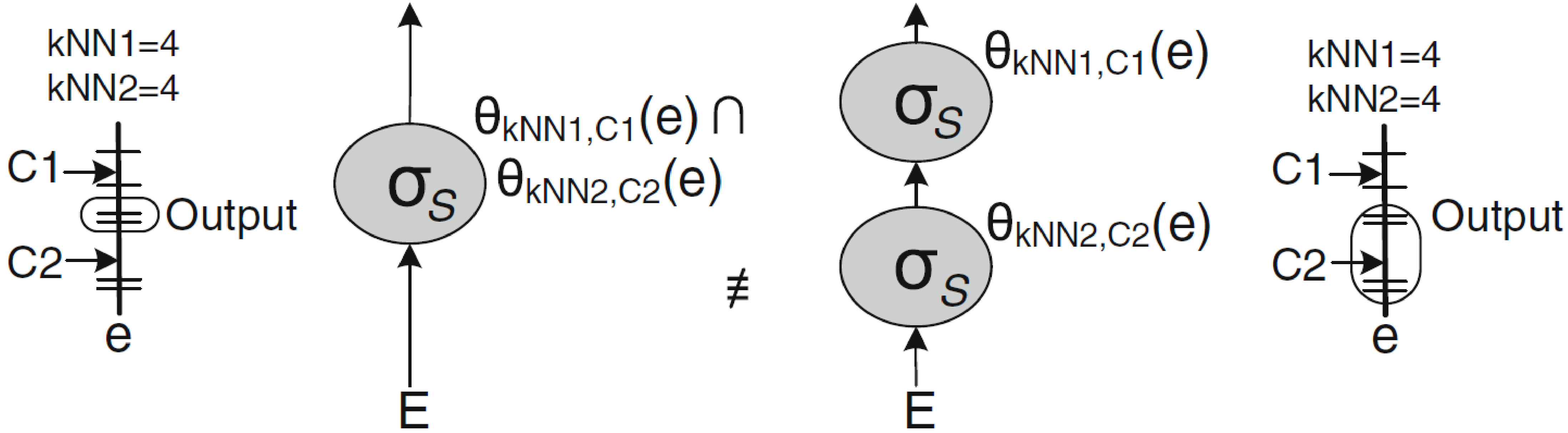
Selection Predicates



- $\sigma_{\Theta_{S1,C1}(x) \wedge \Theta_{S2,C2}(x)}(X) \equiv \sigma_{\Theta_{S1,C1}(x)}\left(\sigma_{\Theta_{S2,C2}(x)}(X)\right)$ iff there is an edge S2→S1

