Considerations for Set Algebra in OpenCypher

This is a proposal written by Thomas Frisendal for extending openCypher with explicit and extensive support for set algebra on the graph level.

WHY DID SET ALGEBRA NOT BECOME MORE POPULAR?

People of my age were taught set algebra at high-school (in my case in the late seventies). Today it is elementary school stuff. And it is indeed a useful tool with applications in many real-life situations.

However, set algebra never made it big time within it tools, applications and databases.

If fact, most of what has been exposed is the SQL set-style operators (UNION, INTERSECTION and DIFFERENCE), which (I believe) made it into the first SQL standard in 1986.

The SQL operators were meant to implement the relational algebra as proposed by Dr. Ted Codd. Unfortunately Dr. Codd based some of his ideas on an ”extended set theory”, which was an idea formulated and described in a 1986 paper:

“Description of a Set-Theoretic Data Structure” by D. L. Childs (https://www.computer.org/csdl/proceedings/afips/1968/5072/00/50720557.pdf).

But Childs’ extensions were not ideally suited, which is explained in quite some detail in this book:

The Algebra of Data - A Foundation for the Data Economy by Professor Gary Sherman & Robin Bloor, Ph.D. © 2015 by The Bloor Group Press, https://www.amazon.com/Algebra-Data-Foundation-Economy/dp/0978979168

The authors argue that mainstream Zermelo-Fraenkel set theory (Cantor), would have been a better starting point. One key issue is that sets should be able to be sets of sets.

WHO ARE ACTUALY USING SETS?

Set operations did not really break out of the SQL zone. I have been writing tons of SQL and in my experience, UNION is used sometimes, whereas INTERSECT and DIFFERENCE are not used very much (and mostly in metadata operations). UNION is most often used as ”the poor DBA’s ETL-tool to consolidate data having different shapes from different systems into one structure, often as part of a view (and often spraying NULL’s all over the place).

For many years I was hoping to see some set algebra in end-user oriented applications. The only product that I know of, and I have been on the lookout for almost 40 years, is a nice product called Set Analyzer, which came into existence in the 80es (?), and was bought by Business

Objects a few years later. The use case was customer segmentation: http://www.crmodyssey.com/Documentation/Documentation_PDF/Business_Objects_Set_Analyser.pdf. The functionality is still alive and kicking now that it belongs to SAP: https://help.sap.com/doc/businessobject_product_guides_boexir31sp3_en_xi31_sp3_sa_user_en_pdf/XI3.1.3/en-US/xi31_sp3_sa_user_en.pdf.

The tool even uses Venn diagrams as part of the UI controls!

Today the product is called ”SAP Business Objects Set Analysis”.

In essence the product maintains lists of customers, who are interesting from a marketing perspective. To me it is interesting that these lists are built, combined, and maintained using the set algebra paradigm. But it is still a tabular solution.

THE BUSINESS CASES

The most obvious use case was/is customer segmentation (customer behavior analysis) and product campaign planning in marketing applications.

Today there are also strong user stories in the contexts of investigative work flows based on sets of suspects in graph based analysis of crime, intelligence, fraud, churn, recommendations and other behavior / networking analytical areas.

I really encourage you to read the BO fact sheet (http://www.crmodyssey.com/Documentation/Documentation_PDF/Business_Objects_Set_Analyser.pdf). What it describes, is essentailly the same as what Business Objects bought around 1985(?). SAP bought BO in 2007.
The basic user story is:

- I (as a business user) can define "sets" as search results

- I can save sets with a name and recall them

- I can use set algebra (just like I learned at school), and also use conditions on sets in order to create new sets

- I use this to plan my campaigns across my different customer segments.

This enables sophisticated, iterative analytics on combinations of sets. Useful in marketing and CRM, which is what the product was designed for. Today social network analysis, fraud, law enforcement and much more promise even better payback than good CRM.

