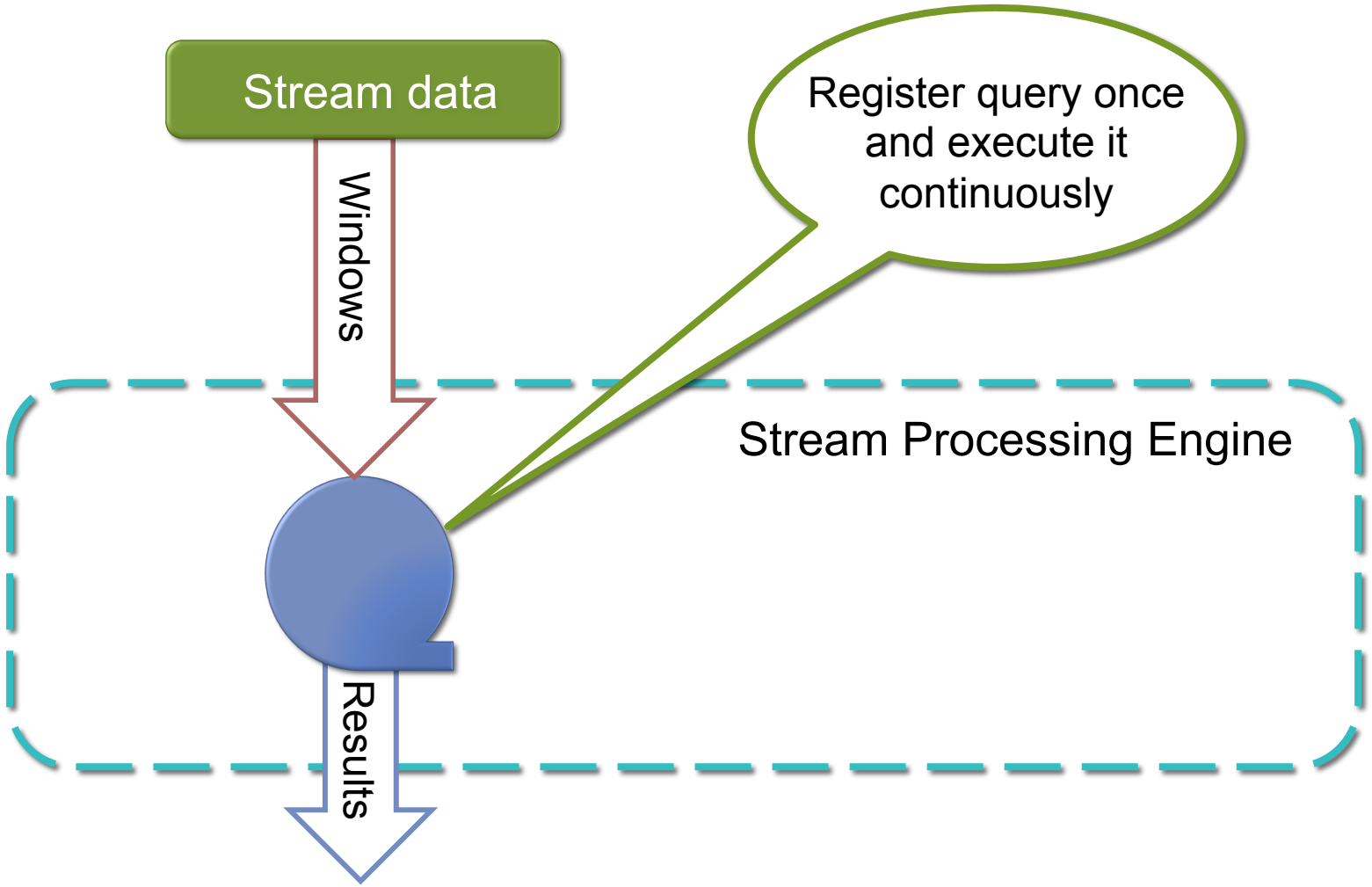
Using Rank Aggregation in Continuously Answering SPARQL Queries on Streaming and Quasi-static Linked Data