Selection Predicates: kNN

- It cannot be established, so must be executed independently

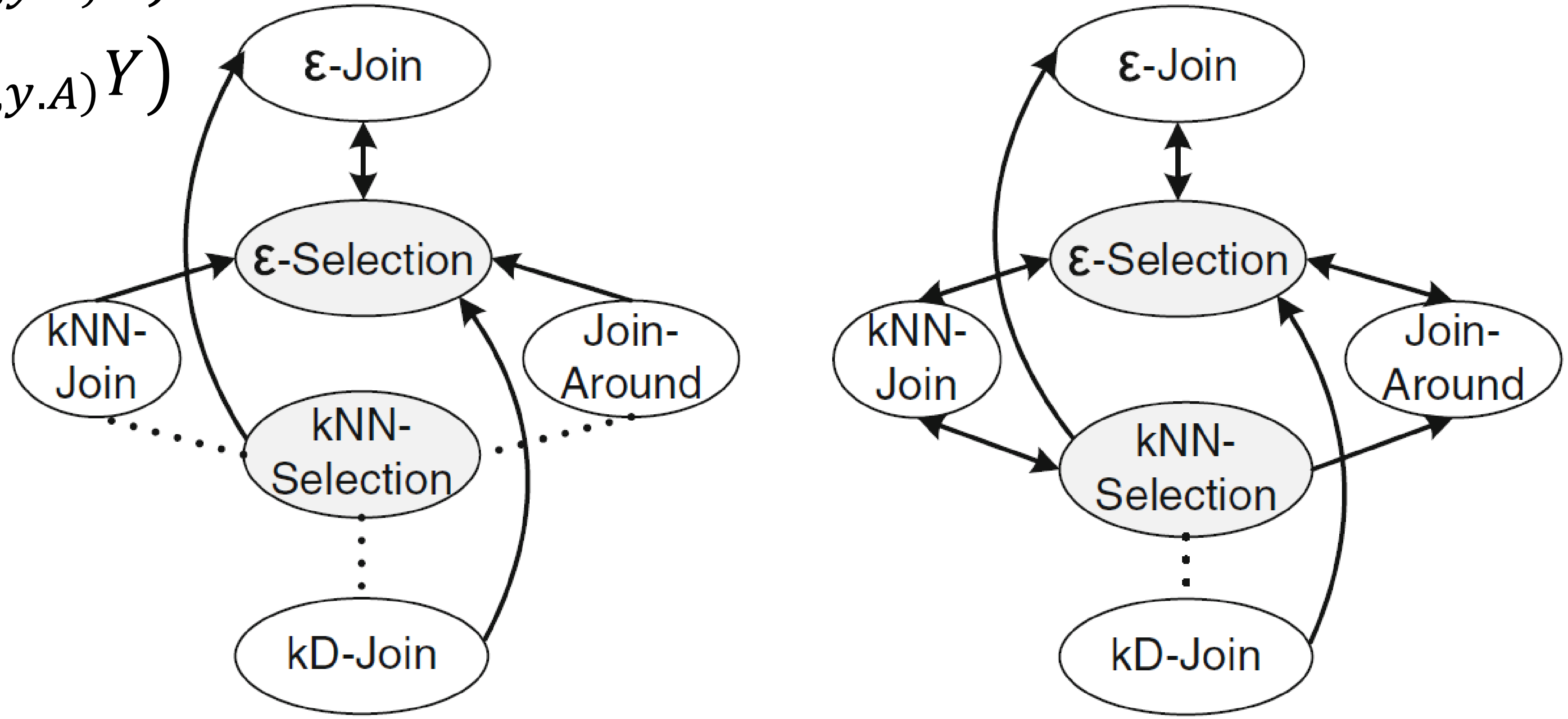


- → Implement a special “multi-kNN” operator?

Selection and Join: Distributivity

a) $\sigma_{\Theta_S(\boldsymbol{y}.\boldsymbol{B},C)}\left(X \bowtie_{\Theta_S(x.A,y.A)} Y\right)$

b) $\sigma_{\Theta_S(\boldsymbol{x}.\boldsymbol{B},C)}\left(X \bowtie_{\Theta_S(x.A,y.A)} Y\right)$



(a) When the sel. attribute is the inner attr. in the join predicate

(b) When the sel. attribute is the outer attr. in the join predicate

Selection and Join: Example Query

- Find the closest three suppliers for every customer within 100 miles from our Chicago headquarters (X,Y)
  - → range query on customers and kNN-join with suppliers

SELECT c_custkey, s_suppkey FROM CUSTOMER c, SUPPLIER s
WHERE c_loc **WITHIN 100 OF** (X,Y) AND s_loc **3 TOP_CLOSEST_NEIGHBOR_OF** c_loc;

$$\sigma_{\theta_{kNN=3}(c\_loc, s\_loc) \cap \theta_{\varepsilon=100, C=(X,Y)}(c\_loc)}(C \times S) \equiv$$
$$\sigma_{\theta_{kNN=3}(c\_loc, s\_loc)}(\sigma_{\theta_{\varepsilon=100, C=(X,Y)}(c\_loc)}(C \times S)) \equiv$$
$$\sigma_{\theta_{\varepsilon=100, C=(X,Y)}(c\_loc)}(\sigma_{\theta_{kNN=3}(c\_loc, s\_loc)}(C \times S)).$$

$$\sigma_{\Theta_{\varepsilon=100, C=(X,Y)}(c\_loc)}(C \bowtie_{\Theta_{kNN=3}(c\_loc, s\_loc)} S)$$
$$\sigma_{\Theta_{\varepsilon=100, C=(X,Y)}(c\_loc)}(C) \bowtie_{\Theta_{kNN=3}(c\_loc, s\_loc)} S$$

Combining Joins

- Commutativity
  - Yes: ε-join, kD-join (and distance function is symmetric)
  - No: kNN-join, Join-Around
- Associativity $E \bowtie_{\Theta_S(e.A,f.A)} F \bowtie_{\Theta_S(f.B,g.B)} G$
  - Yes: ε-join, kNN-join, Join-Around
  - No: kD-join
- "Commutativity" of unrelated datasets: $E \bowtie_{\Theta_S(e.A,f.A)} G \bowtie_{\Theta_S(g.A,f.A)} F$
  - Yes: ε-join
  - No: kNN-join, Join-Around, kD-join

Joins: Example Query

- For every customer, identify its closest 3 suppliers and for each such supplier, identify its closest 2 potential new suppliers

```
SELECT c_custkey, s_suppkey, psu_suppkey FROM CUSTOMER c, SUPPLIER s, POT_SUPPLIER psu
WHERE s_loc 3 TOP_CLOSEST_NEIGHBOR_OF c_loc
   AND psu_loc 2 TOP_CLOSEST_NEIGHBOR_OF s_loc;
```

$$(C \bowtie_{\theta_{kNN1=3}(c\_loc, s\_loc)} S) \bowtie_{\theta_{kNN2=2}(s\_loc, psu\_loc)} PSU \equiv C \bowtie_{\theta_{kNN1=3}(c\_loc, s\_loc)} (S \bowtie_{\theta_{kNN2=2}(s\_loc, psu\_loc)} PSU).$$

