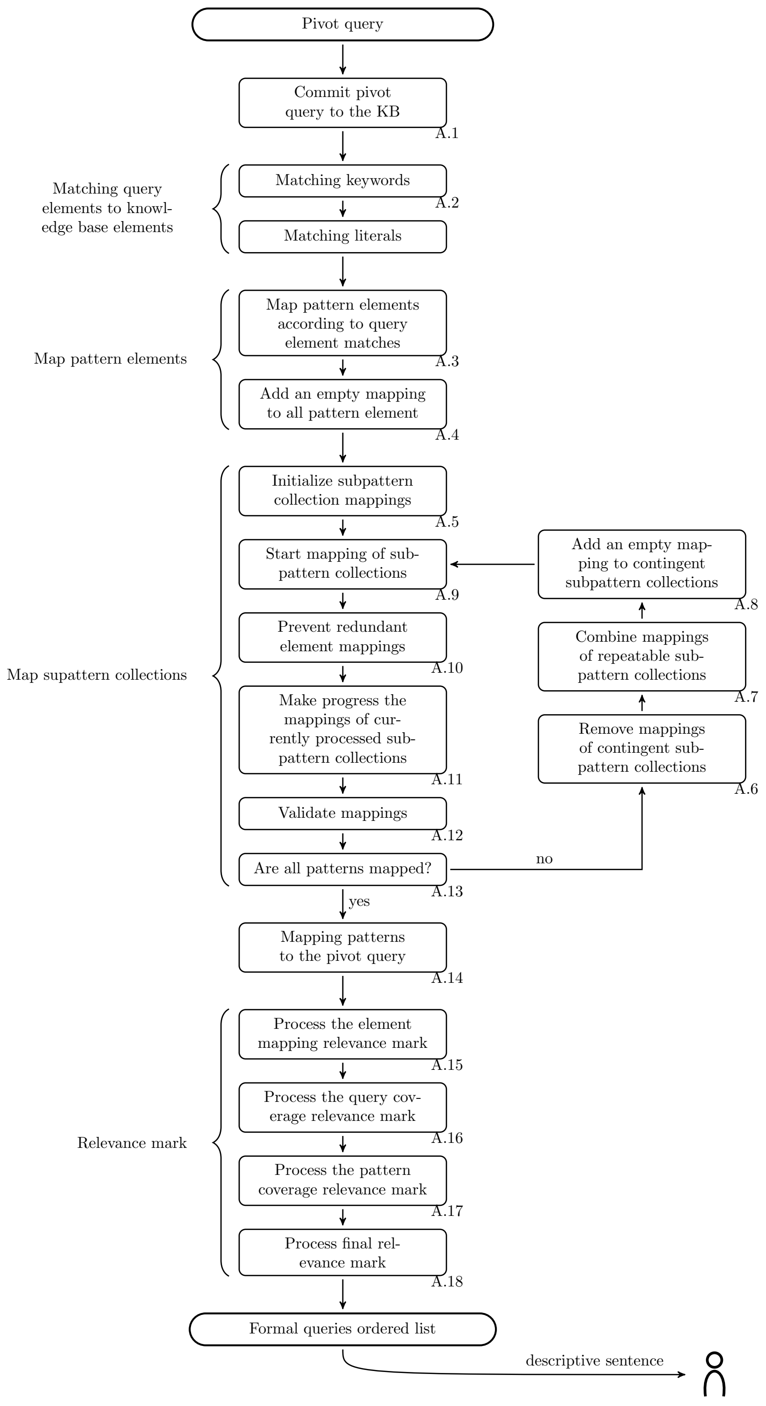
Pivot query
Commit pivot query to the KB
A.1
Matching query elements to knowledge base elements
Matching keywords
A.2
Matching literals
Map pattern elements
Map pattern elements according to query element matches
A.3
Add an empty mapping to all pattern element
A.4
Map supattern collections
Initialize subpattern collection mappings
A.5
Start mapping of subpattern collections
A.9
Prevent redundant element mappings
A.10
Make progress the mappings of currently processed subpattern collections
A.11
Validate mappings
A.12
Are all patterns mapped?
A.13
no
Remove mappings of contingent subpattern collections
A.6
Combine mappings of repeatable subpattern collections
A.7
Add an empty mapping to contingent subpattern collections
A.8
yes
Mapping patterns to the pivot query
A.14
Relevance mark
Process the element mapping relevance mark
A.15
Process the query coverage relevance mark
A.16
Process the pattern coverage relevance mark
A.17
Process final relevance mark
A.18
Formal queries ordered list
descriptive sentence

A SPARQL queries used for pivot query interpretation

This appendix presents the SPARQL queries involved in the pivot query interpretation process.