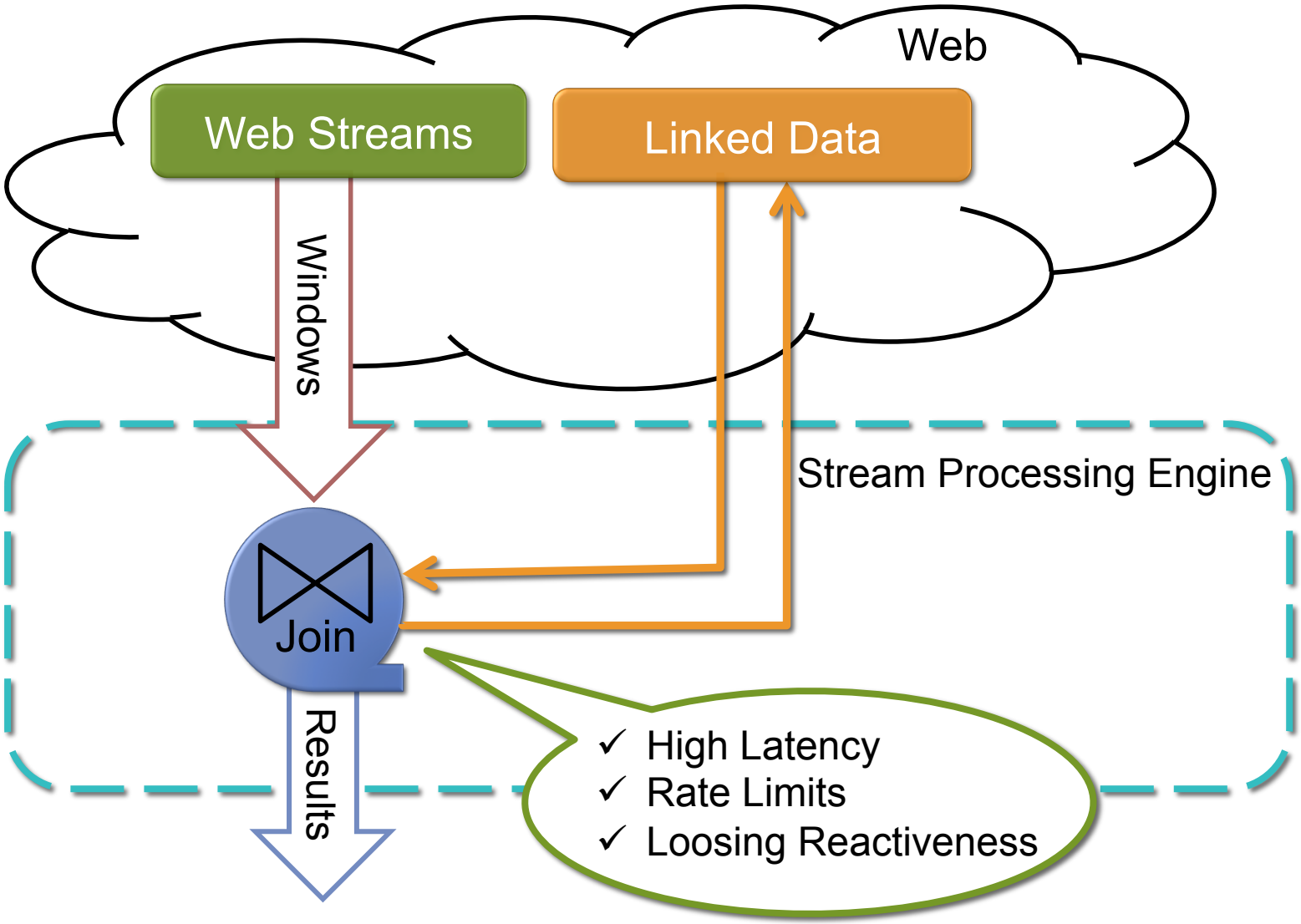
Shima Zahmatkesh, Emanuele Della Valle, and Daniele Dell'Aglio
DEIB - Politecnico of Milano

