Università degli studi di Milano-Bicocca

Big Data in Public and Social Services

GRAPH4DS

A GRAPH DB FOR DATA SCIENCE

Autori:
Beatrice Fumagalli - 784549 - b.fumagalli9@campus.unimib.it
Stefano Rola - 790383 - s.rola@campus.unimib.it

1. Introduzione

Questo *report* presenta **GRAPH4DS**, un sistema che modella il corso di Laurea magistrale in Data Science dell'Università degli Studi di Milano Bicocca come *Graph Database* per l'esecuzione di *graph-based queries* e *social network analyses*. Lo scopo è quello di fornire uno strumento utile alla didattica sia per i docenti che per gli studenti.
Partendo dal *syllabus* del CdL, sono state create le tabelle con estensione .csv da caricare successivamente in Neo4J per la creazione del GraphDB. GRAPH4DS arricchisce inoltre i dati del CdL attraverso le somiglianze tra diversi corsi di studio calcolate attraverso l'indice di *Jaccard*.

2. Informazioni disponibili

Le informazioni disponibili per la creazione del Graph Database sono contenute nel Syllabus e possono essere così sintetizzate:

- Corsi del primo e secondo anno con rispettivi CFU ed eventuale suddivisione in moduli
- 10 tipi di classificazione
- Specifiche di ciascun tipo di classificazione
- Specifiche di ciascun corso/modulo in riferimento alle specifiche dei tipi di classificazione
- Cognome dei docenti di ciascun corso/modulo

Le informazioni che sono state aggiunte a quest'ultime sono nome e e-mail dei docenti di ciascun corso/modulo. Le informazioni mancanti sono tuttavia parecchie. Infatti, molti corsi non presentano le specifiche ed altri invece sono stati compilati in maniera poco accurata.
Si segnala che non presentano la compilazione delle specifiche i seguenti corsi:

- Juridical Issues
- Social Issues
- Digital Economy
- Big data in Economics
- Industry Lab
- Time series analysis
- Decision models

3. Progettazione e implementazione di GRAPH4DS

3.1 Data Model

GRAPH4DS è un'applicazione che consente agli utenti di eseguire *query* basate su grafici sui dati del CdL preso in esame. In fig. 1 è rappresentato il modello di dati che è stato pensato per il sistema, successivamente progettato ed infine implementato con Neo4J. Si parte dai corsi di studio, dove ciascun corso è un nodo e ciascun corso può o non può essere formato da due esami, i cosiddetti moduli. Pertanto la relazione ***splitted_in*** lega il nodo del **corso** ai corrispettivi nodi dei **moduli**. A ciascun modulo o corso, nel caso il corso non sia formato da moduli, è collegato, attraverso la relazione ***taught_by***, un nodo **Prof** che contiene nome, cognome e e-mail del docente che insegna tale materia. Successivamente sono stati implementati i nodi delle specifiche dei corsi come segue. Si parte dal nodo di ciascuna **categoria** principale di ciascun tipo di classificazione, aventi come *label* il tipo di classificazione (*Classification_type*), collegati ai rispettivi moduli/corsi dalla relazione ***is_about***. Se una categoria contiene ulteriori specificazioni, queste sono contenute in un ulteriore nodo avente come *label* il medesimo del nodo categoria più un *label* per la **categoria** di riferimento

e collegati ai primi dalla relazione ***focused_on***. Le specifiche arrivano fino ad un terzo livello, chiamato ***type,*** avente come *label* il tipo di classificazione, la categoria e la sottocategoria e collegato a quest'ultima dalla relazione ***type***. Per concludere è stata implementata la similarità tra corsi come verrà spiegato successivamente nel *report*.