For readability reasons, in each query, the URI of the currently processed query is replaced by the string “[queryUri]” and used prefixes are always the same; their values are given here:

```
PREFIX patterns:  <http://swip.univ-tlse2.fr/ontologies/Patterns#>
PREFIX queries:   <http://swip.univ-tlse2.fr/ontologies/Queries#>
PREFIX graph:   <http://swip.univ-tlse2.fr:8080/musicbrainz/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX pf:    <http://jena.hpl.hp.com/ARQ/property#>
PREFIX xsd:   <http://www.w3.org/2001/XMLSchema#>
```

A.1 Commit a pivot query.

SPARQL Update used to commit the pivot query in the KB.

```
# commit query [queryUri]
PREFIX queries:   <http://swip.univ-tlse2.fr/ontologies/Queries#>
PREFIX graph:   <http://swip.univ-tlse2.fr:8080/musicbrainz/>
INSERT DATA
{
  GRAPH graph:queries
  {
    <[queryUri]> a queries:PivotQuery;
                 queries:queryHasQueryElement graph:person;
                 queries:queryHasQueryElement graph:produce;
                 queries:queryHasQueryElement graph:album;
                 queries:queryHasQueryElement graph:Dookie;
                 queries:queryHasSubquery <[queryUri]/sq1>;
                 queries:queryHasSubquery <[queryUri]/sq2>.
    graph:person a queries:KeywordQueryElement;
                 queries:queryElementHasValue "person".
    graph:produce a queries:KeywordQueryElement;
                  queries:queryElementHasValue "produce".
    graph:album a queries:KeywordQueryElement;
                queries:queryElementHasValue "album".
    graph:Dookie a queries:KeywordQueryElement;
                 queries:queryElementHasValue "Dookie".
    <[queryUri]/sq1> a queries:Q2;
                     queries:subqueryHasE1 <[queryUri]/Dookie>;
                     queries:subqueryHasE2 <[queryUri]/album>.
    <[queryUri]/sq2> a queries:Q3;
                     queries:subqueryHasE1 <[queryUri]/person>;
                     queries:subqueryHasE2 <[queryUri]/produce>;
                     queries:subqueryHasE3 <[queryUri]/Dookie>.
  }
}
```

A.2 Match query elements to KB resources.

SPARQL Update executed in order to find out and store the matched KB elements for each keyword of the query.

```
# matching keywords to KB labels
PREFIX queries:   <http://swip.univ-tlse2.fr/ontologies/Queries#>
PREFIX graph:   <http://swip.univ-tlse2.fr:8080/musicbrainz/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX pf:    <http://jena.hpl.hp.com/ARQ/property#>
PREFIX xsd:   <http://www.w3.org/2001/XMLSchema#>
INSERT
{
  GRAPH graph:queries
  {
    ?matchUri a queries:Matching;
                queries:matchingHasKeyword ?keyword;
                queries:matchingHasResource ?r;
                queries:matchingHasScore ?s;
                queries:matchingHasMatchedLabel ?l.
   ?keyword queries:keywordAlreadyMatched "true"^^xsd:boolean.
  }
}
WHERE
{
  {
    # subquery with DISTINCT to solve unidentified problem apparently due to larq
    SELECT DISTINCT * WHERE
    {
      GRAPH graph:queries
      {
        <[queryUri]> queries:queryHasQueryElement ?keyword.
        ?keyword a queries:KeywordQueryElement;
                 queries:queryElementHasValue ?keywordValue.
        FILTER NOT EXISTS { ?keyword queries:keywordAlreadyMatched "true"^^xsd:boolean. }
      }
      {
        GRAPH graph:ontologies
        {
          (?l ?score) pf:textMatch (?keywordValue 6.0 5).
          ?r rdfs:label ?l.
          BIND ((?score * 2) AS ?s)
        }
      }
      UNION
      {
        GRAPH graph:instances
        {
          (?l ?score) pf:textMatch (?keywordValue 6.0 5).
          ?r rdfs:label ?l.
          BIND ((?score * 1) AS ?s)
        }
      }
    }
  }
  BIND (UUID() AS ?matchUri)
}
```

