# The Odyssey Approach for Optimizing Federated SPARQL Queries

Gabriela Montoya[1], Hala Skaf-Molli[2], and Katja Hose[1]

[1]Aalborg University, Denmark
{gmontoya,khose}@cs.aau.dk

[2]Nantes University, France
hala.skaf@univ-nantes.fr

October 25th, 2017

# Optimizing Federated SPARQL Queries

```
SELECT DISTINCT * WHERE {
  ?film dbo:director ?director .                    (tp1)
  ?film rdf:type dbo:Film .                         (tp2)
  ?movie owl:sameAs ?film .                         (tp3)
  ?movie dcterms:title ?title .                     (tp4)
  ?movie movie:film_subject film_subject:444        (tp5)
}
```

DBpedia, Drugbank, LMDB, ChEBI, Geonames, NYTimes, Jamendo, SWDF, KEGG

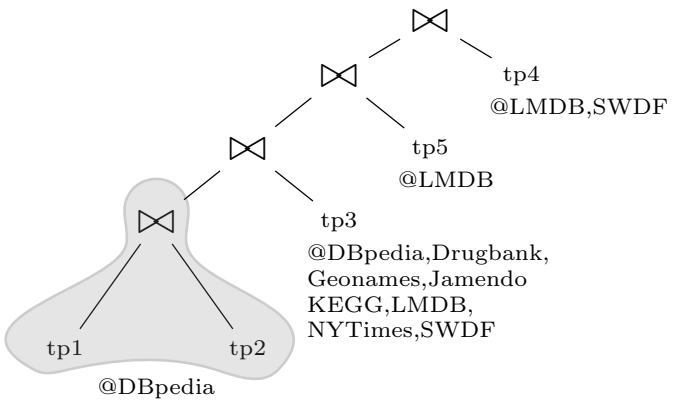
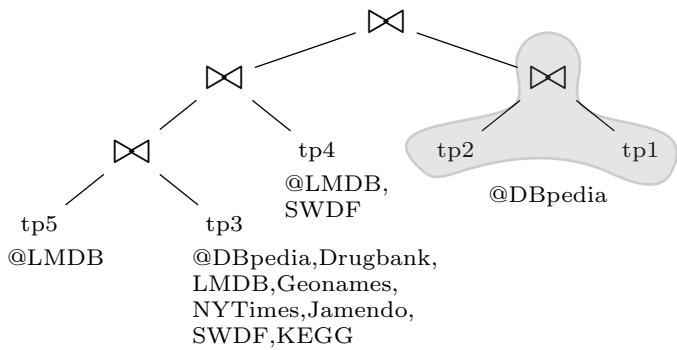
# Optimizing Federated SPARQL Queries

```
SELECT DISTINCT * WHERE {
  ?film dbo:director ?director .                    (tp1)
  ?film rdf:type dbo:Film .                         (tp2)
  ?movie owl:sameAs ?film .                         (tp3)
  ?movie dcterms:title ?title .                     (tp4)
  ?movie movie:film_subject film_subject:444        (tp5)
}
```

DBpedia, Drugbank, LMDB, ChEBI, Geonames, NYTimes, Jamendo, SWDF, KEGG

| Subquery | Relevant Sources |
|---|---|
| ?film dbo:director ?director . | DBpedia |
| ?film rdf:type dbo:Film | DBpedia,Drugbank,LMDB,ChEBI Geonames,NYTimes,Jamendo,SWDF,KEGG |
| ?movie owl:sameAs ?film | DBpedia,Drugbank,LMDB Geonames,NYTimes,Jamendo,SWDF,KEGG |
| ?movie dcterms:title ?title | LMDB,SWDF |
| ?movie movie:film_subject film_subject:444 | LMDB |

# Existing approaches

| | FedX[1] | SemaGrow[2] |
|---|---|---|
| Plan |  |  |
| Optimization Technique | Heuristics | Dynamic Programming |
| Optimization Time (s) | 0.74 | 4.75 |
| Execution Time (s) | 142 | 6.93 |



[1] A. Schwarte et al. "FedX: Optimization Techniques for Federated Query Processing on Linked Data". In: *ISWC'11*.

[2] A. Charalambidis et al. "SemaGrow: optimizing federated SPARQL queries". In: *SEMANTICS'15*.

# Considering joins between triple patterns improves the optimization

| Subquery | Relevant Sources |
|---|---|
| **?movie** owl:sameAs ?film .<br>**?movie** dcterms:title ?title .<br>**?movie** movie:film_subject film_subject:444 | LMDB |

Only entities that satisfy *owl:sameAs*, *dcterms:title*, **and** *movie:film_subject* are part of LMDB!
Entities from all other sources that seem relevant for some triple patterns, actually will not contribute to the result!

# Contributions

- Concise statistics describing links among triples while guaranteeing result completeness.
- A technique to compute such statistics in a federated setup.
- Affordable optimization based on dynamic programming.

# Statistics computation at one location

film:1129 movie:film_subject film_subject:444 .
film:1129 dcterms:title "Kate & Leopold" .
film:1129 owl:sameAs dbr:Kate_&_Leopold .
film:16189 movie:film_subject film_subject:444 .
film:16189 dcterms:title "Journey to the Center of Time" .
film:16189 owl:sameAs dbr:Journey_to_the_Center_of_Time .
dbr:Journey_to_the_Center_of_Time dbo:director dbr:David_L._Hewitt .
dbr:Journey_to_the_Center_of_Time rdf:type dbo:Film .
...

