Relaxed Queries over Data Streams

Barbara Catania Giovanna Guerrini
Maria Teresa Pinto Paola Podestà

[1] Department of Computer and Information Science
University of Genoa, Italy

SEBD, 2012

Outline

The Context - Data

The Past:

- Data with completely known structure
- All data available before processing (stored data)

The Context - Data

The Past:

- Data with completely known structure
- All data available before processing (stored data)

The Present:

- Unknown and dynamic characteristics for data at runtime
- Data dynamically acquired and not persistent (data streams)
- Heterogeneous data

The Context - Data processing

The Past:

- Queries with hard constraints
- Precise answer
- Quite stable execution environments

The Context - Data processing

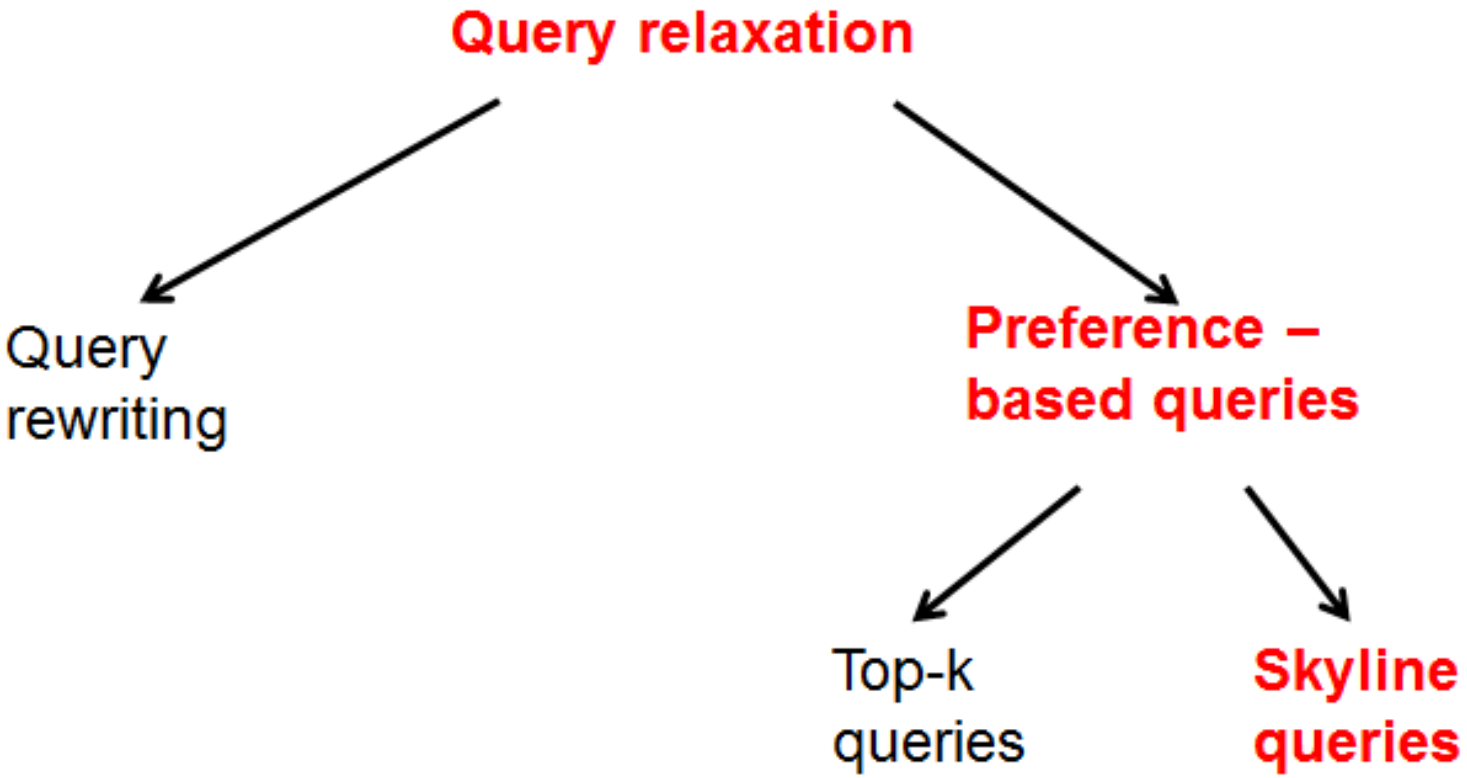
The Past:

- Queries with hard constraints
- Precise answer
- Quite stable execution environments

The Present:

- Hard constraints may lead to unsatisfactory answers (empty aswer problem, many aswers problem)
- Processing conditions (network load...) significantly vary over time
- Relaxed queries with soft constraints

The Context - Approximation



The Context - Approximation

Skyline queries:

- return the best results based on a set of relevance attributes *S* (user preferences) in term of a partial ordering among items
- A dominates B if it is better in at least one attribute and equal or better than B in all the others

Relaxation-skyline queries (r-skyline):

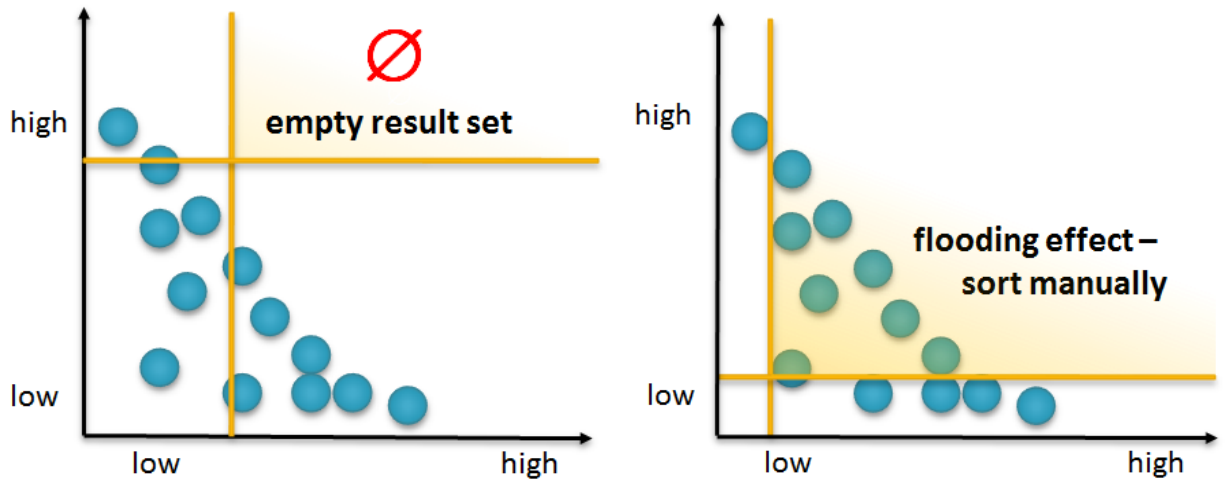
- relaxing function (system-defined preferences) to quantify the distance of each item from the specified query conditions
- rely on a skyline-based semantics to compute the results

	Stored data	**Data streams**
Skyline queries	Hot topic [Börzsönyi et.al, ICDE 2001]	[Tao, Papadias, TKDE 2006]
r-Skyline queries	[Koudas et.al, VLDB 2006]	few proposal

Outline

Focus

- **Main topic:** solutions to the empty answer problem / many answers problem for data streams



- **Here:** preliminary approach for solving the empty answer problem over data streams based on r-skyline

Outline

Contributions

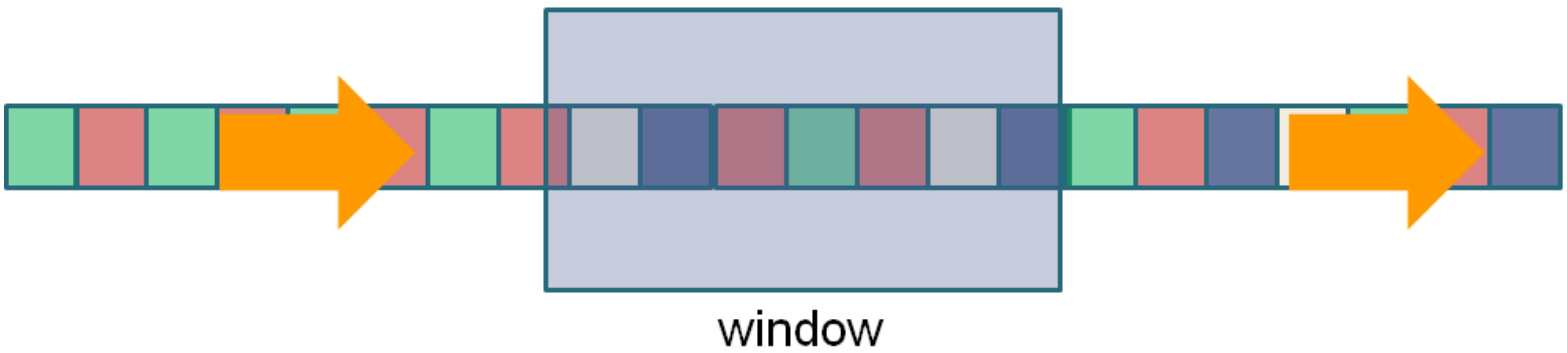
1. Definition of relaxation skyline (r-skyline) for window-based join queries over data streams
2. Processing algorithm for r-skyline queries for window-based join queries over data streams
3. Preliminary experimental evaluation

Contributions - Background - **Data streams**

- **Data stream**: a stream is a continuous and potentially unbounded, real-time, sequence of data elements (e.g., tuples)
- **Window operators**: the way to ”bound” streams
  - count-based
  - time-based

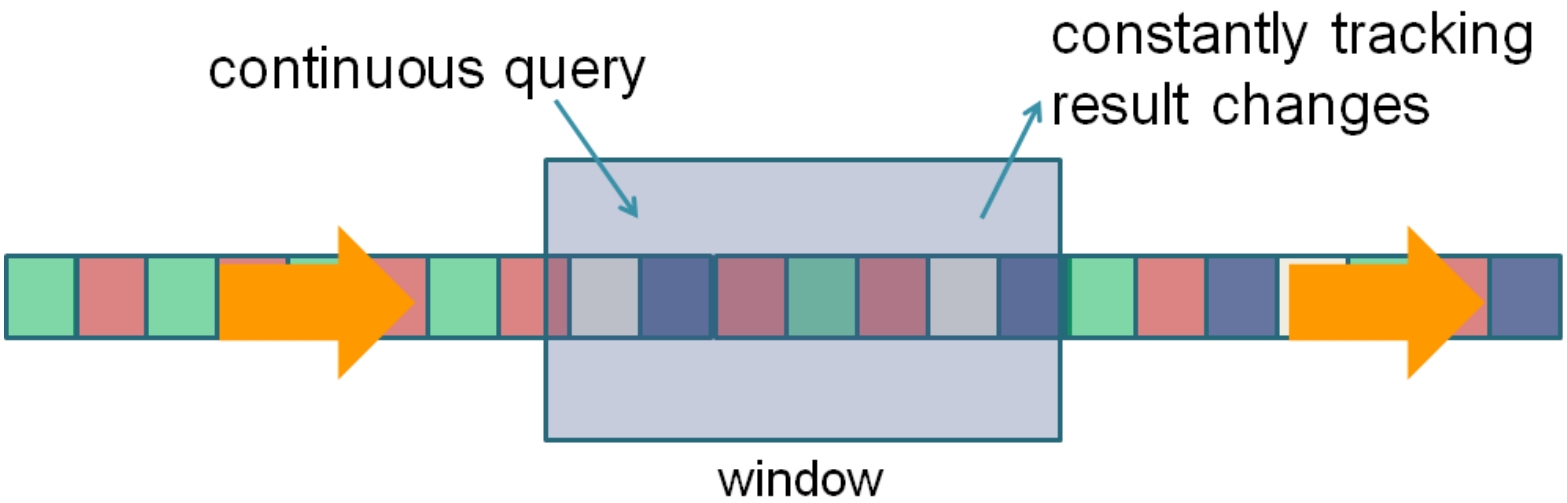
Contributions - Background - **Data streams**

- **Data stream**: a stream is a continuous and potentially unbounded, real-time, sequence of data elements (e.g., tuples)
- **Window operators**: the way to ”bound” streams
  - count-based
  - time-based



Contributions - Background - **Data streams**

- **Data stream**: a stream is a continuous and potentially unbounded, real-time, sequence of data elements (e.g., tuples)
- **Window operators**: the way to ”bound” streams
  - count-based
  - time-based

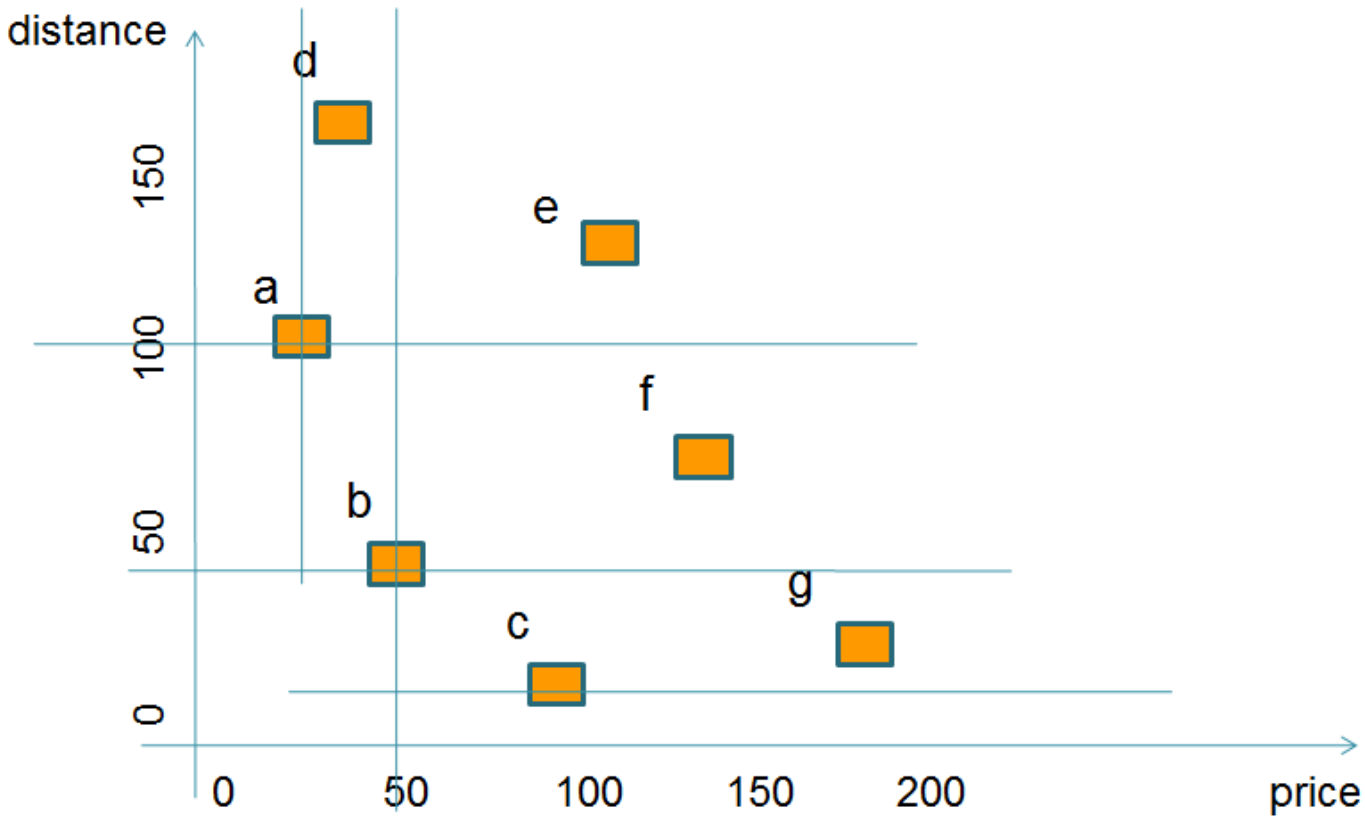


Contributions - Background - **Skyline queries** (Börzsönyi et al., 2001)