A.3 Map pattern elements according to query element matches

SPARQL Update used to

```
# mapping KB pattern elements to keywords
PREFIX patterns:   <http://swip.univ-tlse2.fr/ontologies/Patterns#>
PREFIX queries:   <http://swip.univ-tlse2.fr/ontologies/Queries#>
PREFIX graph:   <http://swip.univ-tlse2.fr:8080/musicbrainz/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
INSERT
{
  GRAPH graph:queries
  {
    ?emUri a queries:ElementMapping;
             queries:emHasPatternElement ?pe;
             queries:mappingHasPatternConstituent ?pe;
             queries:emHasMatching ?matching;
             queries:emHasScore ?score;
             queries:mappingHasQuery <[queryUri]>;
             queries:hasDescriptiveSubsentence ?descSentUri.
    ?descSentUri a queries:DescriptiveSubsentence;
                   queries:hasStringValue ?stringValue.
  }
}
WHERE
{
  GRAPH graph:queries
  {
    <[queryUri]> queries:queryHasQueryElement ?keyword.
    ?matching queries:matchingHasKeyword ?keyword;
              queries:matchingHasResource ?r;
              queries:matchingHasScore ?score;
              queries:matchingHasMatchedLabel ?l.
  }
  GRAPH graph:patterns
  {
    ?p patterns:patternHasPatternElement ?pe.
    ?pe patterns:targetsKBElement ?kbe.
  }
  {
    GRAPH graph:ontologies
    {
      {?r rdfs:subClassOf ?kbe.
      BIND ("some_" AS ?stringValuePrefix)}
      UNION
      {?r rdfs:subPropertyOf ?kbe.}
    }
  }
  UNION
  {
    GRAPH graph:instances
    {
      ?r a ?c.
    }
    GRAPH graph:ontologies
    {
      ?c rdfs:subClassOf ?kbe.
    }
  }
  BIND (UUID() AS ?emUri)
  BIND (UUID() AS ?descSentUri)
  BIND (if (BOUND(?stringValuePrefix), CONCAT(?stringValuePrefix, ?l), STR(?l)) AS ?stringValue)
}
```

A.4 Add an empty mapping to all pattern elements

SPARQL Update used to

```
# add an empty mapping to all pattern elements
PREFIX patterns: <http://swip.univ-tlse2.fr/ontologies/Patterns#>
PREFIX queries: <http://swip.univ-tlse2.fr/ontologies/Queries#>
PREFIX graph: <http://swip.univ-tlse2.fr:8080/musicbrainz/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
INSERT
{
  GRAPH graph:queries
  {
    ?emUri a queries:EmptyElementMapping;
           a queries:ElementMapping;
           queries:emHasPatternElement ?pe;
           queries:mappingHasPatternConstituent ?pe;
           queries:mappingHasQuery <[queryUri]>;
           queries:emHasScore "0"^^xsd:float;
           queries:hasDescriptiveSubsentence ?descSentUri.
    ?descSentUri a queries:DescriptiveSubsentence;
                   queries:hasStringValue ?l.
    ?pe queries:toConsiderInMappingQuery <[queryUri]>.
  }
}
WHERE
{
  GRAPH graph:patterns
  {
    ?pe a patterns:PatternElement;
        patterns:targetsKBElement ?kbe.
    OPTIONAL { ?kbe rdfs:label ?label. }
  }
  BIND (UUID() AS ?emUri)
  BIND (UUID() AS ?descSentUri)
  BIND (IF( BOUND(?label), CONCAT( "(a/an)", ?label), CONCAT( "no_label_found_for_", STR(?kbe))) AS
}
```

A.5 Initialize subpattern collection mappings

SPARQL Update ...

```
# specify for all subpatterns that they have not yet been mapped to the current query
PREFIX patterns:   <http://swip.univ-tlse2.fr/ontologies/Patterns#>
PREFIX queries:   <http://swip.univ-tlse2.fr/ontologies/Queries#>
PREFIX graph:   <http://swip.univ-tlse2.fr:8080/musicbrainz/>
INSERT
{
  GRAPH graph:queries
  {
    ?spc queries:isNotMappedToQuery <[queryUri]>.
  }
}
WHERE
{
  GRAPH graph:patterns
  {
    ?spc a patterns:SubpatternCollection.
  }
}
```

A.6 Remove mappings of contingent subpattern collections containing only empty mappings

SPARQL Update ...

```
# remove mappings of contingent subpattern collections containing only empty mappings
# an empty mapping for the contingent subpattern collection itself will be added later
PREFIX patterns:   <http://swip.univ-tlse2.fr/ontologies/Patterns#>
PREFIX queries:   <http://swip.univ-tlse2.fr/ontologies/Queries#>
PREFIX graph:   <http://swip.univ-tlse2.fr:8080/musicbrainz/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
DELETE
{
  GRAPH graph:queries
  {
    ?m ?a ?b.
  }
}
WHERE
{
  GRAPH graph:patterns
  {
    ?spc patterns:hasCardinalityMin "0"^^xsd:int.
  }
  GRAPH graph:queries
  {
    ?spc queries:toConsiderInMappingQuery <[queryUri]>.
    ?m queries:mappingHasQuery <[queryUri]>;
       queries:mappingHasPatternConstituent ?spc;
       ?a ?b.
    FILTER NOT EXISTS
    {
      ?m queries:mappingContainsMapping+ ?em.
      ?em queries:emHasMatching ?matching.
    }
  }
}
```