Selection and Join: Combining ε-predicates

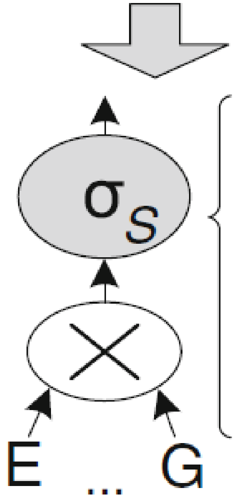
- Selection pull-up & push-down:
  - Used in relational DBMS to further optimize the query
- $\sigma_{\Theta_{\varepsilon 1, C}(x.A)}(X) \bowtie_{\Theta_{\varepsilon 2}(x.A, y.A)} Y \equiv$

$$\sigma_{\Theta_{\varepsilon 1, C}(x.A)}\left(X \bowtie_{\Theta_{\varepsilon 2}(x.A, y.A)} Y\right) \equiv$$

$$\left(\sigma_{\Theta_{\varepsilon 1, C}(x.A)}(X)\right) \bowtie_{\Theta_{\varepsilon 2}(x.A, y.A)} \left(\sigma_{\Theta_{(\varepsilon 1 + \varepsilon 2), C}(y.A)}(Y)\right)$$

Transformation Rules: Example

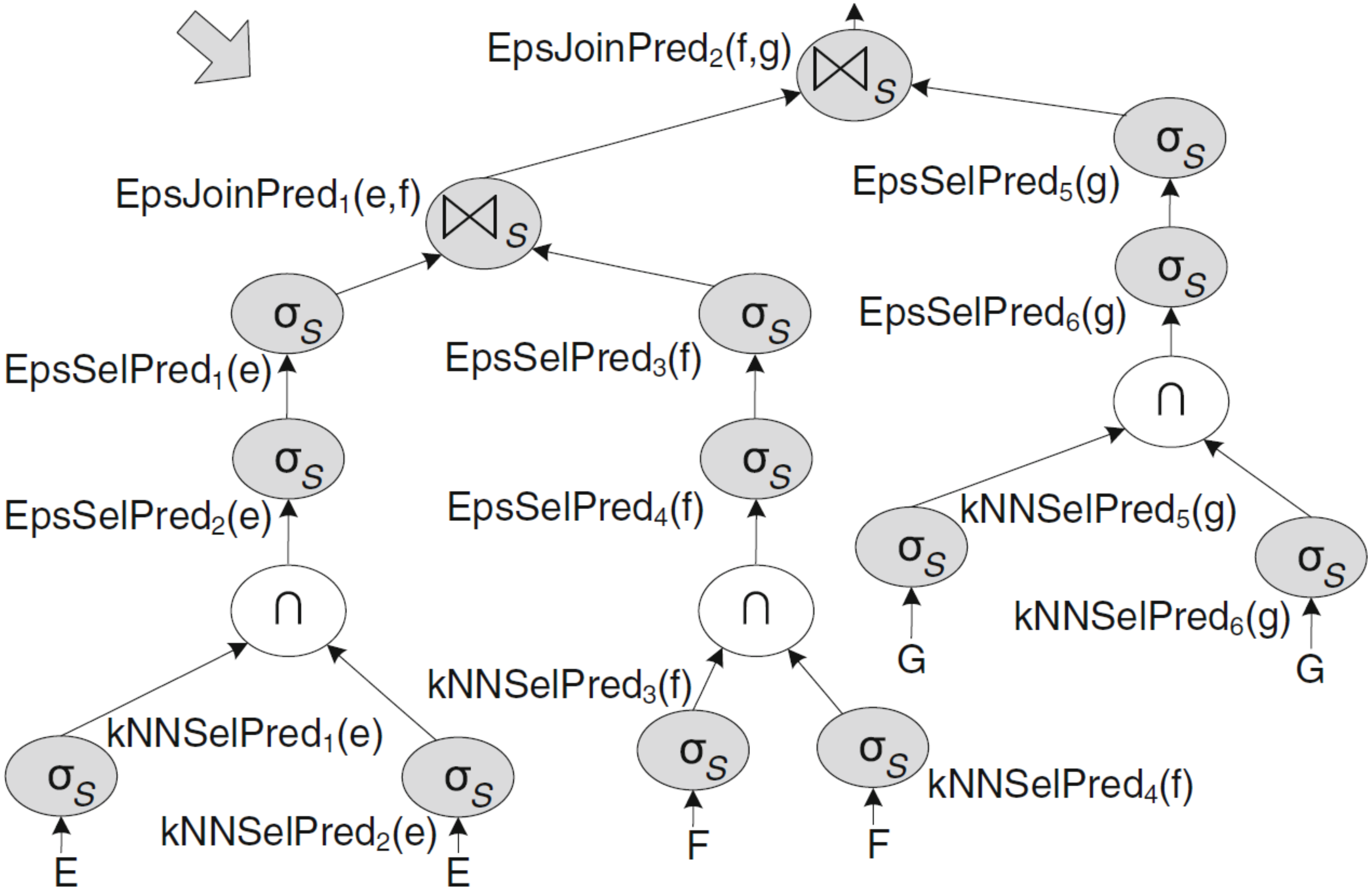
SELECT e, f, g FROM E, F, G WHERE EpsJoinPred$_1$(e,f) AND EpsJoinPred$_2$(f,g) AND EpsSelPred$_1$(e) AND EpsSelPred$_2$(e) AND kNNSelPred$_1$(e) AND kNNSelPred$_2$(e) AND EpsSelPred$_3$(f) AND EpsSelPred$_4$(f) AND kNNSelPred$_3$(f) AND kNNSelPred$_4$(f) AND EpsSelPred$_5$(g) AND EpsSelPred$_6$(g) AND kNNSelPred$_5$(g) AND kNNSelPred$_6$(g)



