Graql Cheatsheet

This cheatsheet is a quick reference guide for those already familiar with Graql. For further information and additional documentation, please see the Graql section on our developer portal [QUICK LINK].

## Variables

Variables start with a `$`, followed by alphanumeric characters.

match (director: $the-director, $theMovie); $theMovie isa movie;

## Queries

## match

A match query will search the graph for any subgraphs that match the given pattern.

**match [pattern; ...] [modifiers]**

match $x isa movie;

# Match several patterns together.

match $x isa movie, has title "Titanic"; (actor: $a, $x);

**Modifiers**

match $m isa movie; (actor: $a1, $m); (actor: $a2, $m);

# Select particular variables from the query

select $a1, $a2;

match (director: $x, $y);

# Skip some results in a query & limit the number returned.

limit 10; offset 20;

match $m isa movie, has title 'Dr. Strangelove'; (actor: $a, $m);

select $a;

# De-duplicate the results of a query.

distinct;

match $x isa person, has name $n;

# Order by variable [ asc | desc ]

order by $n desc;

### ask

An ask query will return whether the given match query has any results.

**[match] ask**

match $x isa person, has name 'James Cameron'; (actor: $x);

ask;

### insert

An insert query will insert the specified variable patterns into the graph.

**insert [ pattern ; ... ]**

insert isa movie, has title 'Finding Dory';

If a match query is provided, the query will insert the given variable patterns for every result of the match query.

match $burton has name "Tim Burton";

$m isa movie; (director: $burton, $m);

$depp has name "Johnny Depp";

# Insert a relation for every

insert (actor: $depp, production-with-cast: $m) isa has-cast;

### delete

A delete query will delete the specified variable patterns for every result of the match query.

**match delete [ pattern ; ... ]**

match $x isa person;

Delete every instance of a type.

delete $x;

## Pattern matching

Match a variable

**identifier [ property, ... ]**

match $x isa person, value "Guillermo del Toro";

Match either the left or right pattern.

**pattern *or* pattern**

match $x isa movie or $x isa person;

Match either the left pattern or all the right patterns.

**{ [ pattern ; ... ] }**

match $x isa movie or { (actor: $x, $y); $y has name 'The Martian'; };

## Type properties

Specify the type of a concept.

**isa type**

match $x isa movie;

Match concepts and their types.

match $x isa $y;

Match the concept with a particular ID.

**id {string}**

match $x id '12345';

Match concepts with a value that contains the given string.

**value [ = | != | < | <= | >= | > | contains ] {value}**

match $m value contains "The Lord of the Rings";

Match concepts with a resource matching a predicate.

**has resource [ = | != | < | <= | >= | > | contains ] {value}**

match $m isa movie, has runtime > 180;

match (director: $p, $m);

Match concepts in a ternary relation.

match (actor: $p, character-being-played: $c, production-with-cast: $m);

## sub

**sub type**

Insert a new type that is a subclass of an existing type.

insert blockbuster sub movie;

## has-role

**has-role role**

Insert a new relation with two roles.

Insert director sub role;

production-with-director sub role;

directorship sub relation, has-role director, has-role production-with-director;

## plays-role

**plays-role role**

Allow instances of a type to play a role in a relation.

insert person plays-role director;

## has-resource

**has-resource resource**

Allow instances of a type to have a resource.

insert person has-resource name;

## datatype

**datatype ( string | long | double | boolean )**

Insert a new resource with the given datatype.

insert name sub resource, datatype string;