**Grakn example using Docker on Windows Command Prompt**

**Download and Install Grakn on Docker (**[**https://dev.grakn.ai/docs/running-grakn/install-and-run**](https://dev.grakn.ai/docs/running-grakn/install-and-run)**)**

docker pull graknlabs/grakn:latest

(docker image ls 🡪 graknlabs/grakn … 697MB)

docker run --name grakn -d -p 48555:48555 graknlabs/grakn:latest

(if you get the error: ‘docker: Error response from daemon: Conflict. The container name "/grakn" is already in use by container …’, it means you have run this command before. 🡪 docker ps -a

if ‘graknlabs/grakn:latest … Status = Exited …’ then

either delete the container 🡪 docker rm -f grakn

then create new container 🡪 docker run --name grakn -d -p 48555:48555 graknlabs/grakn:latest

or try restart 🡪 docker container start grakn

**some server commands**

docker exec -ti grakn bash -c "/grakn-core-all-linux/grakn server status"

docker exec -ti grakn bash -c "/grakn-core-all-linux/grakn server version"

docker exec -ti grakn bash -c "/grakn-core-all-linux/grakn server”

**Use the console (**[**https://dev.grakn.ai/docs/running-grakn/console**](https://dev.grakn.ai/docs/running-grakn/console) **)**

docker exec -ti grakn bash -c "/grakn-core-all-linux/grakn console version"

(version 🡪 get console version 🡪 <https://dev.grakn.ai/docs/running-grakn/console>)

docker exec -ti grakn bash -c "/grakn-core-all-linux/grakn console"

(opens and creates if it does not exist keyspace named grakn)

grakn> exit

(exit 🡪 exit Grakn console 🡪 <https://dev.grakn.ai/docs/running-grakn/console>)

docker exec -ti grakn bash -c "/grakn-core-all-linux/grakn console -k mykeyspace"

(--keyspace <name> 🡪 enter and/or create keyspace 🡪 <https://dev.grakn.ai/docs/running-grakn/console>)

(opens and creates if it does not exist keyspace named mykeyspace)

mykeyspace> keyspace list

(keyspace list 🡪 list keyspace/s in Grakn server 🡪 <https://dev.grakn.ai/docs/running-grakn/console>)

(lists 2 keyspaces: grakn and mykeyspace)

mykeyspace> keyspace delete grakn

(keyspace delete ksp\_name 🡪 delete a keyspace 🡪 <https://dev.grakn.ai/docs/running-grakn/console>)

mykeyspace> keyspace list

(up arrow twice to get previous previous command)

mykeyspace> clean

(clean 🡪 delete current keyspace 🡪 <https://dev.grakn.ai/docs/running-grakn/console>)

docker exec -ti grakn bash -c "/grakn-core-all-linux/grakn console -k demo"

demo> keyspace list

demo> clear

(clear 🡪 clear console 🡪 <https://dev.grakn.ai/docs/running-grakn/console>)

**Define entity, commit, get type of entity, undefine entity, and commit.**

demo> define person sub entity;

(Define an entity 🡪 <https://dev.grakn.ai/docs/schema/concepts#define-an-entity>)

demo> commit

(commit 🡪 commit changes to Grakn server 🡪 <https://dev.grakn.ai/docs/running-grakn/console>)

demo> match $x type person; get;

(A given type 🡪 <https://dev.grakn.ai/docs/query/match-clause#a-given-type>)

demo> undefine person sub entity;

(Undefine 🡪 <https://dev.grakn.ai/docs/schema/concepts#undefine>) Error if ‘person sub entity’ does not exits.

demo> commit

demo> match $x type person; get;

(label 'person' not found 🡪 person entity no longer exists)

**Define entity with attributes, insert two instances of entity, match & get instances of entity, and rollback.**

demo> define name sub attribute, value string;

demo> define hairColour sub attribute, value string;

(Define an attribute 🡪 <https://dev.grakn.ai/docs/schema/concepts#define-an-attribute>)

demo> define person sub entity, has name, has hairColour;

(Assign an attribute to an entity 🡪 <https://dev.grakn.ai/docs/schema/concepts#assign-an-attribute-to-an-entity> )

demo> commit

demo> insert $p isa person, has name "Ann", has hairColour "pink";

demo> insert $p isa person, has name "Bo", has hairColour "red";

(Insert Instances of an Entity Type 🡪 <https://dev.grakn.ai/docs/query/insert-query#insert-instances-of-an-entity-type>)

demo> commit

demo> insert $p isa person, has name "Crazzzzy", has hairColour "none";

demo> match $p isa person; get;