EpsJoinPred$_1$(e,f) $\cap$ EpsJoinPred$_2$(f,g) $\cap$ EpsSelPred$_1$(e) $\cap$ EpsSelPred$_2$(e) $\cap$ kNNSelPred$_1$(e) $\cap$ kNNSelPred$_2$(e) $\cap$ EpsSelPred$_3$(f) $\cap$ EpsSelPred$_4$(f) $\cap$ kNNSelPred$_3$(f) $\cap$ kNNSelPred$_4$(f) $\cap$ EpsSelPred$_5$(g) $\cap$ EpsSelPred$_6$(g) $\cap$ kNNSelPred$_5$(g) $\cap$ kNNSelPred$_6$(g)

Transformation Rules: Example

- No Cartesian product is used
- Datasets reused due to kNN selection

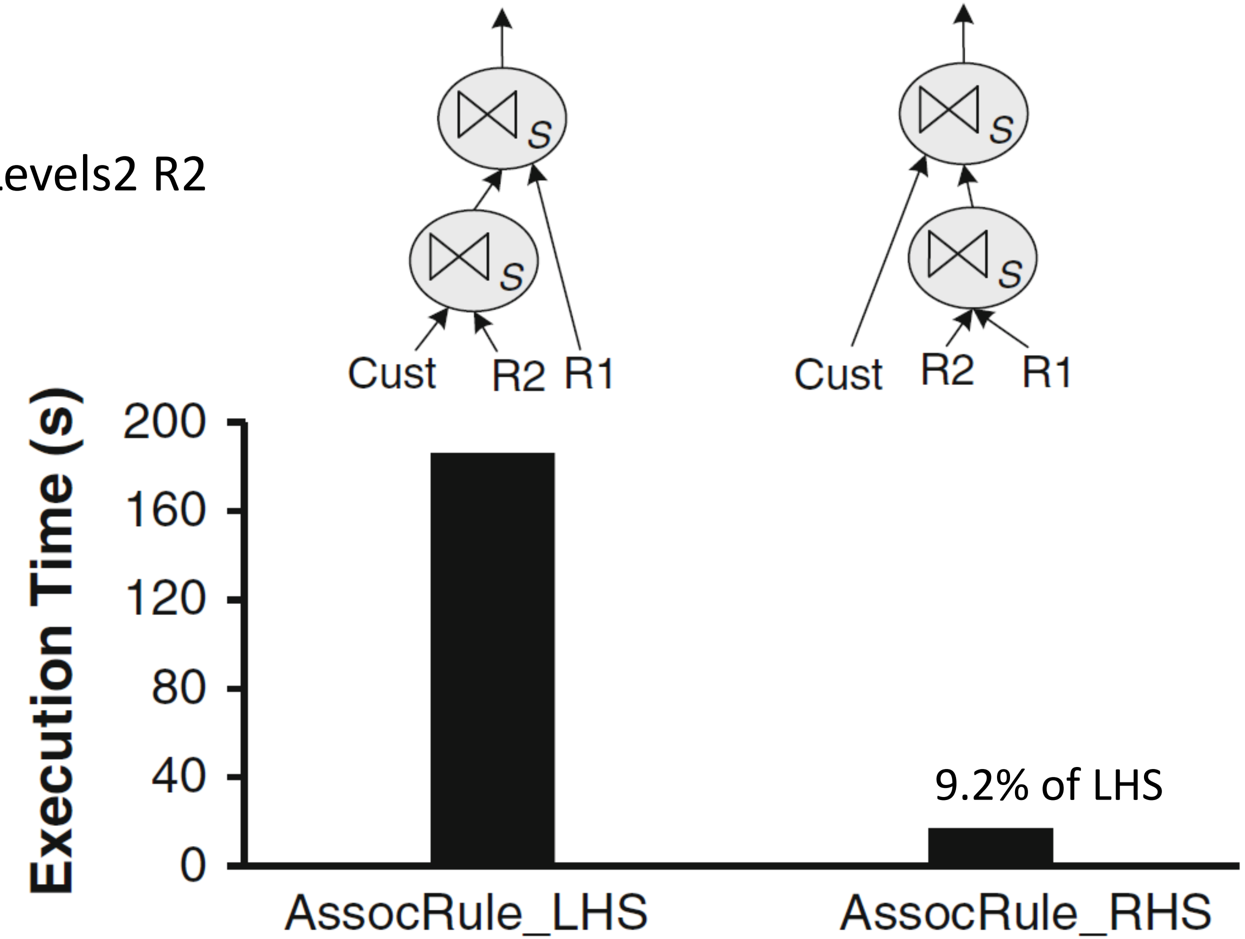


Transformation Rules: Performance

- Associativity of ε-join

  SELECT *
  FROM CUSTOMER C, AccBalLevels1 R1, AccBalLevels2 R2
  WHERE C acctbal **WITHIN 11 OF** R1.refpoint
  AND R1.refpoint **WITHIN 11 OF** R2.refpoint;

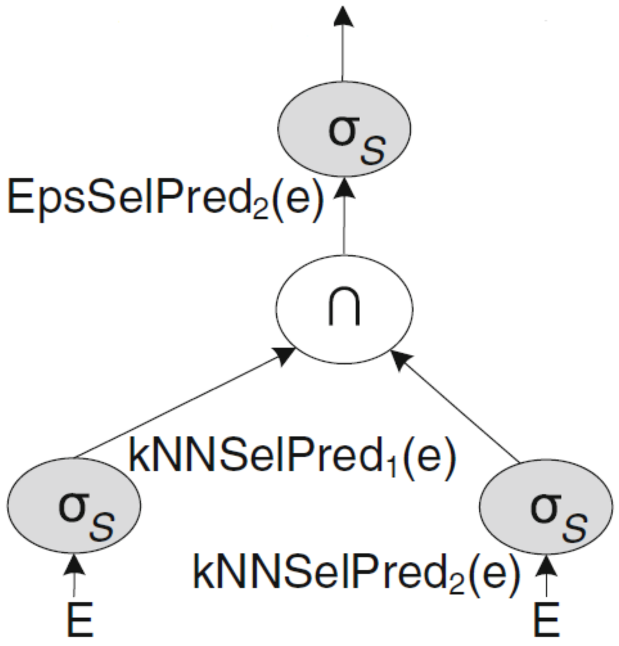
- Data
  - AccBalLevel1
    - 110 different levels of account balance in [0;11000]
  - AccBalLelel2
    - 11,000 dtto
  - Customer
    - 750,000 recs



Similarity Set Operators

- Similarity intersect / union / difference
  - Implemented in relational DB
  - Distance functions defined on regular attributes
  - Identify similar tuples using threshold distance ε
  - Efficient implementation – 100x faster than using regular DB operators

Summary



- Data analytics need multiple operators in a query
  - Consistent query evaluation is important
    - if no priority requested by the user
    - Some operator predicates are not commutative (involving kNN)
  - kNN must be performed as first!