[3] T. Neumann and G. Moerkotte. "Characteristic Sets: Accurate Cardinality Estimation for RDF Queries with Multiple Joins". In: *ICDE'11*.

[4] A. Gubichev and T. Neumann. "Exploiting the query structure for efficient join ordering in SPARQL queries". In: *EDBT'14*.

# Statistics computation at one location

film:1129 movie:film_subject film_subject:444 .
film:1129 dcterms:title "Kate & Leopold" .
film:1129 owl:sameAs dbr:Kate_&_Leopold .
film:16189 movie:film_subject film_subject:444 .
film:16189 dcterms:title "Journey to the Center of Time" .
film:16189 owl:sameAs dbr:Journey_to_the_Center_of_Time .
dbr:Journey_to_the_Center_of_Time dbo:director dbr:David_L._Hewitt .
dbr:Journey_to_the_Center_of_Time rdf:type dbo:Film .
...

[3] T. Neumann and G. Moerkotte. "Characteristic Sets: Accurate Cardinality Estimation for RDF Queries with Multiple Joins". In: *ICDE'11*.

[4] A. Gubichev and T. Neumann. "Exploiting the query structure for efficient join ordering in SPARQL queries". In: *EDBT'14*.

# Statistics computation at one location

Characteristic Sets (CS)[3]

$C_{D,1}$ = { movie:film_subject, dcterms:title, owl:sameAs }

count($C_{D,1}$)=1

film:1129 movie:film_subject film_subject:444 .
film:1129 dcterms:title "Kate & Leopold" .
film:1129 owl:sameAs dbr:Kate_&_Leopold .
film:16189 movie:film_subject film_subject:444 .
film:16189 dcterms:title "Journey to the Center of Time" .
film:16189 owl:sameAs dbr:Journey_to_the_Center_of_Time .
dbr:Journey_to_the_Center_of_Time dbo:director dbr:David_L._Hewitt .
dbr:Journey_to_the_Center_of_Time rdf:type dbo:Film .
...

[3] T. Neumann and G. Moerkotte. "Characteristic Sets: Accurate Cardinality Estimation for RDF Queries with Multiple Joins". In: *ICDE'11*.

[4] A. Gubichev and T. Neumann. "Exploiting the query structure for efficient join ordering in SPARQL queries". In: *EDBT'14*.

# Statistics computation at one location

Characteristic Sets (CS)[3]

$C_{D,1}$ = { movie:film_subject, dcterms:title, owl:sameAs }

count($C_{D,1}$)=2

film:1129 movie:film_subject film_subject:444 .
film:1129 dcterms:title "Kate & Leopold" .
film:1129 owl:sameAs dbr:Kate_&_Leopold .
film:16189 movie:film_subject film_subject:444 .
film:16189 dcterms:title "Journey to the Center of Time" .
film:16189 owl:sameAs dbr:Journey_to_the_Center_of_Time .
dbr:Journey_to_the_Center_of_Time dbo:director dbr:David_L._Hewitt .
dbr:Journey_to_the_Center_of_Time rdf:type dbo:Film .
...

[3] T. Neumann and G. Moerkotte. "Characteristic Sets: Accurate Cardinality Estimation for RDF Queries with Multiple Joins". In: *ICDE'11*.

[4] A. Gubichev and T. Neumann. "Exploiting the query structure for efficient join ordering in SPARQL queries". In: *EDBT'14*.

# Statistics computation at one location

Characteristic Sets (CS)[3]

$C_{D,1}$ = { movie:film_subject, dcterms:title, owl:sameAs }
count($C_{D,1}$)=2

$C_{D,2}$ = { dbo:director, rdf:type } count($C_{D,2}$)=1

film:1129 movie:film_subject film_subject:444 .
film:1129 dcterms:title "Kate & Leopold" .
film:1129 owl:sameAs dbr:Kate_&_Leopold .
film:16189 movie:film_subject film_subject:444 .
film:16189 dcterms:title "Journey to the Center of Time" .
film:16189 owl:sameAs dbr:Journey_to_the_Center_of_Time .
dbr:Journey_to_the_Center_of_Time dbo:director dbr:David_L._Hewitt .
dbr:Journey_to_the_Center_of_Time rdf:type dbo:Film .
...

[3] T. Neumann and G. Moerkotte. "Characteristic Sets: Accurate Cardinality Estimation for RDF Queries with Multiple Joins". In: *ICDE'11*.

[4] A. Gubichev and T. Neumann. "Exploiting the query structure for efficient join ordering in SPARQL queries". In: *EDBT'14*.

# Other basic statistics are also stored

Characteristic Sets (CS)[3]

$C_{D,1}$ = { movie:film_subject, dcterms:title, owl:sameAs }

count($C_{D,1}$)=2; ocurrences(dcterms:title, $C_{D,1}$)=2

$C_{D,2}$ = { dbo:director, rdf:type } count($C_{D,2}$)=1

film:1129 movie:film_subject film_subject:444 .
film:1129 dcterms:title "Kate & Leopold" .
film:1129 owl:sameAs dbr:Kate_&_Leopold .
film:16189 movie:film_subject film_subject:444 .
film:16189 dcterms:title "Journey to the Center of Time" .
film:16189 owl:sameAs dbr:Journey_to_the_Center_of_Time .
dbr:Journey_to_the_Center_of_Time dbo:director dbr:David_L._Hewitt .
dbr:Journey_to_the_Center_of_Time rdf:type dbo:Film .
...

[3] T. Neumann and G. Moerkotte. "Characteristic Sets: Accurate Cardinality Estimation for RDF Queries with Multiple Joins". In: *ICDE'11*.