DEBS 2017 – Barcelona, Spain
23 June 2017

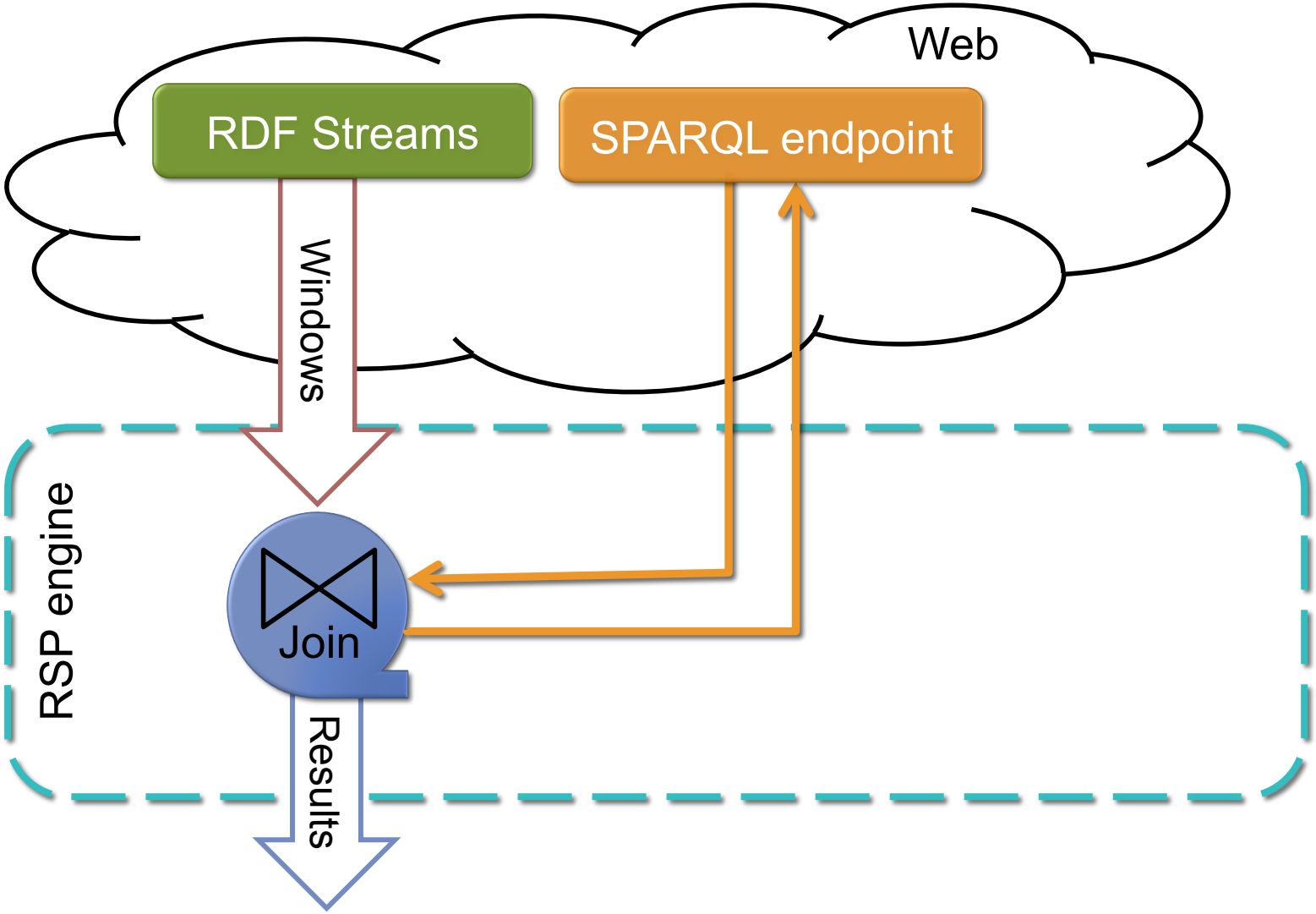
Stream Processing in Nutshell



Web Stream Processing



RDF Stream Processing (RSP) Engine



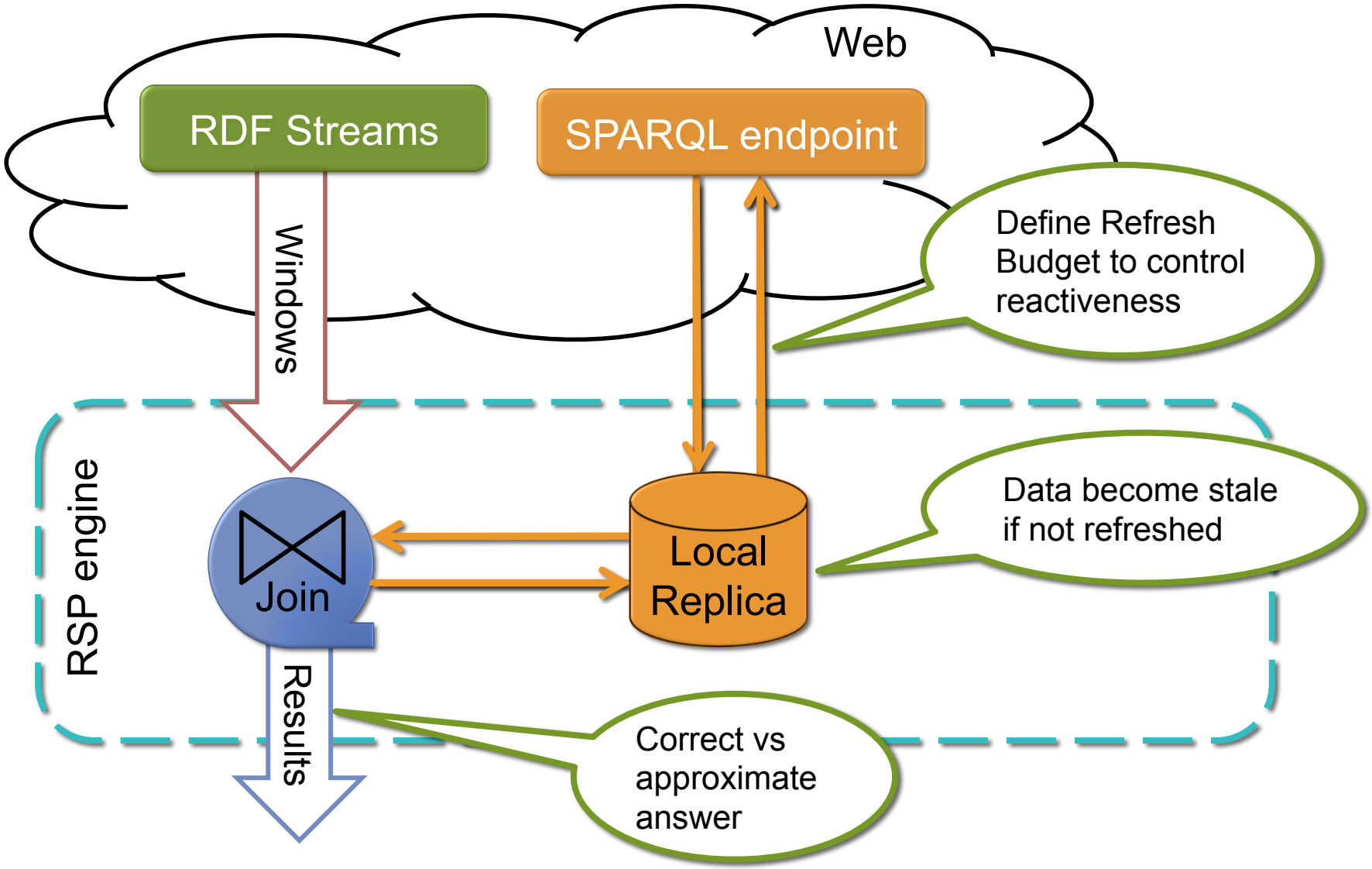
Motivation

The cloth brand ACME wants to persuade influential Social Networks users to post commercial endorsements.

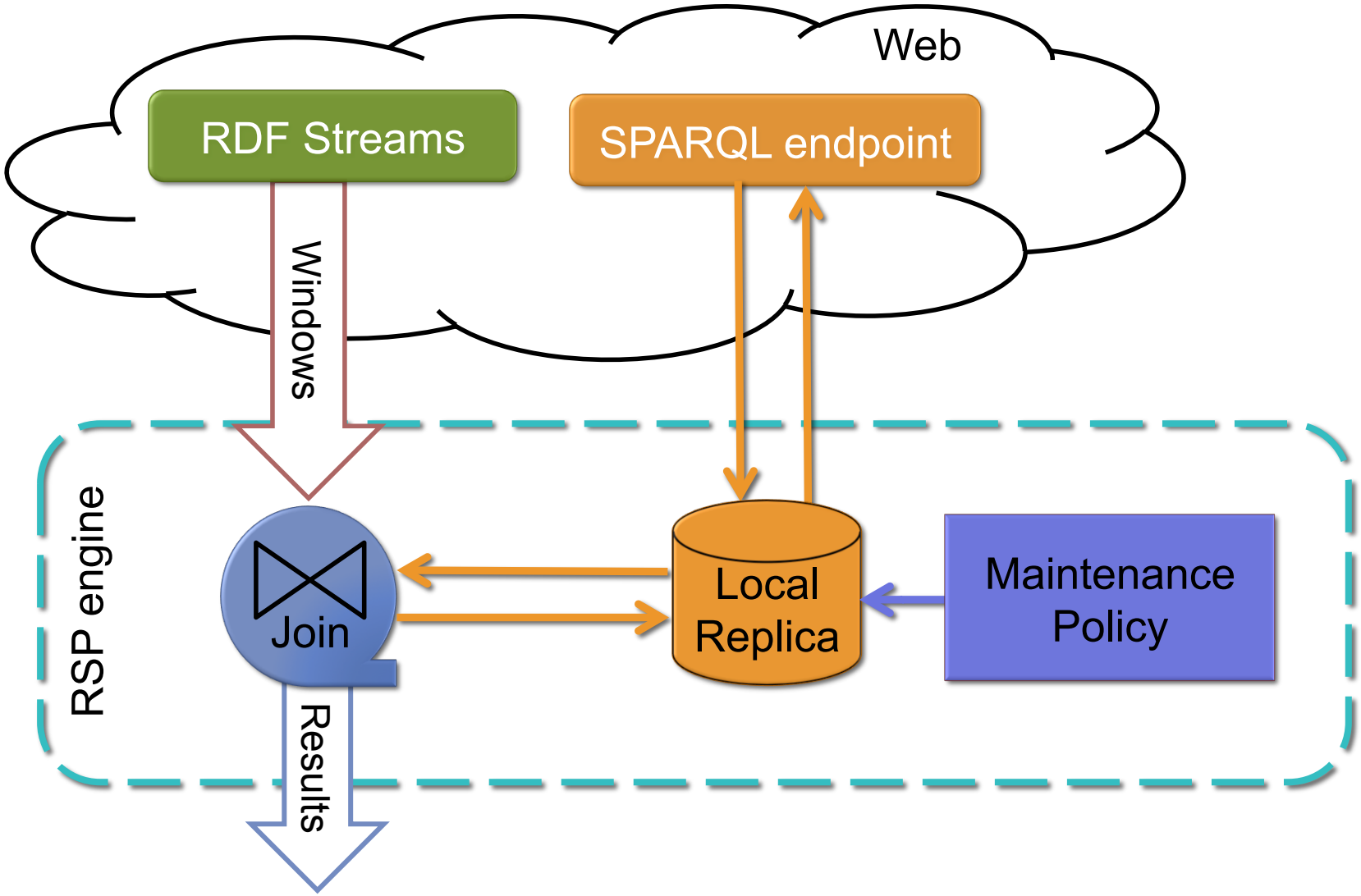
Every minute give me the ID of the users that are mentioned on Social Network in the last 10 minutes whose number of followers is greater than 100,000.

```
REGISTER STREAM <:InfluencersToContact> AS
CONSTRUCT {?user a :influentialUser}
FROM NAMED WINDOW W ON S [RANGE 10m STEP 1m]
WHERE {
        WINDOW W {?user :hasMentions ?mentionsNumber}
        SERVICE BKG  {?user :hasFollowers ?followerCount }
        FILTER (?followerCount > 100,000)
}
```