A.7 Combine mappings

SPARQL Update ...

```
# combine mappings of repeatable subpattern collections
PREFIX patterns:  <http://swip.univ-tlse2.fr/ontologies/Patterns#>
PREFIX queries:  <http://swip.univ-tlse2.fr/ontologies/Queries#>
PREFIX graph:  <http://swip.univ-tlse2.fr:8080/musicbrainz/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd:  <http://www.w3.org/2001/XMLSchema#>
INSERT
{
  GRAPH graph:queries
  {
    ?mappingUri a queries:SubpatternCollectionMapping;
                queries:mappingHasPatternConstituent ?spc;
                queries:mappingContainsMapping ?m1, ?m2;
                queries:mappingHasQuery <[queryUri]>.
    ?spc queries:allreadyCombinedForQuery <[queryUri]>.
    # process the descriptive sentence
    ?mappingUri queries:hasDescriptiveSubsentence ?descSubsent.
    ?descSubsent a queries:DescriptiveSubsentence;
                 queries:isMadeUpOfList ?list;
                 queries:isMadeUpOfList-for ?listFor1, ?listFor2.
  }
}
WHERE
{
  GRAPH graph:patterns
  {
    SELECT DISTINCT ?spc WHERE
    {
      ?spc patterns:hasCardinalityMax "2"^^xsd:int.
    }
  }
  GRAPH graph:queries
  {
    ?spc queries:toConsiderInMappingQuery <[queryUri]>.
    FILTER NOT EXISTS
    {
      ?spc queries:allreadyCombinedForQuery <[queryUri]>.
    }
    ?m1 queries:mappingHasPatternConstituent ?spc.
    ?m2 queries:mappingHasPatternConstituent ?spc.
    FILTER ( STR(?m1) < STR(?m2) )
    # process the descriptive sentence
    ?m1 (queries:hasDescriptiveSubsentence/queries:isMadeUpOfList) ?list.
    ?m1 (queries:hasDescriptiveSubsentence/queries:isMadeUpOfList-for) ?listFor1.
    ?m2 (queries:hasDescriptiveSubsentence/queries:isMadeUpOfList-for) ?listFor2.
  }
  BIND (UUID() AS ?mappingUri)
  BIND (UUID() AS ?descSubsent)
}
```

A.8 Add an empty mapping to contingent subpattern collections

SPARQL Update ...

```
# add an empty mapping to ContingentSubpatternCollections to be considered in next mappings
PREFIX patterns:  <http://swip.univ-tlse2.fr/ontologies/Patterns#>
PREFIX queries:  <http://swip.univ-tlse2.fr/ontologies/Queries#>
PREFIX graph:  <http://swip.univ-tlse2.fr:8080/musicbrainz/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd:  <http://www.w3.org/2001/XMLSchema#>
INSERT
{
  GRAPH graph:queries
  {
    ?mappingUri a queries:EmptySubpatternCollectionMapping;
                a queries:SubpatternCollectionMapping;
```

```
                 queries:mappingHasPatternConstituent ?spc;
                 queries:mappingHasQuery <[queryUri]>;
                 queries:hasDescriptiveSubsentence ?descSentUri.
    ?descSentUri a queries:DescriptiveSubsentence;
                   queries:hasStringValue "(EmptySubpatternCollectionMapping)".
  }
}
WHERE # select all ContingentSubpatternCollections which don't have yet an EmptySubpatternCollectio
{
  GRAPH graph:queries
  {
    ?spc queries:toConsiderInMappingQuery <[queryUri]>.
    FILTER NOT EXISTS
    {
      ?mapping a queries:EmptySubpatternCollectionMapping;
                 queries:mappingHasPatternConstituent ?spc;
                 queries:mappingHasQuery <[queryUri]>.
    }
  }
  GRAPH graph:patterns
  {
    SELECT DISTINCT ?spc WHERE
    {
      ?spc a patterns:SubpatternCollection;
           patterns:hasCardinalityMin "0"^^xsd:int.
    }
  }
  BIND (UUID() AS ?mappingUri)
}
```

A.9 Start mapping of SubpatternCollections whose all contained components are already mapped

