Ontologias para deteção de fraude em meio académico

Tiago Filipe Fernandes dos Santos - a28248

Dissertação apresentada à Escola Superior de Tecnologia e de Gestão de Bragança para obtenção do Grau de Mestre em Informática.

Trabalho orientado por:

Prof. Paulo Jorge Teixeira Matos

Esta dissertação não inclui as críticas e sugestões feitas pelo Júri.

Bragança

2020-2021

Ontologias para deteção de fraude em meio académico

Tiago Filipe Fernandes dos Santos - a28248

Dissertação apresentada à Escola Superior de Tecnologia e de Gestão de Bragança para obtenção do Grau de Mestre em Informática.

Trabalho orientado por:

Prof. Paulo Jorge Teixeira Matos

Esta dissertação não inclui as críticas e sugestões feitas pelo Júri.

Bragança

2020-2021

Dedicatória

Gostaria de dedicar a conclusão deste trabalho ao meu pai, Felisberto dos Santos, à minha mãe, Ana Maria da Silva Fernandes e aos meus irmãos (Mário Rui Fernandes dos Santos, Humberto José Fernandes dos Santos e Liliana de Fátima Fernandes dos Santos), à minha namorada (Cristina), às minhas cunhadas (Marta e Karla), aos meus sobrinhos, aos meus amigos (Ivan Malhão, Fábio André, Diogo Santos, Daniel Campos, Admilson Cruz) e ao meu gatinho (Rodriguinho), também ele importante para mim.

Agradecimentos

Agradeço ao Professor Paulo Jorge Teixeira Matos pela sua orientação e a todos os professores que me permitiram adquirir conhecimentos ao longo deste ciclo académico.

Agradeço também aos meus pais, irmãos, namorada e amigos, que sempre estiveram do meu lado.

Resumo

Este trabalho tem como objetivo principal o desenvolvimento de ontologias para a deteção de fraude académica e a aplicação de métodos quantitativos indicadores do desenvolvimento de um modelo que caminhe no sentido correto para a resolução do problema.

Inicialmente e pelo facto de se tratar de um tema pouco explorado, é feita uma extensiva introdução conceptual e histórica, passando pelos principais conceitos da Web Semântica, por forma a fazer uma interiorização (pessoal e também ao leitor) das bases necessárias para desenvolver e compreender o trabalho desenvolvido, respetivamente. A abordagem teórica passa por um estado da arte, onde são discutidos trabalhos e conceitos relacionados, seguindo depois para a explicação do funcionamento geral da própria Web Semântica, com grande foco em "graph databases" e nos seus conceitos.

É construído um modelo inicial, uma "graph database" populada manualmente, fazendo uso da ferramenta "Neo4j" e depois ajustado conforme as necessidades do problema.

Por fim são aplicados algoritmos sobre esse modelo, obtendo resultados objetivos e que estão intrínsecamente ligados aos objetivos propostos inicialmente, cumprindo-os de uma forma geral, dado que foram encontrados os elementos com mais influência relativamente a outros elementos do sistema, sendo um princípio fundamental constituinte de fraude académica.

Palavras-chave: ontologia, Neo4j, Fraude, Web Semântica.

Abstract

The main objective of this work is the development of ontologies for the detection of academic fraud and the application of quantitative methods that are indicators of the development of a model that moves in the right direction to solve the problem.

Initially, and due to the fact that it is a little explored theme, an extensive conceptual and historical introduction is made, passing through the main concepts of the Semantic Web, in order to make an internalization (personal and also to the reader) of the necessary bases to develop and understand the work developed, respectively. The theoretical approach goes through a state of the art, where works and related concepts are discussed, followed by an explanation of the general functioning of the Semantic Web itself, with great focus on "graph databases" and its concepts.

An initial model is built, a manually populated "graph database", using the "Neo4j" tool and then adjusted according to the needs of the problem.

Finally, algorithms are applied on this model, obtaining objective results that are intrinsically linked to the initially proposed objectives, fulfilling them in a general way, since the elements with more influence over other elements of the system were found, being a fundamental principle constituent of academic fraud.

Keywords: ontology, Neo4j, academic fraud, Semantic Web.

Conteúdo

Lista de Tabelas

Lista de Figuras

Siglas

ACID Atomicity, Consistency, Isolation, and Durability. 28, 44, 45

API Application Programming Interface. 44, 46

APOC Awesome Procedures on Cypher. 49, 50

AQL ArangoDB Query Language. 44

BD Base de Dados. 24, 25, 28, 30, 48, 66, 70

CAFA Centro Anti-Fraude Académica. 2

CQL Cypher Query Language. 47

CRUD Create, Read, Update and Delete. 28, 43, 47

DBMS Data Base Management System. 43, 46, 47

ESTIG Escola Superior de Tecnologia e Gestão. 3

EUA Estados Unidos da América. 15

GB gigabytes. 15

GBAD Graph Based Anomaly Detection. 20–22

GDB Graph Database. x, xii, 6, 24–26, 28–31, 43–45, 47–49, 63

GDBE Graph Database Engine. 27

GDS Graph Data Science. 50, 63

GE Graph Engine. 45

GKG Google Knowledge Graph. 15, 17

GPL3 GNU General Public License v3.0. 44

GSQL TygerGraph Query Language. 46

KG Knowledge Graph. 17–19, 26, 45

LOD Linking Open Data. 19

MI Mestrado de Informática. 3

NDB Network Database. 25

NoSQL Not Only SQL. 24, 25, 29, 44, 47

OWL Ontology Web Language. 40, 86

PB petabytes. 15

PHP PHP: Hypertext Preprocessor. 45

RDF Resource Description Framework. 23, 29, 40, 46

RDFS Resource Description Framework Schema. 14

SGBD Sistema de Gestão de Bases de Dados. 43, 44

SHOE Simple HTML Ontology Extensions. 40

SPARQL SPARQL Protocol and RDF Query Language. 45, 46

SQL Structured Query Language. 44

SVM Support Vector Machine. 21

UC Unidade Curricular. 3, 60

WS Web Semântica. 3, 6, 23, 24

Capítulo 1

Introdução

O conceito de fraude é, nos tempos que correm, um conceito com o qual todos nós enquanto seres humanos estamos familiarizados, direta ou indiretamente. Ou já fomos vítimas de um qualquer tipo de trapaça ou já nós próprios sofremos na pele as piores consequências desse mesmo conceito.

Ao longo dos anos mais recentes, o Mundo evoluiu para um patamar no qual existem muitos benefícios do uso das tecnologias, quer a nível dos conhecimentos ganhos, quer a nível da informação disponível em qualquer lugar, a qualquer momento, de fácil acesso. Embora este facto acarrete muitos mais benefícios do que malefícios, existem sempre desvantagens. Uma das principais desvantagens foi o crescimento do número de possibilidades que pessoas mal-intencionadas possuem para conseguir enganar os outros, estando as fraudes diretamente ligadas a este malefício.

Logicamente, o referido benefício do levantamento e transmissão fácil de informação pode estar intrinsecamente ligado a fraudes. Os intervenientes que desejam contornar de alguma forma o sistema podem facilmente encontrar formas distintas de o fazer, à distância de um clique. Com o surgimento de plataformas de índole comunitário, como redes sociais, hoje em dia existe um chamariz para muitas possibilidades de esquemas de fraude.

Uma das áreas onde a fraude talvez tenha crescido [1] nos últimos tempos foi a academia. Devido ao desenvolvimento das tecnologias, o número de recursos disponíveis

aumentou. Por exemplo, de momento, é mais fácil as cábulas serem de tamanhos reduzidos. Quando impressas, as folhas ocupam menos espaço do que nos casos em que são escritas à mão.

Um outro exemplo de tecnologia que facilita e agiliza resoluções de exercícios de avaliação ilegítimos trata-se dos smartwatches. É perfeitamente possível um aluno tirar partido dos mesmos para verificar soluções de problemas para os quais não tenha adquirido conhecimentos através do seu estudo para dar uma resposta elaborada por si.

Um terceiro caso, de entre muitos, onde pode ocorrer fraude é através da tecnologia e de redes de conhecimento interpessoais estabelecidas pelo aluno, ele obter através de outra pessoa, fora da sala, as respostas. Sejam grupos no "Facebook" ou pessoas a dizerem as respostas através de auscultadores, o facto é que uma vasta gama de utensílios podem ser utilizados e assim, redes complexas de alunos que copiam entre si podem ser formadas.

Com estas possibilidades para a ocorrência de actos ilícitos ao nivel escolar, em todos os ciclos do ensino, existem respostas que têm de começar a ser encontradas, de forma a não fazer do sistema de ensino algo que não se rege pela meritocracia, mas sim pela capacidade social de encontrar os contactos certos e pela agilidade mental de conseguir arranjar formas diferentes para copiar.

Tal como descrito em [1], uma das possibilidades que nos poderia ajudar a encontrar respostas para travar estas redes fraudulentas ao nível das academias, em termos nacionais, seria um Centro Anti-Fraude Académica (CAFA) em cada instituição de ensino. Embora seja uma ideia com valor, seria necessário garantir a idoneidade dos membros integrantes da comissão de gestão do centro.

Outra forma de mitigar a ocorrência de fraude pedagógica, resultando em contenção e prevenção da mesma à base do risco/ benefício dos seus praticantes, que ao serem descobertos rapidamente poderiam deixar de a praticar, seria a utilização da tecnologia a favor de algo positivo como a deteção de irregularidades. Por outras palavras, o uso de software de deteção de plágio poderá ser uma ferramenta eficiente na prevenção da fraude académica, embora não seja o foco deste trabalho.

Um bom exemplo disso mesmo seria a procura de frases exactas num motor de pesquisa

como o "Google". Esta exatidão nas frases escritas (ou transcritas) trata-se de um bom exemplo de padrão de plágio.

Este tipo de padrão, aliado à tecnologia, leva-nos à abertura de novas portas ao nível de conseguirmos combater todo o tipo de fraude, mais concretamente no tema que este trabalho de Dissertação aborda: a fraude académica. Isto abre-nos portas à introdução do conceito de ontologias.

Ontologias são "contextualizações específicas de um domínio" [2] que nos permitem estabelecer relações que à primeira vista não são evidentes, mas que a longo prazo e em comunidades como redes sociais ou, neste caso, contactos estabelecidos entre diferentes alunos de uma escola, nos podem fazer perceber padrões comportamentais que servenm como base de sustentação para a deteção de comunidades dentro da própria comunidade escolar, ou mesmo comportamentos individuais potenciados pelos mesmos presentes numa comunidade suspeita, através de inferência, de praticar fraude académica.

É neste contexto que se insere este trabalho: procurar padrões, através da tecnologia, para deteção de comunidades, ao nível escolar, que possam estar a adulterar a meritocracia em termos de aprovação às unidades curriculares, praticando fraude.

1.1 Enquadramento

Este projeto enquadra-se na Unidade Curricular (UC) de Dissertação, pertencente ao 2º ano do Mestrado de Informática (MI) da Escola Superior de Tecnologia e Gestão (ESTIG).

Surge num contexto de exploração de novas vertentes tecnológicas e foi proposto pelo Professor Paulo Matos, cuja função é de orientador no contexto deste trabalho. Está relacionado com a UC de Web Semântica (WS), uma disciplina inovadora no contexto da instituição e cujo lecionamento se iniciou no presente ano letivo.

Os conceitos englobados nesta área de estudo estão em fase de desenvolvimento e podem ser enquadrados em inúmeros problemas do Mundo real, com grande incidência naqueles que envolvem inferências sobre padrões comportamentais.

Foi neste contexto que o aluno de MI (Tiago Santos) e o coordenador do projeto

(Professor Paulo Matos) chegaram à conclusão de que seria um caminho plausível a escolha de enquadrar esta área de estudo com o contexto académico.

1.2 Objetivos

Este trabalho tem como objetivo a análise da viabilidade e adequação do uso de ontologias, bases de dados orientadas a grafos e outros componentes (metodologias e ferramentas) no âmbito da WS, na deteção de diversas dimensões que possam estar relacionadas com a existência de fraude em contexto académico.

1. **Estudo de bases de dados orientadas a grafos** - Será procurada a aquisição de sapiência sobre o conceito de "graph database" e todos os seus domínios de intervenção, como por exemplo o facto de este tipo de conjuntos de dados estruturados poderem ser utilizados para definir comunidades, metadados com características (atributos e relacões) comuns a todo o domínio, entre outros. Este tipo de bases de dados é compatível com métodos estatísticos, cujo propósito é incidir sobre as suas estruturas.

