SQL/Graph Query Procedures: **an early draft**

oCIG 4 Presentation, 2017-08-17

Peter Furniss, Neo4j

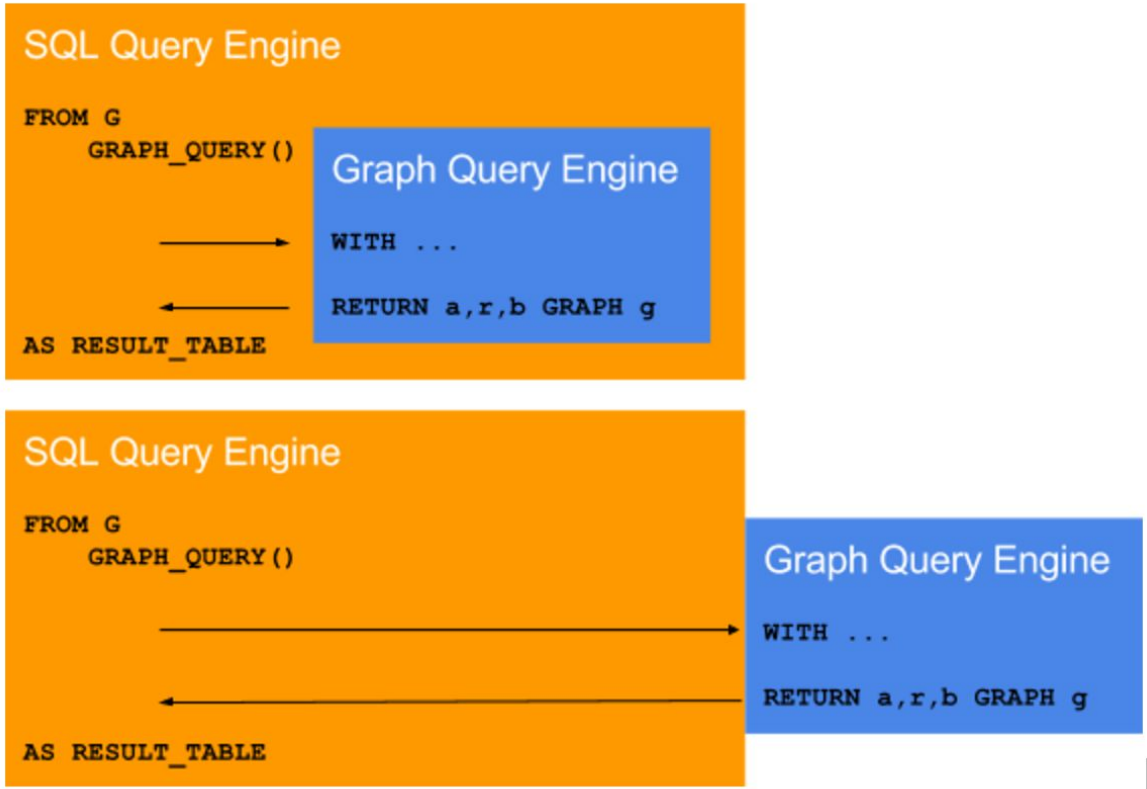
Motivation

Don’t distort SQL with graph querying capabilities

Instead, follow precedents set by SQL/XML and SQL/JSON:

- Extend SQL with a new GRAPH_REFERENCE data type (which is opaque)
- Extend SQL with the ability to invoke procedures to undertake graph query processing

Single and dual engine architecture for the implementation of a graph query engine



Graph query procedures: overview

1) GRAPH_QUERY procedure:

    Returns one or more graphs as GRAPH_REFERENCES

    Returns one or more table columns

2) GRAPH_TABLE procedure:

    Returns one or more table columns only

Graph query procedures: inputs

Mandatory inputs:

- Graph inputs: one or multiple GRAPH_REFERENCES
- Query specification: a string which is a query statement for the target graph query language

Optional inputs:

- Driver table: A table expression, the types of whose columns are capable of being processed by the graph query engine/language
- Query parameters: A list of named scalar values

Graph query procedures: results

GRAPH_QUERY and GRAPH_TABLE return:

> Result table: a named table, containing the data returned by the procedure after the graph query engine has processed the graph query specified in the “Query specification” parameter

Example: Creating and returning a graph using the GRAPH_QUERY procedure

Step 1: Defining the input graph

Assume the following table, SALES_GRAPHS:

NAME [VARCHAR (256)]	YEAR_QUARTER [CHAR (6)]	SALES_GRAPH [GRAPH_REFERENCE]
SalesDetail	<NULL>	[graph_reference_0]

Step 1: Defining the input graph

Query snippet:

```
WITH QUARTERLY_SALES_DETAILS_GRAPH
  AS SELECT RESULT_TABLE_FROM_CYPHER_QUERY_1.*

FROM SALES_GRAPHS G
    GRAPH_QUERY
       (GRAPH_INPUT =>
           (SELECT SALES_GRAPH
              FROM G
             WHERE G.NAME = 'SalesDetail'
               AND G.YEAR_QUARTER IS NULL),
```

Step 2: Defining the graph query

Query snippet:

```
QUERY_SPECIFICATION =>
    OPAQUE endCypherQuery_1

with graph SalesDetail

match (p:Product)-[r:IN]->(o:Order)<-[:HAS]-(s:Store)-[:IN]->(reg:Region)
where datein(o.date, $YEAR_QUARTER_PARAMETER)

return new graph QuarterlySalesDetail

        endCypherQuery_1
```

Step 3: Passing parameters

We assume the existence of a parameter @0 which has been set to ‘2017Q3’

Query snippet:

```
PASSING @0 AS YEAR_QUARTER_PARAMETER
```

Step 4: Returning the new graph from GRAPH_QUERY

Query snippet:

```
COLUMNS
    (QuarterlySalesDetail GRAPH_REFERENCE AS SALES_DETAIL)
...
AS RESULT_TABLE_FROM_CYPHER_QUERY_1
```

Step 5: Post-processing tasks

Query snippet:

```
INSERT INTO SALES_GRAPHS
    (NAME,
     YEAR_QUARTER,
     SALES_GRAPH)
VALUES
    ('SalesDetail',
     @0,
     (SELECT DISTINCT SALES_DETAIL
           FROM QUARTERLY_SALES_DETAILS_GRAPH))
```

After execution, the SALES_GRAPHS table contains:

NAME [VARCHAR (256)]	YEAR_QUARTER [CHAR (6)]	SALES_GRAPH [GRAPH_REFERENCE]
SalesDetail	<NULL>	[graph_reference_0]
SalesDetail	2017Q3	[graph_reference_1]

Putting it all together: the full query

```
WITH QUARTERLY_SALES_DETAILS_GRAPH
  AS SELECT RESULT_TABLE_FROM_CYPHER_QUERY_1.*

FROM SALES_GRAPHS G
    GRAPH_QUERY
       (INPUT_GRAPH =>
           (SELECT SALES_GRAPH
              FROM G
             WHERE G.NAME = 'SalesDetail'
               AND G.YEAR_QUARTER IS NULL), -- the OLTP
"perpetual" graph

        QUERY_SPECIFICATION =>
            OPAQUE endCypherQuery_1 //delimit graph query

// Cypher query 1: extract a quarterly snapshot of sales

with graph SalesDetail

match
(p:Product)-[r:IN]->(o:Order)<-[:HAShas]-(s:Store)-[:IN]->(r
eg:Region)
where datein(o.date, $YEAR_QUARTER_PARAMETER)

return new graph QuarterlySalesDetail

        endCypherQuery_1

        <query continues in right box>
```

```
PASSING @0 AS YEAR_QUARTER_PARAMETER

        COLUMNS
            (QuarterlySalesDetail GRAPH_REFERENCE
AS SALES_DETAIL)
    )

  AS RESULT_TABLE_FROM_CYPHER_QUERY_1

INSERT INTO SALES_GRAPHS -- store graph reference
for quarterly SALES_DETAIL graph
   (NAME,
    YEAR_QUARTER,
    SALES_GRAPH)
VALUES
   ('SalesDetail',
    @0,
   (SELECT DISTINCT SALES_DETAIL
              FROM QUARTERLY_SALES_DETAILS_GRAPH))
```