SPARQL Update ...

```
# start mapping of SubpatternCollections whose all contained components are already mapped
PREFIX patterns:   <http://swip.univ-tlse2.fr/ontologies/Patterns#>
PREFIX queries:   <http://swip.univ-tlse2.fr/ontologies/Queries#>
PREFIX graph:   <http://swip.univ-tlse2.fr:8080/musicbrainz/>
DELETE
{
  GRAPH graph:queries
  {
    ?spc queries:isNotMappedToQuery <[queryUri]>.
  }
}
INSERT
{
  GRAPH graph:queries
  {
    ?spc queries:isBeingMappedToQuery <[queryUri]>.
  }
}
WHERE
{
  {
    # get all SubpatternCollections which are not mapped but whose contained SubpatternCollections
    SELECT ?spc WHERE
    {
      # all SubpatternCollections
      GRAPH graph:patterns
      {
        ?spc a patterns:SubpatternCollection.
      }
      # which are not mapped
      GRAPH graph:queries
      {
        ?spc queries:isNotMappedToQuery <[queryUri]>.
      }
      # whose contained SubpatternCollections are all 'to be considered'
      MINUS
```

```
        {
          # get all SubpatternCollections whose at least one contained SubpatternCollection is being
          SELECT ?spc WHERE
          {
            GRAPH graph:patterns
            {
              ?spc a patterns:SubpatternCollection;
                   patterns:hasDirectSubpattern ?spc2.
              ?spc2 a patterns:SubpatternCollection;
            }
            GRAPH graph:queries
            {
              ?spc2 (queries:isNotMappedToQuery | queries:isBeingMappedToQuery) <[queryUri]>.
            }
          }
        }
      }
    }
}
```

A.10 Prevent redundant element mappings

SPARQL Update ...

```
# prevent from adding in a subpattern collection mapping a element mapping
# relative to a pattern element appearing outside of this subpattern collection
PREFIX patterns:   <http://swip.univ-tlse2.fr/ontologies/Patterns#>
PREFIX queries:   <http://swip.univ-tlse2.fr/ontologies/Queries#>
PREFIX graph:   <http://swip.univ-tlse2.fr:8080/musicbrainz/>
DELETE
{
  GRAPH graph:queries
  {
    ?pc queries:toConsiderInMappingQuery <[queryUri]>.
  }
}
INSERT
{
  GRAPH graph:queries
  {
    ?pc queries:toConsiderNextStepInMappingQuery <[queryUri]>.
  }
}
WHERE
{
  SELECT DISTINCT ?pc WHERE
  {
    GRAPH graph:queries
    {
      ?spc queries:isBeingMappedToQuery <[queryUri]>.
      ?pc queries:toConsiderInMappingQuery <[queryUri]>.
    }
    GRAPH graph:patterns
    {
      ?spc patterns:isMadeUpOf ?pc.
      OPTIONAL
      {
        ?spc2 patterns:isMadeUpOf ?pc.
        FILTER ( !sameTerm(?spc, ?spc2) )
      }
      FILTER NOT EXISTS
      {
        { ?spc patterns:isMadeUpOf ?spc2. }
        UNION
        { ?spc2 patterns:isMadeUpOf ?spc. }
      }
    }
  }
}
```

A.11 Make progress the mappings of currently processed SubpatternCollections

SPARQL Update ...

```
# make progress the mappings of currently processed SubpatternCollections
PREFIX patterns:  <http://swip.univ-tlse2.fr/ontologies/Patterns#>
PREFIX queries:  <http://swip.univ-tlse2.fr/ontologies/Queries#>
PREFIX graph:  <http://swip.univ-tlse2.fr:8080/musicbrainz/>
DELETE
{
  GRAPH graph:queries
  {
    ?pc queries:toConsiderInMappingQuery <[queryUri]>.
    ?pc queries:allreadyCombinedForQuery <[queryUri]>.
    # to not consider these mappings anymore
    ?mPc queries:mappingHasPatternConstituent ?pc.
    ?mSpc queries:mappingHasPatternConstituent ?spc.
  }
}
INSERT
{
  GRAPH graph:queries
  {
    ?pc queries:isMappedToQuery <[queryUri]>.
    ?mappingUri a queries:SubpatternCollectionMapping;
                 queries:mappingHasPatternConstituent ?spc;
                 queries:mappingContainsMapping ?mSpc, ?mPc;
                 queries:mappingHasQuery <[queryUri]>.
    # used for descriptive sentence generation
    ?mPc queries:mappingHadPatternConstituent ?pc.
    ?mSpc queries:mappingHadPatternConstituent ?spc.
  }
}
WHERE
{
  {
    # select a pair (?spc, ?pc) such as
    # - ?spc is a SubpatternCollection made of ?pc
    # - ?spc is being mapped
    # - ?pc is to be considered
    SELECT ?spc ?pc WHERE
    {
      GRAPH graph:queries
      {
        ?pc queries:toConsiderInMappingQuery <[queryUri]>.
        ?spc queries:isBeingMappedToQuery <[queryUri]>.
      }
      GRAPH graph:patterns
      {
        ?spc patterns:isMadeUpOf ?pc.
      }
    } ORDER BY ?spc ?pc LIMIT 1
  }
  GRAPH graph:queries
  {
    OPTIONAL
    {
      ?mSpc queries:mappingHasPatternConstituent ?spc;
            queries:mappingHasQuery <[queryUri]>.
    }
    ?mPc queries:mappingHasPatternConstituent ?pc;
         queries:mappingHasQuery <[queryUri]>.
  }
  BIND (UUID() AS ?mappingUri)
}
```