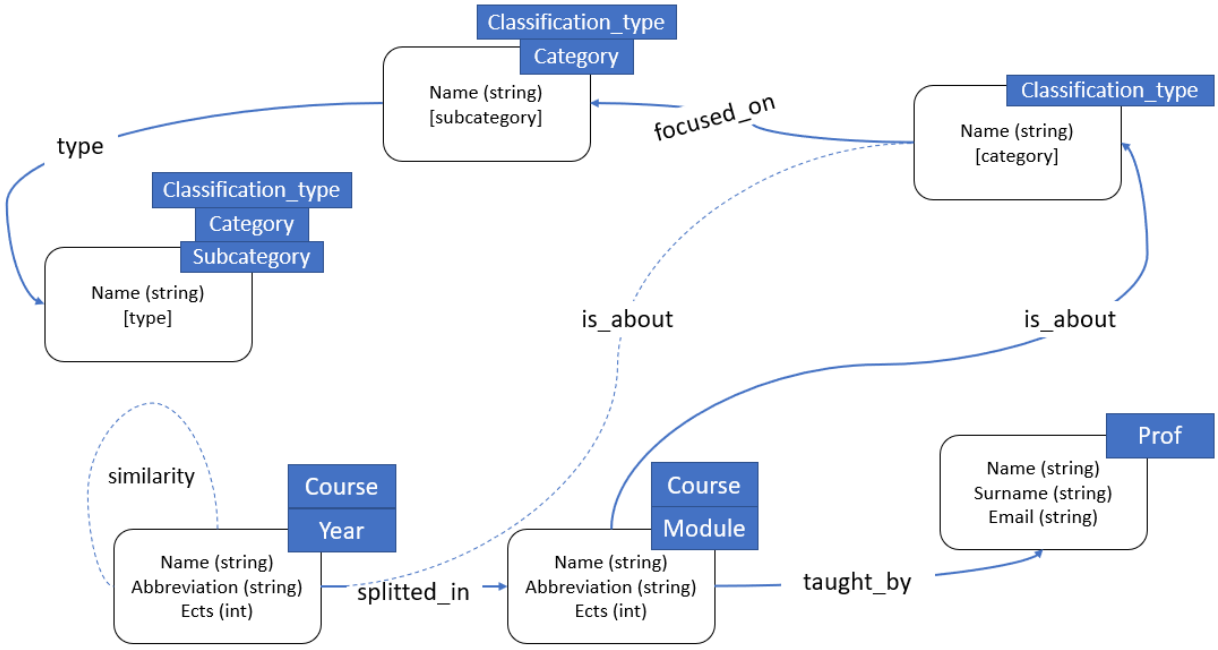


Figura 1 - GRAPH4DS Data Model

3.2 Creazione tabelle .csv

Partendo dal Syllabus del corso di Laurea si è iniziato costruendo i file csv necessari per la costruzione del Graph DB in Neo4J. È stata creata una tabella per ciascuno dei nodi di partenza e le corrispettive tabelle delle relazioni, attraverso cui sono stati creati anche i nodi finali in Neo4J.

CSV NODI

- **course.csv**

Name	Nome del corso
Abbreviation	Abbreviazione del nome del corso
Ects	Corrispondono ai CFU
Year	Anno del CdL in cui viene erogato il corso (primo o secondo anno)

- **module.csv**

Name	Nome del modulo
Abbreviation	Abbreviazione del nome del modulo
Ects	Corrispondono ai CFU

- **prof.csv**

Name	Nome del docente
Surname	Cognome del docente

Email	e-mail universitaria del docente

- **application_domain.csv**

Name	Nome del dominio dell'applicazione di primo livello

- **data_source.csv**

Name	Nome della sorgente dati

- **data_type.csv**

Name	Nome della tipologia dati

- **life_cycle.csv**

Name	Nome della sorgente dati

- **language_and_framework.csv**

Name	Nome dei linguaggi e framework

- **model.csv**

Name	Nome dei modelli

- **technique.csv**

Name	Nome delle tecniche

- **technology.csv**

Name	Nome delle tecnologie

- **vi.csv**

Name	Nome delle 5 V dei Big Data

CSV ARCHI

- **splitted_in.csv**

from	to
course	module

- **taught_by.csv**

From	To
Course	Prof

- **category.csv**

From	To
Course/module	Category (tutte)

