Traceability diagram FAQ

How does DGC select the color of a node?

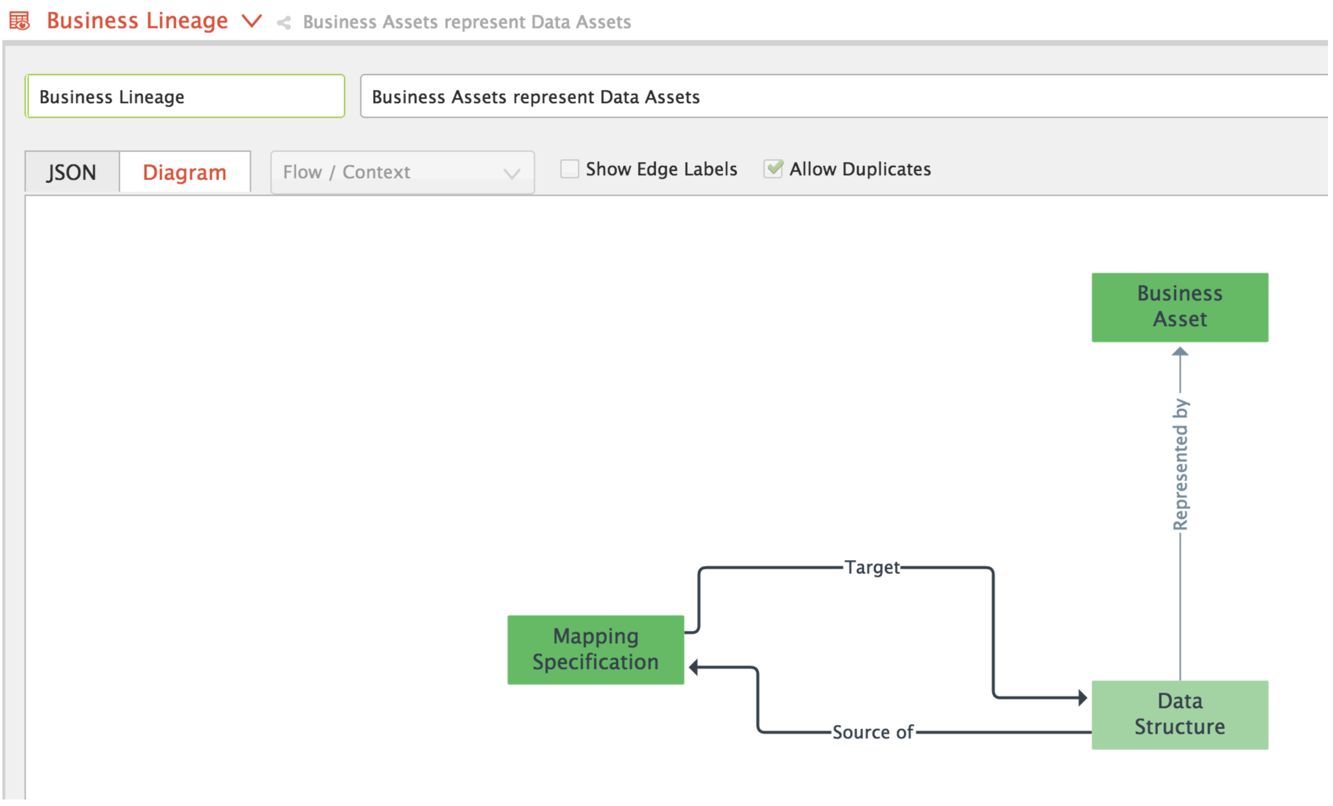
A: The color of each node is automatically selected from a palette, based on the asset type or complex relation type. More specifically, they are based on the name and resource ID of the type.
All assets of the same type have the same color across all diagrams.
Since there are more asset types than colors in the palette, it may happen that, in the same diagram, nodes with two different types nevertheless have the same color.

Why do I see so many diagram views in the dropdown list. Are they all applicable to my asset?