[4] A. Gubichev and T. Neumann. "Exploiting the query structure for efficient join ordering in SPARQL queries". In: *EDBT'14*.

# CSs are connected to other CSs

Characteristic Sets (CS)[3]

$C_{D,1}$ = { movie:film_subject, dcterms:title, owl:sameAs }

count($C_{D,1}$)=2; ocurrences(dcterms:title, $C_{D,1}$)=2

$C_{D,2}$ = { dbo:director, rdf:type } count($C_{D,2}$)=1

film:1129 movie:film_subject film_subject:444 .
film:1129 dcterms:title "Kate & Leopold" .
film:1129 owl:sameAs dbr:Kate_&_Leopold .
film:16189 movie:film_subject film_subject:444 .
film:16189 dcterms:title "Journey to the Center of Time" .
film:16189 owl:sameAs dbr:Journey_to_the_Center_of_Time .
dbr:Journey_to_the_Center_of_Time dbo:director dbr:David_L._Hewitt .
dbr:Journey_to_the_Center_of_Time rdf:type dbo:Film .
...

dbr:Journey_to_the_Center_of_Time → owl:sameAs → $C_{D,1}$

[3] T. Neumann and G. Moerkotte. "Characteristic Sets: Accurate Cardinality Estimation for RDF Queries with Multiple Joins". In: *ICDE'11*.

[4] A. Gubichev and T. Neumann. "Exploiting the query structure for efficient join ordering in SPARQL queries". In: *EDBT'14*.

# CSs are connected to other CSs

Characteristic Sets (CS)[3]

$C_{D,1}$ = { movie:film_subject, dcterms:title, owl:sameAs }

count($C_{D,1}$)=2; ocurrences(dcterms:title, $C_{D,1}$)=2

$C_{D,2}$ = { dbo:director, rdf:type } count($C_{D,2}$)=1

film:1129 movie:film_subject film_subject:444 .
film:1129 dcterms:title "Kate & Leopold" .
film:1129 owl:sameAs dbr:Kate_&_Leopold .
film:16189 movie:film_subject film_subject:444 .
film:16189 dcterms:title "Journey to the Center of Time" .
film:16189 owl:sameAs dbr:Journey_to_the_Center_of_Time .
dbr:Journey_to_the_Center_of_Time dbo:director dbr:David_L._Hewitt .
dbr:Journey_to_the_Center_of_Time rdf:type dbo:Film .

...

dbr:Journey_to_the_Center_of_Time → owl:sameAs → $C_{D,1}$

Characteristic Pairs (CP)[4]

[3] T. Neumann and G. Moerkotte. "Characteristic Sets: Accurate Cardinality Estimation for RDF Queries with Multiple Joins". In: *ICDE'11*.

[4] A. Gubichev and T. Neumann. "Exploiting the query structure for efficient join ordering in SPARQL queries". In: *EDBT'14*.

# CSs are connected to other CSs

Characteristic Sets (CS)[3]

$C_{D,1}$ = { movie:film_subject, dcterms:title, owl:sameAs }

count($C_{D,1}$)=2; ocurrences(dcterms:title, $C_{D,1}$)=2

$C_{D,2}$ = { dbo:director, rdf:type } count($C_{D,2}$)=1

film:1129 movie:film_subject film_subject:444 .
film:1129 dcterms:title "Kate & Leopold" .
film:1129 owl:sameAs dbr:Kate_&_Leopold .
film:16189 movie:film_subject film_subject:444 .
film:16189 dcterms:title "Journey to the Center of Time" .
film:16189 owl:sameAs dbr:Journey_to_the_Center_of_Time .
dbr:Journey_to_the_Center_of_Time dbo:director dbr:David_L._Hewitt .
dbr:Journey_to_the_Center_of_Time rdf:type dbo:Film .

...

dbr:Journey_to_the_Center_of_Time $\rightarrow$ owl:sameAs $\rightarrow$ $C_{D,1}$

Characteristic Pairs (CP)[4]

($C_{D,1}$, $C_{D,2}$, owl:sameAs) count(($C_{D,1}$, $C_{D,2}$, owl:sameAs)) = 1

---

[3] T. Neumann and G. Moerkotte. "Characteristic Sets: Accurate Cardinality Estimation for RDF Queries with Multiple Joins". In: *ICDE'11*.

[4] A. Gubichev and T. Neumann. "Exploiting the query structure for efficient join ordering in SPARQL queries". In: *EDBT'14*.

# Cardinality computation

```
SELECT DISTINCT ?film ?movie WHERE {
  ?film dbo:director ?director .          (tp1)
  ?film rdf:type ?type .                  (tp2)
  ?movie owl:sameAs ?film .               (tp3)
  ?movie dcterms:title ?title .           (tp4)
  ?movie movie:film_subject ?subject      (tp5)
}
```

$$cardinality((P_k, P_l, p)) = \sum_{P_k \subseteq C_i \wedge P_l \subseteq C_j} count((C_i, C_j, p)) \quad (1)$$

$$P_k = \{owl : sameAs, dcterms : title, movie : film\_subject\}$$
$$P_l = \{dbo : director, rdf : type\}$$
$$p = owl : sameAs$$