    - Can kNN be constrained with ε using some statistics?
  - Equivalence rules cannot optimize everything
    - Special “multi-query” operation over one database needed (to combine kNN)
- Similarity group by
  - Applied as the last operator
  - Can be split and pushed down – eager aggregation

Similarity Group By

- Extended syntax of regular group by

$$_{(G_1,S_1),\dots(G_m,S_n)}\Gamma_{F_1(A_1),\dots F_n(A_n)}(X)$$

  S_1 - segmentation of domain of G_1 into non-overlapping groups
  F_1 - aggregation on A_1 of data objects in a group
- Result is a set of objects with regular attributes/properties

- Procedure:
  - Partition all data objects in the result into groups
  - Obtain group representatives
  - Compute aggregates F_i on all objects per group
    - i.e. each combination of segments (values) of all G_js

Similarity Group By: Implementation

- Task: Define segmentation for G_i
  - Using clustering
    - Multiple-pass external algorithms (k-means, hierarchical clustering)
      - very slow, may not be consistent
    - Approximation using sampling or summaries possible (BIRCH, CURE) – one-pass
      - still slow, hard to compute aggregates simultaneously, no pipelining with other operators
  - Special implementation
    - → control on constraint specification
    - → pipelining, aggregations during group-by, no approximations (like sampling)
    - Implemented in a query evaluation engine