A: You see a diagram view in the dropdown list if it is applicable to the asset and if it has been shared with you.
A diagram view is applicable to:

- Each asset of any of the asset types that occur in the diagram view
- Each asset of any child asset type of any of the types that occur in the diagram view

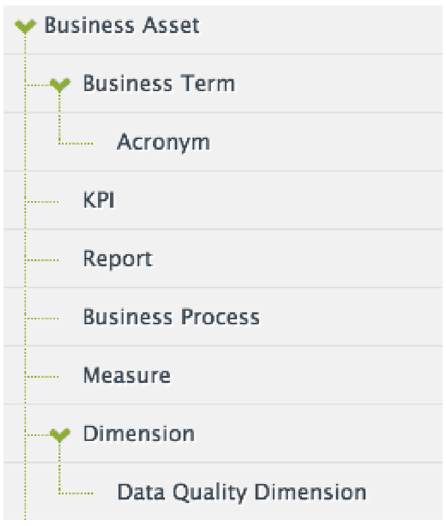
For example, consider this diagram view:



The following asset types occur in the diagram view:

- Business Asset
- Data Structure
- Mapping Specification

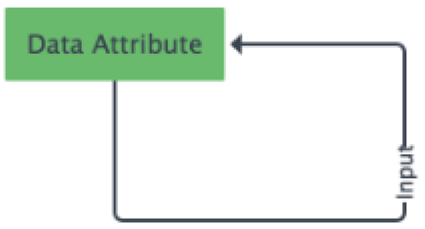
The diagram view applies to any asset of these types. A user can 'start anywhere', either from the asset page of a Business Asset, Data Structure or Mapping Specification.
Not only that, but inheritance of asset types plays a role too. This view applies to any asset of a child asset type of any of the three asset types mentioned above.
In **Settings > Types > Asset types**, you can see the tree of asset types.
For example, Business Asset has the following child asset types.



The view is applicable to all of these, and to the child asset types of Data Structure and Mapping Specification as well.

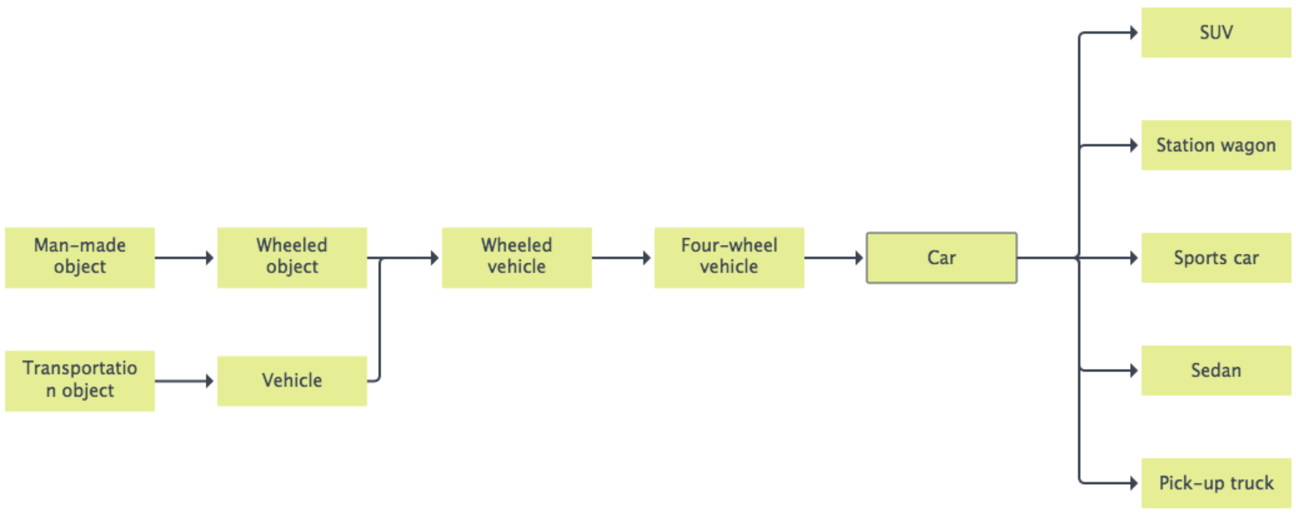
How does DGC decide which edges and nodes to include in the diagram?