2. **Estudo de ontologias** - Esta etapa consiste na aquisição de conhecimentos sobre a principal ferramenta no que toca a detetar possíveis padrões comportamentais de indivíduos, esperando assim compreender melhor a construção de ontologias passíveis de serem validadas por dedutores classificativos.

3. **Análise do domínio do problema** - Estudo sobre possíveis problemas semelhantes que nos possam enquadrar melhor naquele no qual incide este projeto, resultando na aprendizagem de métodos e componentes que eventualmente possam levar na consolidação da estrutura de conhecimento a ser definida neste trabalho. Esta fase também terá como objetivo a obtenção de uma descrição detalhada e alargada dos procedimentos e metodologias de avaliação, na perspectiva dos alunos e dos docentes.

4. **Construção dos modelos de representação do conhecimento** - Esta fase assenta no estabelecimento de critérios que possam permitir a representação de uma base objetiva, através da definição das estruturas de conhecimento.

   (a) **Formulação de base de dados** - Utilizando a tecnologia Neo4j, será construída uma base de dados orientada a grafos. Ela, numa fase inicial, terá a lógica por detrás do problema, assentando-se esta em metadados. Numa fase posterior, será populada, sendo sujeita a métodos estatísticos que poderão demonstrar alguns resultados pretendidos.

   (b) **Construção de ontologias** - Esta etapa consiste na definição de ontologias sujeitas classificadores dedutivos, que têm como função a validação da consistência do modelo definido.

5. **Validação das soluções obtidas** - Consiste na validação dos resultados que podem ser obtidos num contexto próximo ao real, sendo que o objetivo será corroborar as soluções obtidas, por forma a haver material para produção científica.

1.3 Estrutura do Documento

A presente tese encontra-se organizada em seis capítulos: "Introdução", "Estado da Arte", "Web Semântica", "Tecnologias e Ambiente de Desenvolvimento", "Implementação" e "Conclusões e Trabalho Futuro".

No 1º capítulo, está presente uma introdução que visa dar a conhecer o projeto e a tese que nele assenta, bem como o enquadramento, objetivos e estruturação do presente documento.

No 2º capítulo, denominado "Estado da Arte", está presente o estado atual da área da Web Semântica, nomeadamente naquilo em que este projeto é enquadrado, apresentando uma descrição detalhada de ontologias formuladas noutros contextos, que sejam relevantes no contexto atual e nomeadamente no escopo do atual projeto. Estão também

presentes conceitos de outras áreas umbilicalmente ligadas à WS e que podem, inclusive, ser interligadas com esta.

No 3º capítulo, "Web Semântica", está presente uma explicação detalhada do surgimento e utilidades desta área da informática, que contextos podem ser abrangidos e que utilidades, presentes e futuras, podem ser encontradas com a utilização da WS. Nele são explicados conceitos como GDB, os seus tipos, os conceitos presentes na generalidade das GDB e ainda está presente uma abordagem àquilo que são e de quais são os passos mais importantes da construção de ontologias.

No 4º capítulo, "Tecnologias e Ambiente de Desenvolvimento" está presente uma abordagem sobre as alternativas tecnológicas possíveis para o desenvolvimento do trabalho, apresentando alternativas e por fim a tecnologia escolhida, demonstrando a sua instalação.

No 5º capítulo, "Implementação", é demonstrado como o trabalho foi desenvolvido, passando por um metamodelo inicial, populando-o e aplicando algoritmos sobre ele, explicando os ajustes feitos a esse modelo, e novamente aplicando algoritmos sobre o modelo, resultando na obtenção de resultados e na interpretação dos mesmos. Por fim, são feitas considerações gerais sobre o trabalho desenvolvido.

Já no sexto e último capítulo, é dada uma perspectiva generalizada daquilo que foi a experiência de desenvolver o trabalho, bem como uma perspectiva futura do mesmo.

Capítulo 2

Estado da Arte

Serve o presente capítulo como enquadramento relativo à atualidade de alguns trabalhos passíveis de terem a função de contextualização do presente, sendo constituinte de todos estes alguns pontos, sejam estratégias de pesquisa, contexto dos problemas ou métodos utilizados, que poderão servir como base para o desenvolvimento do trabalho relacionado à presente dissertação.

São apresentados e resumidos alguns trabalhos considerados relevantes no contexto dos Sistemas de Apoio à Decisão e que fazem uso de algumas das tecnologias estudadas e introduzidas no capítulo 1.

2.1 Um mundo cada vez mais tecnológico

Nos anos recentes a quantidade de dados produzidos tem sido grande, devido ao facto de vivermos num mundo cada vez mais inter-conectado [3]. Esses dados são gerados em plataformas como blogs e redes sociais, entre outros. Como consequência da enorme quantidade de dados produzidos, e pela natureza dessa consequência, eles muitas vezes são inter-dependentes.

Até há pouco tempo, as entidades nas quais é possível o surgimento destes dados tinham uma abordagem pouco restritiva relativamente ao controlo dos mesmos. Era possível a entidades exteriores obter facilmente acesso a estas informações.

Existem benefícios claros do crescimento do fluxo de dados na internet, resultantes do aumento da qualidade e quantidade de soluções neste âmbito, como por exemplo a acessibilidade imediata a informações relevantes, tais como a ocorrência de determinados eventos ou mesmo assuntos que sejam do nosso interesse [4], ou ainda o facto de conseguirmos comunicar em qualquer local, a qualquer hora e com uma facilidade tremenda com outras pessoas, tornando este processo instantâneo [5].

Também no caso das lojas online, de entre muitos outros exemplos de instantaneidade no contexto da internet, conseguimos à distância de um clique comprar o que nos interessa, a qualquer hora do dia.

Outra vantagem de soluções como redes sociais é a visibilidade que dão a criadores de conteúdo, permitindo-lhes potenciar ao máximo a sua capacidade de inovação e a sua criatividade [6]. O contexto atual, resultante do desenvolvimento e popularidade de plataformas deste género, permite uma visibilidade e reconhecimento para talentos emergentes que outrora não era possível.

No entanto, as desvantagens são também elas consideráveis. Estas desvantagens, na sua vasta maioria, são decorrentes dos perigos que existem no mundo online, onde bases de dados enormes com milhões de registos estão dependentes de uma monitorização constante e sujeitas à invasão da parte de pessoas mal intencionadas, causando prejuízos enormes às instituições e às pessoas.

Estes perigos residem no facto de, como consequência da diversidade de formas de acessibilidade e possibilidades de acesso a dados sensíveis em bancos, redes sociais, websites ou até redes de telecomunicações, muitas atividades consideradas criminosas terem sido potenciadas, resultantes da manipulação de dados, a favor dos malfeitores e de forma a desfeitear o sistema, [7] descobrindo nele falhas que levam ao prejuízo da comunidade em geral.

As atividades criminosas vão desde a predação sexual [8], passando pela manipulação de resultados de eleições, inclusive fora dos próprios países dos malfeitores, até crimes relacionados com finanças, sob a forma de fraude.

2.2 Conceitos relacionados

2.2.1 "Big Data" e Web Semântica

Segundo Aristóteles, existem dois tipos de espécies [9], as gregárias e as solitárias, denominadas respetivamente pelo próprio como "koinonia" e "monadika". Segundo o filósofo, o homem faz parte de ambas as espécies. Os dois grupos são, por esse motivo, passíveis de nova divisão. Existem aquelas que têm tendência a ter uma vida social - "politika" - e as que vivem de uma maneira dispersa - "sporadika". Desta nova divisão, Aristóteles conclui que o homem é uma espécie social, ou seja, que da natureza humana faz parte a socialização.

A socialização transformou o ser humano não só num ser biológico como também numa espécie social, isto é, introduziu em nós a capacidade de transmitir informação através da comunicação [10] e com isso evoluir os nossos conhecimentos ao longo dos milénios, partilhando-os e evoluindo sobre os já existentes, dia após dia.

A comunicação e partilha de determinados conhecimentos comuns define culturas (desde pratos tradicionais de uma região a comportamentos típicos, até às religiões) e aquilo que deu origem a essas culturas em primeira instância foram comunidades, que posteriormente se foram sub-dividindo conforme determinados interesses, sempre baseados em conhecimentos e denominadores comuns, como linguagens, crenças ou localizações geográficas. As comunidades deram origens a cidades e países.´

Através de comunidades e certos padrões, o ser humano foi construíndo a sua história, com base na troca de informações. Comportamentos podem estar associados a denominadores comuns e esses denominadores podem resultar de ou resultar em relações inter-pessoais.

Com a evolução humana, baseada em grande parte nos conceitos nesta secção já referenciados, surgiu aquilo que conhecemos hoje em dia por tecnologia. E com a evolução da tecnologia surgiu, no ano de 1939 o primeiro computador [11]. Foi o primeiro passo para, anos mais tarde, ter surgido aquilo que é mundialmente conhecido e mais uma das ferramentas indispensáveis (até no sentido de comunidade) do Mundo de hoje: a internet.

A partir do surgimento da internet, foi uma questão de tempo até serem encontrados mecanismos de armazenamento de dados, que se tornam cada vez mais massivos na sua quantidade. O conceito que surge em 1997 [12] como consequência destes dados massivos e da necessidade do seu tratamento foi denominado "Big Data". Este termo descreve a área que trata, em grande parte, da análise e interpretação de grandes volumes de dados. Trata-se de um conceito bastante relativo, que envolve o uso de estatísticas para a obtenção de informações sobre dados, por princípio, em grande quantidade.

Estes dados contêm grande variedade e volume (por dia são gerados cerca de 2,5 quintilhões de bytes, estando nós neste momento na era do petabyte [12]). A velocidade a que estas informações são geradas é enormíssima.

Tendo isto em conta, o conceito de "Big Data" pode ser resumido em cinco sub-conceitos, denominados comummente como cinco V's (figura 2.1 [13]:

- Volume: refere-se à quantidade de dados armazenada gerada e operada num determinado sistema, os quais necessitarão de ser explorados.

- Variedade: refere-se ao facto de as fontes de dados terem extrema variação, e com isso a sua complexidade aumentar, bem como a dificuldade das análises também crescer. Esse crescimento da variedade potencia uma quantidade e tipo enormes de ligações entre estas informações, de maneira a sermos capazes de relacioná-las de forma coerente.

- Velocidade: os dados estão continuamente a surgir, constantemente a ser gerados e vêm de todas as direções possíveis. É necessário, para o processo fluir, conseguirmos continuamente analisar essas novas ocorrências.

- Valor: de maneira a termos qualidade na análise, é absolutamente imperioso conseguirmos filtrar a informação que nos vai ajudar a atingir a finalidade que queremos, a informação útil. A informação por si só não tem valor e devemos isolar aquela que é relevante, mantendo o escopo mais fechado mas sem perda de performance.

- Veracidade: refere-se ao quanto uma informação é verdadeira ou não. Ou seja, existe a necessidade de garantir que os dados verdadeiros são realmente aqueles que estão a ser coletados pelo sistema, de maneira a não ficarmos com informação que danifica o processo de análise, tornando-a imprecisa. A qualidade e precisão são difíceis de controlar e este é um dos sub-conceitos mais importantes na referida área de estudo.

