Gremlin Cheat Sheet

Read-Only Traversals

Initial Lookups

Steps	Meaning
`V()`	get all vertices in the graph
`E()`	get all edges in the graph
`V().hasLabel(label1, label2, ⋯)`	get all vertices with the specified labels
`V().has(label, key, value)`	get all vertices with the specified label and the property `key` matching the provided `value`
`V(1)`	get the vertex with the id `1`

Access Properties

Steps	Meaning
`properties(key1, key2, ⋯)`	get all specified properties for the current element
`values(key1, key2, ⋯)`	get all specified property values for the current element
`valueMap(key1, key2, ⋯)`	get all specified property values for the current element as a map

Traversing the Graph

Steps	Meaning
`out(label1, label2, ⋯)`	get all adjacent vertices connected by outgoing edges with the specified labels
`outE(label1, label2, ⋯)`	get all outgoing edges with the specified labels
`inE(label1, label2, ⋯)`	get all incoming edges with the specified labels
`both(label1, label2, ⋯)`	get all adjacent vertices connected by an edge with the specified labels
`bothE(label1, label2, ⋯).otherV()`	traverse to all incident edges with the specified labels and then to the respective other vertices

Filters

Steps	Meaning
`has(key, value)`	keep the current element if the specified property has the given value
`has(key, predicate)`	keep the current element if the specified property matches the given predicate
`filter(traversal)`	keep the current element if the provided traversal emits a result
`not(traversal)`	keep the current element if the provided traversal doesn't emit a result
`where(predicate)`	keep the current element if it matches the predicate referencing another element

Aggregations

Steps	Meaning
`store(key)`	store the current element in the side-effect with the provided key
`aggregate(key)`	store all elements held by all current traversers in the side-effect with the provided key
`group([key]).by(keySelector)`	group all current elements by the provided `keySelector`; group into a side-effect if a side-effect `key` was provided, otherwise emit the result immediately
`fold()`	fold all current elements into a single list
`count()`	count the number of current elements
`min()/max()`	find the min/max value
`sum()`	compute the sum of all current values
`mean()`	compute the mean value of all current values

Branches

Steps	Meaning
`union(branch1, branch2, …)`	execute all branches and emit their results
`choose(condition, true-branch, false-branch)`	`if`/`then`/`else`-based traversal. If the condition matches (yields something), execute the `true-branch`, otherwise follow the `false-branch`.

Steps	Meaning

`choose(selector).`
`  option(opt1, traversal).`
`  option(opt2, traversal).`
`  option(optN, traversal)`
	value-based traversal; If an option value matches the value emitted by the `selector` traversal, the respective option traversal will be executed.

Mutating Traversals

Steps	Meaning
`addV(label)`	add a new vertex
`addE(label).from(source).to(target)`	adds a new edge between the two given vertices
`property(key, value)`	adds or updates the property with the given `key`