1 Query Federation

A query in a query federation system goes through the following stages

1. Parsing
2. Query Planning: finding relevant sources and feasible sub-queries
   (a) Source description
      - Statistical information
        – helps query optimizer to find cost-effective query execution plan

   (b) Build Sub-Queries
      i. Query Rewriting
3. Optimization
   (a) Logical Optimization: uses equalities of query expression to transform a logical query into an equivalent query plan that is likely to be executed faster or with less costs
   (b) Physical Optimization: has the goal to find the best query execution plan among all possible plans. The efficacy of a plan is quantified using a cost model
4. Query Costing
   (a) Cost Model
      - Cardinality
      - Selectivity
5. Query Execution
   - Adaptive Query execution
   - Result Merging

2 DARQ

2.1 Source Description

- describes the data available from a data source in form of capabilities
- capabilities define what kind of triple patterns can be answered by the data source
- definition of capabilities is based on predicate
- capabilities of a data source d is a set C_d where $(p, r) \in C_d$, $r : (T \cup V) \times (T \cup V) \longrightarrow \{True, False\}$, $p \in Voc(d)$ is a predicate in d

- definition of limitations on access patterns for data source d
  - let L_d be a set of limitations on access patterns
  - $(S, O) \in L_d$ be one pattern with S and O where
    - S: set of predicates that must have bound subject
    - O: set of predicates that must have bound object
- $$bound(x) = \begin{cases} False & \text{if } x \in V \\ True & \text{otherwise} \end{cases} \quad (1)$$
- a Source d contribute to the query answer of a query with graph pattern P if Source d satisfies at least of the defined access patterns for d
- an access patterns (S,O) is satisfied if $(\forall p_s \in S \setminus O : \exists (s, p_s, o) \in P : bound(s)) \wedge (\forall p_o \in O \setminus S : \exists (s, p_o, o) \exists P : bound(o)) \wedge (\forall p_b \exists S \cap O : \exists (s, p_b, o) \in P : bound(s) \wedge bound(o))$

2.1.1 Statistical Information

Service description include:

- N_s: total number of triples in data source d
- $n_d(p)$: total number of triples with predicate p in data source d
- $ssel_d(p)$: selectivity of a triple pattern with predicate p if subject is bound (default $\frac{1}{n_d(p)}$)
- $osel_d(p)$: selectivity of a triple pattern with predicate p if object is bound (default 1)

2.2 Query Planning

2.2.1 Source Selection

- query planning is based on information provided in service description
- let $R = \{(d_i, C_i) \mid d_i \in D\}$
- SPARQL query contains one or more Filtered Basic Graph Pattern (FBGP)
- Query planning is performed separately for each FBGP
- algorithm match all triple patterns against capabilities of data sources
- matching for each triple (s, p, o)
  - get capability for predicate p: (p, r) in C_i for data source d_i

  - replace s and o in function r and select data source d_i if $true$ is returned

- Result of source select is a set of data sources for each triple pattern $D_j = \{d \mid (d, C)\ \exists R\langle\ \exists(p_j, r)\ \ \in C : r(s_j, o_j) = true\}$

2.2.2 Building Sub-Queries

- result from source selection are used to build sub-queries that can be answered by data sources
- Sub-queries consist of one filtered basic graph pattern per data source
- A sub-query is a triple (T, C, d) where T is a set of triple patterns, C is a set of value constraints and d is the data source that can answer the sub-query
- if a triple can be answered only by one data source, the triple will be added to the set of sub-query for this data source. All triples in this set can be combined in one sub-query
- If triple matches several data sources, it must be sent individually to all matching data sources in separate sub-queries

2.3 Query Optimization

2.3.1 Logical Optimization

- Use rules based on SPARQL Semantics to transform a logical query plan into an equivalent query plan that is likely to be executed faster or with less costs
- Move possible value constraints into sub-queries to reduce size of intermediate results as early as possible
- Let $Q = T, C, d$ be a sub-query and $FGP = (T', C')$ a filtered basic graph pattern. C' can be moved into the sub-query
  - if all variables in the constraint are also used in the triple patterns of the sub-query
  - if Filters contains variable from more than one sub-queries, it cannot be split and must thus be applied locally inside DARQ

2.3.2 Physical Optimization

- Aim in DARQ is to:
  1. Reduce amount of transferred data
  2. Reduce number of transmission which will lead to less transfer costs and faster query execution

- Two joins implementations are supported:
  1. nested-loop join ($\bowtie$)
  2. bind join ($\bowtie_B$)

2.3.3 Cost Model

- $|R(q_1) \bowtie R(q_2)| = |R(q_1)|c_t + |R(q_2|c_t + 2C_r$ where
  - q_1 and q_2 can be a sub-query or join
  - $|R(q)|$ is the result size of q
  - sel_{12} is the selectivity factor for join attributes
  - $sel_{12} = 0.5$ due to lack of enough information for better estimation
- Transfer cost of a nested loop join
  - $C(q_1 \bowtie q_2) = |R(q_1)|c_t + |R(q_2)|c_t + 2c_r$
- Transfer cost of a bind loop join
  - $C(q_1 \bowtie_B q_2) = |R(q_1)|c_t + |R(q_1)|c_r + |R(q_2')|c_t$
- c_t: transfer cost for one tuple (tuple size is ignored)
- c_r: transfer cost for one query
- q_2': the query with variables bound with values of a result from q_1
- Query result size estimation
  - based on statistics provided in the service description
  - Service description include for each capability:
    - $n_d(p)$: number of triples with predicate p in data source d
    - $ssel_d(p)$: average selectivity if subject is bound
    - $osel_d(p)$: average selectivity if object is bound
  - Cost of a single triple pattern sent to a data source d:
    - $costs_d((s,p,o),b)$
      - if $\neg bound(s,b) \wedge \neg bound(o,b)$ then $costs_d((s,p,o),b) = n_d(p)$
      - if $\neg bound(s,b) \wedge bound(o,b)$ then $costs_d((s,p,o),b) = n_d(p) * osel_d(p)$
      - if $bound(s,b) \wedge \neg bound(o,b)$ then $costs_d((s,p,o),b) = n_d(p) * ssel_d(p)$
      -
      - if $bound(s,b) \wedge bound(o,b)$ then $costs_d((s,p,o),b) = n_d(p) * 0.5$

* Estimating result size of two or more triple pattern is more complex as adding a triple pattern to a query can increase or reduce the size of the result
  - adding more triple with same subject will not introduce new results
  - adding triple patterns with another subject may increase results