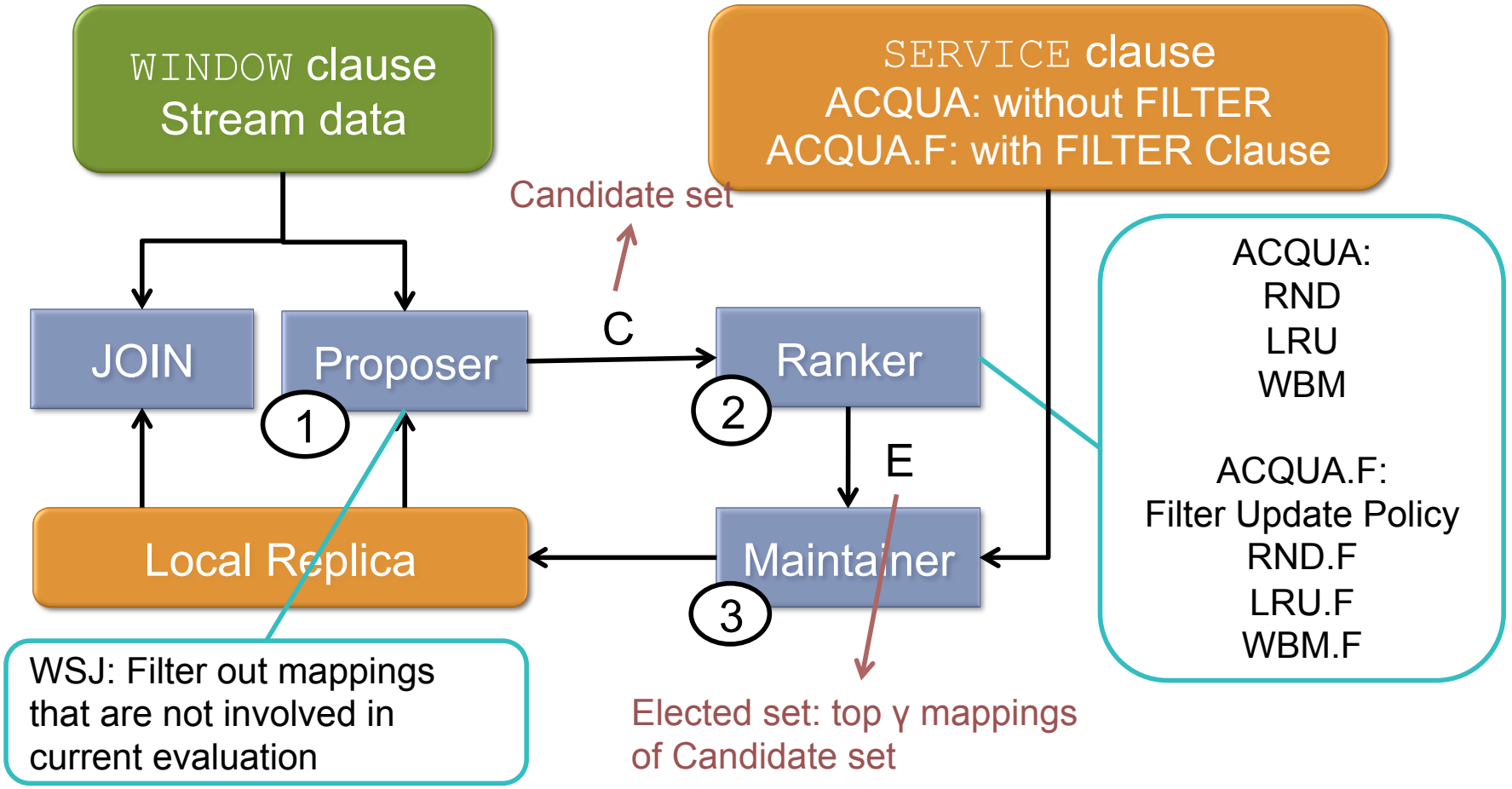
Problem Definition



Problem Definition



ACQUA, ACQUA.F Frameworks



Soheila Dehghanzadeh, et al., Approximate Continuous Query Answering over Streams and Dynamic Linked Data Sets, ICWE 2015.
Shima Zahmatkesh, et al., When a FILTER Makes the Dierence in Continuously Answering SPARQL Queries on Streaming and Quasi-Static Linked Data, ICWE 2016.
Emanuele Della Valle, et al., Taming velocity and variety simultaneously in big data with stream reasoning, DEBS 2016.