- Preferences in terms of a set of relevant attributes S
- Return the best results based on S, independently on any scoring function
- Background concepts:
  - **Dominance:** *A* dominates a point *B* if it is better in at least one attribute and equal or better in all the others, with respect to some ordering
  - **Skyline:** All items that are not dominated by any other item

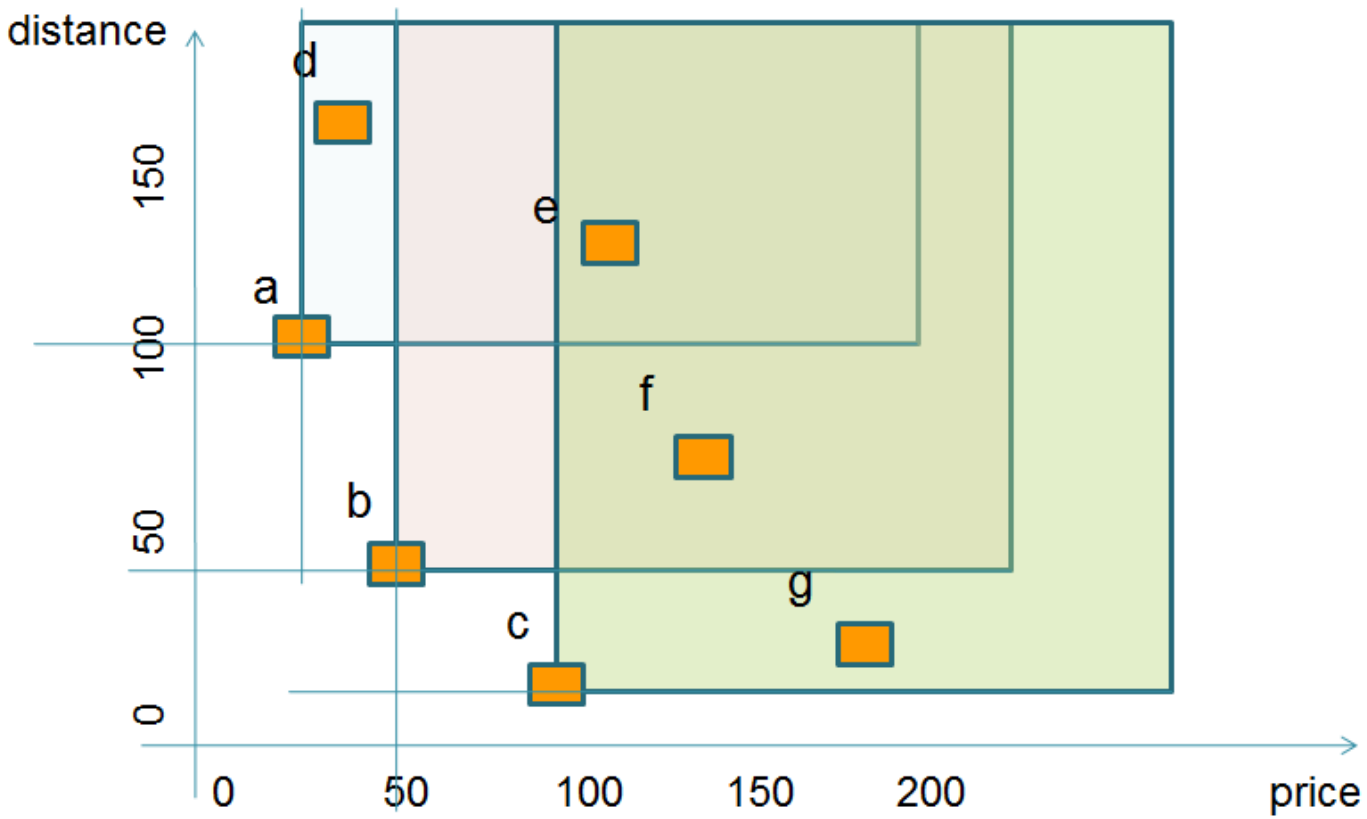
Contributions - Background -
Skyline queries (Börzsönyi et al., 2001)

SELECT *
FROM BBVenice BBV
SKYLINE OF a.price MIN, a.distancefromArtigianelli MIN



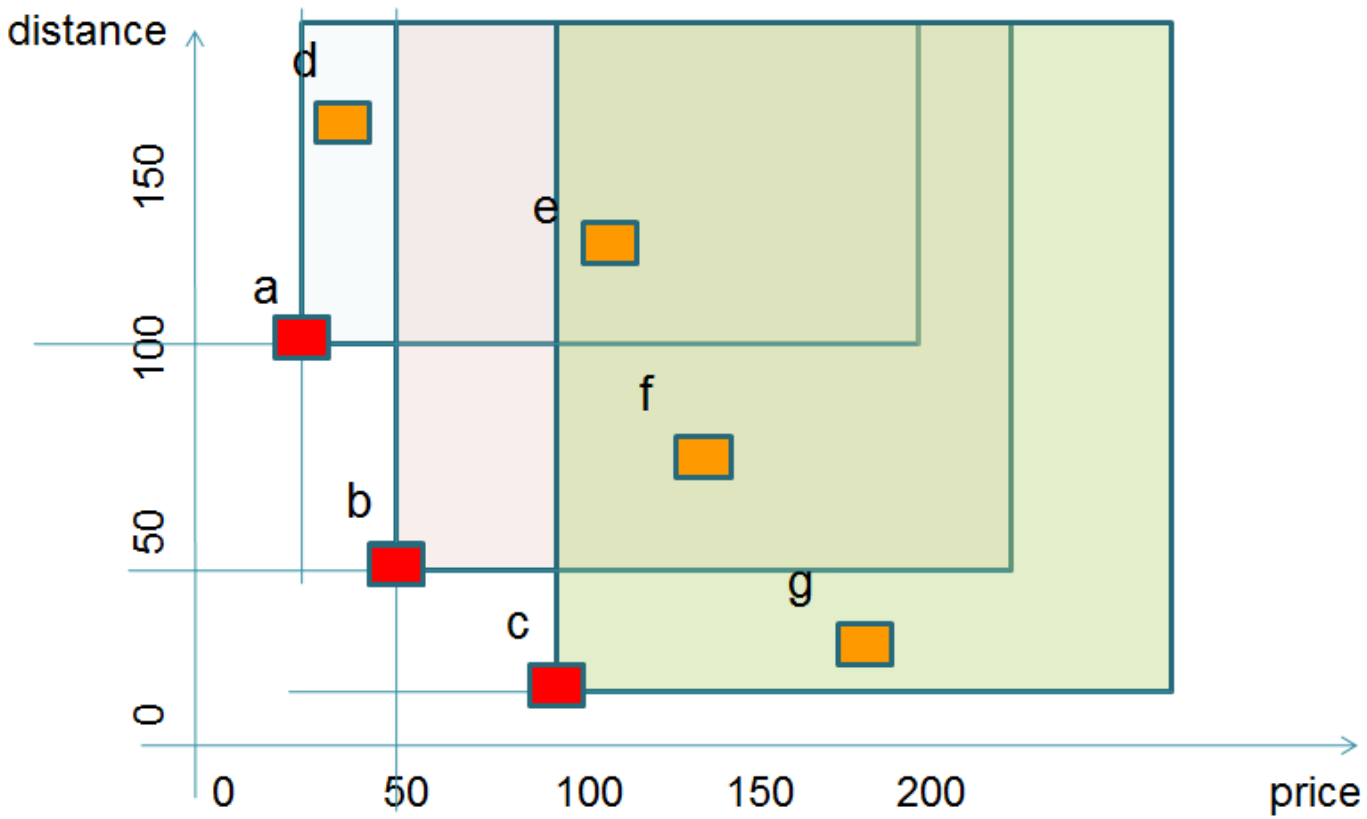
Contributions - Background -
Skyline queries (Börzsönyi et al., 2001)

SELECT *
FROM BBVenice BBV
SKYLINE OF a.price MIN, a.distancefromArtigianelli MIN



Contributions - Background -
Skyline queries (Börzsönyi et al., 2001)

SELECT *
FROM BBVenice BBV
SKYLINE OF a.price MIN, a.distancefromArtigianelli MIN



Contributions - Background -
R-skyline queries [Koudas et.al, VLDB 2006]

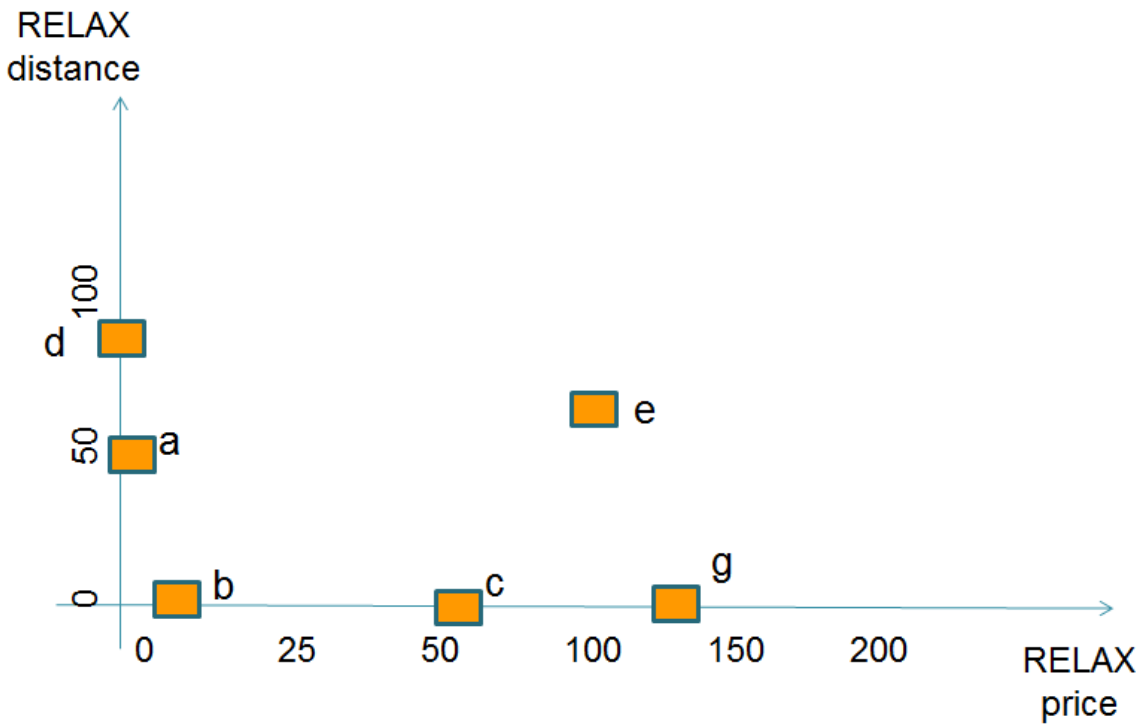
- Extends the concept of skyline queries to deal with derived attributes
- **Derived attribute** represents the distance of the considered tuple (pair of tuples, in case of join) to a condition contained in the query
- **RELAX function** for selection conditions, on tuple t and condition $C : t.A\theta v$:
  - RELAX(t,C)= 0 if t satisfies C
  - RELAX(t,C)= |t.A - v| otherwise

▸ Definitions

Contributions - Background -
R-skyline queries [Koudas et.al, VLDB 2006]

SELECT *
FROM BBVenice BBV
SKYLINE OF a.price <45, a.distancefromArtigianelli <=50

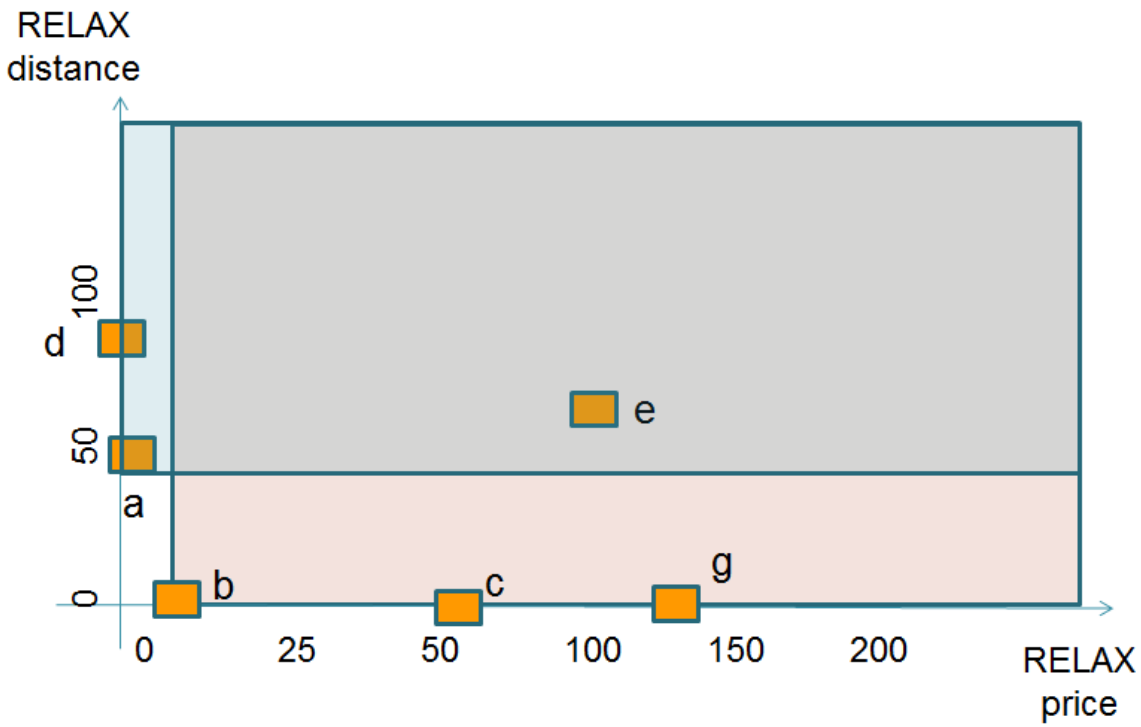
- a.price=30 a.distancefromArtigianelli=100
- RELAX(a, price<45)= 0
- RELAX(a, distancefromArtigianelli<50)= 50



Contributions - Background -
R-skyline queries [Koudas et.al, VLDB 2006]

SELECT *
FROM BBVenice BBV
SKYLINE OF a.price <45, a.distancefromArtigianelli <=50