The current SAP version has the notion of "temporary" which means "as of now” (i.e. do the search again), or "static” meaning as of the last result of the search (which can be done again).

NB: As you may know from projects like the Panama Papers etc. graphs offer a very rich context. It could well be that the SQL set operations never really gained traction, because most of the use cases require a rich context.

In my opinion sets of graphs is a once-in-a-lifetime opportunity that is much more than a revival: Doing set algebra on sets of graphs is potentially several orders of magnitude more powerful than SQL-based set algebra (or the simple lists etc. offered by early set analysis tools such as the BO Set Analyzer).

GRAPHS ARE SETS

All sets are not graphs. But graphs are sets. In the Data Algebra book by Sherman and Bloor (cf. reference above) there is a good argumentation for this. In fact there is a whole chapter (7) called Data Algebra and Graphs. I am not going to repeat that chain of arguments, but basically graphs can be understood as a relationship (which is not Dr. Codd's kind, and which in the data algebra parlance is called a "clan"), which in turn consists of "couplets" (e.g. representing a start node, the relationship and an end node). I refer you to the book for the mathematics about building the hierarchy going from data to couplets over clans to "hordes". If the math holds, and I think it does (Algebraix Data has patented it?), graphs (being "clans" of couplets) can be subjected to:

- Union and cross-unions as well as
- Intersection and cross-intersections.

And sets of graphs have a lot more information than plain sets of the tabular kind. That is the real opportunity here.

In openCypher care should be taken not to repeat too many mistakes of the SQL set behaviors.

In consequence, the set algebra should happen by way of simple, intuitive declarations in openCypher. (Tool vendors can add visuals).

OPENCYPHER 10 AND MULTIPLE GRAPHS

In the openCypher project, openCypher 10 is expected to contain much, sophisticated functionality related to what is necessary for doing "multiple graphs":

- CIP2017-06-18 Querying and constructing multiple graphs
- CIP2016-06-22 Nested, updating, and chained subqueries
- CIP2017-04-20: Query combinators for set operations

- CIP2018-05-03 Creating and managing graphs and views

Some of the issues discussed are: Named query, views, named graphs, working graph, multiple graphs, catalog of graphs etc.

As far as I can see the CIR's are good and will most likely lead to the overall goals of graph closure in openCypher.

I am not arguing against any of the above. I see my proposal as being less "graph-technical" of nature and instead it is focused on ease-of-use at the business level by ordinary business folks, analysts and investigators. At this moment (openCypher 10) a unique opportunity exists to get a rock-solid, intuitive and mathematically closed language serving both aspects of the data structures, which people are interested in:

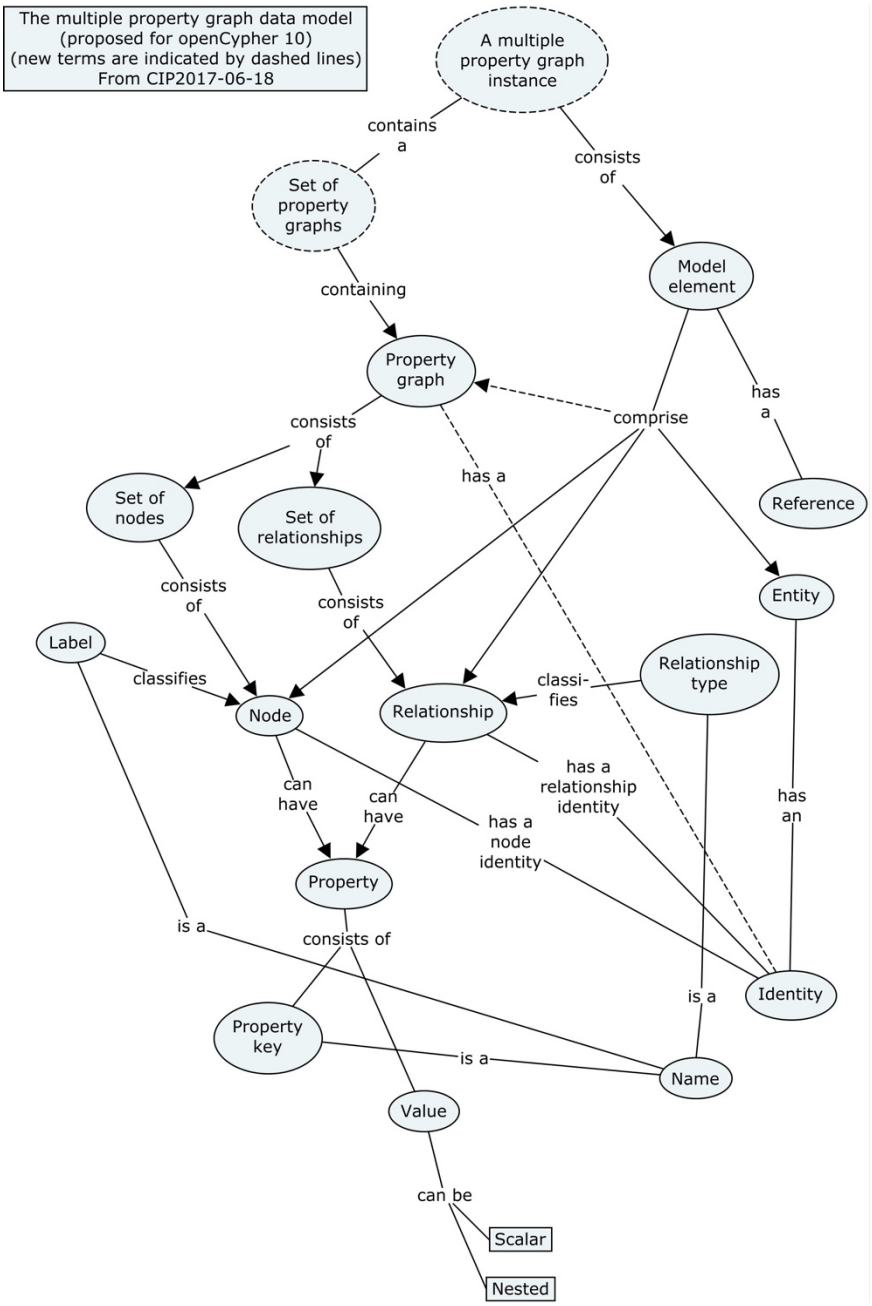
- Data presented as graphs
- Data presented as sets (of graphs)

Set algebra is simply one of the things you can do with graphs. The CIP's above point out many other things, which you can also do with them (graphs).

There are some terminology issues.

DATA MODEL CONSIDERATIONS (CIP2017-06-18)

The concept map below attempts to capture the terminology of CIP2017-06-18 in order to see whether we can find a place for "sets":



Unfortunately, there are already other types of (explicit from a conceptual point of view) “sets” (of nodes and of relationships). Introducing “set” as a specific term for a set of (property) graphs will not work, since “set” is already used in a more general sense.

We will have to stick to “sets of graphs”, which is a concept already in the proposed terminology for openCypher 10 (cf. definition 15 in the CIP and see also the concept map above).

Since any graph can be considered a set in its own right, sets can be members of other sets.

The basic requirement is this:

Requirement 1: There should be support for "sets of graphs", which may include sets of graphs as members.

Another way of looking at is: A name (and identity) on the graph level can designate a set with only one graph as a member, but it can also designate a set of members, some of which are potentially also (multiple member) sets.

WORKING WITH NAMED GRAPHS AND NAMED SETS

"Property graph" is (in OC 10) a model element in its own right and as such it has an identity. Like other model elements, it may have a (user-defined) name. The analytical use cases would also benefit from having a user defined description available:

Requirement 2: Property graphs can have not only a name, defined by the user (supported by CIP2018-05-03), but also a description.