Rankers

- LRU
  - Use Least-Recently Used (LRU) cache replacement algorithm
  - The less recently a mapping have been refreshed in a query, the higher is its rank.

- Filter Update Policy
  - For each mapping in the replica:
    - Computes how close is the value associate to the variable of the mapping to the Filtering Threshold used in Filter clause.
    - Arrange mappings in ascending order.

Filtering Threshold = 100

User	Last Update Time	LRU policy	#followers	Filter Update
Alice	8	1	120	2
Bob	10	2	30	3
Carol	14	3	95	1

Rank Aggregation

- Fairly take into account the opinions of different algorithms.
- Combine the ranking lists obtained from different algorithms by computing aggregated score

$$score_{agg} = \alpha * score_{list-1} + (1 - \alpha) * score_{list-2}$$

List 1

User	Score
Alice	0.8
Bob	0.7
Carol	0.5
David	0.4
Eve	0.1

List 2

User	Score
Bob	0.9
David	0.8
Alice	0.7
Eve	0.4
Carol	0.1

α = 0.5

T = 0.5*0.8 + 0.5*0.9 = 0.85

User	$Score_{agg}$
Bob	0.8
Alice	0.75

Rank Aggregation

- Fairly take into account the opinions of different algorithms.
- Combine the ranking lists obtained from different algorithms by computing aggregated score