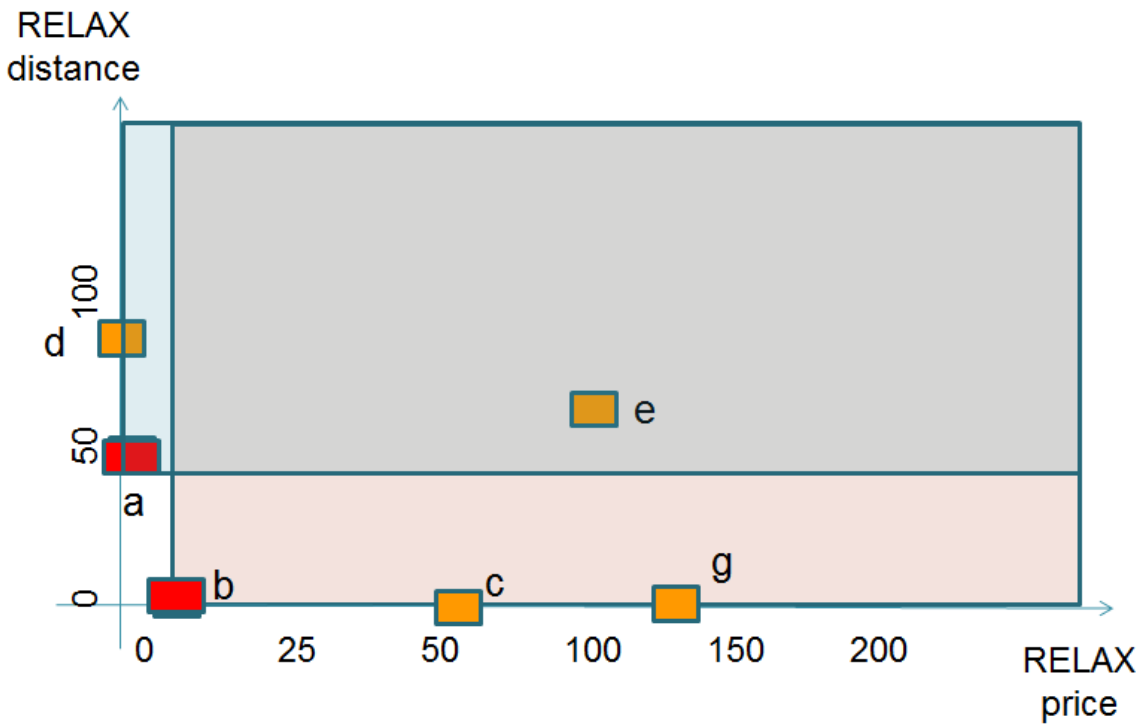
- a.price=30 a.distancefromArtigianelli=100
- RELAX(a, price<45)= 0
- RELAX(a, distancefromArtigianelli<50)= 50



Contributions - Background -
R-skyline queries [Koudas et.al, VLDB 2006]

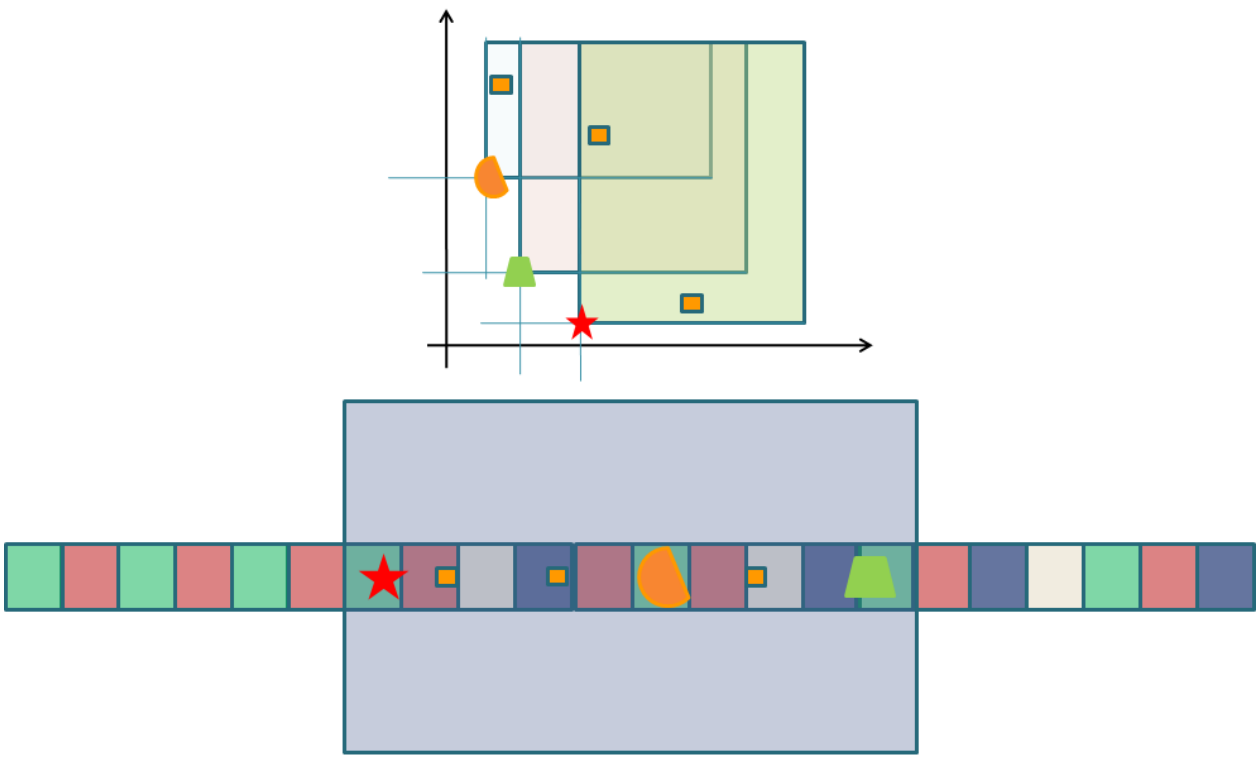
SELECT *
FROM BBVenice BBV
SKYLINE OF a.price <45, a.distancefromArtigianelli <=50

- a.price=30 a.distancefromArtigianelli=100
- RELAX(a, price<45)= 0
- RELAX(a, distancefromArtigianelli<50)= 50



Contributions - Background -
Skyline on Data Stream [Tao, Papadias, TKDE 2006]

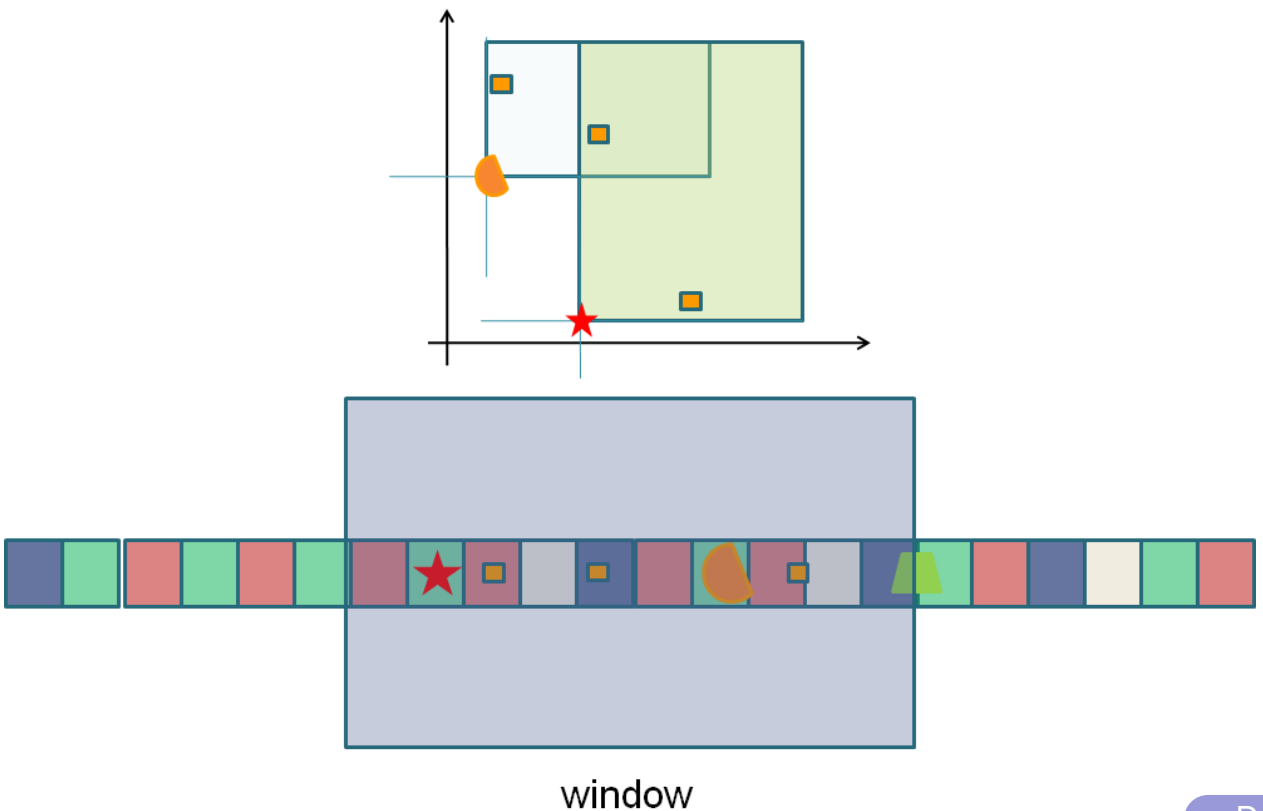
- continously update skyline over the data alive in window W
- r lifespan is $[r.t_{arr}, r.t_{exp})$ where $r.t_{exp} = r.t_{arr} + W$



▸ Definitions

Contributions - Background -
Skyline on Data Stream [Tao, Papadias, TKDE 2006]

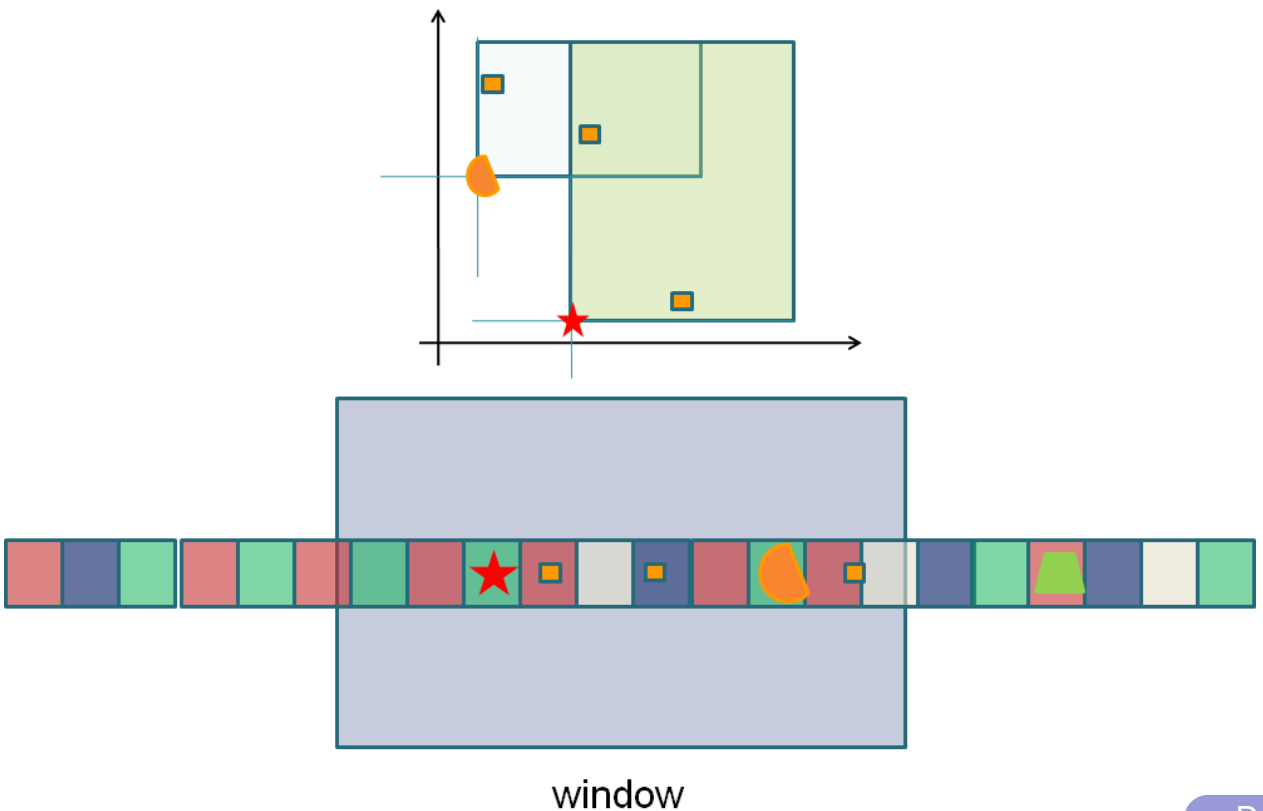
- continously update skyline over the data alive in window W
- r lifespan is $[r.t_{arr}, r.t_{exp})$ where $r.t_{exp} = r.t_{arr} + W$



▸ Definitions

Contributions - Background -
Skyline on Data Stream [Tao, Papadias, TKDE 2006]

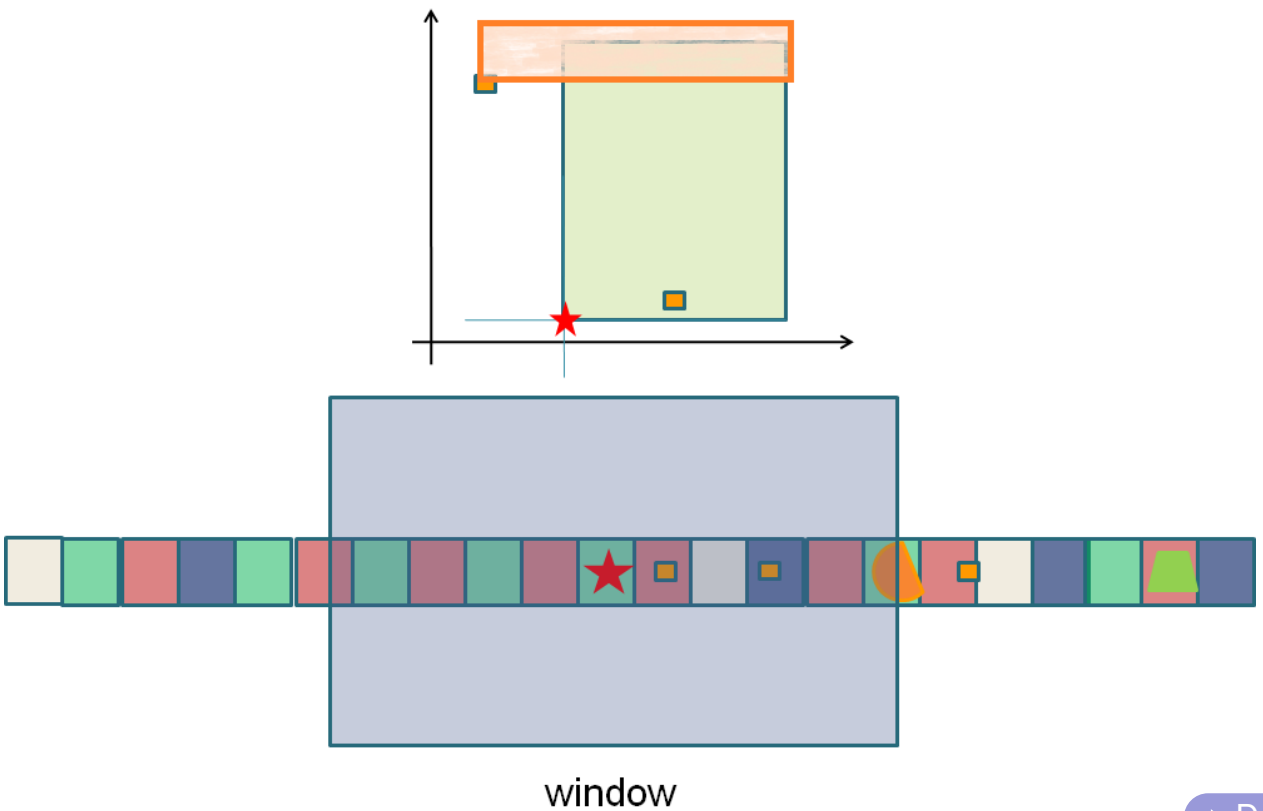
- continously update skyline over the data alive in window W
- r lifespan is $[r.t_{arr}, r.t_{exp})$ where $r.t_{exp} = r.t_{arr} + W$