# Cardinality estimation

```
SELECT DISTINCT * WHERE {
  ?film dbo:director ?director .          (tp1)
  ?film rdf:type ?type .                  (tp2)
  ?movie owl:sameAs ?film .               (tp3)
  ?movie dcterms:title ?title .           (tp4)
  ?movie movie:film_subject ?subject      (tp5)
}
```

$$estimatedCardinality((P_k, P_l, p)) = \sum_{P_k \subseteq C_i \wedge P_l \subseteq C_j} \Big( count((C_i, C_j, p)) * \prod_{p_k \in P_k - \{p\}} \left(\frac{ocurrences(p_k, C_i)}{count(C_i)}\right) * \prod_{p_l \in P_l} \left(\frac{ocurrences(p_l, C_j)}{count(C_j)}\right) \Big) \quad (2)$$

$$P_k = \{owl : sameAs, dcterms : title, movie : film\_subject\}$$

$$P_l = \{dbo : director, rdf : type\}$$

$$p = owl : sameAs$$

# Cardinality estimation

```
SELECT DISTINCT * WHERE {
  ?film dbo:director ?director .                    (tp1)
  ?film rdf:type dbo:Film .                         (tp2)
  ?movie owl:sameAs ?film .                         (tp3)
  ?movie dcterms:title ?title .                     (tp4)
  ?movie movie:film_subject film_subject:444        (tp5)
}
```

$$estimatedCardinality((P_k, P_l, p)) = \max_{P_k \subseteq C_i \wedge P_l \subseteq C_j} \Big( count((C_i, C_j, p)) * \prod_{p_k \in P_k - \{p\}} \left(\frac{ocurrences(p_k, C_i)}{count(C_i)}\right) * \prod_{p_l \in P_l} \left(\frac{ocurrences(p_l, C_j)}{count(C_j)}\right) \Big) \quad (3)$$

$$P_k = \{owl : sameAs, dcterms : title, movie : film\_subject\}$$
$$P_l = \{dbo : director, rdf : type\}$$
$$p = owl : sameAs$$

# Federated computation of statistics

1. At each source the CSs are computed

film:1129 movie:film_subject film_subject:444 .
film:1129 dcterms:title "Kate & Leopold" .
film:1129 owl:sameAs dbr:Kate_&_Leopold .
film:16189 movie:film_subject film_subject:444 .
film:16189 dcterms:title "Journey to the Center of Time" .
film:16189 owl:sameAs dbr:Journey_to_the_Center_of_Time .
...

$C_{LMDB,i}$={movie:film_subject,owl:sameAs,dcterms:title,...}

...

# Federated computation of statistics

1. At each source the CSs are computed
2. At each source local statistics are computed

film:1129 movie:film_subject film_subject:444 .
film:1129 dcterms:title "Kate & Leopold" .
film:1129 owl:sameAs dbr:Kate_&_Leopold .
film:16189 movie:film_subject film_subject:444 .
film:16189 dcterms:title "Journey to the Center of Time" .
film:16189 owl:sameAs dbr:Journey_to_the_Center_of_Time .
...

$C_{LMDB,i}$={movie:film_subject,owl:sameAs,dcterms:title,...}

...

$local\_subjects_{LMDB}(C_{LMDB,i})$={ film:1129, film:16189, ... }
$local\_objects_{LMDB}$(owl:sameAs, $C_{LMDB,i}$)={
dbr:Kate_&_Leopold, dbr:Journey_to_the_Center_of_Time, ...}

# Federated computation of statistics

3. Local statistics and CSs are transferred to the federated query engine

$\text{local\_subjects}_{\text{LMDB}}(C_{\text{LMDB},i})$={ film:1129, film:16189, ... }
$\text{local\_objects}_{\text{LMDB}}$(owl:sameAs, $C_{\text{LMDB},i}$)={ ...,dbr:Journey_to_the_Center_of_Time, ...}
$\text{local\_subjects}_{\text{DBpedia}}(C_{\text{DBpedia},j})$={ ..., dbr:Journey_to_the_Center_of_Time,...}
$\text{local\_objects}_{\text{DBpedia}}$(dbo:director,$C_{\text{DBpedia},j}$)={ dbr:David_L._Hewitt, ...}
...

# Federated computation of statistics

3. Local statistics and CSs are transferred to the federated query engine
4. Set overlap is computed to estimate the statistics of federated CSs and CPs

$local\_subjects_{LMDB}(C_{LMDB,i})$={ film:1129, film:16189, ... }
$local\_objects_{LMDB}$(owl:sameAs, $C_{LMDB,i}$)={ ...,dbr:Journey_to_the_Center_of_Time, ...}
$local\_subjects_{DBpedia}(C_{DBpedia,j})$={ ..., dbr:Journey_to_the_Center_of_Time,...}
$local\_objects_{DBpedia}$(dbo:director,$C_{DBpedia,j}$)={ dbr:David_L._Hewitt, ...}
...

($C_{LMDB,j}$,$C_{DBpedia,j}$,owl:sameAs)

# Odyssey's Query Optimization

```
SELECT DISTINCT * WHERE {
   ?film dbo:director ?director .                    (tp1)
   ?film rdf:type dbo:Film .                         (tp2)
   ?movie owl:sameAs ?film .                         (tp3)
   ?movie dcterms:title ?title .                     (tp4)
   ?movie movie:film_subject film_subject:444        (tp5)
}
```

# Odyssey's Query Optimization

1. Identify the star-shaped subqueries

```
SELECT DISTINCT * WHERE {
   ?film dbo:director ?director .          (tp1)
   ?film rdf:type dbo:Film .               (tp2)
   ?movie owl:sameAs ?film .               (tp3)
   ?movie dcterms:title ?title .           (tp4)
   ?movie movie:film_subject film_subject:444   (tp5)
}
```