$$score_{agg} = \alpha * score_{list-1} + (1 - \alpha) * score_{list-2}$$

List 1

User	Score
Alice	0.8
Bob	0.7
Carol	0.5
David	0.4
Eve	0.1

List 2

User	Score
Bob	0.9
David	0.8
Alice	0.7
Eve	0.4
Carol	0.1

α = 0.5

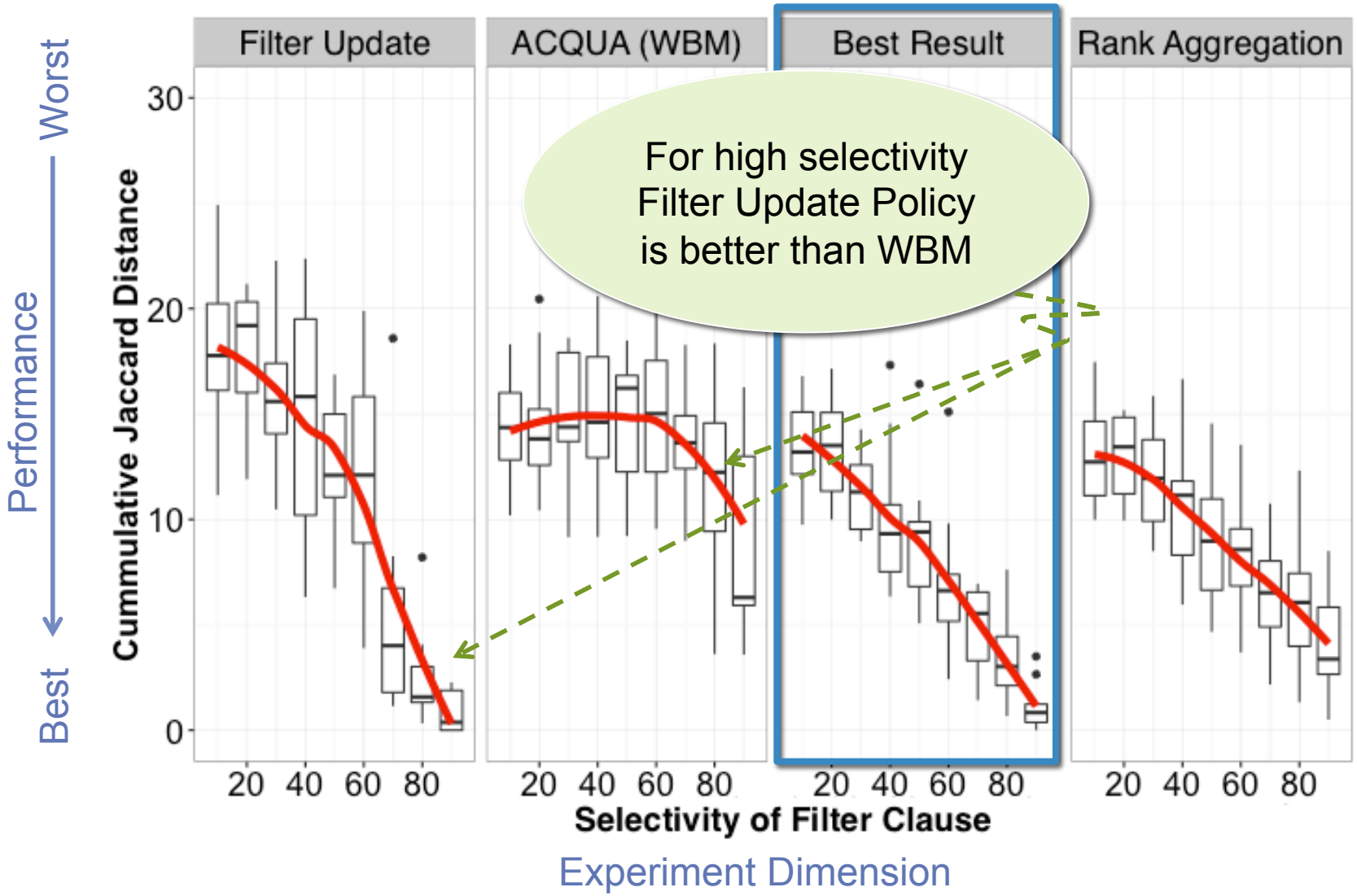
T = 0.5*0.7 + 0.5*0.8 = 0.75

User	$Score_{agg}$
Bob	0.8
Alice	0.75
David	0.6

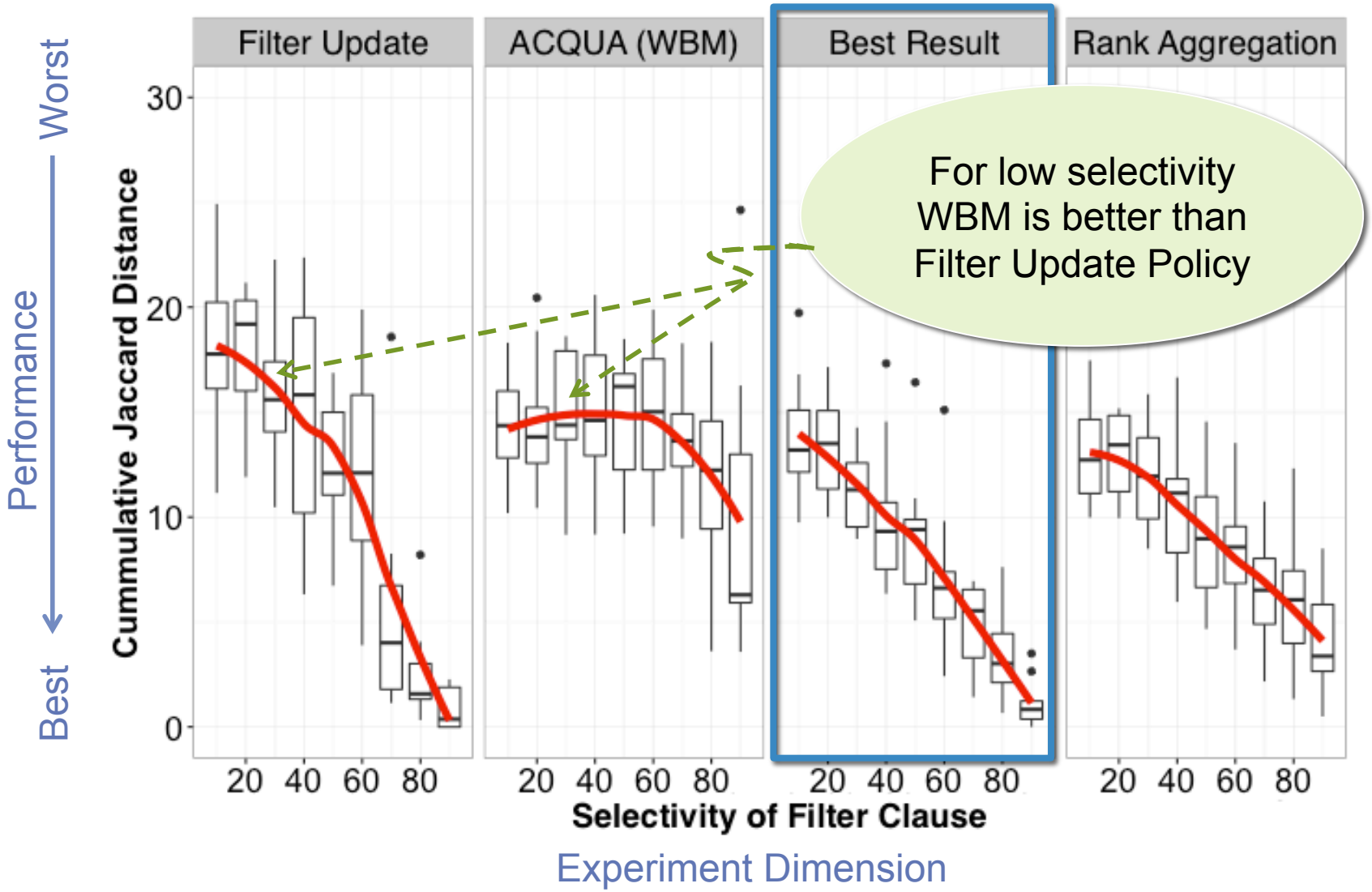
Experimental Evaluation

- Data Sets
  - Streaming data, and realistic background data from real data of Twitter
- Query
  - Contains WINDOW, SERVICE, and FILTER clauses
  - Generate correct answer of the query by an Oracle
- KPIs
  - Measure diversity of the set generated by the query and correct answer
  - Compute cummulative errors over evaluations