▸ Definitions

Contributions - Background -
Skyline on Data Stream [Tao, Papadias, TKDE 2006]

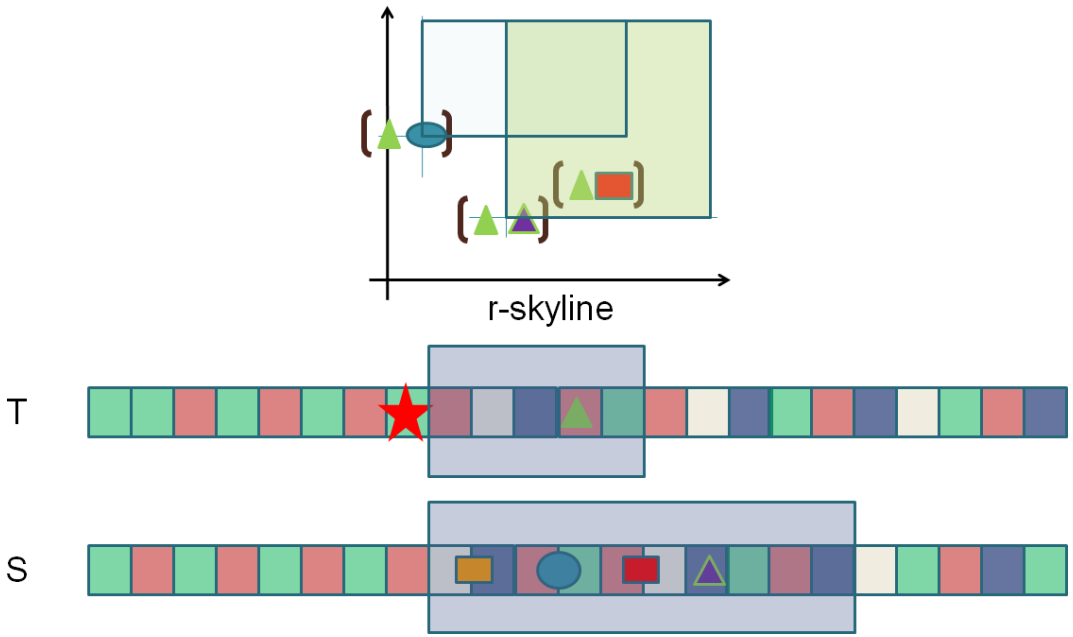
- continously update skyline over the data alive in window W
- r lifespan is $[r.t_{arr}, r.t_{exp})$ where $r.t_{exp} = r.t_{arr} + W$



▸ Definitions

Contributions - **R-skyline window-based join on data stream**

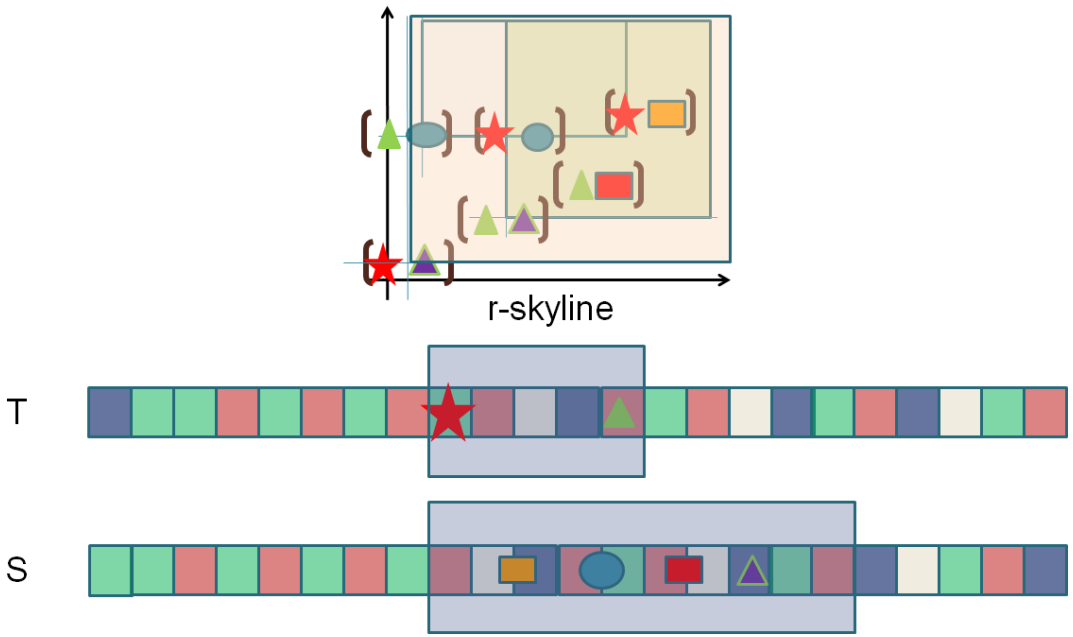
- $\sigma_{C_S}(S \bowtie_{C_J} T)$: precise join condition (C_J) and relaxation only on selection conditions (C_S)
- two data streams S, T with two windows W_S and W_T
- r-skyline
- r=(s,t) lifespan is $r.t_{arr} = max\{s.t_{arr}, r.t_{arr}\}$ $r.t_{exp} = min\{s.t_{exp}, r.t_{exp}\}$



Definitions

Contributions - **R-skyline window-based join on data stream**

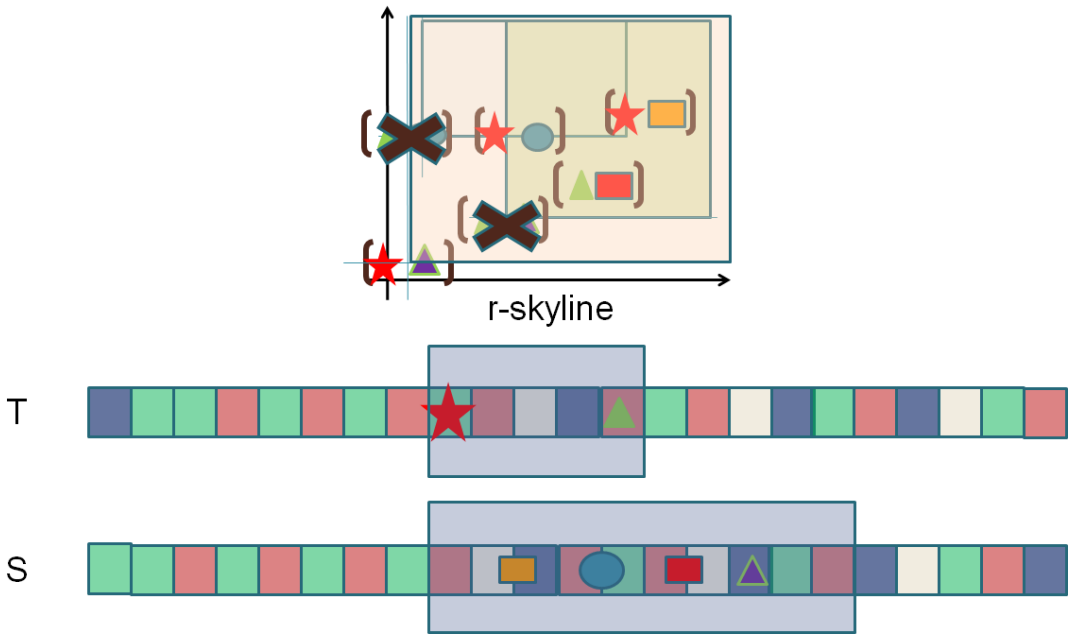
- $\sigma_{C_S}(S \bowtie_{C_J} T)$: precise join condition (C_J) and relaxation only on selection conditions (C_S)
- two data streams S, T with two windows W_S and W_T
- r-skyline
- r=(s,t) lifespan is $r.t_{arr} = max\{s.t_{arr}, r.t_{arr}\}$ $r.t_{exp} = min\{s.t_{exp}, r.t_{exp}\}$



Definitions

Contributions - **R-skyline window-based join on data stream**

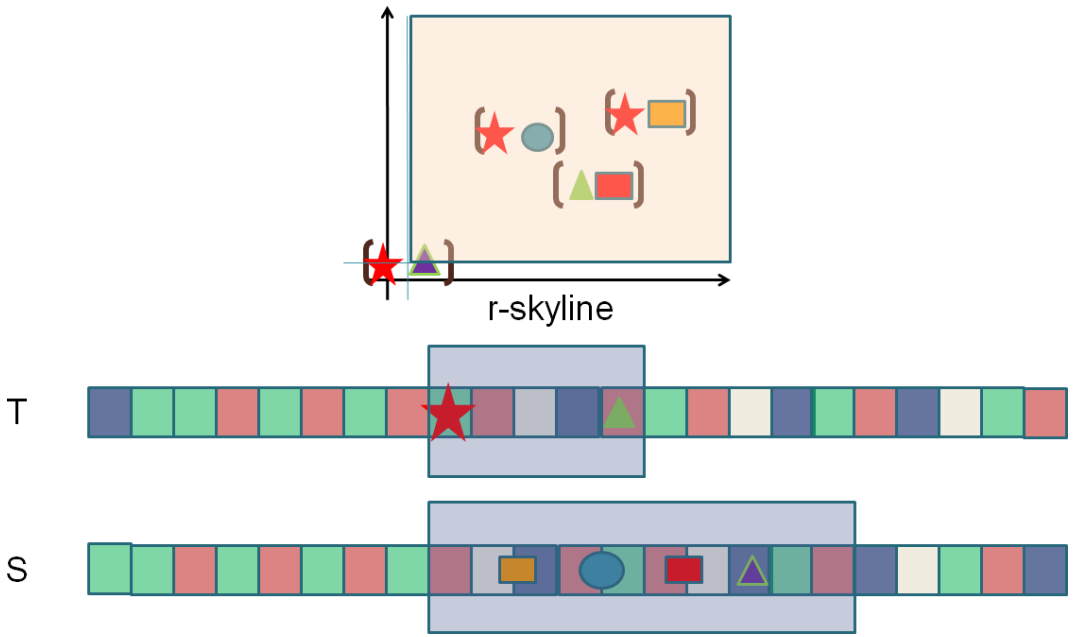
- $\sigma_{C_S}(S \bowtie_{C_J} T)$: precise join condition (C_J) and relaxation only on selection conditions (C_S)
- two data streams S, T with two windows W_S and W_T
- r-skyline
- r=(s,t) lifespan is $r.t_{arr} = max\{s.t_{arr}, r.t_{arr}\}$ $r.t_{exp} = min\{s.t_{exp}, r.t_{exp}\}$



Definitions

Contributions - **R-skyline window-based join on data stream**

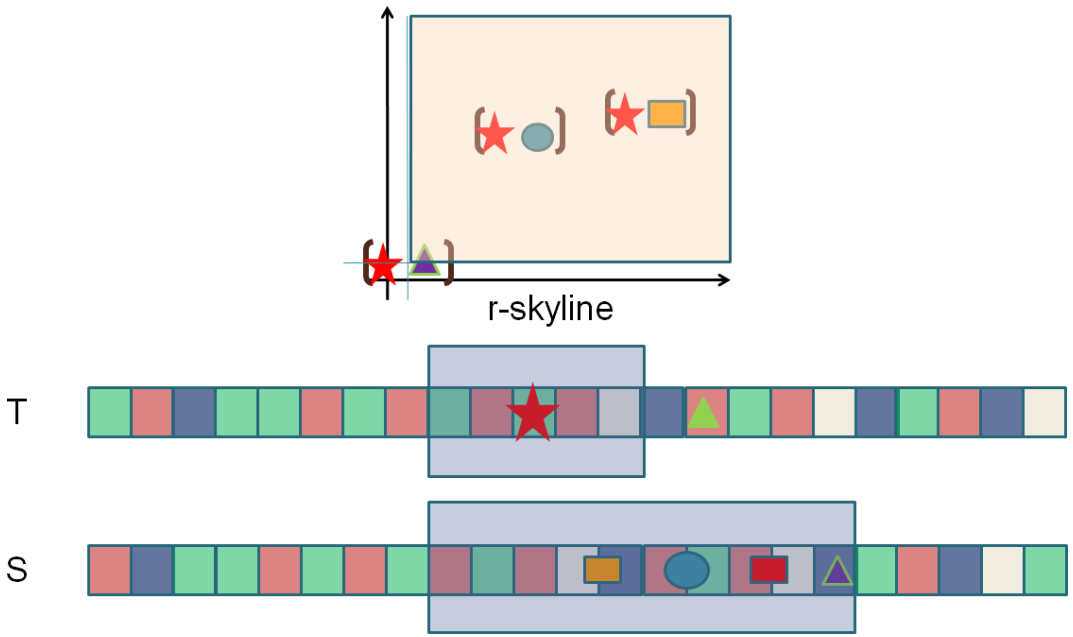
- $\sigma_{C_S}(S \bowtie_{C_J} T)$: precise join condition (C_J) and relaxation only on selection conditions (C_S)
- two data streams S, T with two windows W_S and W_T
- r-skyline
- r=(s,t) lifespan is $r.t_{arr} = max\{s.t_{arr}, r.t_{arr}\}$ $r.t_{exp} = min\{s.t_{exp}, r.t_{exp}\}$



Definitions

Contributions - **R-skyline window-based join on data stream**

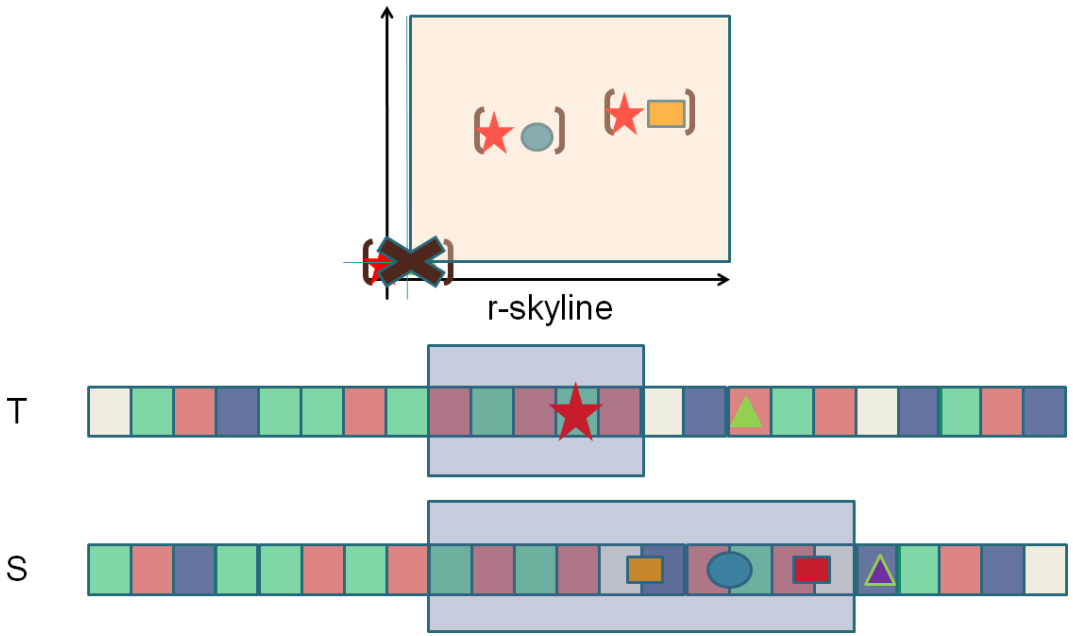
- $\sigma_{C_S}(S \bowtie_{C_J} T)$: precise join condition (C_J) and relaxation only on selection conditions (C_S)
- two data streams S, T with two windows W_S and W_T
- r-skyline
- r=(s,t) lifespan is $r.t_{arr} = max\{s.t_{arr}, r.t_{arr}\}$ $r.t_{exp} = min\{s.t_{exp}, r.t_{exp}\}$