- **ad_focused_on.csv**

From	To
Ad_category	ad_subcategory

- **epid_type.csv**

From	To
Epidemiology	type_epidemiology

- **dt_focused_on.csv**

From	To
dt_category	dt_subcategory

- **lc_focused_on.csv**

From	To
dt_category	dt_subcategory

- **qual_assess_type.csv**

From	To
Quality assessment	type_quality_assessment

- **lf_focused_on.csv**

From	To
Query languages	ql_subcategory

- **mo_focused_on.csv**

From	To
mo_category	mo_subcategory

- **mo_type.csv**

From	To
Probabilistic	prob_subcategory

- **te_focused_on.csv**

From	To
te_category	te_subcategory

3.3 Cypher e Neo4J

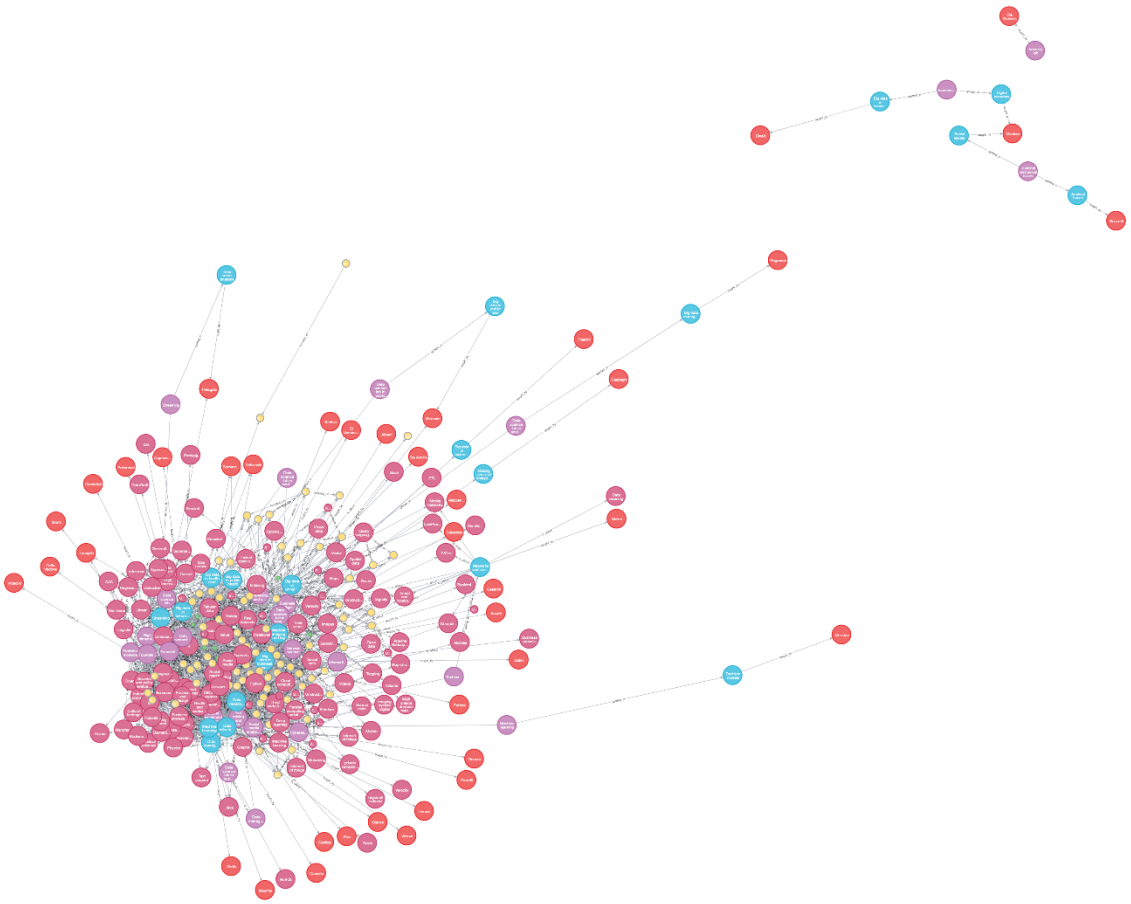
GRAPH4DS è implementato su un sistema di gestione di database grafico open source, **Neo4j** e l'intero codice sorgente per ottenere il set di dati viene reso disponibile in questo *report* in modo che chiunque possa contribuire al miglioramento dell'istanza del grafico del corso di Laurea in Data Science. Una volta create tutte le tabelle necessarie alla costruzione del grafo, queste sono state

caricate in Neo4J per permettere la creazione dei nodi e delle relazioni, come spiegato precedentemente nel modello di dati.
Si specifica che Neo4j è composto da quattro blocchi elementari:

- I **nodi** sono simili alle tuple nei database relazionali comuni. A differenza dei *database* relazionali, i nodi non hanno una struttura fissa ma possono variare di lunghezza contenendo diversi tipi di dati.
- Le **relazioni** possono esistere tra i nodi. Esse hanno sempre un nodo iniziale e uno finale.
- Le **proprietà** sono coppie chiave-valore che possono essere incluse in qualsiasi nodo o relazione.
- Le **etichette** associano un nome comune a un insieme di nodi o relazioni con l'obiettivo di indicizzazione e recupero rapidi. A differenza delle proprietà, le etichette non hanno valori. Un nodo o relazione può avere più di un'etichetta alla volta.