(Match instances of an entity 🡪 <https://dev.grakn.ai/docs/query/match-clause#match-instances-of-an-entity>)

(Get the Variables 🡪 <https://dev.grakn.ai/docs/query/get-query#get-the-variables>)

demo> match $p isa person, has name $n; get;

(Instances of an entity with particular attributes 🡪 <https://dev.grakn.ai/docs/query/match-clause#instances-of-an-entity-with-particular-attributes>)

demo> match $p isa person, has name $n, has hairColour $hc; get;

(Owners with multiple attributes 🡪 <https://dev.grakn.ai/docs/query/match-clause#owners-with-multiple-attributes>)

demo> match $p isa person, has name $n, has hairColour "none"; get;

(Owners with attributes of given values 🡪 <https://dev.grakn.ai/docs/query/match-clause#owners-with-attributes-of-given-values>)

demo> match $p isa person, has name $n; get; limit 2;

(Limit the Answers 🡪 <https://dev.grakn.ai/docs/query/get-query#limit-the-answers>)

demo> match $p isa person, has name $n; get; sort $n desc;

(Order the Answers -> <https://dev.grakn.ai/docs/query/get-query#order-the-answers>)

demo> match $p isa person, has name $n; get; sort $n desc; limit 2;

demo> match $p isa person, has name $n; get; sort $n asc; limit 2;

demo> rollback

(rollback 🡪 undue changes since last commit 🡪 <https://dev.grakn.ai/docs/running-grakn/console>)

demo> match $p isa person, has name $n; get;

(Crazzzzy no longer listed, as not committed to Grakn, so lost in rollback)

demo> match $x "Bo"; get;

(Independent of label 🡪 <https://dev.grakn.ai/docs/query/match-clause#independent-of-label>)

demo> match $n isa name; $n "Ann"; get;

(Independent of owner 🡪 <https://dev.grakn.ai/docs/query/match-clause#independent-of-owner>)

**Migration and backup**

demo> exit

docker exec -ti grakn bash -c "/grakn-core-all-linux/grakn server schema demo > /demo1Schema.gql"

(Schema Migration 🡪 <https://dev.grakn.ai/docs/management/migration-and-backup#schema-migration>)

docker exec -ti grakn bash -c "/grakn-core-all-linux/grakn server export demo /demo1Data.grakn"

(Data Migration 🡪 <https://dev.grakn.ai/docs/management/migration-and-backup#data-migration>)

(Backup 🡪 <https://dev.grakn.ai/docs/management/migration-and-backup#backup>)

docker exec -ti grakn bash -c "ls -l /"

-rw-r--r-- 1 root root 209 Nov 22 21:50 demo1Data.grakn

-rw-r--r-- 1 root root 147 Nov 22 21:50 demo1Schema.gql

docker cp grakn:/demo1Data.grakn c:/temp

docker cp grakn:/demo1Schema.gql c:/temp

docker exec -ti grakn bash -c "rm /demo1Data.grakn /demo1Schema.gql"

docker exec -ti grakn bash -c "rm /demo1Backup.grakn"

docker exec -ti grakn bash -c "ls -l /"

(copy schema backup file and data backup file to windows folder and delete files from container)

**Restore schema and data from backups using demo1Schema.gql and demo1Data.grakn**

docker cp c:/temp/demo1Data.grakn grakn:/demo1Data.grakn

docker cp c:/temp/demo1Schema.gql grakn:/demo1Schema.gql

-rwxr-xr-x 1 root root 209 Nov 22 21:50 demo1Data.grakn

-rwxr-xr-x 1 root root 147 Nov 22 21:50 demo1Schema.gql

(copy schema backup file and data backup file to windows folder and delete files from container)

docker exec -ti grakn bash -c "/grakn-core-all-linux/grakn console -k demo"

demo> clean

demo> confirm

docker exec -ti grakn bash -c "/grakn-core-all-linux/grakn console -k demo -f /demo1Schema.gql"

docker exec -ti grakn bash -c "/grakn-core-all-linux/grakn server import demo /demo1Data.grakn"

docker exec -ti grakn bash -c "/grakn-core-all-linux/grakn console -k demo"

demo> match $p isa person, has name $n; get; sort $n asc;

(Bo and Ann)

**2: continue or start with demo1Schema.gql and demo1Data.grakn**

**Add third instance of entity (person), match & get instances of entity. Add attribute then remove attribute.**

demo> define isObese sub attribute, value boolean;

demo> define person sub entity, has isObese;

(an attribute added to the entity, all other attributes do not need to be redefined)

demo> commit

(it is good to do a commit on every acceptable change, otherwise you might lose your work by making a typo in a later command)

demo> match $p isa person, has name $n, has hairColour $hc; get;

