SPARQL @ TPAC

```
"2023-09-12"^^xsd:date
```

- Incoming
  - Integrate quoted triples
  - Directional language
  - Lang tag formatting
- EXISTS
- Other
  - Implicit timezone
- AOB

EXIST Issues

https://github.com/w3c/sparql-query/issues?q=is%3Aissue+is%3Aopen+exists

1. Some uses of EXISTS are not defined during evaluation
2. Substitution happens where definitions are only for variables
3. Blank nodes substituted into BGPs act as variables
4. Substitution can flip MINUS to its disjoint-domain case
5. Substitution affects disconnected variables

Some Examples

Location must be a variable:

```
EXISTS { BIND ( :e AS ?var ) ?s ?p ?z } , EXISTS { VALUES ?var (1 2 3) . . . }
EXISTS { SELECT ?var { . . . }
EXISTS { . . . BOUND(?var) . . . }
```

Variable scope

```
{ ?s :p ?var } FILTER EXISTS { SELECT ?x { ?s :p ?var } }
```

Blank Nodes moved across BGPs

```
current row ?var = _:b123 FILTER EXISTS { ?var :p ?o } }
```

Related SPARQL Issues

Parameterized Queries

https://github.com/w3c/sparql-dev/issues/89#issuecomment-547352909

https://github.com/w3c/sparql-dev/issues/57

LATERAL join : https://github.com/w3c/sparql-dev/blob/main/SEP/SEP-0006/sep-0006.md

Possible Solutions

1. SEP-0007 : Improved substitution
2. sparql-exists#proposal-a
3. Pattern restrictions (not explored)
4. Other?

Proposal A

Summary
sparql-exists#proposal-a

This proposal for EXISTS emphasizes simplicity and the SPARQL notions of variable scoping and bottom-up evaluation of sub-queries over maximum compatibility with the current SPARQL definition for EXISTS. Its basic idea is to inject values for the variables in-scope just outside a FILTER expression at the beginning of the pattern argument to EXISTS almost as if a VALUES construct was injected there.

. . .

Change the definition of the exists function to:
Let μ be the current solution mapping for a filter, t a token, and P a graph pattern: The value exists(Initial(t),P) given D(G) is true iff eval(D(G),P') is a non-empty multiset of solution bindings, where P' is P with Initial(t) replaced by {μ}.

SEP-0007 : Improved substitution

Intuition - SQL Correlated Subquery.

The current row provides values for to use in the sub-query.

- Don’t substitute. Instead, inject a variable binding into the BGP at the point of use
- Rename variables in hidden scopes.
- Disallow uses of variables that can not be replaced by a value e.g. BIND

https://github.com/w3c/sparql-dev/blob/main/SEP/SEP-0007/sep-0007.md

```
PREFIX : <http://example/>
SELECT * {
    BIND( 53 AS ?z )
    EXISTS { :s :q ?o FILTER (?o < ?z ) }
```

Proposal-A: ?z not in scope at the filter

```
Join ( { ?z = 53 } ,
      { :s :q ?o FILTER (?o < ?z ) }
    )
```

Correlated evaluation: ?z is in scope at the filter

```
   ?z = 53 :s :q ?o FILTER (?o < ?z )
```

```
PREFIX : <http://example/>
SELECT * {
    BIND( 53 AS ?z )
    EXISTS {
        { :s :q ?o FILTER ( ?z < ?o ) }
        UNION
        { :s :r ?z }
    }
}
```

Proposal-A: ?z not in scope at the filter

```
  EXISTS { Join( { ?z = 53 },
                 Union( :s :q ?o FILTER ( ?z > 26 ) ,
                        :s :r ?z
                      )
         }
```

Correlated evaluation: ?z is in scope at the filter

```
EXISTS { UNION(
           { ?z = 53 :s :q ?o FILTER ( ?z > 26 ) },
           { ?z = 53 :s :r ?z }
         ) }
```

Correlated Subquery Approach

How best to express injecting values:

1. Rewrite as BIND/VALUES before BGP etc
   - Logically, as a Join of a refined VALUES clause but only with places variables can be set (BGPs , . . .)
   - ```
     ?z = 53 :s :q ?o FILTER ( ?o < ?z ) ⇒
     Filter ( ?o < ?z ,
             Join(VALUES ?z { 53 }, BGP(:s :q ?o)
            )
     ```

2. New algebra operator “Take value from current row”

3. ??

There are only a few places where variables are associated with values.

BGPs, Paths , Assignment (BIND, VALUES, SELECT, GROUP BY) , GRAPH
     ```