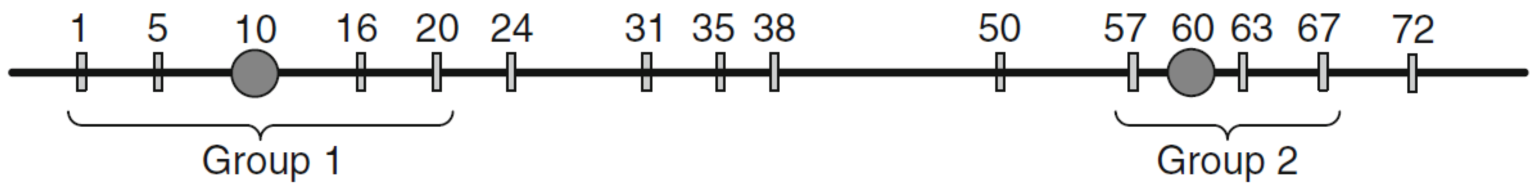
Similarity Group By: Implementation

- Variants:
  - Supervised – segmentation is defined in advance
    - E.g. cluster centers, dividing thresholds
  - Unsupervised – segmentation obtained from data
    - E.g. number of resulting clusters given

- Segmentation properties:
  - Element separation ***s*** – each object has another object within distance ***s***
  - Group diameter ***d*** – distance between the most separated objects ≤ ***d***

Similarity Group By Around

- Supervised: Around – SGB-A
  - Set of **central points** – define groups
    - objects assigned to the group of the closest central point
  - Element separation ***s*** and group diameter ***d*** (=2*r*) – optional

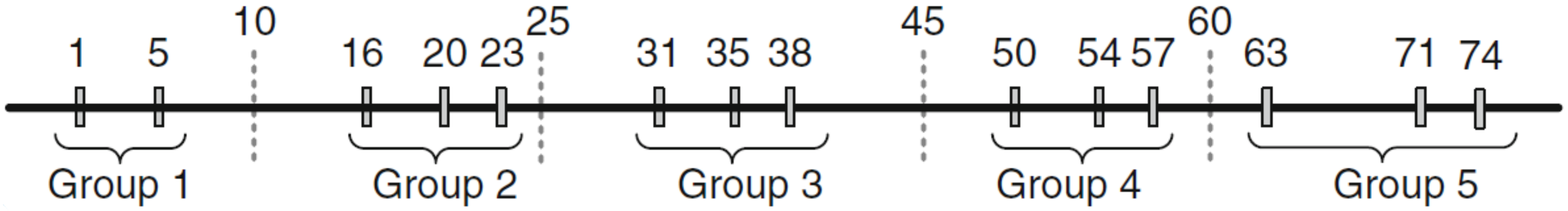



```
SELECT Max(Temperature), Avg(Temperature) FROM SensorsReadings
GROUP BY Temperature AROUND {10,60}
MAXIMUM_ELEMENT_SEPARATION 6 MAXIMUM_GROUP_DIAMETER 20
```

  - Some data objects might be excluded!

Similarity Group By Delimited

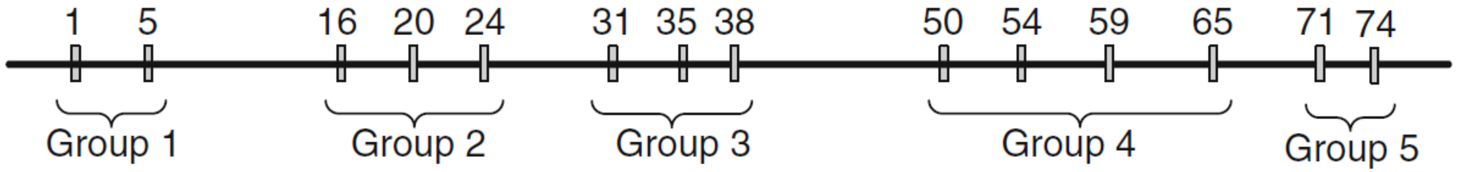
- Supervised: Delimited – SGB-D
  - Set of delimiting **hyperplanes**
    - For nD space: $a_1x_1 + a_2x_2 + \cdots + a_nx_n \leq b$ and $a_1x_1 + a_2x_2 + \cdots + a_nx_n > b$
  - Element separation ***s*** and group diameter ***d*** – optional



SELECT Max(Temperature), Avg(Temperature) FROM SensorsReadings
GROUP BY Temperature **DELIMITED BY** (SELECT Value FROM Thresholds)

Similarity Group By Unsupervised

- Unsupervised – SGB-U
  - Element separation ***s*** – each object has another object within distance ***s***
  - Group diameter ***d*** – distance between the most separated objects ≤ ***d***




```
SELECT Max(Temperature), Avg(Temperature) FROM SensorsReadings
GROUP BY Temperature MAXIMUM_ELEMENT_SEPARATION 6
                     MAXIMUM_GROUP_DIAMETER 20
```