# Odyssey's Query Optimization

1. Identify the star-shaped subqueries
2. Identify the relevant CSs and estimate cardinality

```
SELECT DISTINCT * WHERE {
   ?film dbo:director ?director .          (tp1)
   ?film rdf:type dbo:Film .               (tp2)
   ?movie owl:sameAs ?film .               (tp3)
   ?movie dcterms:title ?title .           (tp4)
   ?movie movie:film_subject film_subject:444   (tp5)
}
```

$C_{DBpedia,j}$={dbo:director,rdf:type,...}
estimatedCardinality({dbo:director,rdf:type})=162
$C_{LMDB,i}$={movie:film_subject,owl:sameAs,dcterms:title,...}
estimatedCardinality({movie:film_subject,owl:sameAs,dcterms:title})=4

# Odyssey's Query Optimization

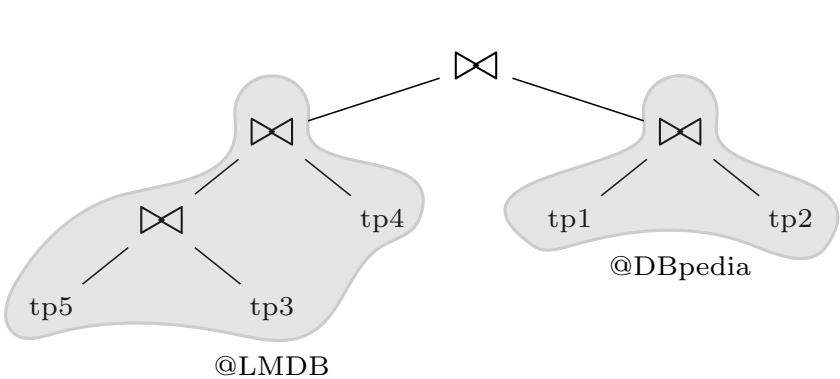
3. Identify the relevant CPs and estimate cardinality.

$(C_{LMDB,i}, C_{DBpedia,j}, \text{owl:sameAs})$
$\text{estimatedCardinality}((C_{LMDB,i}, C_{DBpedia,j}, \text{owl:sameAs})) = 1$

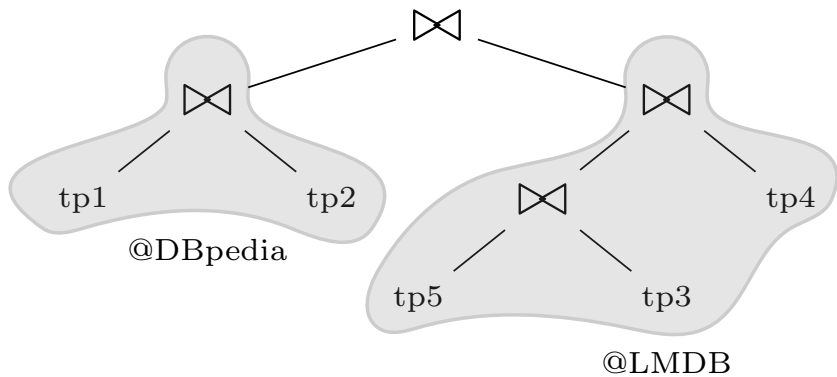
# Odyssey's Query Optimization

3. Identify the relevant CPs and estimate cardinality.
4. A cost function is used to select the plan that leads to transfer the lower number of tuples.

$(C_{LMDB,i}, C_{DBpedia,j}, \text{owl:sameAs})$
$\text{estimatedCardinality}((C_{LMDB,i}, C_{DBpedia,j}, \text{owl:sameAs}))=1$




cost=4+1=5

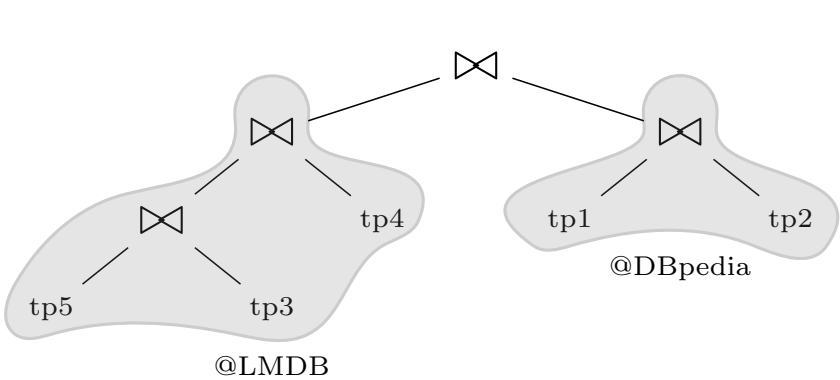



cost=162+1=163

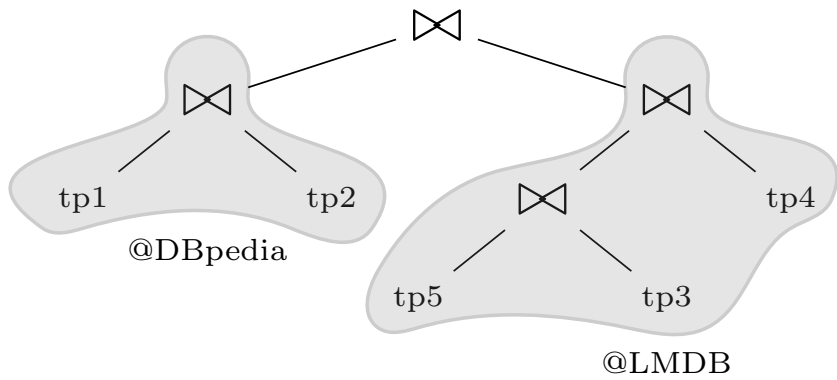
# Odyssey's Query Optimization

3. Identify the relevant CPs and estimate cardinality.
4. A cost function is used to select the plan that leads to transfer the lower number of tuples.
5. If necessary, compute join ordering using Dynamic Programming.