(first two instances displayed)

demo> insert $p isa person, has name "Crazzzzy", has hairColour "none", has isObese true;

demo> insert $p isa person, has name "Crazzzzy2", has isObese false;

demo> commit

(an entity instance does not have to define all attributes of its entity type)

demo> match $p isa person, has name $n, has hairColour $hc; get;

(three instances displayed, Crazzzzy2 does not have attribute hairColour)

demo> match $p isa person, has name $n, has isObese $io; get;

(only Crazzzzy and Crazzzzy2 are instances of person entity with attributes name and isObese)

demo> match $p isa person, has name $n, has hairColour $hc, has isObese $io; get;

(only Crazzzzy is an instances of person entity with attributes name, hairColour and isObese)

demo> match $p isa person, has isObese $io; $io true; delete $p has isObese $io;

(Delete Attribute Ownerships 🡪 <https://dev.grakn.ai/docs/query/delete-query#delete-attribute-ownerships>)

(disassociate isObese = true from person entities 🡪 Crazzzzy)

demo> commit

demo> match $p isa person, has name $n, has isObese $io; get;

(only Crazzzzy2 is an instances of person entity with attributes name and isObese)

demo> match $p isa person, has isObese $io; delete $p has isObese $io;

(disassociate isObese, of any value, from person entities 🡪 Crazzzzy2)

demo> commit

demo> match $p isa person, has isObese $io; get;

(no instances of person entity with attributes name and isObese)

demo> undefine person has isObese;

(Undefine an attribute’s association 🡪 <https://dev.grakn.ai/docs/schema/concepts#undefine-an-attributes-association>)

(disassociate the attribute isObese with the entity person)

demo> commit

demo> insert $p isa person, has name "Dale", has hairColour "green", has isObese true;

(this fails as isObese attribute no longer associated with person entity)

demo> match $io isa isObese; delete $io isa isObese;

(need to delete all instances of attribute before deleting attribute type)

demo> match $x type isObese; get;

(shows that isObese is sub attribute)

demo> undefine isObese sub attribute;

(remove the isObese attribute type from the keyspace)

demo> commit

demo> match $x type isObese; get;

(isObese attribute type no longer exists)

**delete two instances of an entity (person)**

demo> match $p isa person, has name $n; get;

demo> match $p isa person, has name "Crazzzzy"; delete $p isa person;

demo> match $p isa person, has name "Crazzzzy2"; delete $p isa person;

demo> commit

(Delete Instances of a Type 🡪 <https://dev.grakn.ai/docs/query/delete-query#delete-instances-of-a-type>)

demo> match $p isa person, has name $n; get;

(only Ann and Bo left)

demo> exit

docker exec -ti grakn bash -c "/grakn-core-all-linux/grakn server schema demo > /demo2Schema.gql"

docker exec -ti grakn bash -c "/grakn-core-all-linux/grakn server export demo /demo2Data.grakn"

**3: continue or start with demo2Schema.gql and demo2Data.grakn**

docker exec -ti grakn bash -c "/grakn-core-all-linux/grakn console -k demo"

demo> match $p isa person, has name $n, has hairColour $hc; get;

**add four more persons update with age (in years) and gender (male or !male)**

demo> define age sub attribute, value long;

demo> define isMale sub attribute, value boolean;

demo> define person sub entity, has age, has isMale;

demo> commit

demo> insert $p isa person, has name "Charles", has hairColour "red", has age 30, has isMale true;

demo> insert $p isa person, has name "David", has hairColour "brown", has age 32, has isMale true;

demo> insert $p isa person, has name "Emma", has hairColour "brown", has age 28, has isMale false;

demo> insert $p isa person, has name "Fiona", has hairColour "green", has age 45, has isMale false;

demo> commit

demo> match $p isa person, has name "Ann"; insert $p has age 29;

demo> match $p isa person, has name "Ann"; insert $p has isMale false;

demo> match $p isa person, has name "Bo"; insert $p has age 27, has isMale true;

(similar to <https://dev.grakn.ai/docs/query/updating-data#update-attribute-owned-by-a-concept> but no delete, just insert >= one attributes to existing entity)

(also https://dev.grakn.ai/docs/query/insert-query)

demo> commit

demo> match $p isa person, has name $n, has hairColour $hc, has age $a, has isMale $im; get; sort $n asc;

**various match & get examples, update attribute of entity (age of person)**

demo> match $hairColour contains "re"; get;

(With a given subset 🡪 <https://dev.grakn.ai/docs/query/match-clause#with-a-given-subset> )

demo> match $x like "(Bo|Emma)"; get;