Neo4j fa uso del linguaggio dichiarativo ***Cypher*** e in appendice viene riportato il codice per la creazione del GraphDB che risulta avere 306 nodi e 1781 relazioni, con il seguente numero di nodi per ciascun *label* del *Data Model* iniziale:

Course	26
Module	22
Prof	40
Category	121
Subcategory	90
Type	7



3.4 Similarità tra corsi

La relazione SIMILARITY stima la somiglianza tra due corsi accademici sulla base delle loro specifiche. Nello specifico, ciascun corso accademico presenta delle caratteristiche specifiche che lo distinguono ed accomunano ad altri corsi. In questo modo si può calcolare la somiglianza tra due corsi c_1 e c_2 basata sulle caratteristiche che hanno in comune. Si è scelto di utilizzare l'indice di Jaccard calcolato sui corsi.
Sia S_c l'insieme delle specifiche trattate dal corso c, l'indice di Jaccard tra due corsi c_i e c_j è definito come segue:

$$J_{c_i,c_j} = \frac{\left|S_{c_i} \cap S_{c_j}\right|}{\left|S_{c_i} \cup S_{c_j}\right|}$$

In Neo4J viene calcolata utilizzando la funzione *similarity.jaccard()* che si trova all'interno della libreria Graph Algorithms.

4. Calcolo della similarità tra corsi

4.1 Similarity tra due corsi

```
MATCH (c1:Course {name: 'Statistical modeling'})-[:is_about]->(s1)
WITH c1, collect(id(s1)) AS c1subject
MATCH (c2:Course {name: "Foundations of probability and statistics"})-[:is_about]->(s2)
WITH c1, c1subject, c2, collect(id(s2)) AS c2subject
RETURN c1.name AS From, c2.name AS To, round(10000 * algo.similarity.jaccard(c1subject,
        c2subject)) / 10000  AS `Jaccard Similarity`;
```

From	To	Jaccard Similarity
"Machine learning"	"Data management"	0.5862

4.2 Similarity tra tutti i corsi a coppie, in ordine decrescente (primi 10 risultati)

```
MATCH (c:Course)-[:is_about]->(s)
WITH {item:id(c), categories: collect(id(s))} as userData
WITH collect(userData) AS data
CALL algo.similarity.jaccard.stream(data)
YIELD item1, item2, count1, count2, intersection, similarity
RETURN algo.asNode(item1).name AS From, algo.asNode(item2).name AS To, intersection AS
Intersection, round(10000 * similarity) / 10000 AS `Jaccard Similarity`
ORDER BY `Jaccard Similarity` DESC
LIMIT 10
;
```

From	To	Intersection	Jaccard Similarity
"Foundations of probability and statistics"	"Statistical modeling"	63	0.9844
"Statistical modeling"	"High dimensional data analysis"	62	0.8378
"Foundations of probability and statistics"	"High dimensional data analysis"	61	0.8243
"Data science lab"	"Streaming data management"	37	0.8222
"Big data in health care"	"Big data in public health"	23	0.697
"Foundations of computer science"	"Data management"	52	0.65
"Foundations of computer science"	"Machine learning"	54	0.6136
"Data management"	"Machine learning"	51	0.5862
"Text mining and search"	"Machine learning"	68	0.5574
"Data semantics"	"Text mining and search"	78	0.5571

4.3 Corso più simile a uno dato (Machine Learning)

MATCH (c:Course)-[:is_about]->(s)
WITH {item:id(c), categories: collect(id(s))} **AS** userData
WITH collect(userData) **AS** data
CALL algo.similarity.jaccard.stream(data)
YIELD item1, item2, count1, count2, intersection, similarity
WHERE algo.asNode(item1).name = 'Machine learning' **OR** algo.asNode(item2).name = 'Machine learning'
RETURN algo.asNode(item1).name **AS** From, algo.asNode(item2).name **AS** To, intersection **AS** Intersection, round(10000 * similarity) / 10000 **AS** `Jaccard Similarity`
ORDER BY `Jaccard Similarity` **DESC**
LIMIT 1
;