($C_{LMDB,i}$,$C_{DBpedia,j}$,owl:sameAs)
estimatedCardinality(($C_{LMDB,i}$,$C_{DBpedia,j}$,owl:sameAs))=1




cost=4+1=5




cost=162+1=163

# Odyssey provides a better plan

```
SELECT DISTINCT * WHERE {
  ?film dbo:director ?director .                    (tp1)
  ?film rdf:type dbo:Film .                         (tp2)
  ?movie owl:sameAs ?film .                         (tp3)
  ?movie dcterms:title ?title .                     (tp4)
  ?movie movie:film_subject film_subject:444        (tp5)
}
```

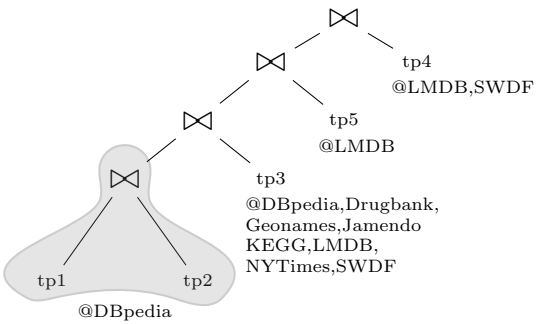
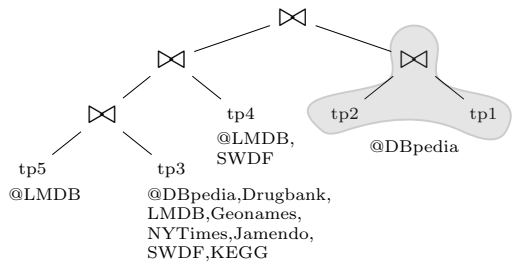
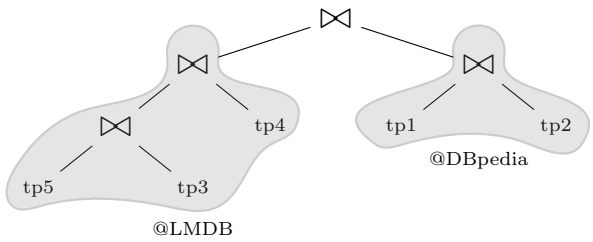
[5] A. Schwarte et al. "FedX: Optimization Techniques for Federated Query Processing on Linked Data". In: *ISWC'11*.

[6] A. Charalambidis et al. "SemaGrow: optimizing federated SPARQL queries". In: *SEMANTICS'15*.

[7] G. Montoya et al. "The Odyssey Approach for Optimizing Federated SPARQL Queries". In: *ISWC'17*.

# Odyssey provides a better plan

```
SELECT DISTINCT * WHERE {
  ?film dbo:director ?director .          (tp1)
  ?film rdf:type dbo:Film .               (tp2)
  ?movie owl:sameAs ?film .               (tp3)
  ?movie dcterms:title ?title .           (tp4)
  ?movie movie:film_subject film_subject:444 (tp5)
}
```

| | FedX[5] | SemaGrow[6] | **Odyssey[7]** |
|---|---|---|---|
| Plan |   |   |   |
| OT | 0.74s | 4.75s | **0.22s** |
| ET | 142s | 6.93s | **1.30s** |

[5] A. Schwarte et al. "FedX: Optimization Techniques for Federated Query Processing on Linked Data". In: *ISWC'11*.

[6] A. Charalambidis et al. "SemaGrow: optimizing federated SPARQL queries". In: *SEMANTICS'15*.

[7] G. Montoya et al. "The Odyssey Approach for Optimizing Federated SPARQL Queries". In: *ISWC'17*.

# Experimental setup

- Queries and datasets from FedBench[8].
- Comparison with existing approaches FedX[9], SemaGrow[10], SPLENDID[11], HiBISCuS[12].
- A Virtuoso 7.2.4.2 endpoint was deployed for each dataset.
- Plots present the average over nine executions with a timeout of 30m.

[8] M. Schmidt et al. "FedBench: A Benchmark Suite for Federated Semantic Data Query Processing". In: *ISWC'11*.