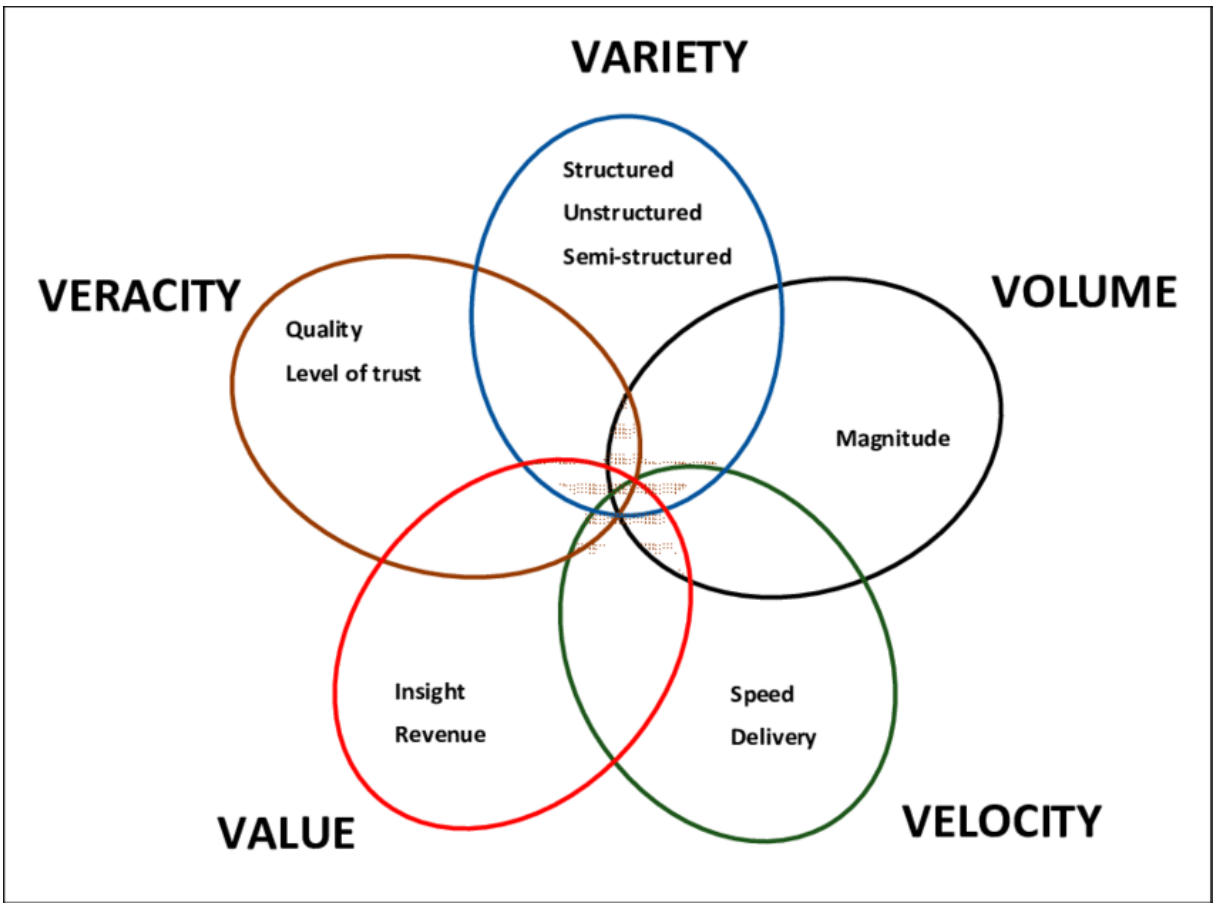

Figura 2.1: Os 5 V's [14]

Estes dados são gerados por diferentes estruturas, podendo tomar três formas diferentes quanto à sua estrutura: Existem os dados estruturados, os semi-estruturados e os não-estruturados.

- Dados estruturados: são informações que podem ser encontradas em bases de dados de SQL. Possuem características bem-definidas, um "esqueleto" que permite associar

cada característica a um valor, sendo que essas propriedades estão armazenadas na base de dados e associadas à "template" de objetos de um tipo (classe) sob a forma de campos.

Um exemplo deste tipo de estruturas é a forma como uma base de dados relacional consegue armazenar informações sobre um determinado aluno numa determinada escola. A escola vai ter características em comum com outras escolas estruturalmente, isto é, vai ter um nome, endereço, id, telefone, entre outros. O aluno estará inserido no contexto desta escola, sendo possível relacioná-los objetivamente. Este estudante iria, também ele, possuir características em comum com os colegas. Um id, nome, curso, número mecanográfico, entre outros.

No entanto, embora a estrutura para cada um dos "templates" (escola e aluno) seja igual entre os objetos do seu tipo, os valores dos campos serão diferentes e isso irá fazer com que uma distinção e individualização seja possível.

- Dados semi-estruturados: são dados que não possuem uma estrutura pré-definida, no entanto contêm certos atributos comuns a todas as informações de um determinado tipo, o que os torna parcialmente estruturados. Exemplos de utilização deste tipo de dados são metadados ou tags de semântica [15].

- Dados não-estruturados: não podem ser armazenados e geridos numa base de dados relacional. Isto porque a informação lá contida não pode ser decifrada de uma forma objetiva. Isto é, o sistema que gere uma base de dados relacional não iria conseguir relacionar cada pedaço de informação lá contido com outra instância.

  Um exemplo prático é uma qualquer palavra da presente dissertação. Mesmo que essa palavra tenha sido escrita pelo autor, ela não possui nenhuma forma de ser associada a ela exclusivamente. Torna-se impossível etiquetar e inserir em contextos todas as palavras de todos os textos, ao nível de uma organização fiável e rentável de informações.

Juntando estes três tipos de estruturas de informação possíveis na área de "Big Data",

temos a variedade suficiente para necessitarmos de uma grande versatilidade na forma como tratamos os dados. Com isto, existem três tipos de dados que podemos considerar, nesta área:

- Enterprise data: tratam-se de dados empresariais. Normalmente usados com a finalidade de optimizar processos e identificar problemas existentes em determinadas secções da empresa. São utilizados pelas equipas de vendas no sentido de ajustar os produtos às necessidades dos utilizadores segundo os dados recolhidos. No âmbito do tema desta dissertação, que consiste na deteção de fraudes, também são importantes porque podem ser obtidas inferências para detectar possíveis anomalias, que resultam em padrões comuns, posteriormente analisados e detetados.

- Personal data: são dados pessoais, que são obtidos através do uso de tecnologias privadas, como por exemplo telemóveis, computadores, entre outros. A nível da identificação de padrões, podem ser feitas ontologias que resultam na deteção de preferências pessoais, que de maneira acumulada levam a preferências coletivas.

- Social data: normalmente são recolhidos a partir de interações online entre os utilizadores, sendo as redes sociais a principal fonte deste tipo de informações. São retiradas informações acerca do comportamento de determinado grupo com características semelhantes, formando padrões comportamentais de vários grupos. Assim, podem ser detectadas comunidades e comportamentos característicos de cada uma delas.

No âmbito do tema abordado nesta dissertação, nomeadamente fraude, especificamente académica, este conceito pode estar presente, pois ao longo de vários anos letivos possuimos milhares de registos gerados, principalmente nas plataformas online.

Estes dados podem ser:

- Estruturados: ao definir por exemplo um grupo de trabalho vamos ter características objetivas e fixas, formando um “esqueleto“ para acomodar esses registos. Campos

como id, número de alunos constintuintes do grupo ou nota global seriam exemplos deste tipo de dados.

- Não-estruturados: as respostas de um determinado aluno a uma pergunta de um determinado teste poderiam ser entidades de uma eventual base de dados NoSQL, sendo que as palavras desses textos são extremamente ambíguas e são impossíveis de contextualizar especificamente a nível lógico, na sua totalidade

No entanto é possível trabalhar estas informações através de modelos semânticos, que resultam em inferências e em dar significado num determinado contexto aos dados, possíveis através de ontologias, existindo neste caso uma ligação de conceitos entre a área de "Big Data" e aquela para a qual esta dissertação está vocacionada, que é a de representação do conhecimento. A proposição de tais modelos semânticos surge como uma alternativa credível para dar significado à forma como estes dados se relacionam.

Por exemplo, Resource Description Framework Schema (RDFS) é uma proposição apontada como uma abordagem adequada para representar entidades de dados, bem como as suas relações, conseguindo nós extrair conhecimento de relações entre quantidades enormes de dados, através de uma representação coerente dos metadados [16].

Existem e existiram motores de busca inteligentes, entre os quais "Ask Jeeves" (ou "Ask.com") e "Wolfram Alpha".

No caso do Wolfram Alpha, lançado em 2009 [17], é computacionado conhecimento de diversas áreas, tais como finanças, matemática, nutrição, entre outras. O objetivo deste mecanismo de conhecimento computacional é conseguir dar respostas precisas às questões efectuadas pelos utilizadores, através de uma limitação progressiva das informações, relembrando-nos do conceito de "veracidade", presente nos 5V's do "Big Data".

Esta tecnologia possui uma infinidade de "datasets", acumulados ao longo de vários anos. Provavelmente, o "dataset" mais conhecido é relativo aos perfis da rede social "Facebook", através do qual e mediante autorização dos utilizadores em causa, podem ser gerados relatórios de análise de dados, com informações resultantes de dados não-estruturados, tais como a frequência de palavras utilizadas em certos contextos e também

de dados estruturados, como é o caso da medição de quantidade de homens e mulheres entre os amigos do utilizador em questão. Este é um exemplo da aplicação da web semântica para retirar informações relevantes de um determinado conjunto de dados.

2.2.2 Knowledge Graphs

São vários trabalhos, desenvolvidos muitos deles por grandes empresas, que relacionam a Web Semântica com o conceito de "Big Data", pois estão presentes, como referido, pontos em comum entre as duas áreas de estudo, nomeadamente no que toca a dar significado a dados recolhidos num determinado contexto.

Muitas das grandes empresas que possuem uma enormidade de dados gerados por dia utilizam os conceitos presentes na Web Semântica para o tratamento desses dados. Este é o caso do Facebook, onde são gerados diariamente 4 petabytes (PB) (equivalente a mais de 4000000000000000 bytes, ou 4000000 gigabytes (GB)) de dados [18], ou da "Google" (onde são realizadas cerca de 1200 trilhões de pesquisas por dia [19]).

Um exemplo de como relacionar quantidades massivas de dados e torná-las inter-dependentes veio da "Google" e é denomidado Google Knowledge Graph (GKG). Esta tecnologia foi adicionada ao motor de busca da Google no ano de 2012 e cujos testes foram realizados na plataforma dos Estados Unidos da América (EUA) [20].

Existem várias definições para grafo de conhecimento, ou "Knowledge Graph", referidas na tabela 2.1:

Knowledge Graph
"Um grafo de conhecimento adquire e integra informações numa ontologia e aplica um "reasoner" para obter novos conhecimentos." [21]
"Um grafo de conhecimento (i) descreve principalmente entidades do mundo real

e as suas inter-relações, organizadas num gráfico, (ii) define possíveis classes e relações de entidades num esquema, (iii) permite potencialmente inter-relacionanar entidades arbitrárias entre si e (iv) abrange vários domínios.“ [22]
“Os grafos de conhecimento são grandes redes de entidades, os seus tipos semânticos, propriedades e relacionamentos entre entidades.“ [23]
“Os grafos de conhecimento podem ser concebidos como uma rede de todos os tipos de coisas que são relevantes a um domínio específico ou a uma organização. Eles não se limitam a conceitos abstratos e a relações, mas também podem conter instâncias de instâncias como documentos e conjuntos de dados. “ [24]
“Definimos um grafo de conhecimento como um grafo RDF. Um grafo RDF consiste num conjunto de RDF “triples“ em que cada RDF “triple“ (s, p, o) é um conjunto ordenado dos seguintes termos RDF: um sujeito $s \epsilon U \vee B$, um predicado $p \epsilon U$, e um objeto $U \vee B \vee L$. Um termo RDF é um URI $u \epsilon U$, um “node“ em branco $b \epsilon B$, ou um literal $l \epsilon L$.“ [25]
“[...] existem sistemas, [...], que utilizam uma variedade de técnicas para extrair novos conhecimentos, sob a forma de factos da web. Estes factos estão inter-relacionados e, portanto, este conhecimento recentemente extraído foi referido como um grafo de conhecimento.“ [26]

Tabela 2.1: Definições de “Knowledge Graph“

Juntando os conceitos introduzidos por estas definições, podemos concluir que um

Knowledge Graph (KG) se trata de uma rede de várias instâncias presentes num sistema, de diferentes tipos, que formam conceitos abstratos sob os quais podem ser extraídos novos conhecimentos, através de uma variedade de técnicas de inferência.

No caso específico da "Google", o seu GKG é utilizado precisamente com o fim de melhorar a sua pesquisa, formando um motor de busca inteligente. Assim, as informações provenientes de várias fontes de dados, tais como a "Wikipédia", "Freebase" ou a "CIA World Factbook", bem como as fontes internas da "Google" [20], são estruturadas e detalhadas.

O objetivo é facilitar a pesquisa aos utilizadores e relacionar entidades que possam estar relacionadas com aquela que foi originalmente pesquisada, através de listas de "links" que nos redirecionam dentro do próprio site para cada uma delas. Também está presente uma caixa informativa sobre aquilo que foi inferido pela pesquisa, com informação vinda diretamente do GKG.

No fundo, cada entidade resultante da pesquisa representa um "node" (ver 3) e é possível navegar para outros "nodes" (mais detalhe em 3) presentes no grafo, clicando em cada um dos "links".

Esta rede semânica é utilizada para atribuir um significado às palavras-chave da busca efectuada. Elas são interpretadas como uma sequência de caracteres, todos arbitrários, que são automaticamente relacionados a outras instâncias do Mundo real, como objetos ou pessoas [20].