A: DGC does not simply add all relations to the diagram, it has logic to decide which relations are relevant. This logic is called the *traversal strategy*.
To understand how this works, we need to differentiate between incoming edges and outgoing edges for a node, and between flow and context nodes.
First look at flow only.
Consider the following simple diagram view:

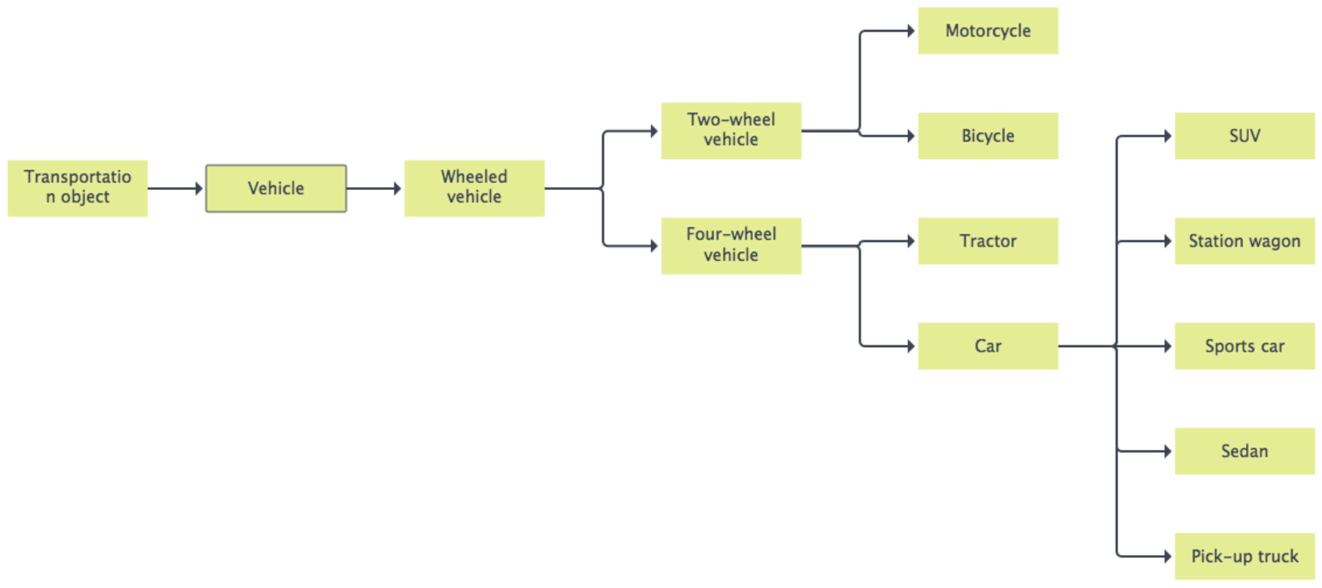


From the start node, DGC adds all directed relations, and their nodes, whose relation type occurs in the view, both in upstream (incoming) and downstream (outgoing) direction.
For each node that was added while traversing downstream, DGC adds all the downstream directed relations and nodes. After all, the upstream relations of a downstream node are irrelevant for the diagram of the start object.
Also for each upstream node that was added, DGC adds all the upstream directed relations and nodes.

In the diagram below, the above view has been applied to **Four-wheel vehicle**, to reveal the 'Groups' hierarchy.
Any downstream node from an upstream node (i.e. **Vehicle**, or **Wheeled vehicle**) is irrelevant for this diagram.



But if we jump to **Vehicle,** we get a different diagram:

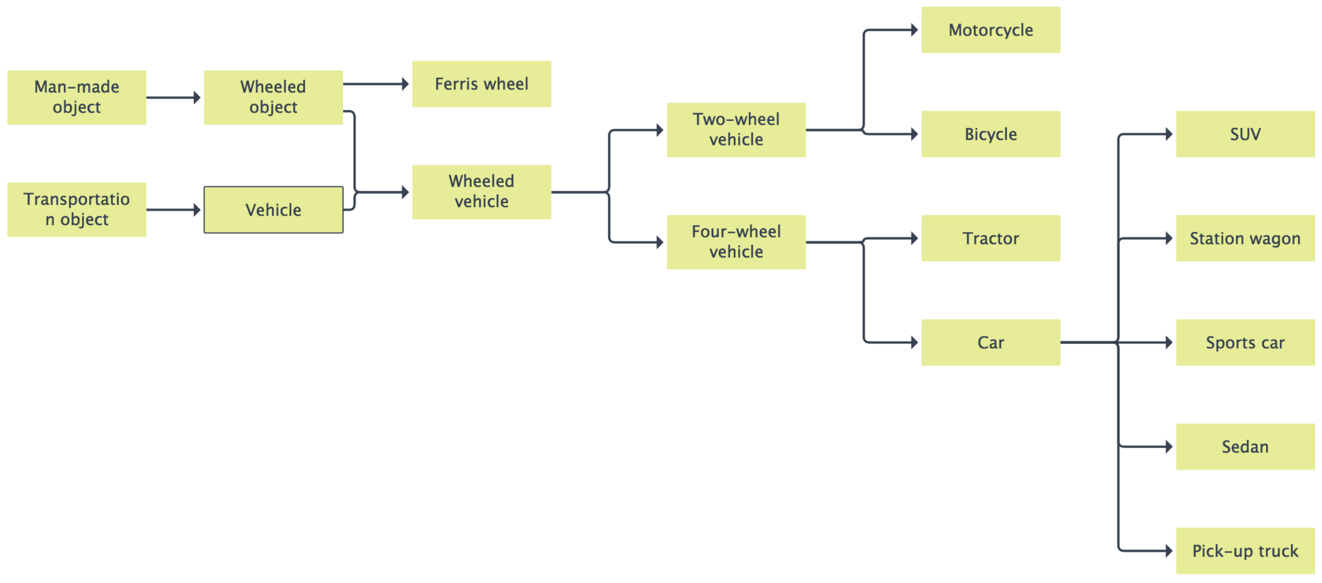


Compared to the first diagram, the nodes that were upstream of **Wheeled vehicle** are removed, and the downstream subtree for **Two-wheel vehicle** was added.

Can I set a different traversal strategy?

All examples in this chapter, unless mentioned otherwise, use the default traversal strategy. We call this the *directed traversal strategy*, and could be paraphrased as 'keep on walking and don't look back'.
You can also switch the traversal strategy to *complete graph*, by always looking in all directions. Depending on your data, this could obviously lead to a much bigger diagram, with more nodes and edges.

Going back to the above example and applying that strategy results in the following diagram:



Note the new upstream branch of **Wheeled vehicle**, which is downstream from the start node. An upstream node in that branch, **Wheeled object**, has a downstream branch: **Ferris wheel**.

You can switch the traversal strategy in the JSON form of the diagram view by adding the following name/value pair to the top-level diagram section

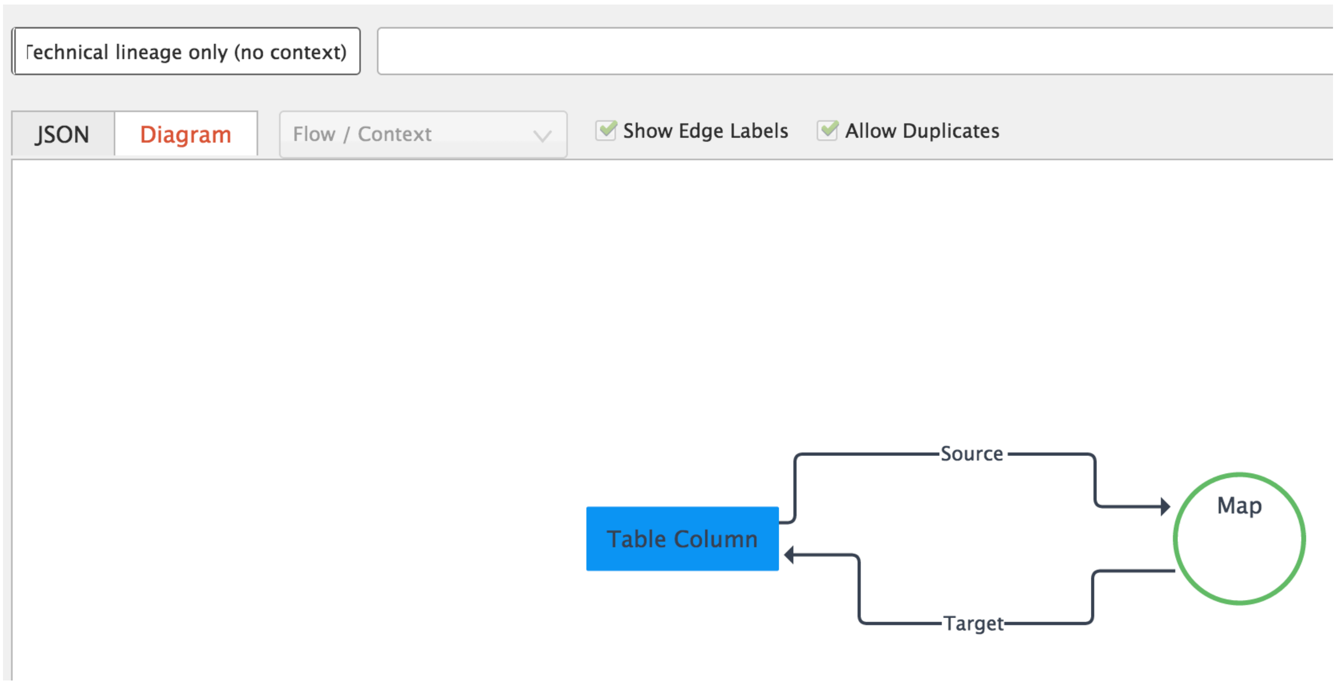
```
"visitStrategy": "completeGraph"
```

In a diagram view, I can assign each node to either the 'Flow' or 'Context' layout region. What does that do?