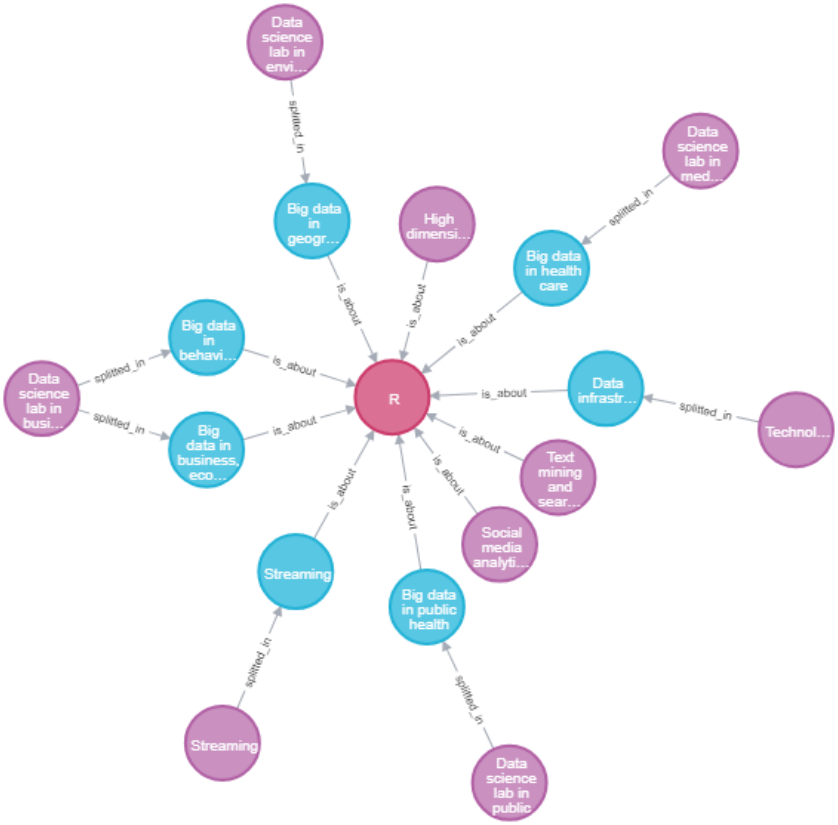
From	To	Intersection	Jaccard Similarity
"Foundations of computer science"	"Machine learning"	54	0.61

5. Esplorazione di GRAPH4DS

5.1 Prima Query

Quali sono i corsi del secondo anno di Laurea magistrale in Data Science che utilizzano come linguaggio di programmazione R?

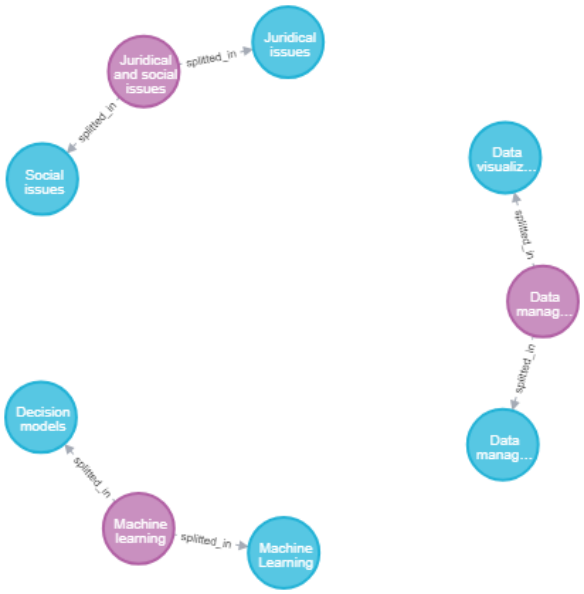
MATCH p=(n:Second)-[r*0..2]-(m:`Language and framework` {name:'R'}) RETURN p



5.2 Seconda Query

Quali sono i corsi del primo anno di Laurea magistrale in Data Science che sono suddivisi in moduli? Indicare corsi e rispettivi moduli

MATCH p=(n:First)-[r:splitted_in]-(m) RETURN p



5.3 Terza Query

Quali sono i corsi inerenti all'ambito medico?

MATCH p=(:Course)--(:`Application domain` {name:'Health and medicine'}) RETURN p

5.4 Quarta Query

Quali sono i corsi che utilizzano come tipologia di dati time series o streaming data?