(With a given regex 🡪 <https://dev.grakn.ai/docs/query/match-clause#with-a-given-regex>)

demo> match $p isa person, has name $n; { $n contains "Ann"; } or { $n contains "Emma"; }; get;

(Disjunction of patterns 🡪 <https://dev.grakn.ai/docs/query/match-clause#disjunction-of-patterns>)

demo> match $p isa person, has name "Ann"; get; 🡪 the next line uses the id of "Ann"

demo> match $x id V4312; get;

(One particular instance 🡪 <https://dev.grakn.ai/docs/query/match-clause#one-particular-instance>)

demo> match $p isa person, has name "Bo", has age $a; delete $p has age $a;

demo> match $p isa person, has name "Bo"; insert $p has age 32;

(Update attribute owned by a concept 🡪 <https://dev.grakn.ai/docs/query/updating-data#update-attribute-owned-by-a-concept>)

demo> commit

demo> match $p isa person, has name $n, has hairColour $hc, has age $a, has isMale $im; get; sort $n asc;

demo> match $s isa person, has age < 32; get;

(Owners with attributes of given values 🡪 <https://dev.grakn.ai/docs/query/match-clause#owners-with-attributes-of-given-values>)

**aggregate-query: count, sum, maximum, minimum, mean, median, group answers**

demo> match $p isa person, has age $a, has isMale $im; get; sort $im asc;

demo> match $p isa person, has age $a; get; count;

demo> match $p isa person, has age $a; $a > 40; get; count;

(Count 🡪 <https://dev.grakn.ai/docs/query/aggregate-query#count>)

demo> match $p isa person, has age $a; get; sum $a;

demo> match $p isa person, has hairColour "brown", has age $a; get; sum $a;

(Sum 🡪 <https://dev.grakn.ai/docs/query/aggregate-query#sum>)

demo> match $p isa person, has age $a; get; max $a;

demo> match $p isa person, has isMale true, has age $a; get; max $a;

(Maximum 🡪 <https://dev.grakn.ai/docs/query/aggregate-query#maximum>)

demo> match $p isa person, has age $a; get; min $a;

demo> match $p isa person, has isMale true, has age $a; get; min $a;

(Minimum 🡪 <https://dev.grakn.ai/docs/query/aggregate-query#minimum>)

demo> match $p isa person, has age $a; get; mean $a;

(Mean 🡪 <https://dev.grakn.ai/docs/query/aggregate-query#mean>)

demo> match $p isa person, has age $a; get; median $a;

(Median 🡪 <https://dev.grakn.ai/docs/query/aggregate-query#median>)

demo> match $p isa person, has hairColour $hc, has age $a; get; group $hc;

demo> match $p isa person, has hairColour $hc, has age $a; get; group $hc; mean $a;

(Grouping Answers 🡪 <https://dev.grakn.ai/docs/query/aggregate-query#grouping-answers>)

**compute query: count, sum, maximum, minimum, mean, median, and standard deviation**

demo> match $p isa person, has age $a; get; sort $a asc;

demo> compute count in person;

demo> compute count;

(Count 🡪 <https://dev.grakn.ai/docs/query/compute-query#count>)

demo> compute sum of age, in person;

(Sum 🡪 <https://dev.grakn.ai/docs/query/compute-query#sum>)

demo> compute max of age, in person;

(Maximum 🡪 <https://dev.grakn.ai/docs/query/compute-query#maximum>)

demo> compute min of age, in person;

(Minimum 🡪 <https://dev.grakn.ai/docs/query/compute-query#minimum>)

demo> compute mean of age, in person;

(Mean 🡪 <https://dev.grakn.ai/docs/query/compute-query#mean>)

demo> compute median of age, in person;

(Median 🡪 <https://dev.grakn.ai/docs/query/compute-query#median>)

demo> compute std of age, in person;

(Standard Deviation 🡪 <https://dev.grakn.ai/docs/query/compute-query#standard-deviation>)

demo> exit

docker exec -ti grakn bash -c "/grakn-core-all-linux/grakn server schema demo > /demo3Schema.gql"

docker exec -ti grakn bash -c "/grakn-core-all-linux/grakn server export demo /demo3Data.grakn"

**4: continue or start with demo3Schema.gql and demo3Data.grakn**

demo> match $p isa person, has hairColour $hc, has age $a; get; group $hc;

**Traditional marriage relation**

demo> define marriage sub relation;

(Define a relation 🡪 <https://dev.grakn.ai/docs/schema/concepts#define-a-relation>)

demo> undefine marriage sub relation;