A: The default traversal strategy ('keep on walking and don't look back') gives the required results when you depict a set of assets that are related through some kind of (data) flow or dependency. But when the relation represents a link to *context* rather than *flow*, the edge between two nodes should always be traversed, even if the related node is upstream of a downstream node, or vice versa. You can also regard an edge between a flow node and a context node as *bidirectional:* it is always traversed.

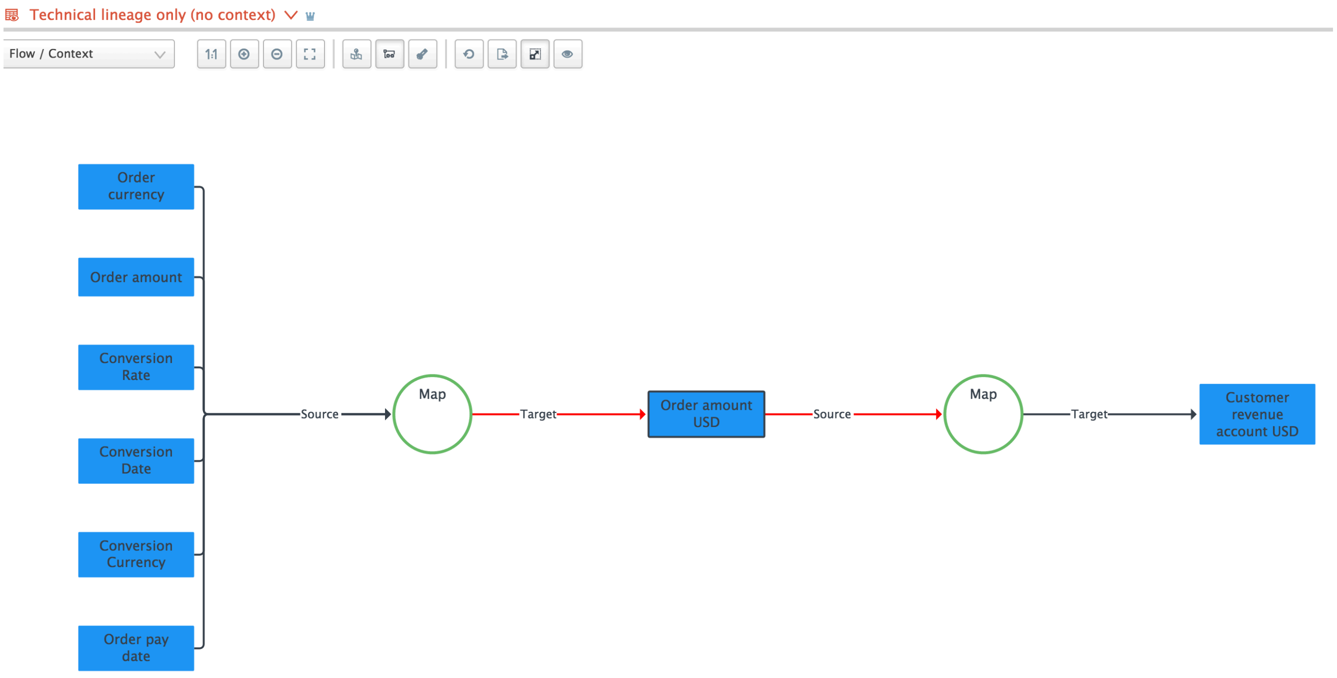
This notion will become clearer when you look at the example below.
When the flow depicts transformations of table columns, the tables that contain the columns, and the business terms that are related to them, provide context to the columns. The context nodes are always relevant, regardless of whether the column was encountered while going upstream or downstream. The context nodes are always added to the diagram.
For this traversal strategy, you can assign a node in the diagram view to be a 'context' node.

Consider the following example diagram view:



This view contains flow only.