MATCH p=(:Course)-[*0..2]-(m:`Data type`)
WHERE m.name = 'Time series' OR m.name = 'Streaming data'
RETURN p

Appendice

Codice Cypher

- COURSE AND MODULES

```
USING PERIODIC COMMIT LOAD CSV WITH HEADERS FROM "file:///Graph4DS/course.csv" as row FIELDTERMINATOR ';'
CALL apoc.create.node(['Course',row.year], {name:row.name, abbreviation:row.abbreviation, ects:row.ects}) YIELD node
RETURN node
;

USING PERIODIC COMMIT LOAD CSV WITH HEADERS FROM "file:///Graph4DS/module.csv" as row FIELDTERMINATOR ';'
CREATE (m:Course:Module {name:row.name, ects:row.ects})
RETURN m
;

USING PERIODIC COMMIT LOAD CSV WITH HEADERS FROM "file:///Graph4DS/splitted_in.csv" as row FIELDTERMINATOR ';'
MATCH (c:Course {name:row.from})
MATCH (m:Module {name:row.to})
CREATE (c)-[:splitted_in]->(m)
;
```

- PROF

```
USING PERIODIC COMMIT LOAD CSV WITH HEADERS FROM "file:///Graph4DS/prof.csv" as row FIELDTERMINATOR ';'
CREATE (p:Prof {surname:row.surname, name:row.name, email:row.email})
RETURN p
;

USING PERIODIC COMMIT LOAD CSV WITH HEADERS FROM "file:///Graph4DS/taught_by.csv" as row FIELDTERMINATOR ';'
MATCH (c:Course {name:row.from})
MERGE (m:Prof {surname:row.to})
MERGE (c)-[:taught_by]->(m)
;

USING PERIODIC COMMIT LOAD CSV WITH HEADERS FROM "file:///Graph4DS/taught_by.csv" as row FIELDTERMINATOR ';'
MATCH (c:Module {name:row.from})
MERGE (m:Prof {surname:row.to})
MERGE (c)-[:taught_by]->(m)
;
```

- APPLICATION DOMAINS

```
USING PERIODIC COMMIT LOAD CSV WITH HEADERS FROM
"file:///Graph4DS/AD/application_domain.csv" as row FIELDTERMINATOR ';'
CREATE (a:`Application domain` {name:row.name})
RETURN a
;

USING PERIODIC COMMIT LOAD CSV WITH HEADERS FROM
"file:///Graph4DS/AD/ad_focused_on.csv" as row FIELDTERMINATOR ';'
MERGE (c:`Application domain` {name:row.from}) WITH c,row
CALL apoc.create.node(['Application domain',row.from], {name:row.to}) YIELD node
MERGE (c)-[:focused_on]->(node)
;

USING PERIODIC COMMIT LOAD CSV WITH HEADERS FROM
"file:///Graph4DS/AD/epid_type.csv" as row FIELDTERMINATOR ';'
MERGE (c:`Application domain` {name:row.from}) WITH c,row
CALL apoc.create.node(['Application domain','Health and medicine',row.from], {name:row.to})
YIELD node
MERGE (c)-[:type]->(node)
;
```

- DATA SOURCES

```
USING PERIODIC COMMIT LOAD CSV WITH HEADERS FROM
"file:///Graph4DS/DS/data_source.csv" as row FIELDTERMINATOR ';'
CREATE (a:`Data source` {name:row.name})
RETURN a
;
```

- DATA TYPES

```
USING PERIODIC COMMIT LOAD CSV WITH HEADERS FROM
"file:///Graph4DS/DT/data_type.csv" as row FIELDTERMINATOR ';'
CREATE (a:`Data type` {name:row.name})
RETURN a
;

USING PERIODIC COMMIT LOAD CSV WITH HEADERS FROM
"file:///Graph4DS/DT/dt_focused_on.csv" as row FIELDTERMINATOR ';'
MERGE (c:`Data type` {name:row.from}) WITH c,row
CALL apoc.create.node(['Data type',row.from], {name:row.to}) YIELD node
MERGE (c)-[:focused_on]->(node)
;
```