(Undefine a relation 🡪 <https://dev.grakn.ai/docs/schema/concepts#undefine-a-relation>)

demo> define marriage sub relation, abstract, relates husband, relates wife;

(Define an abstract relation 🡪 <https://dev.grakn.ai/docs/schema/concepts#define-an-abstract-relation>)

(add husband and wife roles to the marriage relation)

demo> commit

demo> define contractDate sub attribute, value datetime;

demo> define marriage sub relation, has contractDate;

(Assign an attribute to a relation 🡪 <https://dev.grakn.ai/docs/schema/concepts#assign-an-attribute-to-a-relation>)

demo> commit

demo> define contractRefNum sub attribute, value string;

demo> define marriage sub relation, key contractRefNum;

(Assign an attribute to a relation as a unique identifier 🡪 <https://dev.grakn.ai/docs/schema/concepts#assign-an-attribute-to-a-relation-as-a-unique-identifier>)

demo> commit

demo> define secularMarriage sub marriage, relates partner1 as wife, relates partner2 as husband;

demo> define religousMarriage sub marriage, relates spouse1 as wife, relates spouse2 as husband;

(Subtype a relation 🡪 <https://dev.grakn.ai/docs/schema/concepts#subtype-a-relation>)

demo> commit

demo> match religousMarriage relates $x as husband; get;

(Subroles of a given role in a super-relation 🡪 <https://dev.grakn.ai/docs/query/match-clause#subroles-of-a-given-role-in-a-super-relation>)

demo> define marriageCeremony sub relation, relates marriageToBe, relates celebrant;

demo> define marriage sub relation, plays marriageToBe;

(Define a relation to play a role 🡪 <https://dev.grakn.ai/docs/schema/concepts#define-a-relation-to-play-a-role>)

demo> commit

demo> define family sub relation, relates father, relates mother, relates daughter, relates son;

(A relation with many role players 🡪 <https://dev.grakn.ai/docs/schema/concepts#a-relation-with-many-role-players>)

demo> commit

demo> match family relates $x; get;

(Roles of a given relation 🡪 <https://dev.grakn.ai/docs/query/match-clause#roles-of-a-given-relation>)

demo> define person sub entity, plays celebrant, plays spouse1, plays spouse2;

(Entity to play a role 🡪 <https://dev.grakn.ai/docs/schema/concepts#entity-to-play-a-role>)

demo> commit

demo> match $w isa person, has name "Ann"; $h isa person, has name "Bo"; insert $newReligiousMarriage (spouse1: $w, spouse2: $h) isa religousMarriage; $newReligiousMarriage has contractDate 2020-02-27; $newReligiousMarriage has contractRefNum "abc123";

(Insert Instances of a Relation Type 🡪 <https://dev.grakn.ai/docs/query/insert-query#insert-instances-of-a-relation-type>)

demo> commit

demo> match $c isa person, has name "Fiona"; $m isa marriage, has contractRefNum "abc123"; insert $newMarriageCeremony (marriageToBe: $m, celebrant: $c) isa marriageCeremony;

demo> commit

demo> define person sub entity, plays father, plays mother, plays daughter, plays son;

demo> commit

demo> match person plays $x; get;

demo> match $f isa person, has name "Bo"; $m isa person, has name "Ann"; $d isa person, has name "Emma"; $s isa person, has name "David"; insert $newFamily (father: $f, mother: $m, daughter: $d, son: $s) isa family;

demo> commit

demo> match $x plays mother; get;

(Role players of a given role 🡪 [https://dev.grakn.ai/docs/query/match-clause #role-players-of-a-given-role](https://dev.grakn.ai/docs/query/match-clause%20#role-players-of-a-given-role))

demo> match $x has contractRefNum; get;

(Owners of a given attribute 🡪 <https://dev.grakn.ai/docs/query/match-clause#owners-of-a-given-attribute>)

demo> match $x sub! marriage; get;

(Direct subtypes of a given type 🡪 <https://dev.grakn.ai/docs/query/match-clause#direct-subtypes-of-a-given-type>)

demo> match $m (wife: $w, husband: $h) isa marriage; get;

(Match instances of a relation 🡪 <https://dev.grakn.ai/docs/query/match-clause#match-instances-of-a-relation>)

demo> match (father:$f, mother: $m, daughter: $d, son: $s); get;

(Leave the relation instance unassigned 🡪 <https://dev.grakn.ai/docs/query/match-clause#leave-the-relation-instance-unassigned>)

demo> match (father:$f, mother: $m, daughter: $d, son: $s); $f isa person, has name $fn; get;

