Scalar Subqueries and List Subqueries

CIP 2017-03-29

Tobias Lindaaker
tobias@neo4j.com

Filter the contents of a list

```
...
WITH … AS my_list
RETURN [
    UNWIND my_list AS item
    WHERE item.size < 15
    RETURN item
] AS small_items
```

Collect Lists of matching subpatterns

```
MATCH (me:Person {name: $my_name})
RETURN me.user_id, [
    MATCH (me)-[:FRIEND]-(friend)
    RETURN friend{.name, .user_id}
] AS friends, [
    MATCH (me)-[:ENEMY]-(enemy)
    RETURN enemy{.name, .user_id}
] AS enemies
```

Sort a list

```
...
WITH … AS my_list
WITH [
    UNWIND my_list AS item
    RETURN item
    ORDER BY item.price ASCENDING
] AS my_sorted_list
...
```

Aggregate a list

```
...
WITH … AS my_numbers
RETURN SCALAR {
    UNWIND my_numbers AS number
    RETURN sum(number)
}
```

we might want to come up with shorthand syntax for this simple case

Unpacking a single element from a list

```
...
WITH … AS my_singleton_list
RETURN SCALAR {
    UNWIND my_singleton_list
} AS my_value
```

Unpacking single expected match

```
...
WITH … AS my_list
RETURN SCALAR {
    UNWIND my_list AS item
    WHERE item.id = $item_id
} AS my_single_value
```

the lack of **`RETURN`** *signals the single-or-null semantics*

Scalar subqueries

```
MATCH (dtor:Director)
WHERE dtor.age < SCALAR {
    MATCH (dtor)-[:DIRECTED]->(m:Movie),
          (a:Actor)-[:ACTED_IN]->(m)
    RETURN min(a.age)
}
RETURN dtor.name AS young_director
```

Grammar and Semantics

List Subqueries

- Contained in **`[ … ]`**
- Starts with **`MATCH`**, **`UNWIND`**, or **`CALL`**
- Returns *a single expression*

In both cases, initial **`MATCH`** can be omitted if only matching a single connected pattern

Scalar Subqueries

- Contained in **`SCALAR{ … }`** *or other suggestions welcome!*
- Starts with **`MATCH`**, **`UNWIND`**, or **`CALL`**
- Returns *a single aggregation*, or is of special form: **`SCALAR{ UNWIND`** *`<var>`* **`}`**

Omitting the initial MATCH

```
MATCH (me:Person {name: $my_name})
RETURN me{.name, .user_id, friends: [
     (me)-[:FRIEND]-(friend)
     RETURN friend{.name, .user_id}
], enemies: [
     (me)-[:ENEMY]-(enemy)
     RETURN enemy{.name, .user_id}
]}
```

Using a *Projection Map* in a List Subquery

```
MATCH (dtor:Director)
RETURN dtor.name, [
     (dtor)-[:DIRECTED]->(movie:Movie)
     RETURN {movie, actors:[
          (a:Actor)-[:ACTED_IN]->(m)
          RETURN a
     ]}
] AS movies
```