Definitions

Contributions - **R-skyline window-based join on data stream**

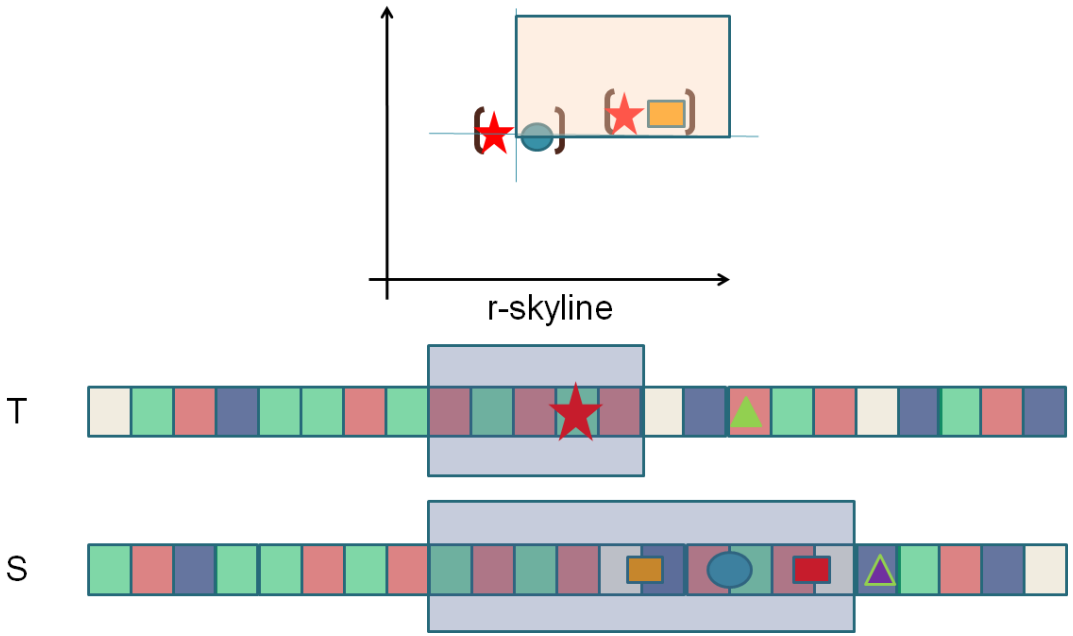
- $\sigma_{C_S}(S \bowtie_{C_J} T)$: precise join condition (C_J) and relaxation only on selection conditions (C_S)
- two data streams S, T with two windows W_S and W_T
- r-skyline
- r=(s,t) lifespan is $r.t_{arr} = max\{s.t_{arr}, r.t_{arr}\}$ $r.t_{exp} = min\{s.t_{exp}, r.t_{exp}\}$



Definitions

Contributions - **R-skyline window-based join on data stream**

- $\sigma_{C_S}(S \bowtie_{C_J} T)$: precise join condition (C_J) and relaxation only on selection conditions (C_S)
- two data streams S, T with two windows W_S and W_T
- r-skyline
- r=(s,t) lifespan is $r.t_{arr} = max\{s.t_{arr}, r.t_{arr}\}$ $r.t_{exp} = min\{s.t_{exp}, r.t_{exp}\}$



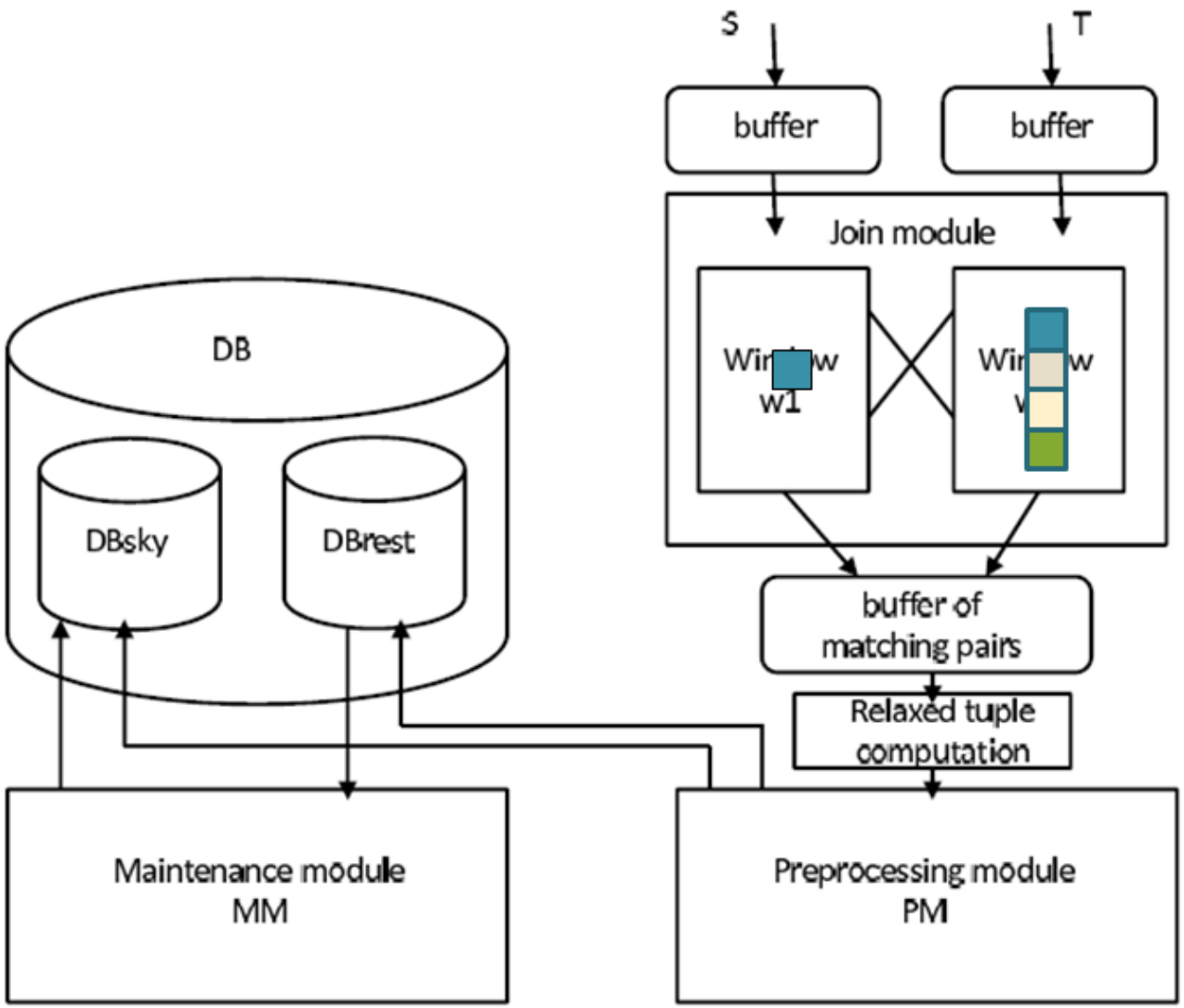
Definitions

Contributions - **NRJL Algorithm**

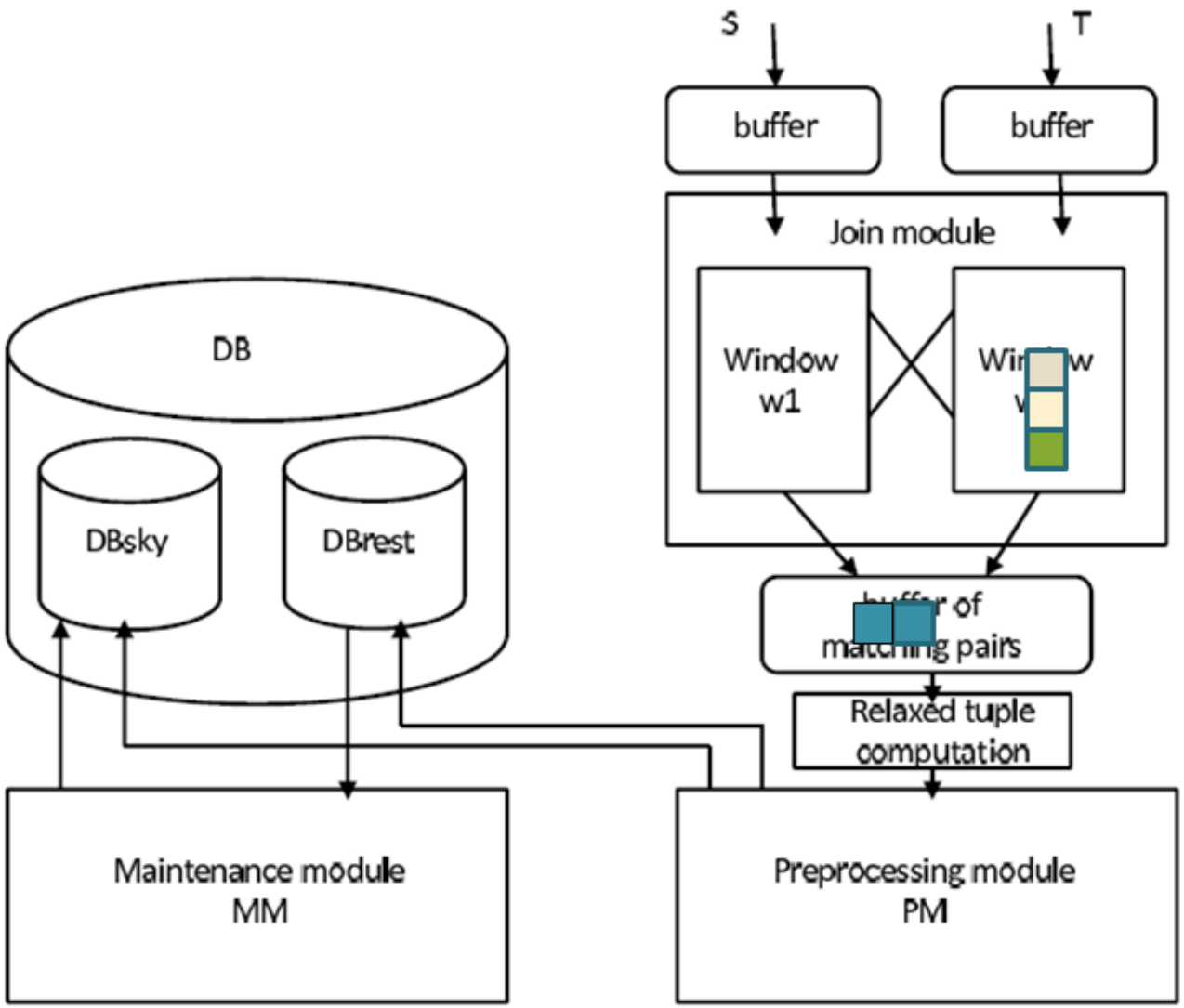
Non Relaxed Join Lazy (NRJL) is obtained by merging:

- The **Pruning Join** algorithm presented in [Koudas et.al, VLDB 2006]
  - stored relational data
  - relaxation on both selection and join conditions
- The **Lazy method** presented in [Tao, Papadias, TKDE 2006]
  - single data stream
  - different computation of tuple life span
  - do not relax query conditions