An example diagram for this view is below:

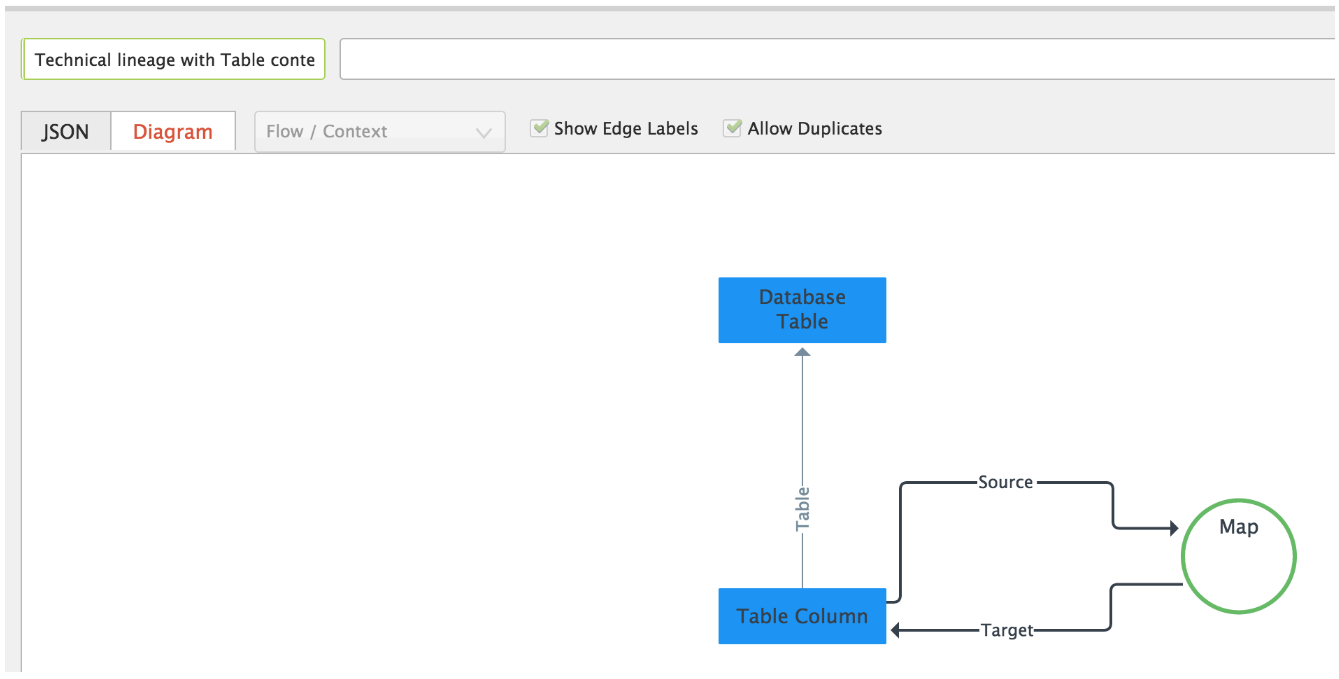


We started from the asset in the middle (**Order amount USD**).

Now assume that you also want to see to which tables these columns belong.
You could add a relation from column to table to the view (in other words, a downstream edge), but this relation would only be considered on the downstream side, and if we add the column/table relation in the reverse direction, it would only be considered on the upstream side. To

solve this problem, we designate the Table as a *context* node, so that the relation from Column to Table is always traversed, both for Columns that are upstream as well as downstream.