Por exemplo, se colocarmos as palavras-chave "Marcelo Rebelo de Sousa" (figura 2.2) na plataforma, é-nos apresentada uma caixa informativa com factos relevantes sobre o Presidente da República Portuguesa. Nela, vemos informações sobre esta entidade e podemos ver com mais detalhe as restantes informações ao carregarmos no primeiro link.

No entanto, são-nos apresentadas entidades relacionados a esta pessoa, através de "links". Podemos automaticamente, a partir daqui, ver qual a cidade e informar-nos sobre ela, perceber quem é a esposa do político, ou mesmo com quem ele já teve outros relacionamentos. Conseguimos ainda informar-nos sobre quem são os filhos e até mesmo os irmãos do Presidente.

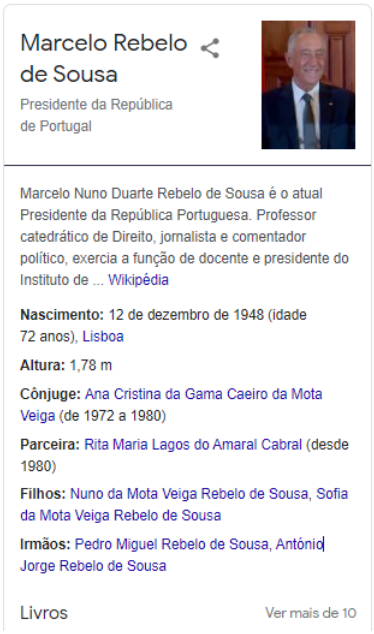

Figura 2.2: “Infobox“ para as palavras-chave “Marcelo Rebelo de Sousa“

Para além das pessoas relevantes na vida de Marcelo é-nos possível verificar, por exemplo, quais os livros dos quais ele é autor. Em último lugar, é-nos apresentado um conjunto de pesquisas relacionadas a estas palavras-chave, inferidas pela “Google“, comprovadamente de forma precisa.

Ou seja, temos “nodes“ relacionados uns com os outros, que representam entidades do Mundo real de qualquer tipo. São-nos apresentadas informações relativamente a localidades, pessoas e livros, provando que apenas três palavras, que isoladamente não possuem qualquer contexto, quando relacionadas podem trazer muita informação extremamente útil relativas a uma entidade e a todas as outras que a si se relacionam.

KG é hoje em dia um termo altamente popularizado e com algumas aplicações famosas, para além da já referida da “Google“.

Alguns exemplos são:

- DBPedia: trata-se de um sistema para extrair conteúdo da Wikipédia de forma estruturada. Baseia-se numa ontologia (3) que tem cobertura enciclopédica sobre entidades como pessoas, casas, locais, entre outros. Este “dataset“ está no centro do

movimento Linking Open Data (LOD), tendo tido um grande contributo no que toca a ajudar as organizações a iniciarem os seus próprios KG internos [27], promovendo assim a amplificação das redes de conhecimento a nível global.

- Geonames: é um "dataset" referente a entidades geográficas com mais de oito milhões de nomes geográficos, correspondentes a 25 milhões de entidades [27] existentes. Lá, conseguimos obter informações tais como latitude, longitude e população dessas localidades [28].

- Wordnet: é um "dataset" lexical da língua inglesa. Substantivos, verbos, adjetivos e advérbios são agrupados em conjuntos de sinônimos cognitivos (denominados "synsets"), com cada um deles a representar um conceito diferente. As "synsets" são interligadas através de relações conceituais-semânticas e lexicais. Cada conjunto de palavras relacionadas entre si, pode ser navegado e explorado através de "links".

- FactForge: trata-se de um repositório de LOD, que junta mais de mil milhões de factos [29] de "datasets" como os já referidos "DBPedia" e "Geonames", mas também "Wordnet", "Panorama Papers", entre outros, e ainda ontologias como a "Financial Industry Business" sobre entidades do universo real como localizações, pessoas e organizações. Nesta solução, está incluída a funcionalidade de notícias em tempo real, associando-as a entidades/ conceitos através de metadados.

2.3 Trabalhos sobre deteção de padrões anormais

Como referido no início do presente capítulo, é absolutamente crucial com o desenvolvimento de melhores e mais complexas técnicas para a prática de fraude, saber mitigá-la ou, melhor ainda, antecipar a potencial ocorrência deste tipo de prática, inclusive no meio académico. Foram desenvolvidos trabalhos neste âmbito, abordados e sintetizados nesta secção.

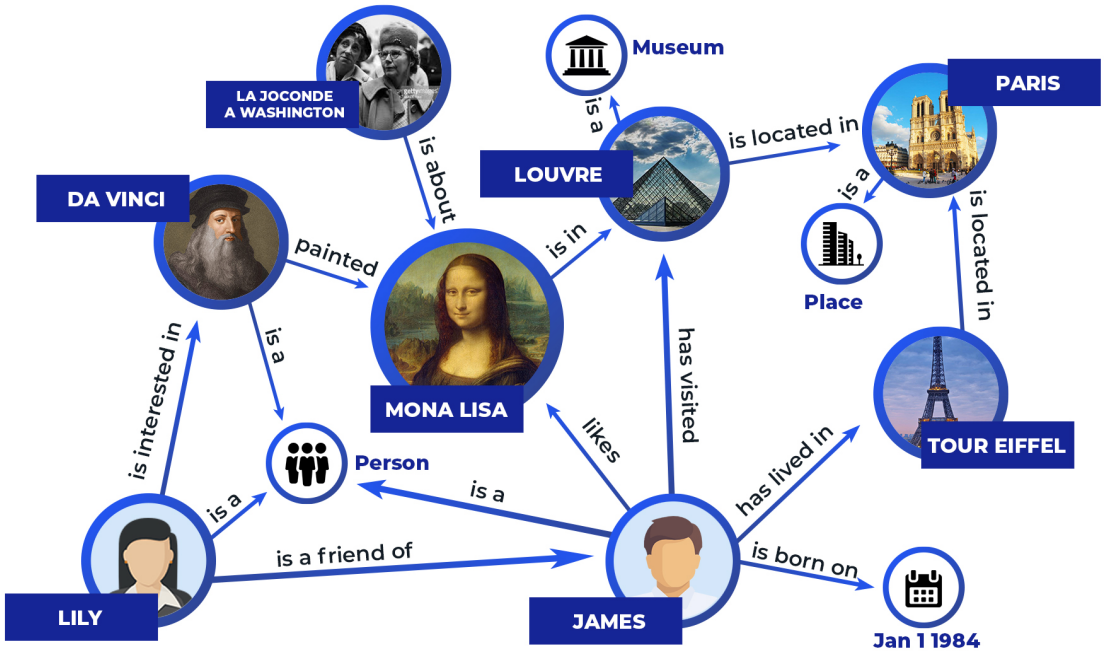

Figura 2.3: Knowledge Graphs [30]

O conceito de Graph Based Anomaly Detection (GBAD) refere-se a descobrir nos padrões de dados estruturados instâncias anormais, sendo que os dados representam entidades, ações e relacionamentos, representados na forma de um grafo. O "input" consiste num grafo etiquetado, no qual as entidades são representadas por vértices rotulados e os relacionamentos entre estas entidades tratam-se de arestas que unem esses vértices.

De maneira a serem detectadas anomalias, GBAD faz uso de diferentes algoritmos para detectar se os dados são atualizados, se há novos registos ou se houve remoções. A partir daqui, cada um desses algoritmos descobre atividades que parecem normais mas não o são, ou seja, o padrão é próximo ao normal mas encontra-se alterado. Desta forma, é possível detectar atividades potencialmente fraudolentas em diversos contextos, entre os quais o académico (se pensarmos nos relacionamentos entre alunos).

Em [8], [31], [32] e [33], são realizados trabalhos nos quais são aplicados os conceitos de GBAD para a deteção de anomalias:

- No primeiro caso [8], é realizado um levantamento de técnicas computacionais existentes para detetar anomalias em redes sociais. Estas anomalias são caracterizadas

como estáticas ou dinâmicas, rotuladas e não-rotuladas. São sugeridos dois sub-processos para a deteção do desvio de padões nas redes sociais: a seleção e cálculo dos recursos de rede e a classificação de observações a partir desse mesmo espaço de recursos.

- Em [31], foi feita uma pesquisa com a intenção de fornecer uma visão geral e abrangente dos métodos de GBAD. É pelos autores oferecida uma estrutura generalizada, com as seguines categorias: abordagens supervisionadas (e semi-supervisionadas) versus não-supervisionadas, relacionadas com grafos estáticos versus grafos dinâmicos, para grafos atribuídos versus grafos simples. É feita uma síntese relativa à deteção de anomalias e são destacadas algumas técnicas para descobrir a causa das mesmas. São ainda apresentados alguns trabalhos que utilizam grafos para detetar anomalias no mundo real, em diversos âmbitos e áreas da sociedade.

- O trabalho [32] é uma pesquisa sobre comportamentos anormais, desviantes da maioria dos utilizadores, numa rede social. É dada uma visão geral sobre as técnicas existentes para esta deteção de anomalias e ainda a separação entre duas técnicas: aquelas que são baseadas em estrutuas e aquelas que são baseadas em comportamentos, para além de discutir os principais problemas nesta área de pesquisa.

- Por último, em [33] existe a intenção, da parte dos autores em fornecer uma visão geral daquilo que é a detecção de anomalias em redes dinâmicas. São descritos quatro tipos de anomalias que surgem neste contexto. É construída uma taxonomia de duas camadas, a primeira consistindo no particionamento dos métodos com base na intuição e a segunda focando-se na sub-divisão dos mesmos com base nos tipos de anomalias pelos métodos detetadas. São destacadas as principais semelhanças e diferenças entre estes métodos.

Nos trabalhos [34], [35] e [36] este conceito é aproximado ao tema da deteção de fraudes em particular, estando GBAD umbilicalmente ligado a essa deteção.

- [34] consiste em estudar abordagens avançadas de mineração de dados, Support

Vector Machine (SVM), entre outros conceitos como florestas aleatórias, com a intenção de controlar e processar com mais qualidade fraudes no mundo das finanças, nomeadamente relativas a cartões de crédito.

- No caso de [35], é-nos apresentado um estudo sobre utilizadores com intenções de fraude, que utilizam diversas abordagens para corromperem o sistema de comércio eletrónico. Estas abordagens, como são cada vez mais complexas, fazem os sistemas de prevenção de fraude ficar obsoletos. Numa abordagem que consiste numa visão sistematizada e bastante abrangente do problema, são selecionados cinco sistemas de comércio eletrónico: cartão de crédito, telecomunicações, seguro saúde, seguro automóvel e leilões online. São introduzidas abordagens atuais de sistemas de prevenção de fraude e feito um prognóstico sobre o futuro próximo da área.

- Por último, em [36] é-nos apresentada uma revisão da literatura académica sobre a aplicação de técnicas de mineração de dados para a deteção de fraude no mundo financeiro. Foram agrupados pelos autores 49 artigos, publicados entre 1997 e 2008 e divididos em quatro categorias:

  - Fraude bancária.
  - Fraude de seguros.
  - Fraude de títulos e "commodities".
  - Outras fraudes relacionadas.

  São ainda inferidas várias conclusões sobre o futuro na área.

Em [37] é realizado um trabalho de síntese sobre o tema, no qual são investigadas as tendências atuais e identificados os principais desafios que requerem esforços de pesquisa significativos para aumentar a credibilidade da GBAD, sendo posteriormente fornecidas recomendações para lidar com variados desafios no que toca ao uso desta técnica.

Capítulo 3

Web Semântica

No ano de 2001, por Tim Berners-Lee, James Hendler e Ora Lassila [38] e como resposta a uma internet cada vez mais em crescimento, surgiu o conceito de Web Semântica. À data, os três idealizadores do conceito escreveram "Web Semântica: um novo formato de conteúdo para a Web que tem significado para computadores vai iniciar uma revolução de novas possibilidades"[39].

Existe uma ideia estrutural de classificação, armazenamento, recuperação e disseminação da informação, à semelhança do que acontece com as enciclopédias. Quase como que se se tratasse de uma rede global de documentação.

A intenção da WS é a de organizar a informação de forma a ser entendida, pelos nossos sistemas, através de padrões de formatação de dados (como Resource Description Framework (RDF)).

Com isto, criam-se grupos específicos de conhecimento ao longo do tempo, sendo que estes podem retirar significado de toda esta informação, utilizando-a determinada tarefa ou contexto específico. Com recurso a ontologias estas tarefas e contextos podem ser relacionados, aumentando ainda mais a profundidade da rede global de conhecimento.