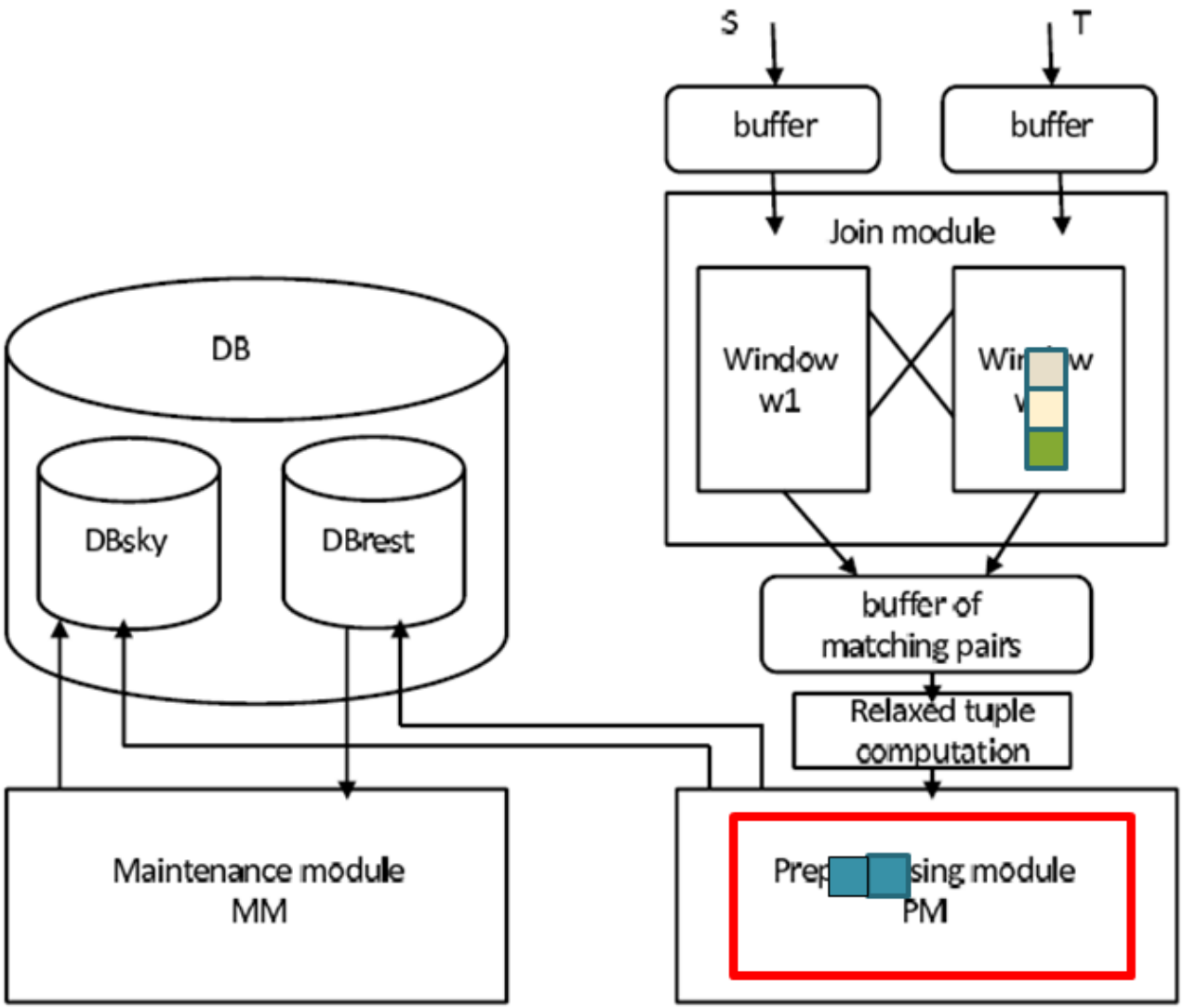
Contributions - **NRJL Algorithm**



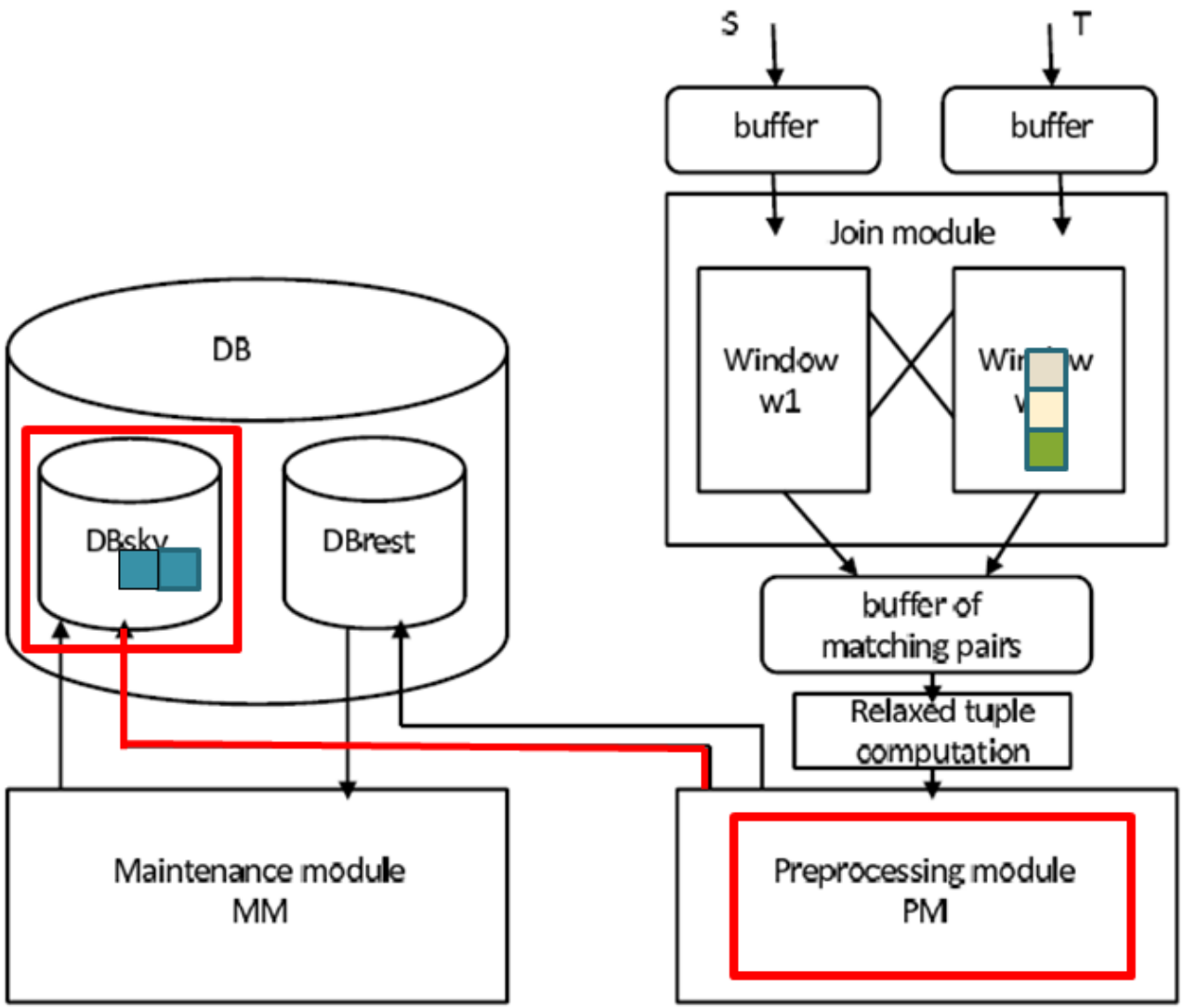
Contributions - **NRJL Algorithm**



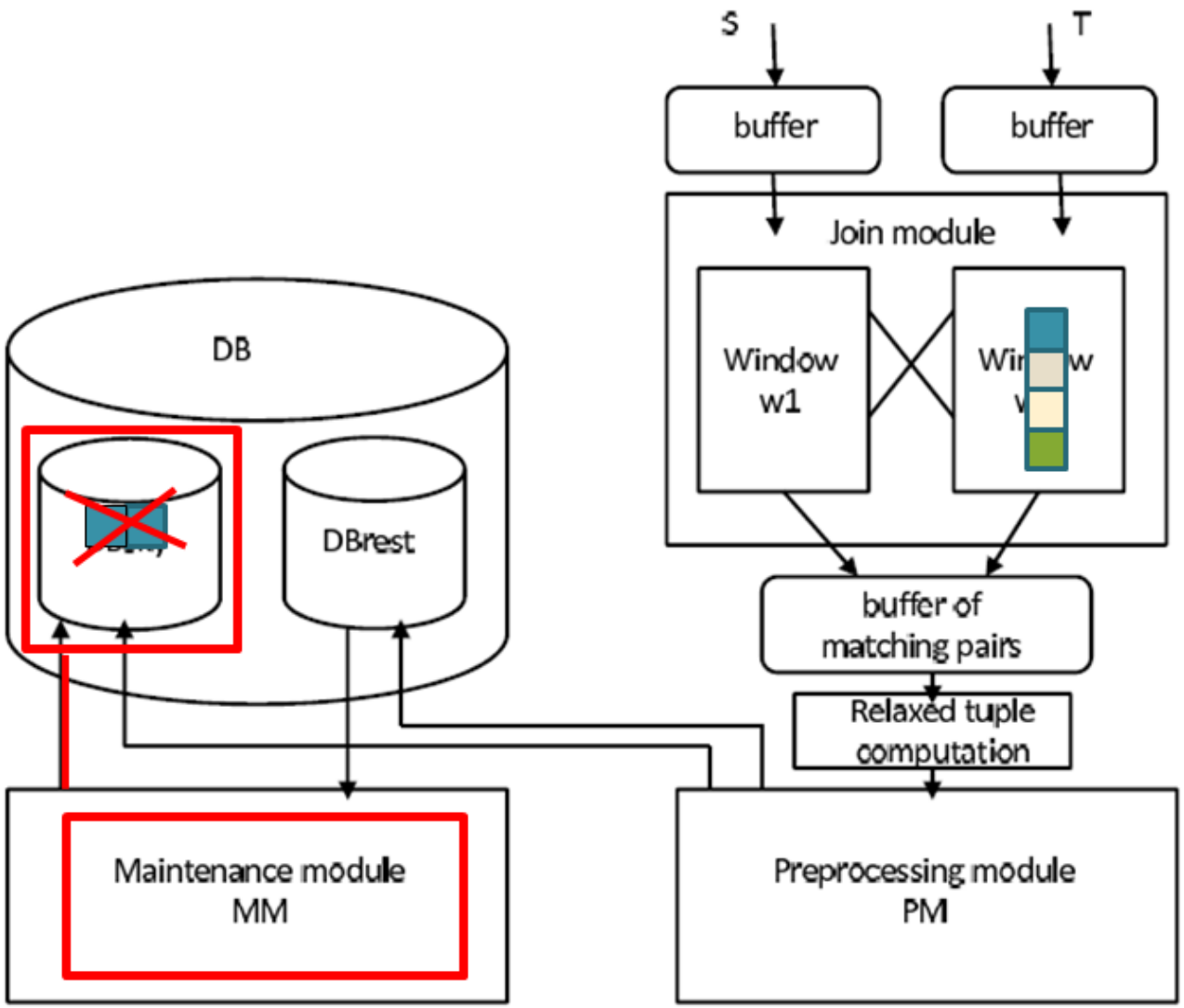
Contributions - **NRJL Algorithm**



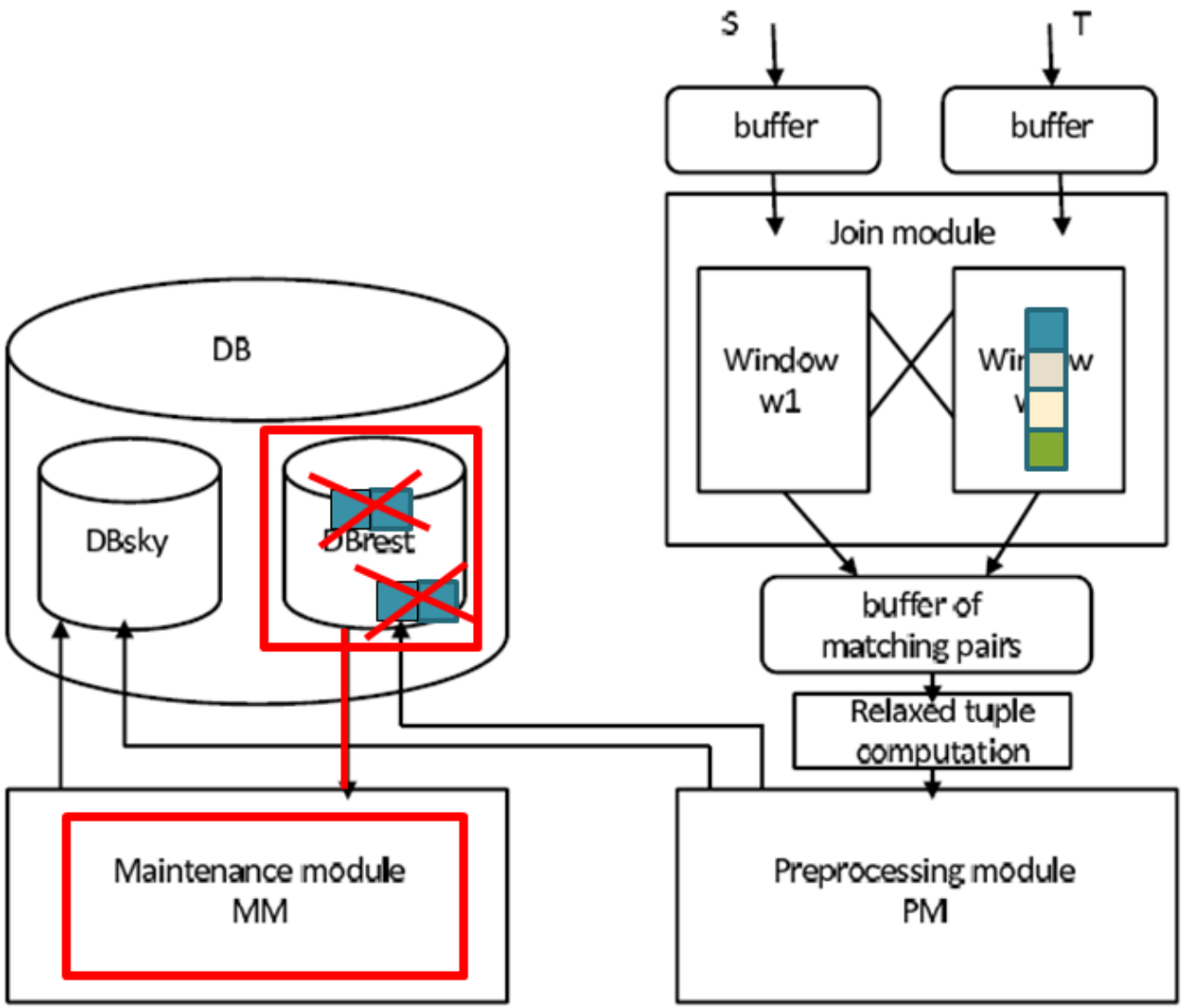
Contributions - **NRJL Algorithm**



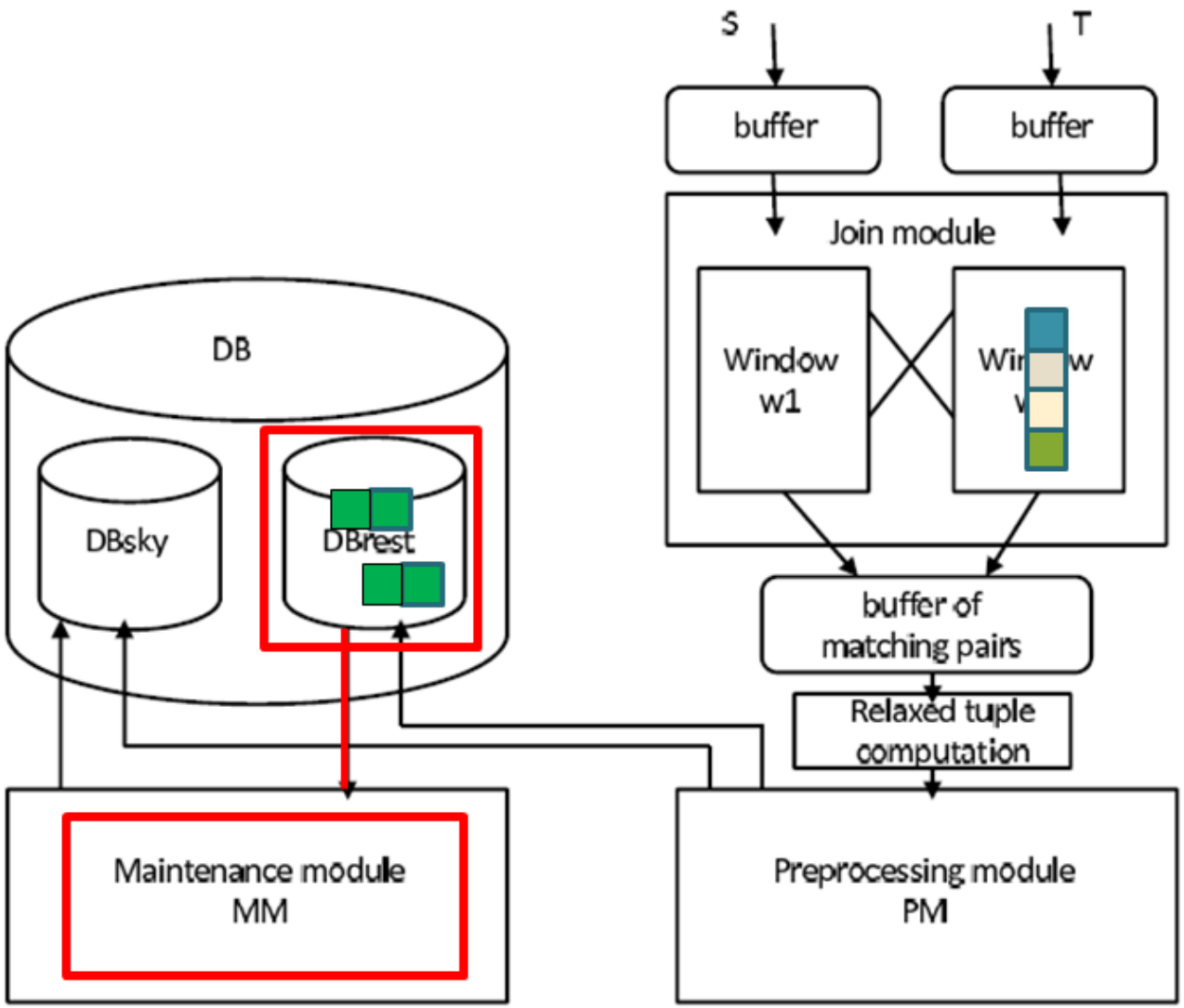
Contributions - **NRJL Algorithm**



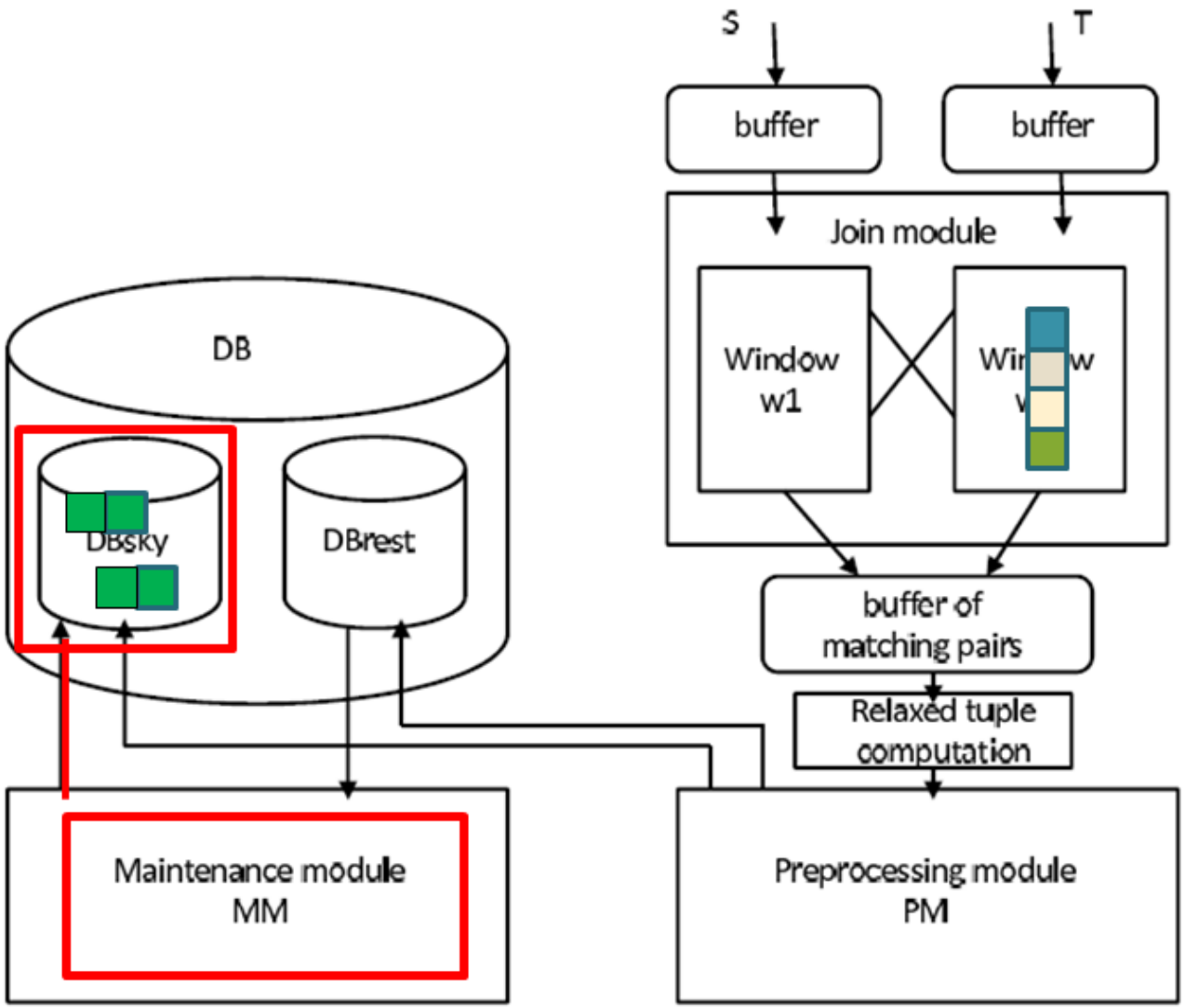
Contributions - **NRJL Algorithm**



Contributions - **NRJL Algorithm**



Contributions - **NRJL Algorithm**



Contributions - **Experimental Evaluation**

Goal of the experimental evaluation:

- investigate the impact of **relaxation overhead** on performance, then we consider both precise and relaxed queries

- Considered parameters:
  - the average size of the r-skyline in each time instant
  - Processing time of individual tuples
  - Amortized processing time: (sum of processing times of individual tuples) / (number of processed tuples)

Contributions - **Experimental Evaluation**

- Two streams of 3D tuples, with uniform and anticorrelated distribution
- A_j, A_{s1}, $A_{s2} \in [0,1]$
- Query:
  - Join condition on A_j
  - Selection conditions on A_{s1}, A_{s2} like $A_{sj} = 0$
- Window size: 400, 800, 1600, 3200 and 6400
- 30 windows for each stream
- Alternate arrival between the two streams every 5 seconds

Contributions - **Experimental Evaluation**

R-skyline size

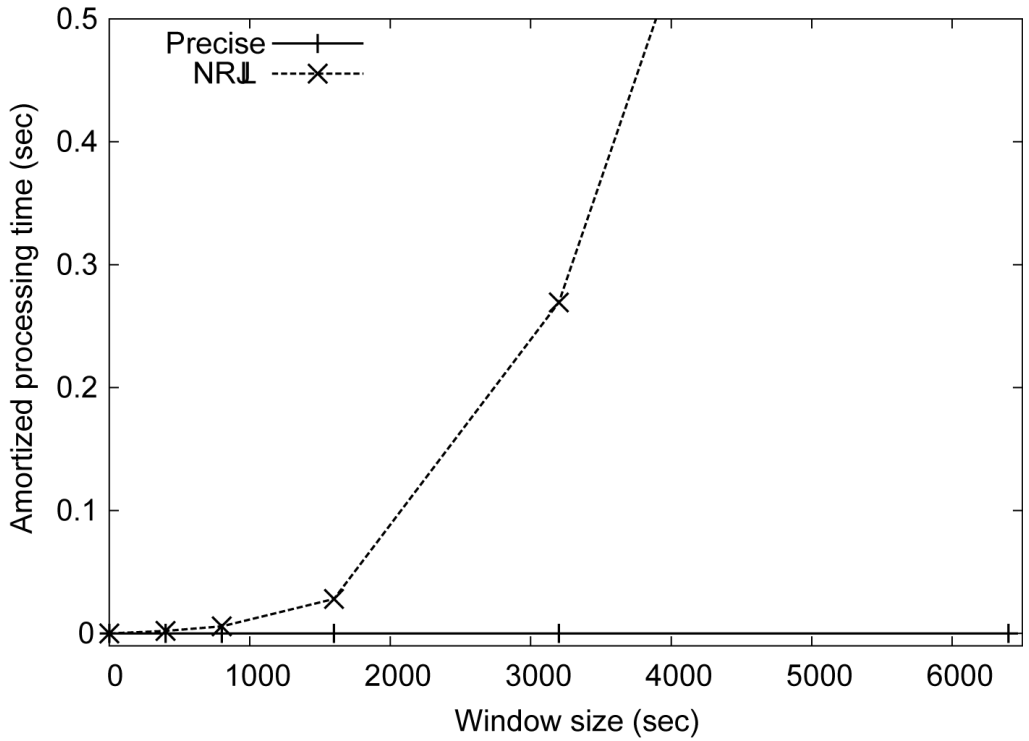
Query	W	400	800	1600	3200	6400
Q_2	**uniform**	2	2	3	3	7
Q_2	**anticorrelated**	2	7	11	11	10
Q_4	**uniform**	7	7	12	10	8
Q_4	**anticorrelated**	16	19	21	21	22

Table: DB_{sky} average sizes corresponding to varying window sizes

- The number of skyline tuples for the uniform distribution is always less than or equal to that for the anticorrelated distribution
- The size grows in the number of relaxed conditions