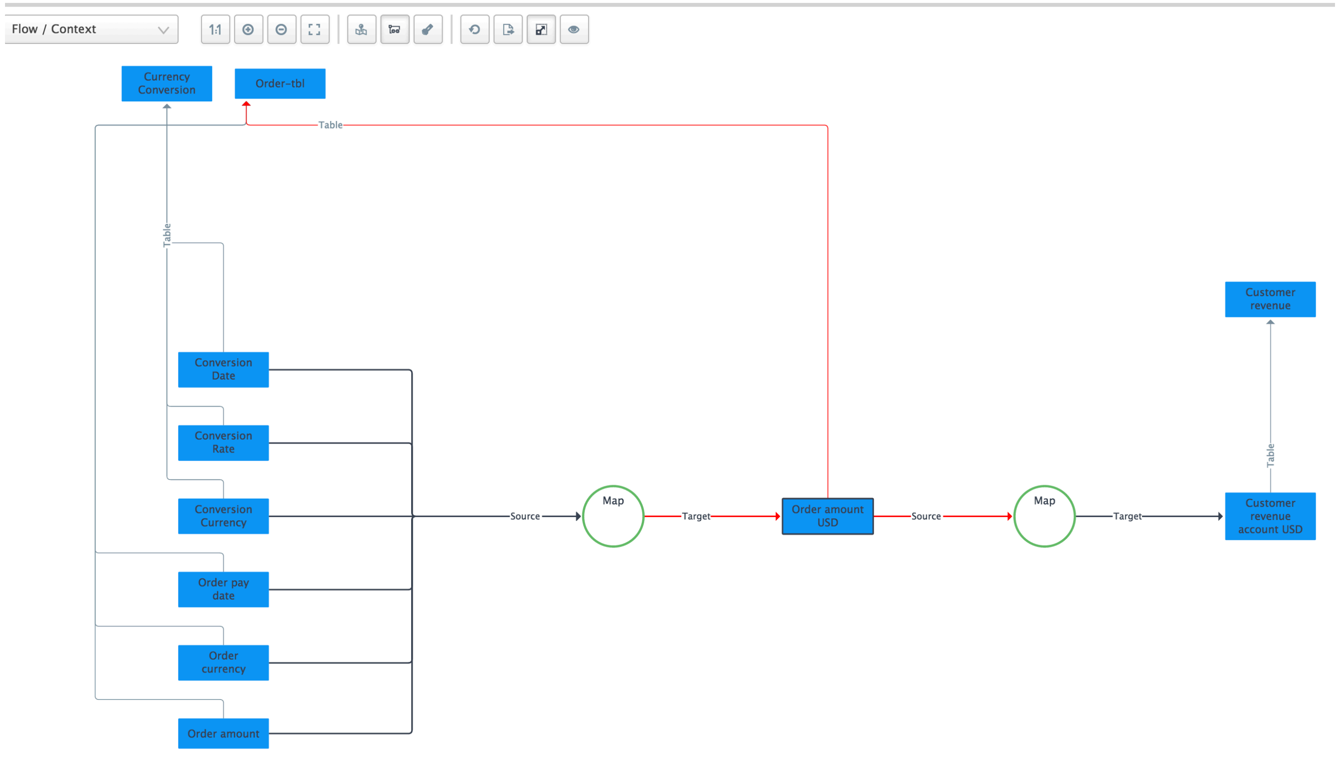
Technical lineage with Table context.


Technical lineage with Table conte

JSON | Diagram | Flow / Context | ✓ Show Edge Labels | ✓ Allow Duplicates

Resulting in the diagram below:

Technical lineage with Table context.



Both upstream and downstream Columns show the Table that contains them. The edge from Column to Table was always traversed.

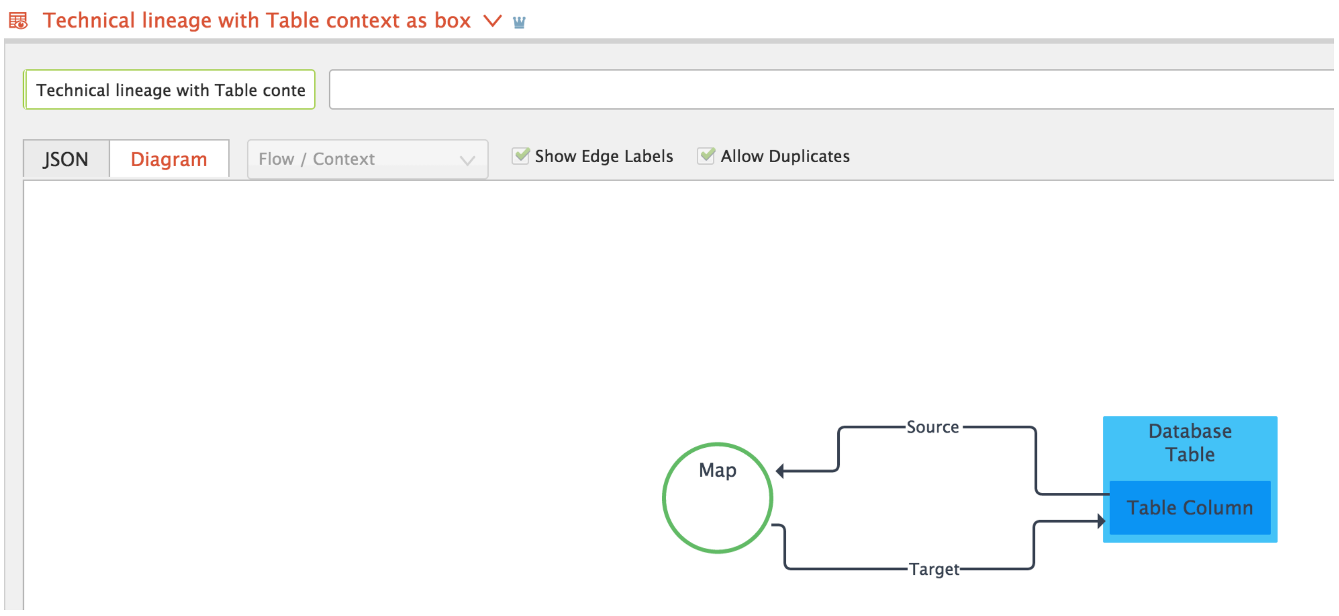
Among the list of diagram layout options, 'Flow/Context' seems to be special. What is so special about it?