(get father’s name in family)

demo> match (father:$f, mother: $m, daughter: $d, son: $s); $f isa person, has name $fn; $m isa person, has name $mn; $d isa person, has name $dn; $s isa person, has name $sn; get;

demo> define person sub entity, plays partner1, plays partner2;

demo> commit

demo> match $p2 isa person, has name "Charles"; $p1 isa person, has name "Emma"; insert $newSecularMarriage (partner1: $p1, partner2: $p2) isa secularMarriage; $newSecularMarriage has contractDate 2020-02-28; $newSecularMarriage has contractRefNum "def456";

demo> commit

demo> match $c isa person, has name "Fiona"; $m isa marriage, has contractRefNum "def456"; insert $newMarriageCeremony (marriageToBe: $m, celebrant: $c) isa marriageCeremony;

demo> commit

demo> match $m (wife: $w, husband: $h) isa marriage; get;

demo> match $m (wife: $w, husband: $h) isa marriage, has contractRefNum "def456"; get;

(Instances of a relation with particular attributes 🡪 <https://dev.grakn.ai/docs/query/match-clause#instances-of-a-relation-with-particular-attributes>)

demo> match $f isa family; get;

demo> match $m isa marriage; get;

demo> match $mc isa marriageCeremony; get;

demo> match $p isa person, has name $n; get; sort $n asc;

demo> compute path from V16416, to V40968296; (David’s id to Charles’ id, will probably be different for every database.)

(Compute the Shortest Path 🡪 <https://dev.grakn.ai/docs/query/compute-query#compute-the-shortest-path>)

(David (son) to family-relation to Emma (daughter, partner1) to secularMarriage-relation to Charles (partner2))

(I could not work out how to get id from the person’s name in console and put it in compute. It can be done via a client, such as Node.js)

demo> compute path from V20512, to V16504; (Emma’s id to Fiona’ id)

(Emma (partner1) to secularMarriage-relation to marriageCeremony to Fiona (celebrant))

demo> compute path from V20512, to V16504, in [person, family, religousMarriage, marriageCeremony];

(Emma’s id to Fiona’ id, exclude secularMarriage-relation by making a whitelist)

(Specify a whitelist 🡪 <https://dev.grakn.ai/docs/query/compute-query#specify-a-whitelist>)

(Emma (daughter) to family-relation to Ann (mother, spouse1) to religousMarriage-relation to marriageCeremony-relation to Fiona (celebrant))

(Emma (daughter) to family-relation to Bo (father, spouse2) to religousMarriage-relation to marriageCeremony-relation to Fiona (celebrant))

demo> compute centrality using degree;

(Compute centrality using degree 🡪 <https://dev.grakn.ai/docs/query/compute-query#compute-centrality-using-degree>)

demo> compute centrality in [person, family, religousMarriage, marriageCeremony], using degree;

(Degree in a subgraph 🡪 <https://dev.grakn.ai/docs/query/compute-query#in-a-subgraph>)

demo> compute centrality of religousMarriage, in [person, family, religousMarriage, marriageCeremony], using degree;

(Degree of a given type 🡪 <https://dev.grakn.ai/docs/query/compute-query#of-a-given-type>)

demo> compute centrality using k-core;

(Compute centrality using k-core 🡪 <https://dev.grakn.ai/docs/query/compute-query#compute-centrality-using-k-core>)

demo> compute centrality using k-core, where min-k=3;

(Specify the minimum k value 🡪 <https://dev.grakn.ai/docs/query/compute-query#specify-the-minimum-k-value>)

demo> compute cluster in [person, family, religousMarriage, marriageCeremony], using connected-component;

(Compute clusters using connected component 🡪 <https://dev.grakn.ai/docs/query/compute-query#compute-clusters-using-connected-component>)

demo> compute cluster in [person, family, religousMarriage, marriageCeremony], using connected-component, where contains=V16504; (Fiona’s id)

(Retrieve the cluster that contains a given instance 🡪 <https://dev.grakn.ai/docs/query/compute-query#retrieve-the-cluster-that-contains-a-given-instance>)

demo> compute cluster in [person, family, marriage], using k-core;

(Compute clusters using k-core 🡪 <https://dev.grakn.ai/docs/query/compute-query#compute-clusters-using-k-core>)

demo> compute cluster in [person, family, marriage], using k-core, where k=2;

(Compute clusters specify the k value 🡪 <https://dev.grakn.ai/docs/query/compute-query#specify-the-k-value>)

(not a clear example as k=2 is both the maximum and minimum values for this cluster)

demo> exit

docker exec -ti grakn bash -c "/grakn-core-all-linux/grakn server schema demo > /demo4Schema.gql"

docker exec -ti grakn bash -c "/grakn-core-all-linux/grakn server export demo /demo4Data.grakn"