Contributions - **Experimental Evaluation**

Amortized processing time: relaxed vs precise execution

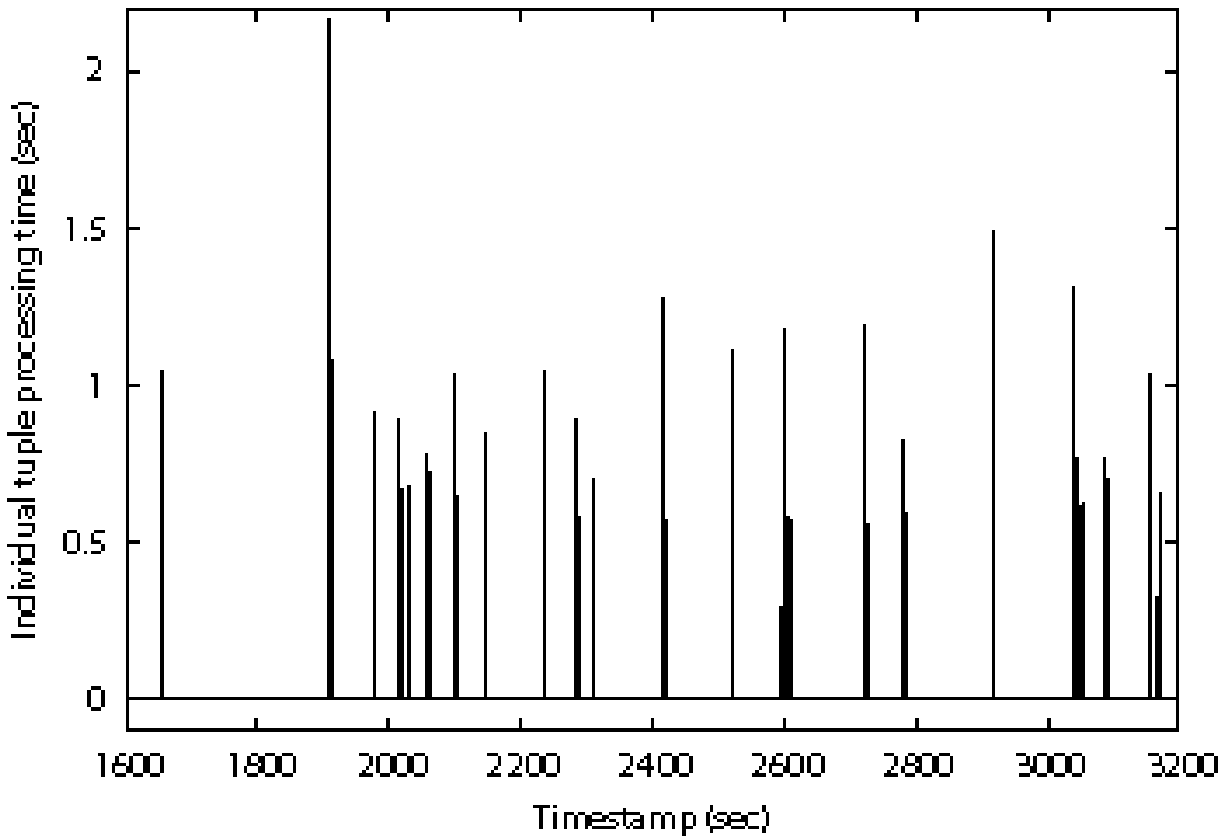


As expected relaxing queries penalizes performance, since in precise queries:

- No relaxation is applied on tuples
- No auxiliary data structures need to be maintained

Contributions - Experimental Evaluation

Individual tuple processing time



- All time are not null (scale problem)
- Peaks represent activation of the MM module

Outline

Conclusions

Current work:

- Extension of r-skyline queries to data streams
- Processing algorithm has been provided
- Preliminary experimental results has been provided

Conclusions

Current work:

- Extension of r-skyline queries to data streams
- Processing algorithm has been provided
- Preliminary experimental results has been provided

Future work:

- Relaxed queries performance are worst than precise queries (as expected) ...How to improve performance?

Conclusions - Ongoing and Future Work

Implementative issues:

- Anticipate some skyline computation during window-based join execution [Catania, Guerrini, Pinto, Podesta', ADBIS 2012]
- Reduce the size of the maintained state information through index usage for DB_{rest} and DB_{sky} (R-tree or grid) [Catania, Guerrini, Pinto, Podesta', ADBIS 2012]

Conclusions - Ongoing and Future Work

Implementative issues:

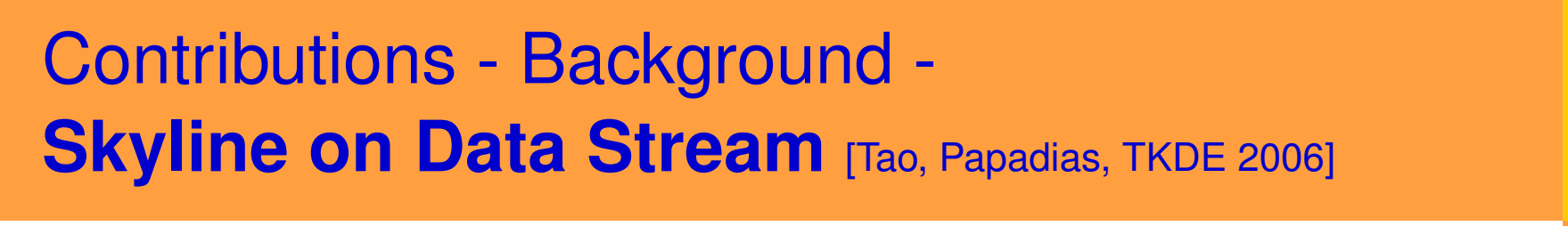
- Anticipate some skyline computation during window-based join execution [Catania, Guerrini, Pinto, Podesta', ADBIS 2012]
- Reduce the size of the maintained state information through index usage for DB_{rest} and DB_{sky} (R-tree or grid) [Catania, Guerrini, Pinto, Podesta', ADBIS 2012]

Theoretical issues:

- adaptive processing approach in order to switch from skyline-based execution to a precise ones (and vice versa) using QoD parameters