Since many sets of graphs will contain more than one graph, also sets of graphs will have to be able to have a name:

Requirement 3: Sets of graphs can have a name, defined by the user, who may also maintain a description of the set. This even applies to empty (at the moment) sets of property graphs.

The assignment of names and descriptions of graphs and sets of graphs will, as proposed, happen:

- within the "construction clauses", however:
    - is this what CIP2017-06-18, section 6.2.2, suggests?
    - none of the examples of construct-clauses in chapter 7 name the graph?
    - should graphs always be created in a preparatory step? (CIP2017-06-18, section 7.3.1, CREATE GRAPH "PersonCityEvents")
- and, obviously, also in conjunction with the proposed "catalog" (CIP2018-05-03).

We definitely need the provenance tracking and construction features of CIP2017-06-18, also in the set context.

Working with sets (of graphs) means that whenever the current proposals make reference to a specific graph (using FROM, for example), a set of graphs should be possible to use at that place.

What about syntax, then? In traditional mathematics sets are described in these manners:

A is a member of the set X:

$$A \in X$$

The set X has the following members:

X = {A, B, C}

One of the hallmarks of the openCypher language is the use of "ASCII Art" (in path definitions).

It would be in the spirit of the language to also use intuitive graphics in the area of set algebra. So let me propose that openCypher recognizes the following as an inclusion of a set as a member in another set:

A (: X

So, at a point in time we have a graph (as a result) and we want to call it A. That is defined already in the CIP's.

Since A is a graph, A is also a set (with itself as member). Specifying A (: A is redundant and implied, but should be possible.

But, what if we, at the same time, know that this graph A is also to be included in two other sets, which we want to point out in a "sets of graphs name list":

A (: [X, Y]

Note: not having the little "open e" available in ASCII, I chose "(:" as an approximation. Contending suggestions are welcome.

Requirement 4: openCypher should include syntax for specifying set membership in place (where the set is named anyway). This includes membership in more than one named set.

The term "graph-name-list" (CIP2017-06-18) can be debated. It can refer to graphs and/or to sets of graphs.

An example:

CREATE GRAPH BigSpenders2018 (: [BigSpenders, VIPCardTargets]

SET OPERATIONS CONSIDERATIONS (CIP2017-04-20)

I have been boggling my mind trying to come up with some ASCII art for set operations. The point is that set operations will probably (mostly) be performed in a set context:

- Set operations are possibly not that common in describing the pattern matching and filtering that goes on in the first-time creation of a graph (that part is more or less plain "old" openCypher + subqueries + views etc.), but

- once a “library” of graphs has been built, users will want to look at combinations of graphs in sets.

That means that the syntax to be used should be compact and intuitive (w/o command like syntax). This calls for some ASCII art, but I think ASCII (7-bit) has run out of steam?

Let me describe the challenge as a user story.

As a marketing analyst I am trying to establish potential churners among the customers. I have some graphs established for various kinds of behavior:

A. No purchases last 6 months

B. VIP card owners

C. Logins last 12 months

D. High income residentials

So, I want to target some potential churners, whom I would hate to loose as customers. Thus creating a new set called “Hate to loose” comprising:

“In set A, but only members of set D, who are in set B and who are not in set C.”

In classis math notation, it is something like:

$$\text{Hate to loose} = \text{No purchases last 6 months} \cap$$

$$((\text{High income residentials} - \text{Logins last 12months}) \cup \text{VIP card owners})$$

Or something like that. How about this humble attempt at ASCII art:

```
CREATE GRAPH {Hate to loose} = {No purchases last 6 months} >+< (({High
income residentials} >-< {Logins last 12 months}) >:< {VIP card owners})

   (>+< ~ UNION, >++< ~ UNION ALL, >-< ~ DIFFERENCE, >:< ~ INTERSECT)
```

Alternative ASCII art is welcome!