"A Web semântica pode ser encarada como um constructo abrangente cujo objetivo último será o de tornar a WWW numa "base de dados global", tornando possível, por um lado, uma pesquisa transversal (independente do tipo, formato e fonte dos dados) e, por outro, a obtenção de dados semanticamente inter-relacionados, i.e., informação e não

apenas uma listagem de documentos, muitas vezes sem ligação semântica entre si, como acontece na actual web de documentos.“[40]

Neste capítulo serão abordados e sintetizados os conceitos mais importantes daquilo que é a Web Semântica, com o objetivo de fornecer um enquadramento teórico naquilo que é a lógica por detrás da criação de ontologias, inclusive a relacionada com a temática da utilização de ontologias para a deteção de fraude académica. Alguns conceitos relacionados com a WS podem ser vistos na figura 3.1.

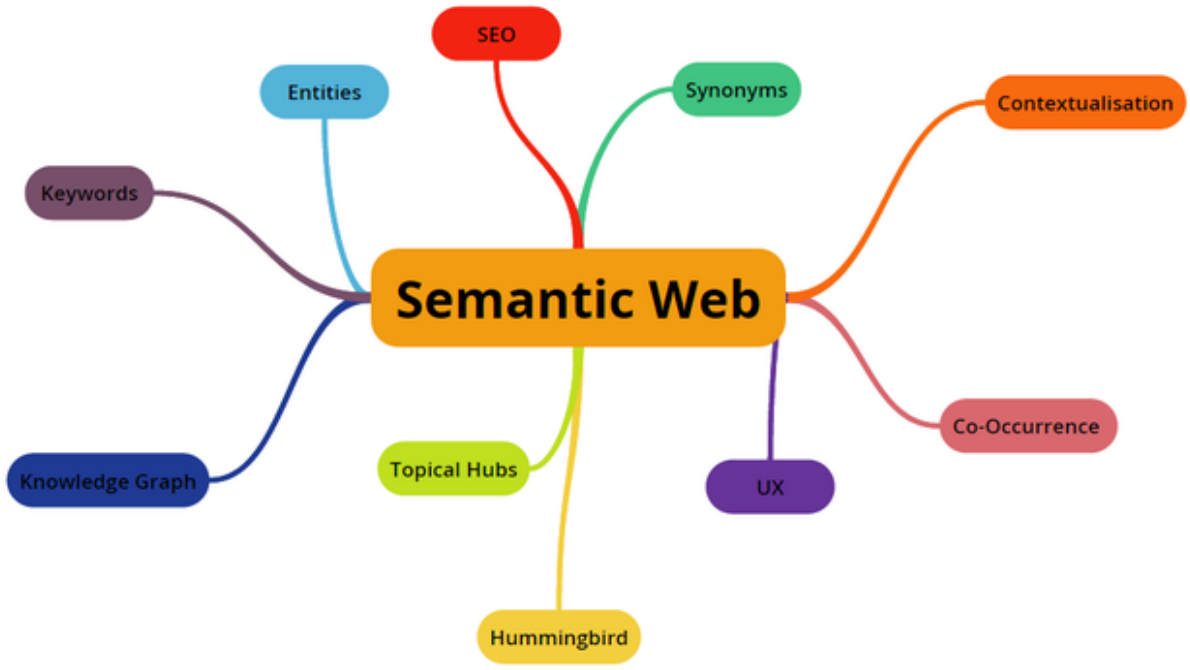

Figura 3.1: Web Semântica [41]

3.1 Graph Databases

Uma GDB é uma Base de Dados (BD) usualmente do tipo Not Only SQL (NoSQL) que faz uso de estruturas típicas de um grafo como “nodes“, arestas e propriedades, por forma a representar e guardar informações. Uma GDB geral tem a capacidade de guardar qualquer tipo de grafo, ao contrário de outras bases de dados especializadas, como é o caso

das “Network Database (NDB)“ ou de “triplestores“. Este tipo de BD tem a capacidade de guardar dados como uma coleção de informações organizada [42].

NoSQL é muito usado por estas bases de dados, por forma a combater as limitações das bases de dados relacionais. Um os objetivos principais é o de reduzir a complexidade da montagem da base de dados. Os dados são guardados através de conexões implícitas. As relações podem ser rotuladas, direcionadas entre os “nodes“ correspondentes e podem também possuir propriedades.

Um exemplo disto seria a relação entre um estudante e uma disciplina. Numa base de dados relacional, tradicionalmente, como um estudante pode estar inscrito em muitas disciplinas e uma disciplina tem muitos estudantes inscritos, seria necessária a criação de uma tabela associativa, por causa da formação de uma relação n-n.

Neste caso, essa complexidade desfaz-se porque na relação entre estudantes e disciplinas podemos colocar características tais como, por exemplo, o ano de inscrição na própria unidade curricular.

Uma base de dados relacional requer, para um armazenamento e utilização confiáveis, uma grande normalização da sua estrutura, pelo que quando conseguida esta normalização, nos garante uma grande consistência. No entanto, pela complexidade da sua estrutura, são mais lentas quando em comparação com uma GDB no que toca a “datasets“ mais associativos [43].

Uma grande mais-valia da abordagem que contempla a utilização de grafos é o tempo de computação das suas “querys“, quando em comparação com uma base de dados relacional.

“O verdadeiro valor da abordagem de grafos torna-se evidente quando se realiza pesquisas com mais de um nível de profundidade. Por exemplo, considere uma pesquisa por utilizadores que possuem “assinantes“ (uma tabela que vincula utilizadores a outros utilizadores) no código de área “311“. Nesse caso, uma BD relacional deve primeiro pesquisar todos os utilizadores com um código de área em “311“, depois pesquisar na tabela de assinantes qualquer um desses utilizadores e, finalmente, pesquisar na tabela de utilizadores para recuperar os utilizadores correspondentes. Em contraste, uma GDB procuraria todos

os utilizadores em "311" e, em seguida, seguiria os backlinks através do relacionamento do assinante para localizar os usuários assinantes. Isso evita várias pesquisas, consultas e o uso de memória envolvido na retenção de todos os dados temporários dos vários registos necessários para construir o "output". Em termos de notação "Big O", esta query levaria $O(log(n)) + O(1)$ tempo - ou seja, proporcional ao logaritmo do tamanho dos dados. Em contraste, a versão relacional levaria várias pesquisas $O(log(n))$, mais o tempo necessário para juntar todos os registos de dados."[43]

No entanto, no caso de querermos rapidez a retornar um grande conjunto de dados e uma estrutura mais estática, constante ao longo do tempo, as bases de dados relacionais sobrepõem-se às GDB.

As GDB são comumente utilizadas com a finalidade de guardar dados relacionados a KG (2.2.2), bem como a sua lógica, predicados e descrições, complementando assim o conceito de KG e funcionando em conjunto e não representando, portanto, o mesmo conceito. A nível conceptual, uma GDB está representada na figura 3.2.

3.1.1 Tipos de Graph Database

As GDB são classificadas em diferentes tipos, baseando-nos na forma como os dados são guardados e também no seu modelo de dados.

Relativamente ao armazenamento de dados, existem três categorias principais [45]:

- Armazenamento nativo: é especificamente direcionado a guardar e gerir os grafos em disco, com as unidades de medida a serem os vértices e arestas. Exemplos deste tipo de armazenamento são "Neo4j" e "TigerGraph".

  Existem algumas regras para este tipo de armazenamento, que podem ser consultadas em [46]:

  - Modelação e Armazenamento nativos - consiste em guardar e modelar a informação como um grafo de relações, ao invés de colunas e linhas ou qualquer outro tipo de estrutura.

Figura 3.2: Graph DB [44]

– Gestão nativa - refere-se à gestão de dados e relações estritamente através de propriedades e capacidades de grafos.

– Relações de primeira classe - a regra consiste no tratamento das relações presentes num grafo como elementos com propriedades quantitativas e usadas pelo Graph Database Engine (GDBE).

– Disponibilidade em tempo real - conseguir fazer "queries" à informação em tempo real, independentemente da quantidade ou complexidade das relações entre as entidades.

– Adjacência livre de índices - consiste em ligar cada elemento diretamente aos seus relacionamentos, sejam direcionados para si ou a partir de si.

  - Gestão de dados abrangente - conseguir gerir por completamente as operações Create, Read, Update and Delete (CRUD), seja dos dados ou dos seus relacionamentos subjacentes.
  - Gestão discreta - ter a capacidade de adição e modificação de dados/ relações sem ter que fazer alterações estruturais à BD.
  - Suportar "Cypher" - conseguir suportar a linguagem utilizada pelos líderes na indústria.
  - Não-subversão: proibir tentativas de acesso ou modificação de dados através de conseguir contornar regras de integridade e restrições impostas pela GDB.
  - Transações Atomicity, Consistency, Isolation, and Durability (ACID) - garantir que todas as transações são ACID, ou seja, seguem as regras de atomicidade, consistência, isolamento e durabilidade, por forma a garantir consistência e durabilidade.
  - Leituras consistentes - assegurar que os utilizadores que lêm os dados conseguem sempre ver os mesmos registos.
  - Escritas consistentes - assegurar que os utilizadores que escrevem e atualizam os dados veêm sempre a última versão dos mesmos.
  - Independência de integridade - Armazenar as restrições de integridade de dados no catálogo de dados gráficos, não em programas de aplicação.
  - Independência de dados - As aplicações não são afetadas logicamente quando as representações de armazenamento de dados gráficos subjacentes ou os métodos de acesso mudam.
  - Apresentação perfeita - armazenar e mostrar o grafo de uma maneira unificada, de maneira a providenciar aos utilizadores uma visão completa do grafo, independentemente da localização dos dados.

- Armazenamento do tipo chave-valor: este tipo de armazenamento consiste no uso de um par de chaves-valor, com um valor a corresponder a cada chave. É visto

como uma adaptação das “graph databases“ ao conceito de “Big Data“. A ideia de procurar pela chave e retornar registos rapidamente e a capacidade de suportar grandes conjuntos e dados são as melhores características desta forma de guardar os dados [47]. Este tipo de armazenamento faz uso de bases de dados NoSQL como “Cassandra“ e “HBase“.

- Armazenamento do tipo relacional: “GraphX“ é um tipo de GDB que faz uso deste armazenamento. São usados modelos relacionais de forma a guardar as informações representadas por vértices e arestas, em “edge tables“ e “vertex tables“. Os vértices representam as entidades e as arestas são as conexões entre elas.

  Por exemplo, numa rede de computadores, os vértices são os computadores e as arestas são as ligações entre si [48].

Depois de abordada a forma como os dados são armazenados nos principais tipos de GDB, passemos à abordagem baseada no seu modelo de dados.

- “Property Graphs“: trata-se de um dos tipos de modelo de dados mais importantes para o desenvolvimento do presente trabalho. Faz uso de “nodes“, relações e propriedades, na sua organização de dados (3.1.2).

- “Hypergraphs“: tem a sua maior utilidade nos casos em que, através do nosso modelo de dados, temos a obrigatoriedade de representar uma vasta gama de relações n-n (muitos para muitos). Uma relação (tecnicamente chamada “hyperedge“, neste contexto) entre entidades, neste tipo de estruturas, pode ligar um número qualquer de registos. Um exemplo de “hypergraph“ está presente na figura 3.3.

- “Triple Store“: também conhecido como RDF, tem como característica o armazenamento de dados numa estrutura conhecida como “sujeito-predicado-objeto“. Por exemplo, “O Tiago namora com a Cristina“, em que “o Tiago“ seria o sujeito, “namora“ o predicado, representado por um verbo no presente, e “a Cristina“ o objeto. A estrutura baseia-se em dois “nodes“, um para o sujeito e outro para o objeto, e um “arco“, que representa a relação entre ambos.

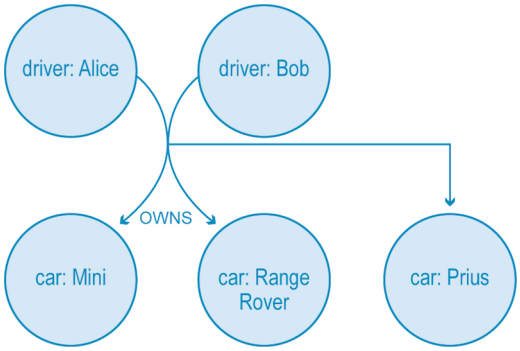

Figura 3.3: Exemplo de “hypergraph“ [45]

3.1.2 Nodes, Edges e Relationships

Tal como acontece em todos os âmbitos de estudo em qualquer área, nomeadamente na tecnológica, existem conceitos sobre os quais todas as soluções para os problemas têm que, obrigatoriamente, se basear.