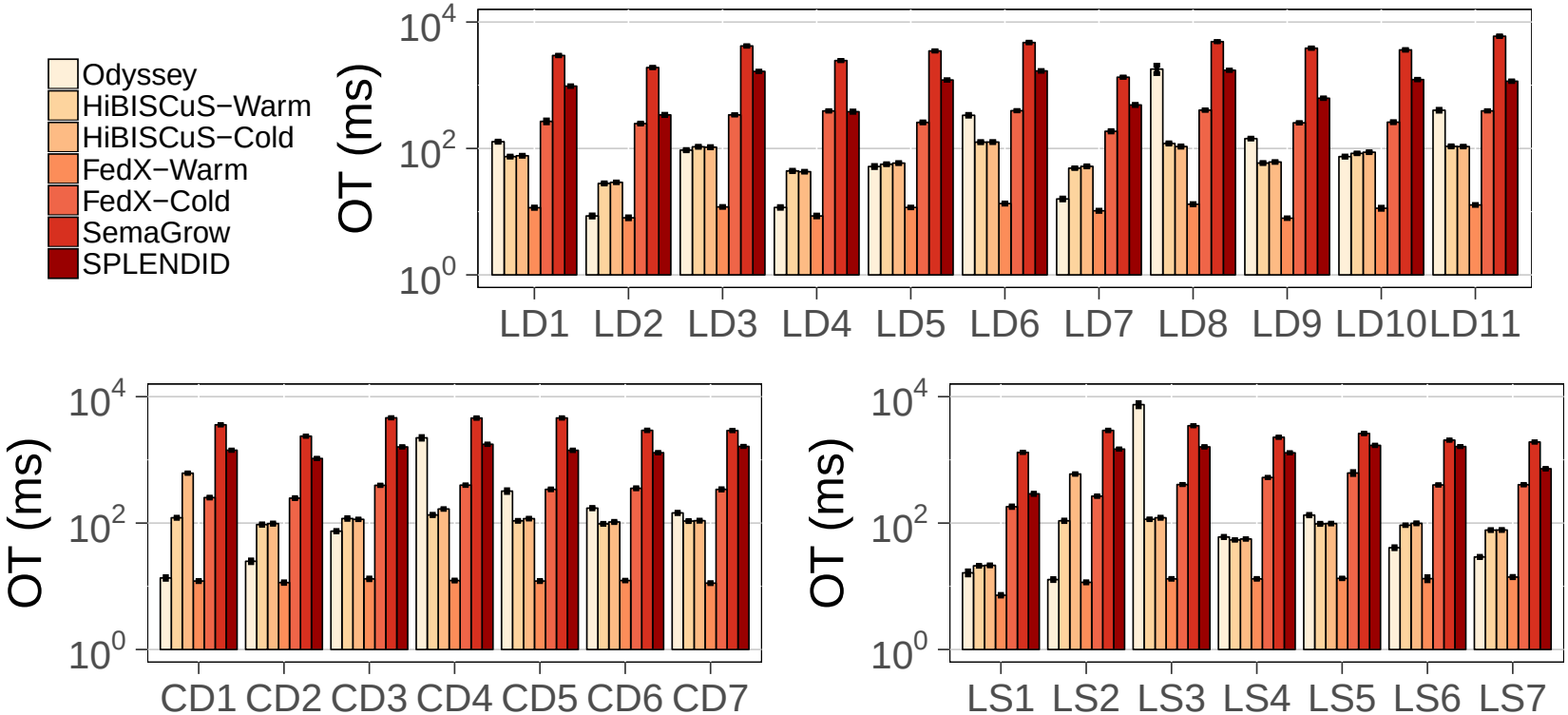
[9] A. Schwarte et al. "FedX: Optimization Techniques for Federated Query Processing on Linked Data". In: *ISWC'11*.

[10] A. Charalambidis et al. "SemaGrow: optimizing federated SPARQL queries". In: *SEMANTICS'15*.

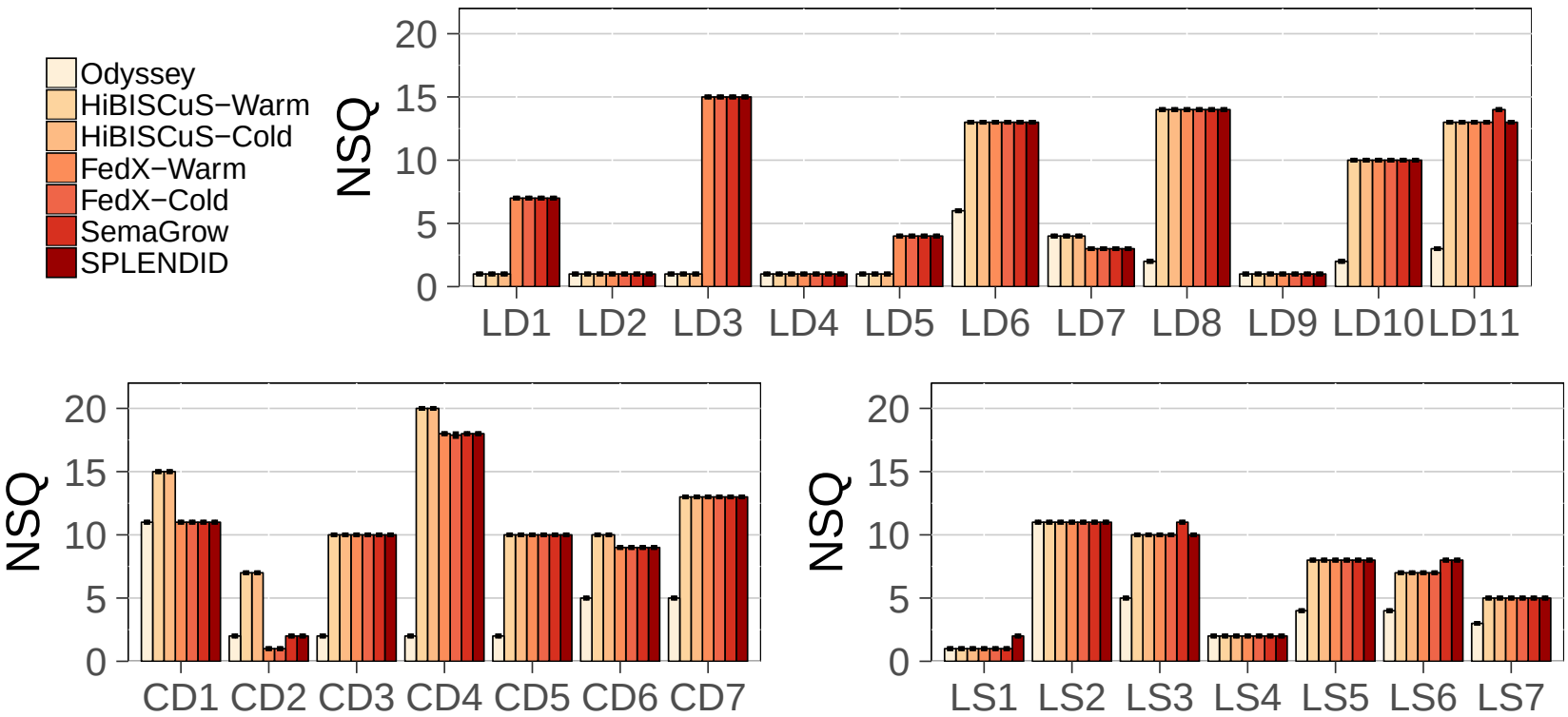
[11] O. Görlitz and S. Staab. "SPLENDID: SPARQL Endpoint Federation Exploiting VOID Descriptions". In: *COLD'11*.