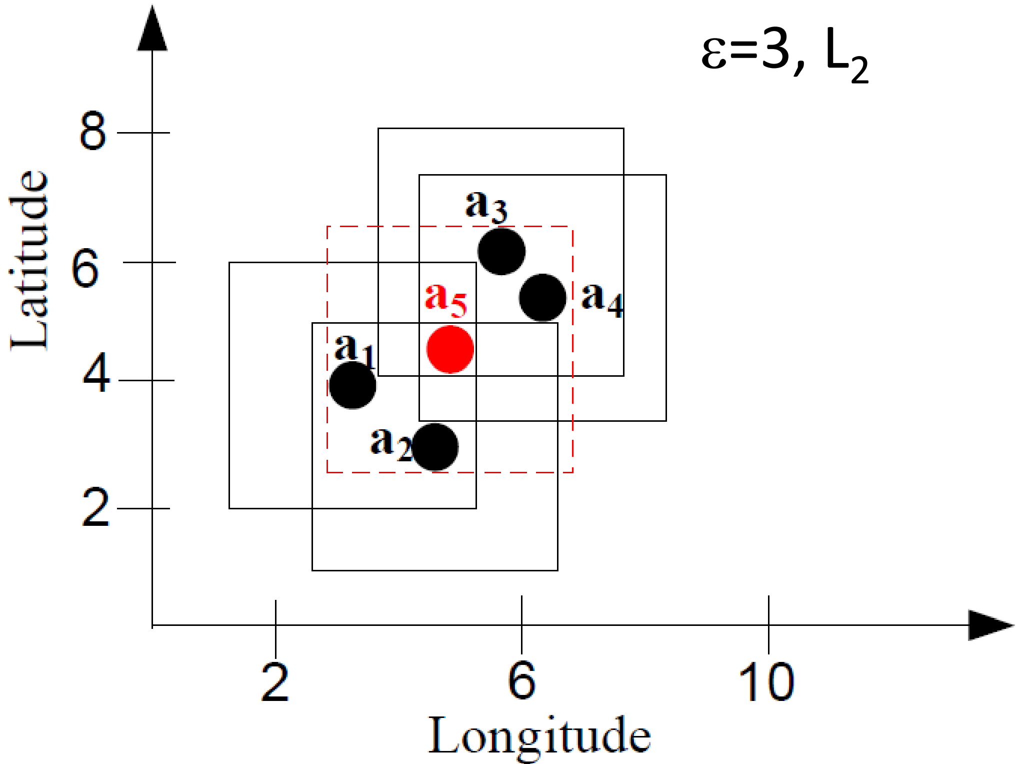
Similarity Group By Unsupervised

- Unsupervised SGB extended to vector spaces
  - Element separation ***s*** – each object has another object within distance ***s***
  - Distance function ***d*** – L_2 or L_∞, but not limited to

  - SGB-All
  - SGB-Any

Similarity Group By: Vector Spaces

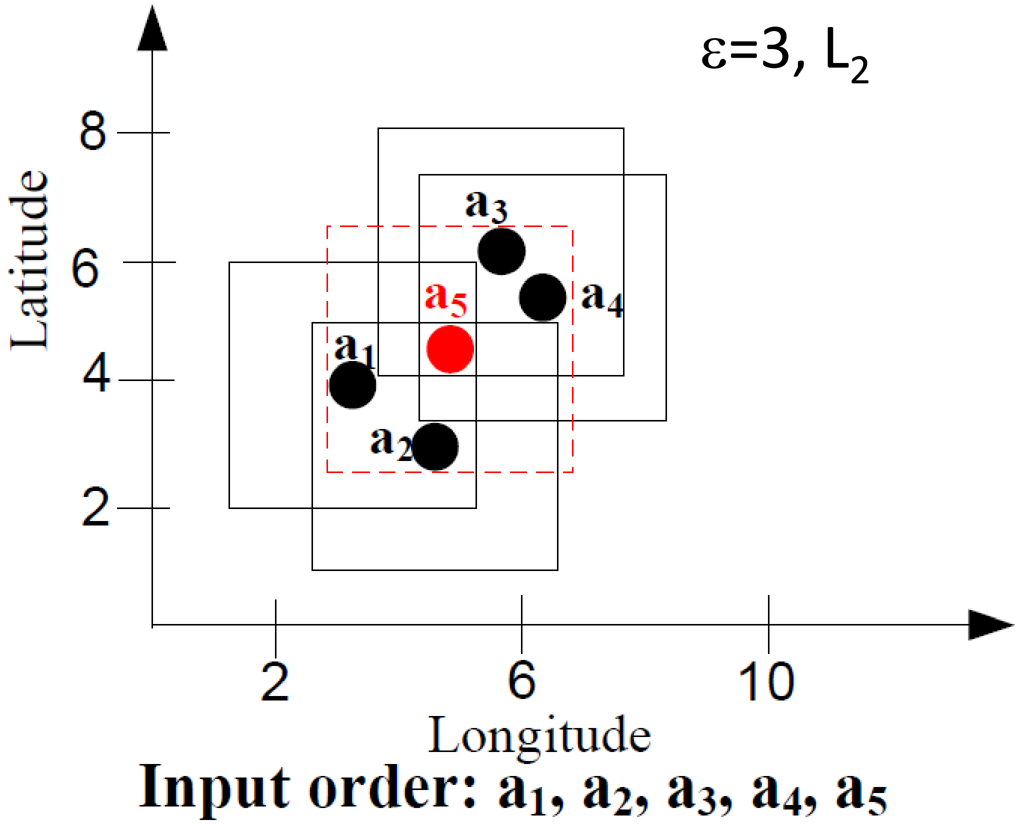
- SGB-All
  - An object o is in the group if its distance to **all** objects in the group is ≤ ***s*** (ε)
  - An object may be part of more groups, so the *on-overlap* functions:
    - **Join-any** – a group out of the matching groups is picked at random
    - **Eliminate** – the object is eliminated from grouping
    - **New-group** – all objects overlapping more groups form <u>a new group</u>
  - Algorithm $\mathcal{O}(n \log|G|)$ for vector spaces