Requirement 5: openCypher should include syntax for declaring set operations as part of the construction of sets of graphs. It should be done in an “ASCII art” style.

CATALOG CONSIDERATIONS (CIP2018-05-03)

Working on the set level: what I would like to see is (in loose terms) that the graph content be saved under a name, and that it can be recalled as either the graph or the paths constructing the result graph (again).

Requirement 6: There is a need for being able to create a "dynamic set of graphs" or a "dynamic graph".

In CIP2018-05-03 et al the materialized result graph is persisted, which is also fine in many use cases. However, some use cases need to be able to "re-run" the graph construction at a later point in time (based on the then current data content in the paths specified for the graph construction). *Maybe a graph "construction type" ("static" or "dynamic")?*

The CIP2018-05-03 solution is activation of views: Obviously views can be used for dynamic reconstruction of a graph, so that could be an interim solution for OC 10. However, a more symmetric solution would be preferable. With that I mean that if I (as a business analyst) work primarily with graphs and sets of them, then I do not need to know about views. All I want is to be able to "re-fresh" the content of a graph based on current production data. So, if I choose "dynamic", I get current data (running on a view without my needing to know that).

On the other hand: I might also want to make a snapshot of the graph as it is now. In that case I would copy the dynamic graph (set) to another one ("BigSpenders 2018-07-05") and mark it as static. (Which corresponds to transforming a (result set of a) view to a named graph).

We are looking at the catalog, where we probably will need to handle scope of "namespaces", which span multiple users / groups or other hierarchical organisations of named objects.

CIP2017-06-18 mentions "fully qualified graph names" (definition 44). I would like to put more emphasis on the business user orientation in the sets of graphs handling. Some business users will be using openCypher directly, whereas other business users will use an application (graph browser or graph analysis apps), which interact with the database using openCypher. In both scenarios, the business users need to have their own "folders" of named graphs and named sets of graphs, just as if they were document objects:

Requirement 7: The catalog should support end-user ownership and maintenance of "libraries" ("folders" of named graphs and named sets of graphs).

Think fraud investigators as the business users and what they would like to be able to do.

CONCLUSION

There are not many additional things to do in order to provide a good platform for building solutions, which employ set algebra style applications on top of sets of graphs.

The business benefits of being able to do so are substantial, so I hope that openCypher 10 will be able to accommodate this functionality, too.

Here is the list of the 7 identified requirements:

Requirement 1: There should be support for "sets of graphs", which may include sets of graphs as members.

Requirement 2: Property graphs can have not only a name, defined by the user (supported by CIP2018-05-03), but also a description.

Requirement 3: Sets of graphs can have a name, defined by the user, who may also maintain a description of the set. This even applies to empty (at the moment) sets of property graphs.

Requirement 4: openCypher should include syntax for specifying set membership in place (where the set is named anyway). This includes membership in more than one named set.

Requirement 5: openCypher should include syntax for declaring set operations as part of the construction of sets of graphs. It should be done in an "ASCII art" style.

Requirement 6: There is a need for being able to create a "dynamic set of graphs" or a "dynamic graph".

Requirement 7: The catalog should support end-user ownership and maintenance of "libraries" ("folders" of named graphs and named sets of graphs).

#7 could be implementation specific.

#1, named sets of graphs will have to plug in in many places of the language, but from what I have seen, I don't believe there are any conflicts. If users define meaningless sets of graphs, the (lack of) matching results will quickly show them that matching apples and pears generally is only a good idea in a shared context.

The latest years have shown that business oriented analysts, data scientists and investigators can use openCypher. Not least because of the design of the language to be intuitive by way of "ASCII art". By doing that again, this time on set algebra, these communities will be able to work even more productively. Workflows will focus on finding, cataloguing and reusing named sets of graphs to produce rock-solid and documentable contexts of patterns of behavior.