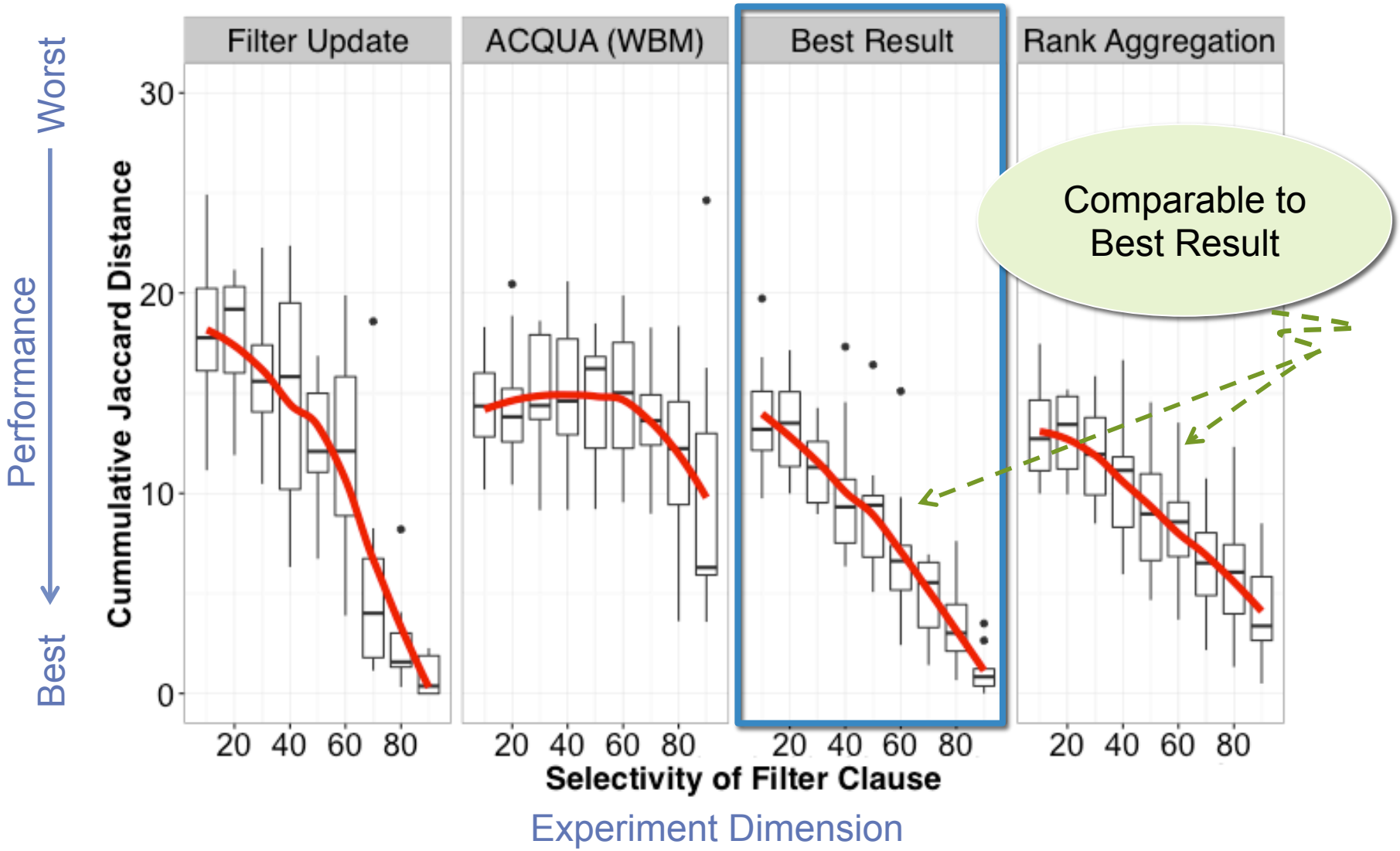
Experimental Results



Experimental Results



Experimental Results



Conclusion

- Problem of continuously evaluating queries over data stream and background data.
- The results of experiments show that proposed policies have the same accuracy of the best result achieved without using any assumption.
- They also show that the proposed policies are not sensitive to the value of alpha used in rank aggregation formula.

Future works

- Broaden the class of queries
  - Multiple filtering
  - Filtering condition formulated as a ranking clause
- Pushing the FILTER clause into the SERVICE clause and considering caching instead of local replica
- Study the effect of different trends in the data

Thank you!
Any Question?

Using Rank Aggregation in Continuously Answering SPARQL Queries on Streaming and Quasi-static Linked Data

Shima Zahmatkesh
shima.zahmatkesh@polimi.it
DEIB - Politecnico of Milano