Input order: a_1, a_2, a_3, a_4, a_5

Similarity Group By: Vector Spaces

- SGB-Any
  - An object o is in the group if its distance to **at least one** object in the group is ≤ ***s*** (ε)
  - All objects can be assigned uniquely
  - Algorithm $\mathcal{O}(n \log n)$ for vector spaces



Summary: SGB for 1D-nD spaces

- Efficient algorithms
  - comparable to regular Group-By, faster than clustering
- Group representatives
  - SGB-Around
    - Given by default
  - SGB-Delimited
    - Order on boundaries (1D)
  - SGB-U/All/Any
    - Centroid of group
    - Regular aggregations
      - On a component of all points in a group (min/max/avg) – 1D
      - On points (polygon/convex hull) – nD

Similarity Group By: Distance Spaces

- Parameters:
  - Element separation ***s*** – each object has another object within distance ***s***
  - Group diameter $\boldsymbol{2\varepsilon}$ – distance between the most separated objects ≤ $\boldsymbol{2\varepsilon}$
- Variants:
  - SGB-Around
    - Voronoi partitioning using pivots
    - Element separation ***s***, Group diameter $\boldsymbol{2\varepsilon}$
      - *Some objects might not be assigned to any group*
      - *Some objects might belong to more groups*
  - SGB-Delimited
    - A set of hyperplanes defined by pivots $\langle p_1, p_2 \rangle$; similar to Voronoi partitioning
    - Any practical application?

Similarity Group By: Distance Spaces

- Variants:
  - SGB-U
    - SGB-All → parameters $\boldsymbol{s}=\varepsilon$, diameter=$\varepsilon$
    - SGB-Any → parameters $\boldsymbol{s}=\varepsilon$, diameter=$\infty$
  - SGB-P – permutation
    - Group is formed by objects having the same k-nearest pivots
      - Recursive application of SGB-Around
    - Parameters *separation* and *diameter* not applicable
    - Alternative: …. k-furthest pivots?
  - SGB-S – subset of pivots
    - “permutation” without ordering

Similarity Group By: Distance Spaces

- SGB conflict handling when parameters (separation, diameter) given
  - on overlap
    - ELIMINATE
    - ASSIGN_TO_ANY - might not be random, due to consistency
    - ASSIGN_TO_ALL
    - FORM_NEW_GROUP
    - SET_NULL
  - on miss
    - ELIMINATE
    - FORM_NEW_GROUP
    - SET_NULL
- Handle missing descriptors → "NULL" group
  - e.g. when more SGB attributes are combined

Similarity Group By: Distance Spaces

- SGB with roll-up/drill-down on parameters
  - Gradually extend/shrink separation/diameter
    - Multiple assignment necessary
- SGB with roll-up/drill-down/dice on attributes
- Group representatives
  - Cluster medoid
  - Surrogate key
  - Result of an aggregate function

Similarity Group By: Distance Spaces

- Aggregate functions:
  - COUNT
  - MIN
    - The object closest to the group representative
  - MAX
    - Furthest object from group representative
  - AVG
    - Artificial object (center)?
  - MEDOID
  - SUM
    - A concatenation of all objects?
  - SET
    - A bag of all original objects
  - COVER
    - a set of boundary objects
    - “skyline” objects (or “convex hull”)

Future Work

- Combination of multiple predicates
  - Optimized algorithms
- Similarity group by variants defined
    - Introduce parameter refinement – "drill-down" feature
    - Incorporate *diversity* in result
  - Need for new aggregate functions
  - Efficient algorithms for metric spaces
    - Many existing rely on sorting the attribute values
- New query types
  - "Multi-query" kNN
  - Recursive self-join