A.12 Validate mappings

SPARQL Update ...

```
# validate mappings of SubpatternCollections which are being mapped and whose contained Subpattern
# change the status of toConsiderNextStepInMappingQuery SubpatternCollections into toConsiderInMapp
PREFIX patterns:  <http://swip.univ-tlse2.fr/ontologies/Patterns#>
PREFIX queries:  <http://swip.univ-tlse2.fr/ontologies/Queries#>
PREFIX graph:  <http://swip.univ-tlse2.fr:8080/musicbrainz/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
DELETE
{
  GRAPH graph:queries
  {
    ?spc queries:isBeingMappedToQuery <[queryUri]>.
    ?comp queries:toConsiderNextStepInMappingQuery <[queryUri]>.
  }
}
INSERT
{
  GRAPH graph:queries
  {
    ?spc queries:toConsiderInMappingQuery <[queryUri]>.
    ?comp queries:toConsiderInMappingQuery <[queryUri]>.
    # process the descriptive sentence
    ?spcm queries:hasDescriptiveSubsentence ?descSubsent.
    ?descSubsent a queries:DescriptiveSubsentence;
                 ?madeUpRelation ?listSerialized .
    ?oneListSerialized rdf:first ?oneElementSerializedSsit ;
                       rdf:first ?oneElementSerializedFlit ;
                       rdf:first ?oneElementSerializedOther ;
                       rdf:firsttt ?plop ;
                       rdf:rest ?nextListSerialized .
    ?oneElementSerializedSsit queries:hasStringValue ?ssitValue.
    ?oneElementSerializedFlit queries:insertForSubsentenceOf ?spc.
  }
}
WHERE
{
  {
    # get all SubpatternCollections which are being mapped and whose contained SubpatternCollection
    SELECT ?spc WHERE
    {
      # all SubpatternCollections
      GRAPH graph:patterns
      {
        ?spc a patterns:SubpatternCollection.
      }
      # which are being mapped
      GRAPH graph:queries
      {
        ?spc queries:isBeingMappedToQuery <[queryUri]>.
      }
      # whose contained SubpatternCollections are all mapped or to be mapped in next step
      MINUS
      {
        # get all SubpatternCollections whose at least one contained component is not mapped
        SELECT DISTINCT ?spc WHERE
        {
          GRAPH graph:patterns
          {
            ?spc a patterns:SubpatternCollection;
                 patterns:isMadeUpOf ?comp.
          }
          GRAPH graph:queries
          {
            ?comp queries:toConsiderInMappingQuery <[queryUri]>.
          }
        }
      }
    }
  }
  # change the status of toConsiderNextStepInMappingQuery SubpatternCollections into toConsiderInMa
  OPTIONAL
  {
    GRAPH graph:patterns
    {
      ?spc patterns:isMadeUpOf ?comp.
    }
    GRAPH graph:queries
    {
      ?comp queries:toConsiderNextStepInMappingQuery <[queryUri]>.
    }
  }
  # process the descriptive sentence
  GRAPH graph:queries
  {
    ?spcm queries:mappingHasPatternConstituent ?spc.
  }
  GRAPH graph:patterns
  {
    ?sst (patterns:sstTargetsPc|^patterns:hasSentenceTemplate) ?spc ;
         patterns:isMadeUpOfList ?list ;
    # when dealing with spc containing a for loop, two subsentences are generated (one for the inne
    OPTIONAL {
      ?sst a patterns:ForLoopInTemplate.
      BIND (queries:isMadeUpOfList-for AS ?madeUpRelation).
    }
    OPTIONAL {
      ?sst a patterns:SpcInTemplate.
      BIND (queries:isMadeUpOfList AS ?madeUpRelation).
    }
    ?list rdf:rest* ?oneList .
    ?oneList rdf:first ?oneElement ;
             rdf:rest ?nextList .
  }
  # oneElement is a static string
  OPTIONAL
  {
    GRAPH graph:patterns { ?oneElement patterns:ssitHasValue ?ssitValue.
                           BIND (UUID() AS ?oneElementSerializedSsit) }
  }
  # oneElement is a ForLoopInTemplate
  OPTIONAL
  {
    GRAPH graph:patterns { ?oneElement a patterns:ForLoopInTemplate.
                           BIND (UUID() AS ?oneElementSerializedFlit) }
  }
  # oneElement is a PeInTemplate or a SpcInTemplate
  OPTIONAL
  {
    GRAPH graph:patterns { {?oneElement a patterns:PeInTemplate.} UNION {?oneElement a patterns:Spc
                           ?oneElement patterns:sstTargetsPc ?pc. }
    GRAPH graph:queries { ?spcm queries:mappingContainsMapping+ ?sspcm.
                          ?sspcm queries:mappingHadPatternConstituent ?pc;
                                 queries:hasDescriptiveSubsentence ?oneElementSerializedOther.
                          FILTER NOT EXISTS # because of combined mappings which maps same spc as t
                          # FIXME: this part makes the query very slow (for something that looks si
                          {
                            ?spcm queries:mappingContainsMapping+ ?sspcm2.
                            ?sspcm2 queries:mappingHadPatternConstituent ?pc;
                                    queries:mappingContainsMapping ?sspcm.
                          }}
    BIND (<SpcInTemplate> AS ?plop)
  }
  BIND (IRI(CONCAT(str(?spcm), "_descSubsent")) AS ?descSubsent)
  BIND (IRI(CONCAT(str(?spcm), str(?list))) AS ?listSerialized)
  BIND (IRI(CONCAT(str(?spcm), str(?oneList))) AS ?oneListSerialized)
  BIND (IRI(CONCAT(str(?spcm), str(?nextList))) AS ?nextListSerialized)
}
```