**5: continue or start with demo4Schema.gql and demo4Data.grakn**

**Friends**

demo> define friendship sub relation, relates friend;

demo> define sportFriendship sub friendship, relates sportFriend as friend;

demo> define rugbyFriendship sub sportFriendship, relates rugbyFriend as sportFriend;

demo> define person sub entity, plays friend, plays sportFriend, plays rugbyFriend;

demo> commit

demo> match person plays $x; get;

demo> match $d isa person, has name "David"; $c isa person, has name "Charles"; insert $newFriendship (friend: $d, friend: $c) isa friendship;

(Duplicate Role Players 🡪 <https://dev.grakn.ai/docs/query/insert-query#duplicate-role-players>)

demo> commit

demo> match $b isa person, has name "Bo"; $f isa person, has name "Fiona"; insert $newSportFriendship (sportFriend: $b, sportFriend: $f) isa sportFriendship;

demo> commit

demo> match $e isa person, has name "Emma"; $f isa person, has name "Fiona"; insert $newRugbyFriendship (rugbyFriend: $e, rugbyFriend: $f) isa rugbyFriendship;

demo> commit

demo> match $fr isa friendship; get;

(Direct and indirect subtypes of a given type 🡪 <https://dev.grakn.ai/docs/query/match-clause#direct-and-indirect-subtypes-of-a-given-type>)

demo> match $fr isa! friendship; get;

(Instances of a direct type 🡪 <https://dev.grakn.ai/docs/query/match-clause#instances-of-a-direct-type>)

demo> match $fr isa rugbyFriendship; get;

demo> match $fr ($x, $y) isa rugbyFriendship; get;

(Leave the roles out 🡪 <https://dev.grakn.ai/docs/query/match-clause#leave-the-roles-out> )

demo> exit

docker exec -ti grakn bash -c "/grakn-core-all-linux/grakn server schema demo > /demo5Schema.gql"

docker exec -ti grakn bash -c "/grakn-core-all-linux/grakn server export demo /demo5Data.grakn"

**6: continue or start with demo5Schema.gql and demo5Data.grakn**

demo> match $p isa person, has name $n; get; sort $n asc;

demo> match (rugbyFriend: $rf); $rf isa person, has name $n; get; sort $n asc;

**Company information**

demo> define shareValue sub attribute, abstract, value long;

(Define an abstract attribute 🡪 <https://dev.grakn.ai/docs/schema/concepts#define-an-abstract-attribute>)

demo> define votingShareValue sub shareValue;

demo> define nonVotingShareValue sub shareValue;

(Subtype an attribute 🡪 <https://dev.grakn.ai/docs/schema/concepts#subtype-an-attribute>)

demo> commit

demo> define Aud2Gbp sub attribute, value double;

demo> insert $x isa Aud2Gbp; $x 0.55;

(Insert Instances of an Attribute Type 🡪 <https://dev.grakn.ai/docs/query/insert-query#insert-instances-of-an-attribute-type>)

demo> commit

demo> define shareCurrency sub attribute, value string, regex "^(AUD,GBP)$ ";

(Restrict attribute’s value by Regex 🡪 <https://dev.grakn.ai/docs/schema/concepts#restrict-attributes-value-by-regex>)

demo> define ABN sub attribute, value string;

demo> define company sub entity, abstract, key ABN, has name;

(Define an abstract entity 🡪 <https://dev.grakn.ai/docs/schema/concepts#define-an-abstract-entity>)

(Assign an attribute to an entity as a unique identifier 🡪 <https://dev.grakn.ai/docs/schema/concepts#assign-an-attribute-to-an-entity-as-a-unique-identifier>)

demo> define privateCompany sub company;

(Subtype an entity 🡪 <https://dev.grakn.ai/docs/schema/concepts#subtype-an-entity>)

demo> define numberOfShares sub attribute, value long;

demo> define publicCompany sub company, has numberOfShares;

demo> commit

demo> define language sub attribute, value string;

demo> define employmentContract sub attribute, value string, has language;

(Assign an attribute to another attribute 🡪 <https://dev.grakn.ai/docs/schema/concepts#assign-an-attribute-to-another-attribute>)

demo> define employmentContractCreation sub relation, relates employmentContractWriter, relates employmentContractWritten;

demo> match employmentContractCreation relates $x; get;

demo> define person sub entity, plays employmentContractWriter;

demo> define language sub attribute, plays employmentContractWritten;

