

Gremlin Cheat Sheet

Read-Only Traversals

Initial Lookups

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

Access Properties

Steps	Meaning
<code>properties(key1, key2, ...)</code>	get all specified properties for the current element
<code>values(key1, key2, ...)</code>	get all specified property values for the current element
<code>valueMap(key1, key2, ...)</code>	get all specified property values for the current element as a map

Traversing the Graph

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

Filters

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

Aggregations

Steps	Meaning
<code>store(key)</code>	store the current element in the side-effect with the provided key
<code>aggregate(key)</code>	store all elements held by all current traversers in the side-effect with the provided key
<code>group([key]).by(keySelector)</code>	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
<code>fold()</code>	fold all current elements into a single list
<code>count()</code>	count the number of current elements
<code>min()/max()</code>	find the min/max value
<code>sum()</code>	compute the sum of all current values
<code>mean()</code>	compute the mean value of all current values

Branches

Steps	Meaning
<code>union(branch1, branch2, ...)</code>	execute all branches and emit their results
<code>choose(condition, true-branch, false-branch)</code>	if/then/else -based traversal. If the condition matches (yields something), execute the true-branch , otherwise follow the false-branch .
<code>choose(selector). option(opt1, traversal). option(opt2, traversal). option(optN, traversal)</code>	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
<code>addV(label)</code>	add a new vertex
<code>addE(label).from(source).to(target)</code>	adds a new edge between the two given vertices
<code>property(key, value)</code>	adds or updates the property with the given key