A.13 Are all patterns mapped?

SPARQL Ask used to

```
# are all patterns mapped to the current query?
PREFIX patterns:   <http://swip.univ-tlse2.fr/ontologies/Patterns#>
PREFIX queries:    <http://swip.univ-tlse2.fr/ontologies/Queries#>
PREFIX graph:   <http://swip.univ-tlse2.fr:8080/musicbrainz/>
ASK
```

```
{
  GRAPH graph:patterns
  {
    ?p a patterns:Pattern.
  }
  FILTER NOT EXISTS
  {
    GRAPH graph:queries { ?p queries:toConsiderInMappingQuery <[queryUri]>. }
  }
}
```

A.14 Mapping patterns to the pivot query

```
# perform pattern mapping
PREFIX patterns:  <http://swip.univ-tlse2.fr/ontologies/Patterns#>
PREFIX queries:  <http://swip.univ-tlse2.fr/ontologies/Queries#>
PREFIX graph:  <http://swip.univ-tlse2.fr:8080/musicbrainz/>
DELETE
{
  GRAPH graph:queries
  {
    ?p queries:toConsiderInMappingQuery <[queryUri]>.
  }
}
INSERT
{
  GRAPH graph:queries
  {
    ?p queries:isMappedToQuery <[queryUri]>.
    ?pm a queries:PatternMapping.
  }
}
WHERE
{
  GRAPH graph:queries
  {
    ?pm queries:mappingHasPatternConstituent ?p;
        queries:mappingHasQuery <[queryUri]>.
    FILTER NOT EXISTS {?pm2 queries:mappingContainsMapping ?pm.}
  }
  GRAPH graph:patterns
  {
    ?p a patterns:Pattern.
  }
}
```

A.15 Element mapping relevance mark

```
# process the element mapping relevance mark
# step 1: process the average score of involved matchings
PREFIX patterns:  <http://swip.univ-tlse2.fr/ontologies/Patterns#>
PREFIX queries:  <http://swip.univ-tlse2.fr/ontologies/Queries#>
PREFIX graph:  <http://swip.univ-tlse2.fr:8080/musicbrainz/>
INSERT
{
  GRAPH graph:queries
  {
    ?pm queries:hasEmAvg ?avgscore;
  }
}
WHERE
{
  SELECT ?pm (AVG(?score) AS ?avgscore) WHERE
    {
    GRAPH graph:queries
    {
      ?pm a queries:PatternMapping;
          queries:mappingHasQuery <[queryUri]>;
          queries:mappingContainsMapping+ ?em.
      ?em queries:emHasScore ?score.
    }
  } GROUP BY ?pm
};
# step 2: disavantage query mappings involving several times a same keyword
INSERT
{
  GRAPH graph:queries
  {
    ?pm queries:hasEmFactor ?factor;
  }
}
WHERE
{
  SELECT ?pm (COUNT(distinct ?kw) / COUNT(distinct ?em) AS ?factor ) WHERE
  {
    GRAPH graph:queries
    {
      ?pm a queries:PatternMapping;
          queries:mappingHasQuery <[queryUri]>;
          queries:mappingContainsMapping+ ?em.
      FILTER NOT EXISTS {?em a queries:EmptyElementMapping.}
      ?em (queries:emHasMatching / queries:matchingHasKeyword) ?kw.
    }
  } GROUP BY ?pm
};
INSERT
{
  GRAPH graph:queries
  {
    ?pm queries:hasEmrMark ?emrmark;
  }
}
WHERE
{
  GRAPH graph:queries
  {
    ?pm queries:hasEmAvg ?avgscore.
    ?pm queries:hasEmFactor ?factor.
    BIND (?factor * ?avgscore AS ?emrmark)
  }
}
```