No caso das GDB(“property graphs“), os três conceitos basilares para a representação de um modelo são:

- “Nodes“: representam entidades do modelo. São o equivalente nas BD tradicionais, as relacionais, a instâncias como escolas, alunos, professores ou grupos. Possuem características como nome, idade ou número mecanográfico. Os “nodes“ podem ter zero ou mais (“labels“) para definir (classificar) que tipo de “nodes“ são. Por exemplo, um trabalho de grupo é um “trabalho“ e ao mesmo tempo um “elemento de avaliação“.

- “Edges“: também podendo ser denominadas “relações“, são linhas no grafo que conectam entidades, ou “nodes“. São uma peça chave no que toca obter padrões relativos a um determinado conjunto de dados. Podem ser uni-direcionais ou bi-direcionais, sendo que no primeiro caso, propriedades que ligam dois “nodes“ podem ter significado diferente conforme a origem e o destino. No caso das “edges“ direcionadas, cada propriedade tem apenas um significado. Todas as relações possuem um “target node“ e um “source node“, ou seja, um nó de origem e um de destino. Também possuem um tipo que as caracteriza e direção.

Imaginemos que um aluno tem aulas com um professor. Na perspectiva do aluno, ele "é lecionado" por aquele professor, enquanto que o professor "leciona" o aluno. Estamos na presença de uma relação com um significado diferente conforme o "node" de origem.

Para o segundo caso, um exemplo é um aluno estar inscrito numa unidade curricular. A disciplina apenas pode atuar como "target node" (destino), sendo que estamos na presença de uma relação unidirecional, sempre com o mesmo significado.

À semelhança das entidades, também as relações podem ser classificadas com propriedades, definidas por pares de "chave-valor".

Este conceito é diferenciador entre as GDB e os outros tipos de bases de dados.

- Propriedades são características associadas aos nós ou às relações. Por exemplo se um aluno tiver um número mecanográfico, isso é representado por um par de chave-valor em nos "nodes" em que essa propriedade estiver presente, em que a chave é o nome da propriedade e o valor é o correspondente daquela propriedade naquele aluno (tal como: {"nmecanografico": "a28248"}) Também no caso das relações, estas propriedades estão presentes.

3.2 Ontologias

Inicialmente, o conceito de "ontologia" surgiu fora do âmbito das tecnologias, ou sequer da ciência em geral. Apareceu como um conceito filosófico e foi definido como o ramo geral da metafísica (ao contrário da teologia, psicologia ou cosmologia, que são ramos específicos)[49].

O termo ganhou notoriedade devido ao filósofo de nacionalidade alemã Christian Wolff e surge da junção de duas palavras gregas "ontos" (referente ao ser) e "logia" (estudos), formando assim a corrente de pensamentos relacionada ao significado do ser. Ou seja, é o ramo da metafísica que estuda os tipos de coisas existentes no Mundo [50].

As ontologias começaram a ser usadas em termos práticos para a classificação, por exemplo, de espécies.

3.2.1 Ontologias na Informática

Na literatura, estão presentes várias definições para aquilo que representa uma ontologia, sendo que uma descrição generalista é a de que se trata de uma especificação de uma conceitualização, ou seja, uma descrição de vários conceitos e as relações existentes entre os mesmos [51].

Uma definição mais concreta, ou seja mais específica, pode ser encontrada em [51]. Aqui, é afirmado pelo autor que uma ontologia se trata de uma espeficicação "formal e explícita" de uma "conceitualização compartilhada". Por "especificação formal", entende-se que algo é legível para os dispositivos informáticos, enquanto que por "explícita" se entende pelos conceitos, axiomas, funções ou restrições definidos de uma forma explícita. Neste caso, uma "conceitualização" tem como função representar um modelo representativo de um contexto real, abstrato. Já "compartilhada", tem como significado o conhecimento consensual.

Como tal, o propósito de uma ontologia é dar-nos a capacidade de compartilhar conhecimento, bem como a sua reutilização.

Para ser conseguido tal objetivo, é necessário um mecanismo concreto e exato, seguido por todos e com um padrão comum e bem-definido. Para alcançar essa meta, surgiu a necessidade da criação de uma especificação formal. Em [52] é sugerida uma forma de criar essas definições formais para os conceitos. Seguindo a lógica sugerida em [52], as ontologias são meios de realizar uma "especificação parcial", ou seja, apenas tendo a função de representar conhecimento numa determinada área de intervenção (domínio), não podendo representar a totalidade do conhecimento em todos os domínios.

Neste sentido, uma ontologia no contexto da informática define uma linguagem, representada por um conjunto de termos, que será utilizada para formular consultas[50].

Esta construção ideológica pode ser encontrada em [53].

Depois de feita a contextualização sobre aquilo que é uma ontologia, nomeadamente no mundo da informática, é importante destacar algumas das suas vantagens, abaixo descritas:

- Permitem a partilha de conhecimento: isto é, caso uma ontologia represente com precisão um determinado domínio de conhecimento, ela pode ser usada por indivíduos que desenvolvam um projeto nesse âmbito.
- Acarretam um vocabulário de representação do conhecimento: isto é, o vocabulário traz uma conceitualização que lhe confere credibilidade e permite a sua sustentação. É específico e por isso não existe ambiguidade na representação de conhecimento.
- Fornecem uma descrição exata do conhecimento: ou seja, é retirada a ambiguidade existente na linguagem natural. Uma palavra, na linguagem natural, pode ter várias interpretações conforme o seu contexto e quando é retirada do mesmo, tem um significado ambíguo. Uma ontologia contextualiza "à priori" o significado de palavras, à partida, ambíguas.
- Existe a possibilidade de extender o uso de uma ontologia de caráter mais genérico, podendo ajustá-la a um domínio mais restrito. Por exemplo, dada uma ontologia genérica de deteção de fraudes, ela pode ser utilizada para a deteção de fraude académica, sempre e quando seja adequada com as características do problema em específico.

Devido a estas vantagens, a utilização de ontologias já abrange várias áreas da informática, tais como [53]:

- Recuperação de informações na internet;
- Processamento de linguagem natural;
- Gestão do Conhecimento;
- Web Semântica.

No entanto, a maior desvantagem reside no facto de ainda não haver uma lógica definida no que toca às metodologias de desenvolvimento, pois ainda não existem muitos trabalhos neste âmbito. Isso faz com que a criação de uma ontologia sobre um determinado contexto, ou domínio, se trate de uma atividade sem muitas diretrizes pelas quais nos guiar, o que dificulta a tarefa.

Uma ontologia, quanto ao seu tipo e função, pode ser classificada como:

- Genérica: procuram representar uma gama mais ampla de domínios, mais abstrata. Não se focam num problema ou tarefa em específico, mas sim em conceitos mais generalistas. Procuram apresentar características comuns a cada domínio.

- De domínio: são ontologias que se focam numa determinada área de intervenção, sendo mais específicas do que as genéricas, isolando-se do restante Mundo e focando-se apenas naquele domínio em específico. Um exemplo seria uma ontologia para deteção de fraude.

- De tarefas: representam atividades aplicáveis a qualquer domínio, transversais entre eles e que podem usar a resolver algum tipo de problema.

- De aplicação: são ontologias que, dado um domínio D e uma tarefa T, representam $D \cap T$, ou seja, representam especializações num determinado domínio e associadas a uma determinada tarefa, em específico.

As ontologias também podem ser classificadas quanto ao seu formalismo, sendo que uma "altamente informal" represeta algo expresso em linguagem natural e aquelas que são "rigorosamente formais" são constituidas por semânticas formais, provas e teoremas. Também podem ser classificadas quanto à sua classificação e conteúdo [53].

Quanto ao seu conteúdo, elas podem ter as seguintes classificações:

- De informação: procuram especificar algo estrutural, algo comum representado sempre da mesma forma, possuindo informação aplicável a todas as entidades abrangidas.

- Terminológicas: procuram especificar termos para utilização final num determinado problema/domínio específico.
- De modelação de conhecimento: representam os conceitos, propriedades, relações, funções, restrições e axiomas, ou seja, a conceitualização do conhecimento.
- De aplicação: procuram contextualizar um domínio com uma tarefa específica.
- De domínio: reresentam algo genérico num determinado contexto, algo abrangente a todos os sub-contextos do mesmo.
- Genéricas: representam algo genérico e independente de contextos, algo que seja abrangente a tudo no Mundo e que seja independente da área de intervenção, domínio ou tarefa.
- De representação: explicam todos os formalismos que formam a representação do conhecimento, através da sua conceitualização.

3.2.2 Conceitos nas Ontologias

De modo a conseguirmos construir uma ontologia coerente, é necessaria a escolha de um dominio de atuação (no caso deste trabalho, a fraude académica). Também é necessária a definição prévia de uma metodologia e das tecnologias, sendo uma ferramenta e uma linguagem quesitos obrigatórios para a definição da sua estrutura, tendo em conta que a grande maioria das ontologias tem na sua estrutura básica os seguintes elementos:

- Classes: Representam o tipo de um conjunto de entidades, com caracteristicas semelhantes num determinado contexto, neste caso na ontologia correspondente a um domínio. Por exemplo, no caso de uma avaliação, todos os elementos de avaliação podem corresponder a uma classe, sendo ela dividida em sub-classes quando queremos especificar mais o tipo de avaliação. Imaginemos um trabalho de grupo. Ele é um “elemento de avaliação“, mas também é um “trabalho“, pertencendo, portanto, a ambas as classes. Na figura 3.4 podemos ver outro exemplo:

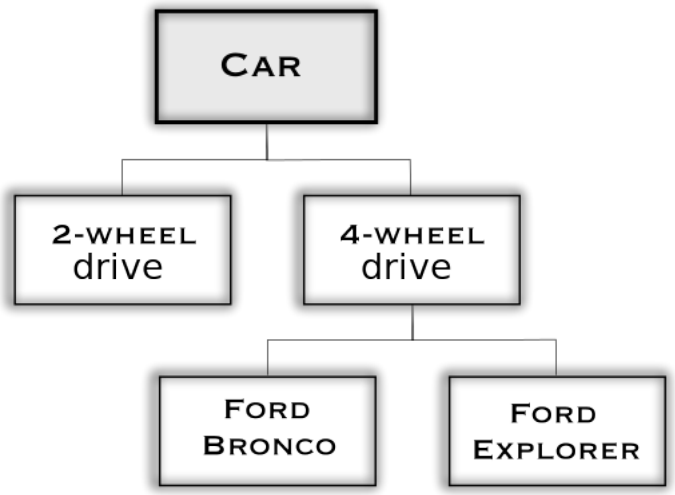

Figura 3.4: Classes e subclasses [54]

- Axiomas: Representam condições que, quando respeitadas, são sempre verdadeiras. Quando estas condições são respeitadas, automaticamente outro facto é inferido automaticamente pela ontologia. Por exemplo, caso o aluno A tenha participado no grupo A do trabalho Y e o aluno B tenha participado no grupo A do trabalho Y, podemos dizer que eles ficaram "no mesmo grupo", estabelecendo automaticamente essa relação, caso dois alunos tenham participado no mesmo grupo.

- Funções: são eventos que ocorrem dentro de uma determinada ontologia. Ou seja, por exemplo, quando um trabalho de grupo é concluído e as notas são distribuídas a cada elemento do grupo A, ao calcularmos a média estamos na presença de uma função.

- Relações: representam determinadas relações que existem entre elementos de classes diferentes. Elas ligam as classes, mas verdadeiramente têm efeito nos indivíduos. Por exemplo, dado um aluno A que frequenta uma disciplina B, a relação com origem em A e destino em B é "frequenta", enquanto que a relação com origem em B e destino em A é "é frequentada". Na verdade, estas relações são definidas entre as classes, mas têm efeito sobre os indivíduos (instâncias) das mesmas.

- Entidades: são utilizadas para representar a individualidade dentro de uma ontologia. Ou seja, o aluno A, o trabalho Y ou a disciplina C. Por esse motivo, representam os dados propriamente ditos no contexto da ontologia. Cada instância é um registo pertencente a uma classe e com relações com outras instâncias, que podem formar axiomas.