A: Indeed, 'Flow/Context' is a special layout, with specific behavior.
If you set the diagram layout to 'Flow/Context', the flow nodes (nodes that are in the 'Flow' layout region) are depicted as a left-to-right

hierarchy in the lower half of the diagram and the context nodes are all in the top half of the diagram.

Can I direct a diagram view to show certain related assets as 'boxes in boxes'?

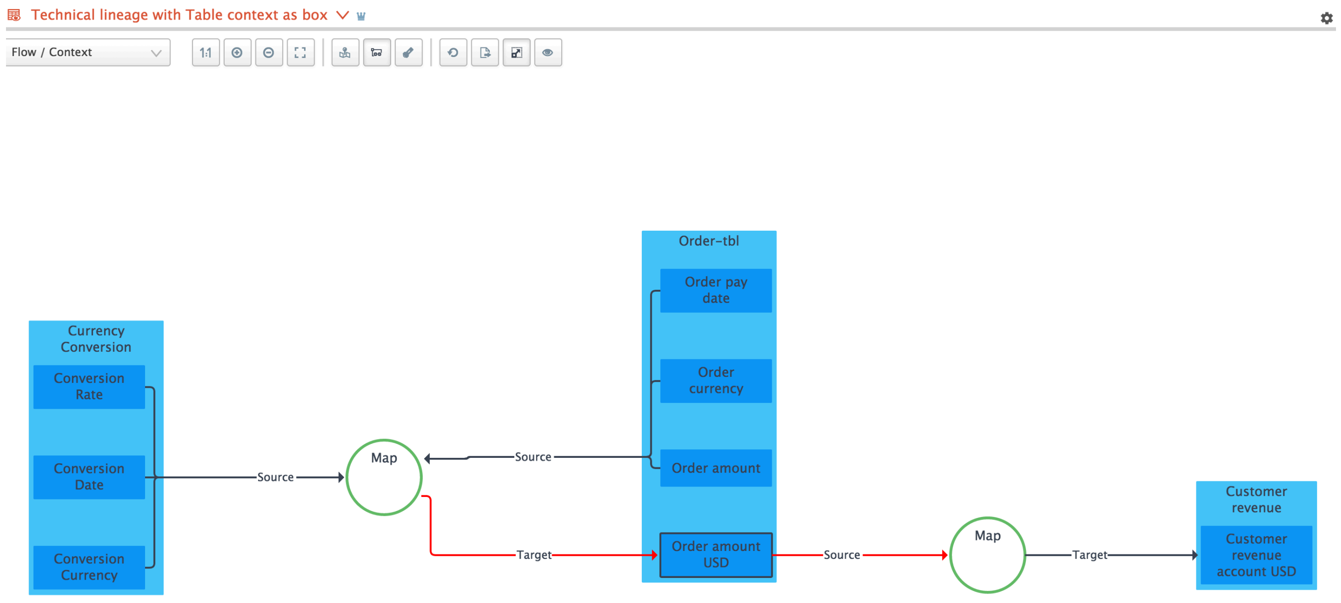
A: Yes. For example, you can further enhance the layout of the above diagram by depicting the Table as a box around its columns:



You can achieve this view by editing the view in JSON form. In the section for the edge from Table Column to Database Table, you add the following name/value pair:

```
"style": "reversed-box"
```

This view results in below diagram:



Note that the flow is no longer purely left to right. The Order-tbl contains both columns that are a source, as well as columns that are a target.

To keep all columns of Order-tbl in a single box in the middle of the diagram, one arrow now goes from right to left.