A.16 Query coverage relevance mark

```
# process the query coverage relevance mark
PREFIX patterns:  <http://swip.univ-tlse2.fr/ontologies/Patterns#>
PREFIX queries:  <http://swip.univ-tlse2.fr/ontologies/Queries#>
PREFIX graph:  <http://swip.univ-tlse2.fr:8080/musicbrainz/>
INSERT
{
  GRAPH graph:queries
  {
    ?pm queries:hasQcrMark ?qcrmark;
  }
}
WHERE
{
  {
    SELECT (COUNT(?qe) AS ?nbqe) WHERE
    {
      GRAPH graph:queries
      {
        <[queryUri]> queries:queryHasQueryElement ?qe.
      }
    }
  }
  {
    SELECT ?pm (COUNT(DISTINCT ?qe) AS ?nbmappedqe) WHERE
    {
      GRAPH graph:queries
      {
        ?pm a queries:PatternMapping;
            queries:mappingHasQuery <[queryUri]>;
            queries:mappingContainsMapping+ ?em.
        ?em (queries:emHasMatching / queries:matchingHasKeyword) ?qe.
      }
    } GROUP BY ?pm
  }
  BIND ( ?nbmappedqe / ?nbqe AS ?qcrmark )
}
```

A.17 Pattern coverage relevance mark

```
# process the pattern coverage relevance mark
# in two step probably because of a bug in ARQ
# step 1: find out for each query mapping the number of element mapping with distinct pattern eleme
PREFIX patterns:  <http://swip.univ-tlse2.fr/ontologies/Patterns#>
PREFIX queries:  <http://swip.univ-tlse2.fr/ontologies/Queries#>
PREFIX graph:  <http://swip.univ-tlse2.fr:8080/musicbrainz/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
INSERT
{
  GRAPH graph:queries
  {
    ?pm queries:numberOfSignificantElementMappings ?nbmappedpe;
  }
}
WHERE
{
  select ?pm (count(distinct ?pe) as ?nbmappedpe) where
  {
      GRAPH graph:queries
      {
        ?pm a queries:PatternMapping;
             queries:mappingHasQuery <[queryUri]>;
             queries:mappingHasPatternConstituent ?p;
             queries:mappingContainsMapping+ ?em.
        ?em queries:emHasPatternElement ?pe.
        FILTER NOT EXISTS { ?em a queries:EmptyElementMapping. }
      }
  } group by ?pm
};
# step 2: find out for each query mapping the total number of element mappings
PREFIX patterns:  <http://swip.univ-tlse2.fr/ontologies/Patterns#>
PREFIX queries:  <http://swip.univ-tlse2.fr/ontologies/Queries#>
PREFIX graph:  <http://swip.univ-tlse2.fr:8080/musicbrainz/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
INSERT
{
  GRAPH graph:queries
  {
    ?pm queries:numberOfElementMappings ?nbmappedpe;
  }
}
WHERE
{
  select ?pm (count(distinct ?pe) as ?nbmappedpe) where
  {
      GRAPH graph:queries
      {
        ?pm a queries:PatternMapping;
             queries:mappingHasQuery <[queryUri]>;
             queries:mappingHasPatternConstituent ?p;
             queries:mappingContainsMapping+ ?em.
        ?em queries:emHasPatternElement ?pe.
      }
  } group by ?pm
};
# step 3:
INSERT
{
  GRAPH graph:queries
  {
    ?pm queries:hasPcrMark ?pcrmark;
  }
}
WHERE
```

```
{
  GRAPH graph:queries
  {
    ?pm queries:mappingHasQuery <[queryUri]>;
        queries:numberOfElementMappings ?nbem;
        queries:numberOfSignificantElementMappings ?nbsem.
  }
  BIND (?nbsem / ?nbem AS ?pcrmark)
}
```

A.18 Final relevance mark

```
# process final relevance mark
PREFIX patterns:  <http://swip.univ-tlse2.fr/ontologies/Patterns#>
PREFIX queries:  <http://swip.univ-tlse2.fr/ontologies/Queries#>
PREFIX graph:  <http://swip.univ-tlse2.fr:8080/musicbrainz/>
INSERT
{
  GRAPH graph:queries
  {
    ?pm queries:hasRelevanceMark ?rmark;
  }
}
WHERE
{
  GRAPH graph:queries
  {
    ?pm a queries:PatternMapping;
        queries:mappingHasQuery <[queryUri]>;
        queries:hasEmrMark ?emrmark;
        queries:hasQcrMark ?qcrmark;
        queries:hasPcrMark ?pcrmark.
  }
  BIND ( ?emrmark * ?qcrmark * ?pcrmark AS ?rmark )
}
```

A.19 Clear KB

```
# clear all mappings whose rank is over a given threshold
```