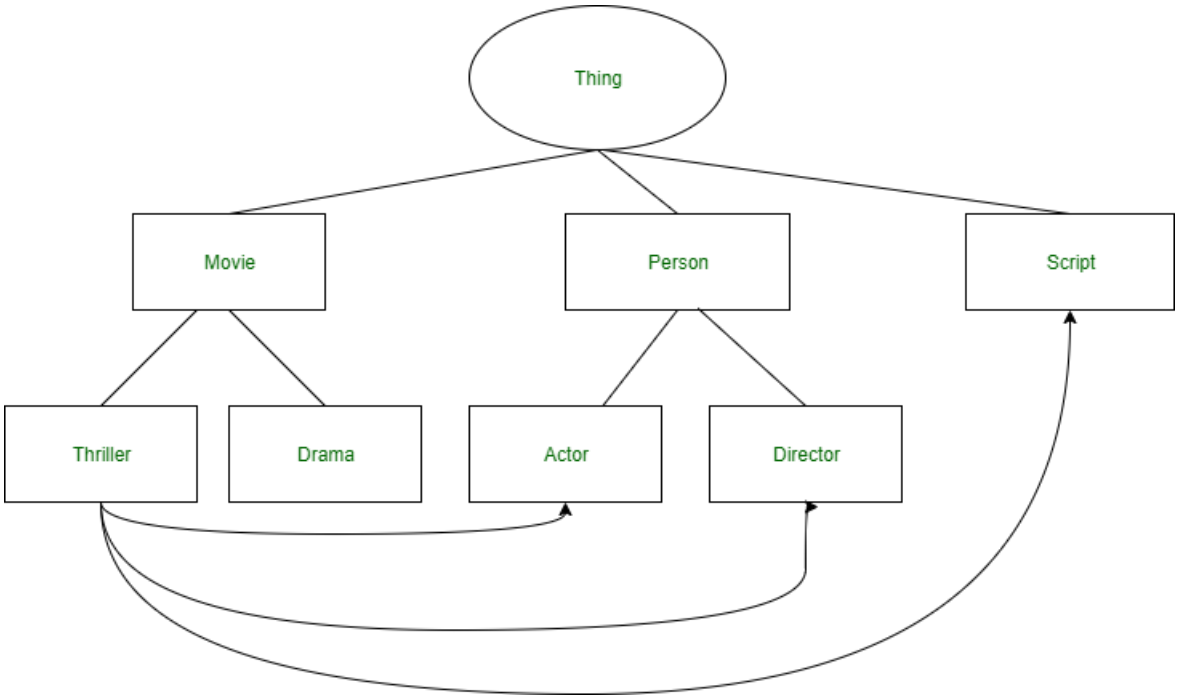

Figura 3.5: Conceitos de ontologias [55]

3.2.3 Metodologias para Construção de Ontologias

De maneira a chegarmos aos componentes necessários para a composição de uma ontologia (axiomas, funções, relações e entidades), é necessário que haja uma estrutura bem definida de como montá-la. Esta secção aborda as metodologias e fases de construção de uma ontologia.

As ontologias podem ser construídas através de três metodologias: abordagem de ontologia única, abordagem de ontologias múltiplas e abordagem de ontologia híbrida [56].

- Abordagem de ontologia única: neste caso, é usada uma única ontologia para todas as fontes de informação. Assim, o vocabulário e terminologia são partilhados tendo em vista a especificação semântica. Tem a limitação de não permitir a construção de ontologias individuais [56] e é normalmente utilizada para a construção de sistemas genéricos (imagem 3.6).

- Abordagem de ontologias múltiplas: neste método, cada fonte de informação tem a sua própria ontologia associada. É uma abordagem oposta à de ontologia única, porque se baseia na individualidade e na independência e não na globalidade. Cada ontologia é algo isolado (imagem 3.7) [57].

- Abordagem de ontologia híbrida: a semântica de cada ontologia é baseada na sua própria fonte de informação. Mas também existe, à semlhança da abordagem de ontologia única, um vocabulário global, de forma a que as ontologias não estejam alheadas das outras e separadas por contextos diversos (imagem 3.8) [57].

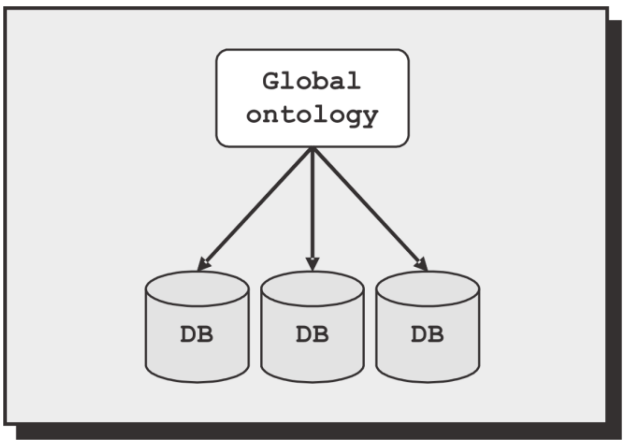

Figura 3.6: Abordagem de ontologia única [58]

Em termos gerais, a construção de uma ontologia envolve seis passos básicos:

- Definição do escopo: nesta fase, existem dois objetivos: a identificação da necessidade e do propósito para a construção da ontologia. Existe a preocupação de definir se a ontologia deve ser construída de raíz ou se devemos reutilizar uma ontologia já existente. São identificadas as questões que a ontologia deve abranger.

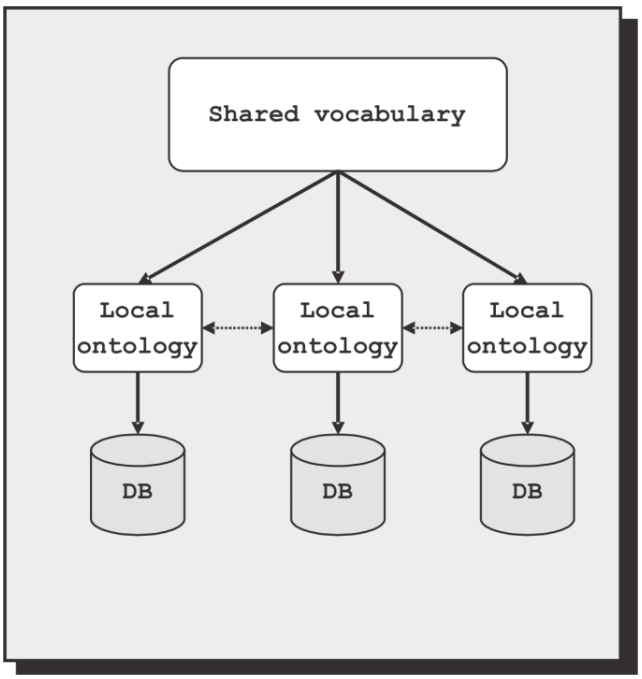

Figura 3.7: Abordagem de ontologias múltiplas [58]

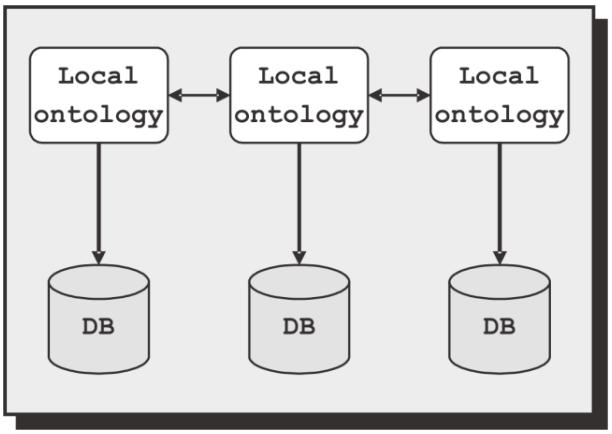

Figura 3.8: Abordagem de ontologia híbrida [58]

- Definição da captura: é a fase que corresponde à identificação de conceitos-chave e das relações enquadradas com o domínio de interesse. Os conceitos são modelados como classes ou sub-classes e entre eles existem relações.

- Codificação: refere-se à escolha de uma linguagem de representação da ontologia.

Linguagens como Simple HTML Ontology Extensions (SHOE), Ontology Web Language (OWL) ou RDF são exemplos disso. Aqui, as ideias são transformadas em código entendível pelos sistemas informáticos, o que retira a ambiguidade.

- Integração: trata-se de combinar a ontologia construída com aquela que já é existente, no caso de ser aplicável.

- Avaliação: devem ser definidos critérios de avaliação, sejam eles genéricos ou específicos e a ontologia deve ser avaliada com base neles. No caso dos critérios genéricos, temos fatores como consistência ou reutilização. Já no caso dos critérios específicos, a ideia é haver um encurtamento do escopo para a finalidade específica da solução e para a verificação da correspondência com essa finalidade e capacidade de a cumprir.

- Documentação: é a última fase para a construção de uma ontologia. É importante para a reutilização da ontologia. Envolve colocar tudo no papel e registar explicitamente todas as especificações da solução proposta. É uma fase abrangente a todos os projetos de "software".

Estes passos podem ser encontrados em [59] e estão descritos visualmente na figura 3.9.

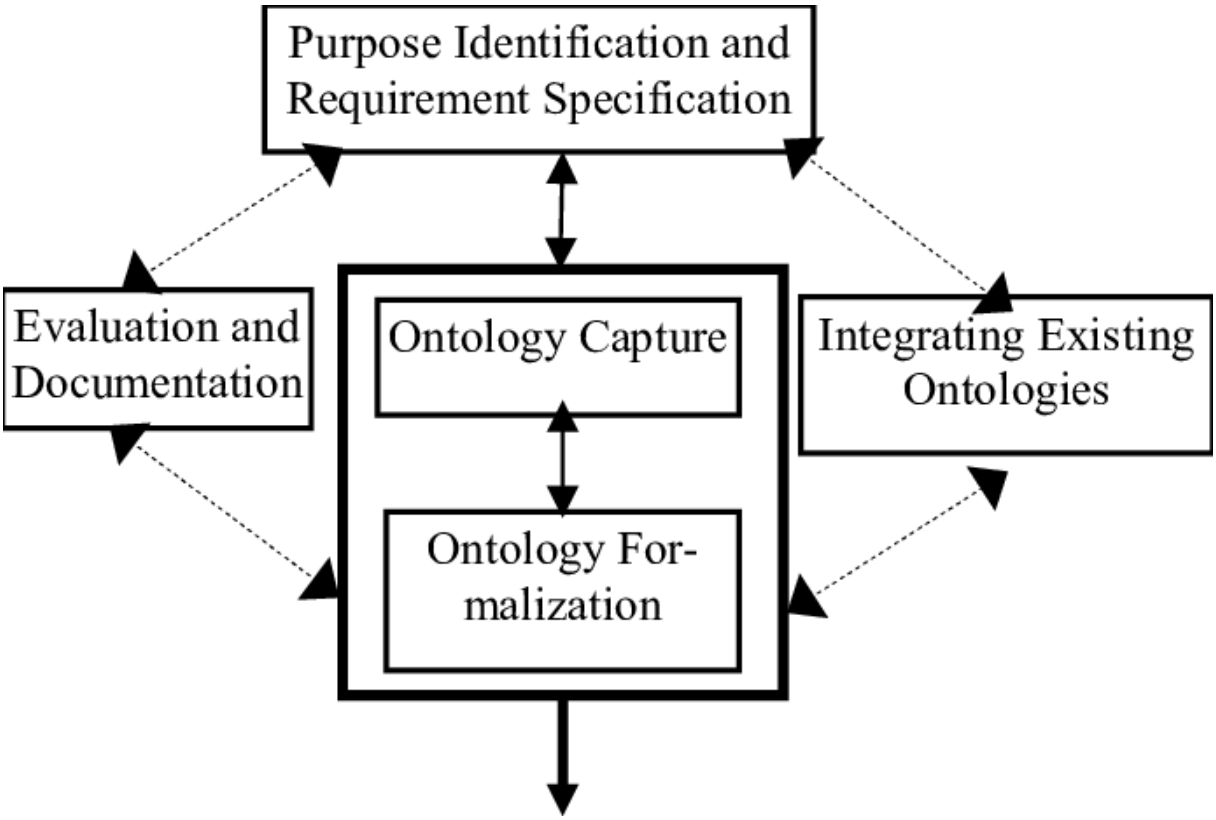

Figura 3.9: Processo de desenvolvimento de uma ontologia [60]

Capítulo 4

Tecnologias e Ambiente de Desenvolvimento

No presente capítulo, é feita uma abordagem sobre as alternativas tecnológicas no que toca à gestão de bases de dados baseadas em grafos (4.1). Posteriormente e relacionadas a esta secção, dando-lhe continuidade, serão apresentadas em profundidade quais as alternativas tecnológicas possíveis e a escolhida para este trabalho(4.2). Em 4.3, é demonstrado como instalar e configuar o ambiente de desenvolvimento.