[12] M. Saleem and A. N. Ngomo. "HiBISCuS: Hypergraph-Based Source Selection for SPARQL Endpoint Federation". In: *ESWC'14*.

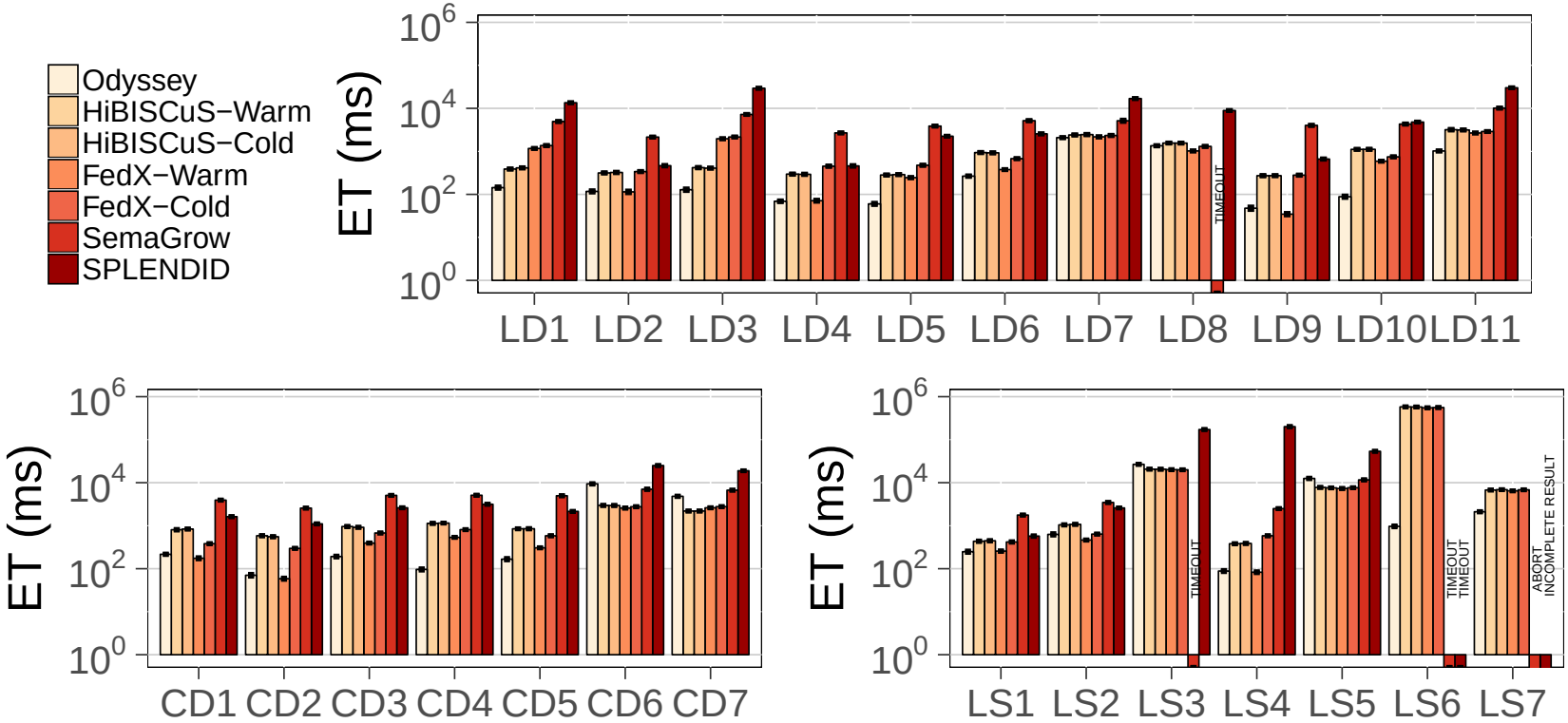
# Odyssey's Optimization Time is comparable with other approaches'



# Odyssey's plans have less subqueries than other approaches' plans



# Odyssey's plans are overall executed faster than other approaches' plans



# Conclusions

- Odyssey uses cardinality estimations that allow for better optimizations.
- Local statistics allow to discover connections between datasets in a federated setup.
- Odyssey's plans are in general better than existing approaches' plans.

# Future works

- Odyssey's optimization can be improved using with runtime optimizations (ASK queries).
- Reduce the local statistics computation times and sizes.
- Provide efficient strategies to update the statistics.

# Questions?