- LIFE CYCLES

```
USING PERIODIC COMMIT LOAD CSV WITH HEADERS FROM "file:///Graph4DS/LC/life_cycle.csv"
as row FIELDTERMINATOR ';'
CREATE (a:`Life cycle` {name:row.name})
RETURN a
;

USING PERIODIC COMMIT LOAD CSV WITH HEADERS FROM
"file:///Graph4DS/LC/lc_focused_on.csv" as row FIELDTERMINATOR ';'
MERGE (c:`Life cycle` {name:row.from}) WITH c,row
CALL apoc.create.node(['Life cycle',row.from], {name:row.to}) YIELD node
MERGE (c)-[:focused_on]->(node)
;

USING PERIODIC COMMIT LOAD CSV WITH HEADERS FROM
"file:///Graph4DS/LC/qual_assess.csv" as row FIELDTERMINATOR ';'
MERGE (c:`Life cycle` {name:row.from}) WITH c,row
CALL apoc.create.node(['Life cycle','Information system',row.from], {name:row.to}) YIELD node
MERGE (c)-[:type]->(node)
;
```

- LANGUAGES AND FRAMEWORKS

```
USING PERIODIC COMMIT LOAD CSV WITH HEADERS FROM
"file:///Graph4DS/LF/language_and_framework.csv" as row FIELDTERMINATOR ';'
CREATE (a:`Language and framework` {name:row.name})
RETURN a
;

USING PERIODIC COMMIT LOAD CSV WITH HEADERS FROM
"file:///Graph4DS/LF/lf_focused_on.csv" as row FIELDTERMINATOR ';'
MERGE (c:`Language and framework` {name:row.from}) WITH c,row
CALL apoc.create.node(['Language and framework',row.from], {name:row.to}) YIELD node
MERGE (c)-[:focused_on]->(node)
;
```

- MODELS

```
USING PERIODIC COMMIT LOAD CSV WITH HEADERS FROM "file:///Graph4DS/MO/model.csv"
as row FIELDTERMINATOR ';'
CREATE (a:Model {name:row.name})
RETURN a
;

USING PERIODIC COMMIT LOAD CSV WITH HEADERS FROM
"file:///Graph4DS/MO/mo_focused_on.csv" as row FIELDTERMINATOR ';'
MERGE (c:Model {name:row.from}) WITH c,row
CALL apoc.create.node(['Model',row.from], {name:row.to}) YIELD node
MERGE (c)-[:focused_on]->(node)
```

;

USING PERIODIC COMMIT LOAD CSV WITH HEADERS FROM
"file:///Graph4DS/MO/mo_type.csv" as row FIELDTERMINATOR ';'
MERGE (c:Model {name:row.from}) **WITH** c,row
CALL apoc.create.node(['Model','Information',row.from], {name:row.to}) **YIELD** node
MERGE (c)-[:type]->(node)
;

- TECHNIQUES

USING PERIODIC COMMIT LOAD CSV WITH HEADERS FROM
"file:///Graph4DS/TECH/technique.csv" as row FIELDTERMINATOR ';'
CREATE (a:Technique {name:row.name})
RETURN a
;

USING PERIODIC COMMIT LOAD CSV WITH HEADERS FROM
"file:///Graph4DS/TECH/te_focused_on.csv" as row FIELDTERMINATOR ';'
MERGE (c:Technique {name:row.from}) **WITH** c,row
CALL apoc.create.node(['Technique',row.from], {name:row.to}) **YIELD** node
MERGE (c)-[:focused_on]->(node)
;

- TECNOLOGIES

USING PERIODIC COMMIT LOAD CSV WITH HEADERS FROM
"file:///Graph4DS/TECN/technology.csv" as row FIELDTERMINATOR ';'
CREATE (a:Technology {name:row.name})
RETURN a
;

- VEES

USING PERIODIC COMMIT LOAD CSV WITH HEADERS FROM "file:///Graph4DS/VI/vi.csv" as row FIELDTERMINATOR ';'
CREATE (a:Vi {name:row.name})
RETURN a
;

- COURSE/MODULE-> CATEGORY

USING PERIODIC COMMIT LOAD CSV WITH HEADERS FROM "file:///Graph4DS/category.csv" as row FIELDTERMINATOR ';'
MATCH (n {name:row.from})
MATCH (m {name:row.to})
MERGE (n)-[:is_about]->(m)
;

```
LOAD CSV WITH HEADERS FROM "file:///Graph4DS/category.csv" as row FIELDTERMINATOR ';'
WITH row
MATCH (n)
WHERE row.to in labels(n)
WITH row.from as f, row.to as t, n
MATCH (a:Course {name:f})
MERGE (a)-[:is_about]->(n)
;
```