GEO - Graph Evolution Operation

Dominique Hausler

University of Regensburg
Data Engineering Group, Faculty of Informatics and Data Science
dominique.hausler@ur.de

November 28, 2023

```
(*add operation*)
add ::= addReplicable | addUnique
addReplicable ::= addEntities | addFeatures
addEntityTypes ::= "add" entityTypes "to" database
// addFeatures ::= "add" features "to" entityTypes
addLabels ::= "add" labels "of multi-labeled" nodes "to other" nodes
addNodeProps ::= "add node" properties "from" nodes "to other" nodes
addNodeProps ::= "add relationship" properties "from" relationships "to other" relationships
addUnique ::= (addEntityTypes | addFeatures) ("if created add" properties)? ("if unique add"
     properties)?

(*delete operation*)
delete ::= restrictedDelete | cascadeDelete | remove
restrictedDelete ::= "delete" entityTypes "from" database
cascadeDelete ::= "delete" connectedNodes "from" database
remove ::= "remove" features "from" entityTypes

(*rename operation*)
rename ::= "rename" features "to new" name
name ::= String

(*transform operation*)
transform :: = nodeToRel | relToNode | propToNode
nodeToRel ::= "transform" node "with its" features "into a" relationship
relToNode ::= "transform a" relationship "with its" features "into a" node
propToNode ::= "select a" property "of n" nodes "and transform them into a" node "connected by a"
     relationship "to the nodes prior containing this property"

(*merge operation*)
join ::= innerJoin | fullOuterExclusive | fullOuterInclusive | leftJoin | rightJoin
innerJoin ::= "merge duplicate" properties "of" entityTypes "," add (labels | types) "and" (
     cascadeDelete | restrictedDelete) "originals"
fullOuterExclusive ::= "merge different" properties "of" entityTypes "," remove "duplicate"
     properties "and" (copy labels | add labels)
fullOuterInclusive ::= copy properties "of" entityTypes "to" entityTypes "and" (cascadeDelete |
     restrictedDelete) "originals"
leftJoin ::= "keep all features of left" entityTypes "add duplicates from right" entityTypes
rightJoin ::= "keep all features of right" entityTypes "add duplicates from right" entityTypes
```

```
(*copy operation*)
copy ::= copyEntityTypes | copyFeatures | copySubgraph
copyEntityTypes ::= "copy" (connectedNodes | nodes) ("keep" relationships | delete relationships)
//copyFeatures ::= "copy" features "to" entityTypes
copyFeatures ::= copyNodeFeatures | copyRelFeatures
copyNodeFeatures ::= "copy" (labels | "node" properties) "to other" nodes
copyRelFeatures ::= "copy relationship" properties "to other" relationships
copySubgraph ::= "copy" connectedNodes "linked as" subgraph

(*split operation*)
split ::= splitNodes | splitRelationships
splitNodes ::= "split" nodes "at" propertyName ("keep" relationships | delete relationships | "keep"
     relationships "of" (partA | partB))
splitRelationships ::= "split" relationships "at" propertyName (copy endNodes | remove properties "
     from" endNodes "of" (partA | partB) | "overwrite" properties "of" endNodes "of" (partA | partB))
partA ::= properties "till" propertyName
partB ::= properties "from" propertyName

(*move operation*)
move ::= moveSubgraph | moveNodes | moveRelationships | moveFeatures | moveDirection
moveSubgraph ::= "move" multiConnectedNodes "to new" node
moveEntityTypes ::= moveNodes | moveRelationships
moveNodes ::= "move" connectedNodes "to new" node
moveRelationships ::= "move" realtionships "from old" ((startNode "to new" startNode) | (endNode "to
     new" endNode))
// moveFeatures ::= "move" features "from original" entityTypes "to new" entityTypes
moveFeatures ::= moveLabels | moveNodeProps | moveRelProps
moveLabels ::= "move" labels "from original" nodes "to other" nodes
moveNodeProps ::= "move node" properties "from original" nodes "to other" nodes
moveRel Props ::= "move relationship" properties "from original" relationships "to other"
     relationships
moveDirection ::= "move" relationships "from" ((rightDirected "to" leftDirected) | (leftDirected "to"
      rightDirected))


entityType ::= node | relationship
entityTypes ::= nodes | relationships
node ::= (labels (properties)*)*
connectedNodes ::= node "through" relationships (rightDirected | leftDirected) "to" node
subgraph ::= connectedNodes (connectedNodes)* "of" database
direction ::= "-"
rightDirected ::= "→"
leftDirected ::= "←"
nodes ::= node ("," node )*
startNode ::= "outgoing relationship from" node
endNode ::= "ingoing reationship towards" node

relationship ::= (type (properties)*)*
relationships ::= relationship ("," relationship )*
variable ::= String | Char
feature ::= label | type | property
features ::= labels |properties
label ::= String
labels ::= label ("," label )*
type ::= String
property ::= key propertyValue
key ::= String
```

```
propertyValue ::= String | Char | Integer | Long | Float | Double | Boolean | Date | Array;
properties ::= property ("," property )*
```