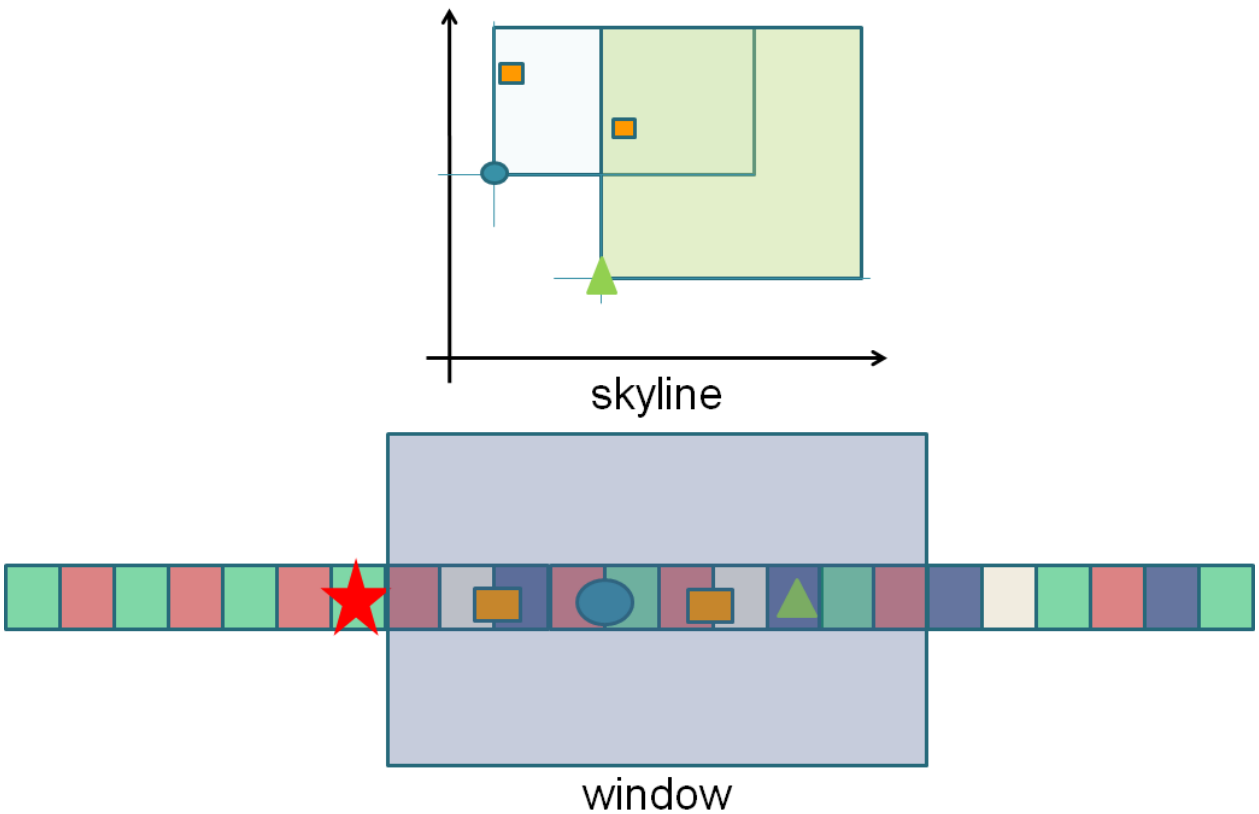
Outline

Contributions - Background -
Skyline on Data Stream [Tao, Papadias, TKDE 2006]



- a new tuple r' arrives
- an existing skyline tuple r' expires.

$$T_{exp}^{sky} = min\{r.t_{exp} | t \in SKYLINE\}$$

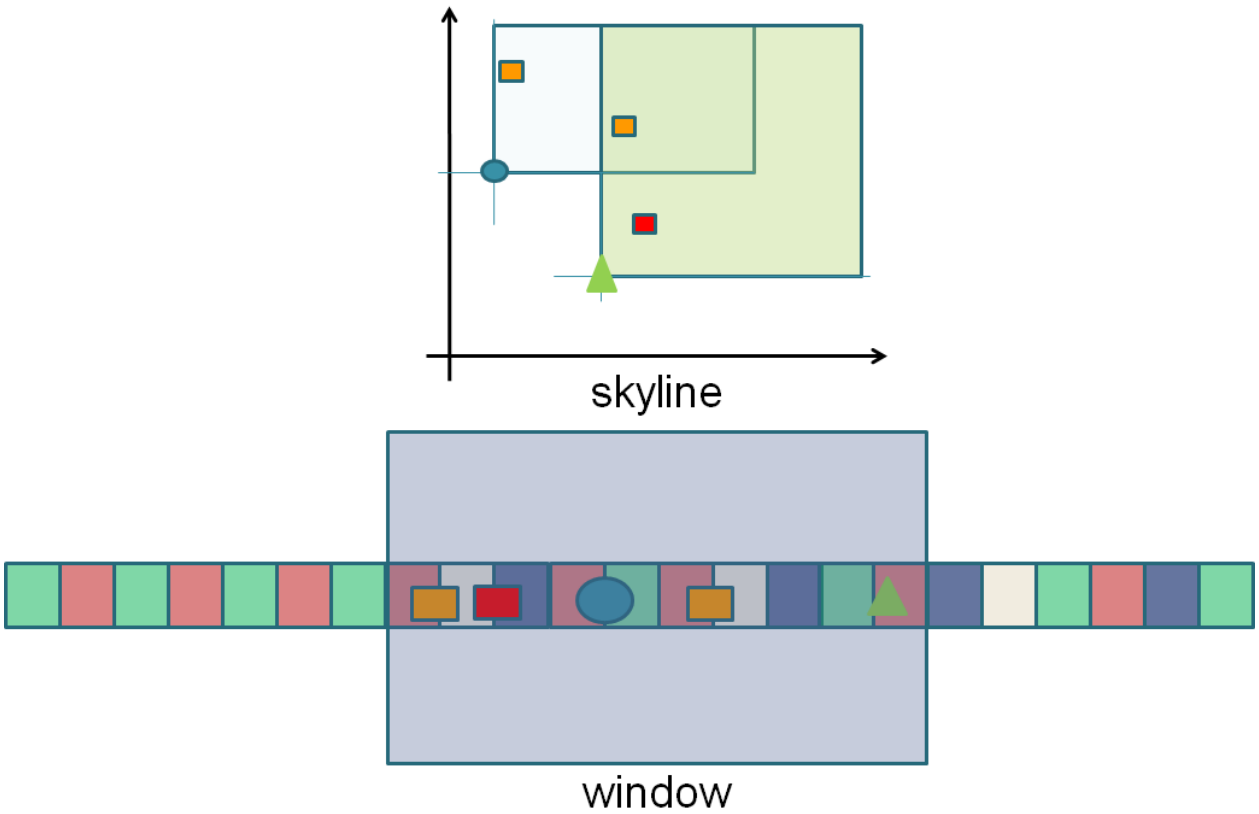


Appendix

Return

Contributions - Background -
Skyline on Data Stream [Tao, Papadias, TKDE 2006]

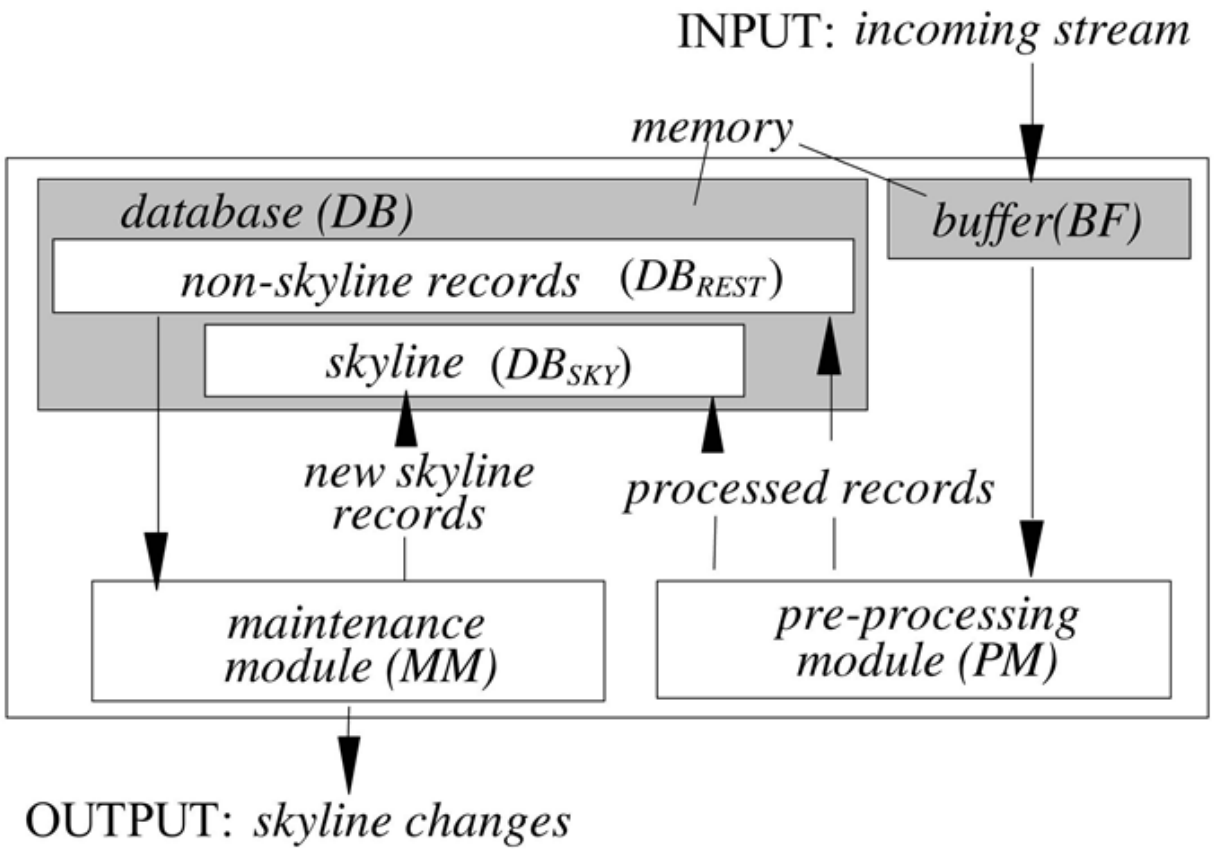
- a new tuple r' arrives
- an existing skyline tuple r' expires.
  $T_{exp}^{sky} = min\{r.t_{exp} | t \in SKYLINE\}$



Appendix

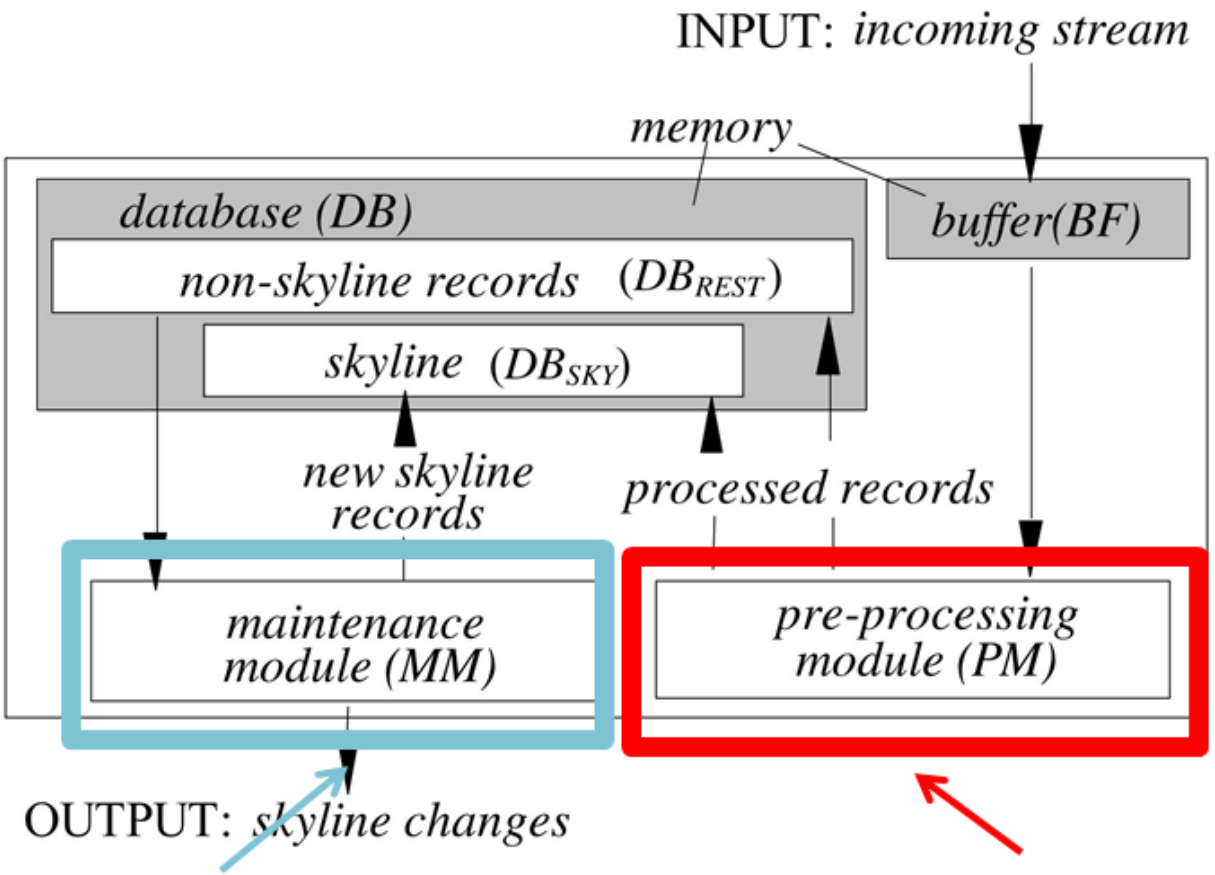
Return

Contributions - Background -
Skyline on Data Stream [Tao, Papadias, TKDE 2006]



Return

Contributions - Background -
Skyline on Data Stream [Tao, Papadias, TKDE 2006]



Appendix

Return

Contributions - Background -
R-skyline queries [Koudas et.al, VLDB 2006]

Appendix

- **Relaxing selection conditions:** Let R a relation, Q a query, $r \in R$ and $C : R.A\theta v$ a selection condition. *RELAX*, applied to r and C is defined as follows:
  - RELAX(r,C) = 0, if r satisfies C
  - RELAX(r,C) = |r.A - v|, otherwise

- **Relaxing join conditions:** Let R, S two relations, Q a query, (r, s) a pair of records and $J : R.A\theta S.B$ a join condition. *RELAX*, applied to (r, s) and J is defined as follows:
  - RELAX(r,s,J) = 0, if r, s satisfy C
  - RELAX(r,s,J) = |r.A - s.B|, otherwise.

Return

Contributions - Background - **R-skyline queries** [Koudas et.al, VLDB 2006]

Appendix

- **Dominance:** We say $RELAX(r_1, s_2, Q)$ dominates $RELAX(r_2, s_2, Q)$ if the relaxations in $RELAX(r_1, s_1, Q)$ are equal or smaller than the corresponding relaxations in $RELAX(r_2, s_2, Q)$ for all the conditions and smaller in at least one conditions.

- **Relaxation skyline:** The relaxation skyline of a query Q on two relations R and S, denoted by $SKYLINE(R, S, Q)$, is the set of all the tuple pairs, (r, s), $r \in R$ and $s \in S$, each of which has its relaxations, with respect to Q not dominated by any other tuple pair (r', s'), $r' \in R$ and $s' \in S$.

◂ Return

Contributions - **R-skyline window-based join on data stream**

Relaxation function: Let S and T be two data streams. Let $C_1 = S.A_i \theta v$ and $C_2 = S.A_i \theta\ T.A_j$. Let s be a tuple in S and t a tuple in T.

- RELAX(s, C_1) = 0 if $s.A_i \theta v$, $|s.A_i - v|$ otherwise
- RELAX(s, t, C_2) returns 0 if $s.A_i \theta t.A_j$, $|s.A_i - t.A_j|$ otherwise

Dominance: Let Q be a query; S and T be two data streams; s_1 and s_2 be tuples of S; t_1 and t_2 tuples of T. The pair of tuples $\langle s1, t1 \rangle$ dominates ($\preceq$) the pair $\langle s2, t2 \rangle$ if:

- all relaxed values in $RELAX(s_1, t_1, Q) \preceq$ corresponding relaxed values in $RELAX(s_2, t_2, Q)$
- at least one relaxed value in $RELAX(s_1, t_1, Q) \prec$ corresponding relaxed value in $RELAX(s_2, t_2, Q)$

Return

Contributions - **R-skyline window-based join on data stream**

R-skyline on data streams: Let Q be a query; S and T be two data streams; w_1 and w_2 be two window operators. The r-skyline of S and T with respect to Q, w_1 and w_2, denoted as $rs(Q, S, T, w_1, w_2)$ at time τ contains the tuples in $(S[w_1] \bowtie T[w_2])(\tau)$ that are not dominated by any tuple in $(S[w_1] \bowtie T[w_2])(\tau)$

Appendix

Return