(Define an attribute to play a role 🡪 <https://dev.grakn.ai/docs/schema/concepts#define-an-attribute-to-play-a-role>)

demo> commit

demo> insert $x isa language; $x "BritishEnglish";

demo> match $la isa language; get;

demo> match $la isa language; insert $x isa employmentContract, has language $la; $x "casual";

demo> commit

demo> match $ec isa employmentContract; get;

demo> match $ecwr isa person, has name "Fiona"; $ecwn isa language; insert $ecc (employmentContractWriter: $ecwr, employmentContractWritten: $ecwn) isa employmentContractCreation;

demo> commit

demo> match $ecc isa employmentContractCreation; get;

demo> insert $prc isa privateCompany, has ABN "12345678901", has name "One";

demo> insert $puc isa publicCompany, has ABN "12345678902", has name "Two", has numberOfShares 2000;

demo> commit

demo> match $c isa company, has ABN $a, has name $n; get;

demo> define employment sub relation, relates employer, relates employee;

demo> commit

demo> match $x type employment; get;

demo> define person sub entity, plays employee;

demo> commit

demo> match person plays $x; get;

demo> define company sub entity, plays employer;

demo> commit

demo> match company plays $x; get;

demo> match $co isa company, has name "One"; $b isa person, has name "Bo"; insert $newEmployment (employer: $co, employee: $b) isa employment;

(Bo employed by One)

demo> match $co isa company, has name "One"; $c isa person, has name "Charles"; insert $newEmployment (employer: $co, employee: $c) isa employment;

(Charles employed by One)

demo> match $co isa company, has name "One"; $d isa person, has name "David"; insert $newEmployment (employer: $co, employee: $d) isa employment;

(David employed by One)

demo> commit

demo> match $co isa company, has name "Two"; $e isa person, has name "Emma"; insert $newEmployment (employer: $co, employee: $e) isa employment;

(Emma employed by Two)

demo> match $co isa company, has name "Two"; $c isa person, has name "Charles"; insert $newEmployment (employer: $co, employee: $c) isa employment;

(Charles employed by Two)

demo> match $co isa company, has name "Two"; $f isa person, has name "Fiona"; insert $newEmployment (employer: $co, employee: $f) isa employment;

(Fiona employed by Two)

demo> commit

demo> match $emp isa employment; get;

demo> match $co isa company, has name "Two"; $f isa person, has name "Fiona"; $oldEmployment (employer: $co, employee: $f) isa employment; delete $oldEmployment isa employment;

(Fiona no longer employed by Two)

demo> commit

demo> match $emp isa employment; get;

demo> insert $puc isa publicCompany, has ABN "12345678903", has name "Three", has numberOfShares 3000;

demo> commit

demo> match $co isa company, has name $n; get;

demo> match $co isa company, has name "Two"; $nco isa company, has name "Three"; $oldEmployment (employer: $co, employee: $p) isa employment; insert $oldEmployment (employer: $nco);

(Extending a relation with a new role player 🡪 <https://dev.grakn.ai/docs/query/updating-data#extending-a-relation-with-a-new-role-player>)

(Emma and Charles employed by Two are now also employed by Three)

demo> commit

demo> match $emp isa employment; get;

demo> match $co isa company, has name "Two"; $oldEmployment (employer: $co, employee: $p) isa employment; delete $oldEmployment (employer: $co);

(Delete Role Players from Relations 🡪 <https://dev.grakn.ai/docs/query/delete-query#delete-role-players-from-relations>)

(Modifying a relation’s role player 🡪 <https://dev.grakn.ai/docs/query/updating-data#modifying-a-relations-role-player>)

(Emma and Charles now don’t have Two as their employer, but still have Three has their employer)

demo> commit

demo> match $emp isa employment; get;

demo> match $co isa company, has name $n, has ABN $a; get; sort $n asc;

demo> match $co isa $company, has ABN $a; {$a contains "903"; }; get;

demo> match $co isa $company, has ABN $a; {$a contains "903"; }; insert $co has ABN "12345678904";

demo> match $co isa $company, has ABN $a; {$a contains "903"; }; delete $a isa ABN;

(Update all instances of a given attribute 🡪 <https://dev.grakn.ai/docs/query/updating-data#update-all-instances-of-a-given-attribute>)

(first insert new ABN, then delete old ABN)

demo> commit

demo> match $co isa company, has name $n, has ABN $a; get; sort $n asc;

demo> exit

docker exec -ti grakn bash -c "/grakn-core-all-linux/grakn server schema demo > /demo6Schema.gql"

docker exec -ti grakn bash -c "/grakn-core-all-linux/grakn server export demo /demo6Data.grakn"

**7: continue or start with demo6Schema.gql and demo6Data.grakn**

demo> match $co isa company, has name $n, has ABN $a; get; sort $n asc;

… to be continued.