4.1 Sistemas de Gestão de GDB

Um Data Base Management System (DBMS) ou, em português Sistema de Gestão de Bases de Dados (SGBD) tradicional é qualquer "software" com a capacidade de gerir uma ou mais bases de dados. Tem como objetivo providenciar facilidade de assesso, manipulção, eliminação e criação de dados (CRUD) e para isso é disponibilizada ao gestor uma interface gráfica de maneira a ser proporcionada uma interação com os dados "user-friendly" [61].

Este conceito também é transversal às GDB, sendo que existem várias alternativas para utilização no caso de querermos construir uma GDB e gerir os seus dados:

- Neo4j: trata-se de um SGBD desenvolvido pela Neo4j, Inc. Os seus desenvolvedores descrevem-no como uma "graph database" compatível com ACID, com o armazenamento e processamento de grafos nativo. Está disponível uma "community edition" "open source" licenciada pela GNU General Public License v3.0 (GPL3).

- Wikibase: é o software onde corre a "Wikidata". Tem como objetivo o fornecimento de uma base de dados editata colaborativamente, que oferece suporte para projetos famosos como a "Wikipédia". É otimizado para dados em forma de grafo, não para dados tabulares ou fortemente pré-estruturados.

- ArangoDB: consiste numa solução NoSQL "open-source" que possui um modelo de dados flexível para grafos, documentos e pares de chave-valor. Tem uma linguagem de consulta semelhante a Structured Query Language (SQL), chamada ArangoDB Query Language (AQL).

- Titan Database: é uma base de dados escalável que visa armazenar e permitir consultas a grafos que possuem um número enorme de vértices, que se encontram distribuídos ao longo de um "cluster" e múltiplas máquinas.

- RedisGraph: é uma "property database" na forma de módulo para o Redis[62]. Faz uso de álgebra linear em matrizes de adjacência para implementar operações sobre os grafos. É possível usar "queries" "OpenCypher".

- Bitsy Graph Database: implementa a Application Programming Interface (API) "Blueprint" [63]. Tem como características a sua pequena dimensão, rapidez e capacidade de ser incorporável. Suporta transições ACID.

- Orient DB: tem como característica destacada a capacidade de gravar 220000 registos por segundo em "hardware" comum. É "open source" e escrito em Java. É do tipo NoSQL.

- Nebula Graph: é uma GDB "open source" capaz de hospedar grafos muito grandes, em larga escala, com milhões de vértices (ou "nodes") e um número também enorme

de relações entre os mesmos. Utilizado por empresas como a "Vivo" ou a "Huawei".

- Dgraph: as suas transações são ACID e tem como características mais realçáveis são a sua rapidez e o facto de ser uma GDB distribuída.

- AllegroGraph: é uma GDB distribuída horizontalmente, que suporta linguagens como SPARQL Protocol and RDF Query Language (SPARQL) ou Prolog e tem como objetivo permitir às empresas tomar boas decisões relativamente a determinados assuntos, abordados e transpostos para o KG, ajudando a sintetizar e atribuir significado a uma quantidade grande de dados [64].

- TigerGraph: é uma GDB escalável, sendo que inter-conecta um número gigante de vértices e arestas de um grafo, servindo assim para o uso da parte de grandes empresas. Muitos bancos utilizam esta tecnologia de forma a detetar atividades como a fraude bancária.

- Trinity Graph Engine: Graph Engine (GE) é um sistema de grafos abragente, existente numa "memory cloud". Os registos são guardados em pares de chaves-valor, distribuídos ao longo de um "cluster" de máquinas. Suporta grafos enormes, na escala dos biliões de "nodes" [65].

- RecallGraph: guarda os registos de forma versionada isto é, é possível verificar as mudanças de estado ou características que mudaram nos dados durante a sua existência até ao seu ponto atual. É possível fazer uma "query" ao objeto do passado e do presente [66].

- Graph Commons: é uma ferramenta que permite construir grafos, desenhando-os de forma colaborativa e inserindo colaborativamente as informações e relações necessárias.

- Amazon Neptune: propriedade da Amazon, é uma GDB comercial, que suporta linguagens como Java, Go, JavaScript, PHP: Hypertext Preprocessor (PHP), Python,

Ruby ou Scala. Tem como objetivos ter uma grande disponibilidade e durabilidade, computando um grande número de registos, se necessário, em simultâneo.

Estas alternativas podem ser encontradas em [67].

Neste trabalho, era necessária a escolha de um DBMS de entre estas opções, que pudesse enquadrar-se no problema, estar documentado, ter facilidade de uso e que oferecesse características que, em conjunto, fossem superiores às dos outros sistemas semelhantes quando contextualizadas com o presente trabalho.

Posto isto, de todas as ferramentas referidas, havia três candidatas: "TigerGraph", porque tem um "background" muito interessante no que toca à deteção de fraudes noutro contexto, "Neo4j", pela facilidade de uso e pelas suas bibliotecas e "Amazon Neptune", pela sua consistência e rapidez.

A escolha recaiu sobre a ferramenta "Neo4j", pelos seguintes motivos:

- O seu suporte a API's e outros métodos de acesso: enquanto que a "Amazon Neptune" e a "TigerGraph" têm um suporte bastante específico, com SPARQL e RDF 1.1 a serem suportados pela "Amazon Neptune" e TygerGraph Query Language (GSQL), Kafka e RESTful HTTP/JSON API pela "TygerGraph", a ferramenta "Neo4j" providencia-nos uma quantidade de métodos de acesso muito maior e mais abrangente: suporta o protocolo "Bolt", a Cypher Query Language, dá suporte a Java API, entre outras.

- Número de sistemas operativos: enquanto que no caso da ferramenta propriedade da "Amazon", tudo é feito com "host" na sua "cloud", a TygerGraph apenas é suportada em sistemas "Linux". Já no caso do "Neo4j", o suporte existe em quatro diferentes sistemas operativos, "Linux", "Windows", "OS X" e "Solaris".

- O facto de ser "open source": "Neo4j" não precisa de licença e é gratuita para trabalhar, enquanto que as outras duas ferramentas são licenciadas e, por isso, as suas funcionalidades não conseguem ser potenciadas ao máximo, pois são limitadas a pagamentos mediante consumo.

4.2 Neo4j

Abordadas as diferentes alternativas de DBMS neste contexto, numa descrição sintetizada e abrangente e escolhida aquela a ser usada, esta secção tem como função o fornecimento de uma visão mais pormenorizada daquilo que é o "Neo4j", passando por uma abordagem às suas características, benefícios e funcionalidades.

A ferramenta "Neo4j" foi construída usando a linguagem Java e lançada no ano de 2007 [68], sendo que os adjetivos escalável e não-esquemático (NoSQL) são aplicáveis relativamente à sua caracterização.

Esta tecnologia, que é líder em termos de "open-source" [69] a nível global no âmbito das GDB, tem claras vantagens decorrentes do seu uso:

- Providencia resultados em tempo real, com os resultados a serem baseados em dados, também eles em tempo real.
- Está acessível em qualquer instante, caracterizando-se por uma alta disponibilidade e por isso pode garantir as operações CRUD de modelos bastante grandes de forma consistente em tempo real.
- O seu modelo de dados é bastante simples de entender, por forma a começar a ser trabalhado e também é simples modificar a solução apresentada em qualquer instante, sem comprometer aquilo que já foi feito nem a sua consistência (do modelo).
- Os dados semi-estruturados e não-estruturados podem ser representados com facilidade no "Neo4j", algo muito importante para qualquer projeto que consista na implementação basilar de um mecanismo de inferência, como é o caso deste.
- Quando comparando "Neo4j" com outras bases de dados, retira-se à conclusão de que se trata de uma GDB onde podemos navegar entre "nodes" e arestas muito rapidamente [69].
- Cypher Query Language (CQL) - ou simplesmente "Cypher" - é uma linguagem muito poderosa e de bastante alto nível, o que representa uma vantagem no que

toca a fazermos “queries“ à base de dados.

- É fácil retornar instâncias conectadas a outras instâncias sem precisarmos de fazer uma “query“ de alta complexidade, como seria necessário num esquema relacional (através de um ou mais “joins“, por exemplo).

Um exemplo de utilização da linguagem é o seguinte:

```
MATCH (tiago:Student)-[:FRIENDS*2]->(fof)
WHERE tiago.name = "Tiago Santos"
AND NOT((tiago)-[:FRIENDS]-(fof))
RETURN DISTINCT fof.name;
```

Em “Cypher“, a instrução “MATCH“ procura selecionar um determinado conjunto de dados da BD. Neste caso, a “query“ representa um qualquer estudante (variável “tiago“) que tenha uma relação de amizade (”friends”) em segundo grau (daí o *2 depois de “friends“) na direção (representada, em “Cypher“ pela direção da seta) de algum outro “node“ (“fof“). Depois, é utilizada a instrução “WHERE“ para filtrar dentro de todos os estudantes (“tiago“) aqueles que têm o nome especificado “Tiago Santos“ e não sejam (“AND NOT“) amigos de primeiro grau com o mesmo. Por fim, são retornados todos os amigos em segundo grau (ou seja, amigos de amigos) do estudante em específico, pelo seu nome (“fof.name“).

A ferramenta “Neo4j“ possui um modelo nativo usando grafos, com “nodes“ e relações a serem guardados como um par de chave-valor, chamado de “propriedade“. Estas propriedades podem ser removidas para uma determinada instância, porque o esquema não tem uma estrutura fixa. Por exemplo, um “node“ do tipo (ou seja, “label“) aluno A, pode ter a si associada a propriedade de chave “trabalhador-estudante“ com o valor a “sim“ e o aluno B pode não ter nada associado relativamente a essa propriedade. Se quisermos apenas retornar os alunos que são trabalhadores-estudantes, poderemos fazer uma “query“ e retornar os alunos que possuem essa propriedade associada. São assim, aplicados os conceitos de GDB (3.1.2). A lógica pode ser visualizada na figura 4.1.

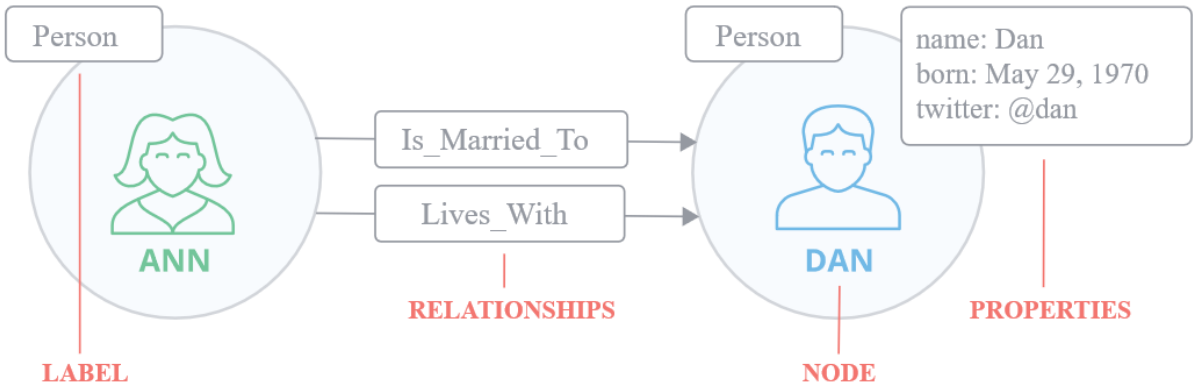

Figura 4.1: Neo4j e os conceitos aplicados de GDB [70]

Outra das funcionalidades a destacar do "Neo4j" é o seu "browser", no qual podemos ver graficamente o grafo construído e podemos, a partir dessa interface, consultar de maneira pormenorizada e ter uma noção esquemática dos dados presentes, seja no esquema a nível dos metadados (estrutural) ou nos dados que o populam. Essas consultas podem ser feitas através de "Cypher", tal como é exemplificado na figura 4.2.

O Neo4j possui ainda bibliotecas que ajudam em termos de funcionalidades e de facilitar a consulta dos dados, entre outros, como é o caso da Awesome Procedures on Cypher (APOC), criada com o objetivo de ser uma biblioteca utilitária para "procedures" e funções.

Sendo uma das maiores bibliotecas e também a mais usada, inclui mais de 450 "procedures" standard que providenciam funcionalidades para atualizações de grafos, conversões, entre outros [72]. Estas instruções podem ser incluidas nas queries "Cypher", simplificando muitas vezes o código.

4.3 Instalação do Neo4j

Passando da teórica à prática, esta secção tem como objetivo a demonstração dos passos para instalar o "Neo4j" e ainda alguns exemplos básicos do uso da ferramenta, bem como das suas bibliotecas.

Em primeiro lugar, devemos aceder ao portal de "download" da ferramenta, cujo "link"

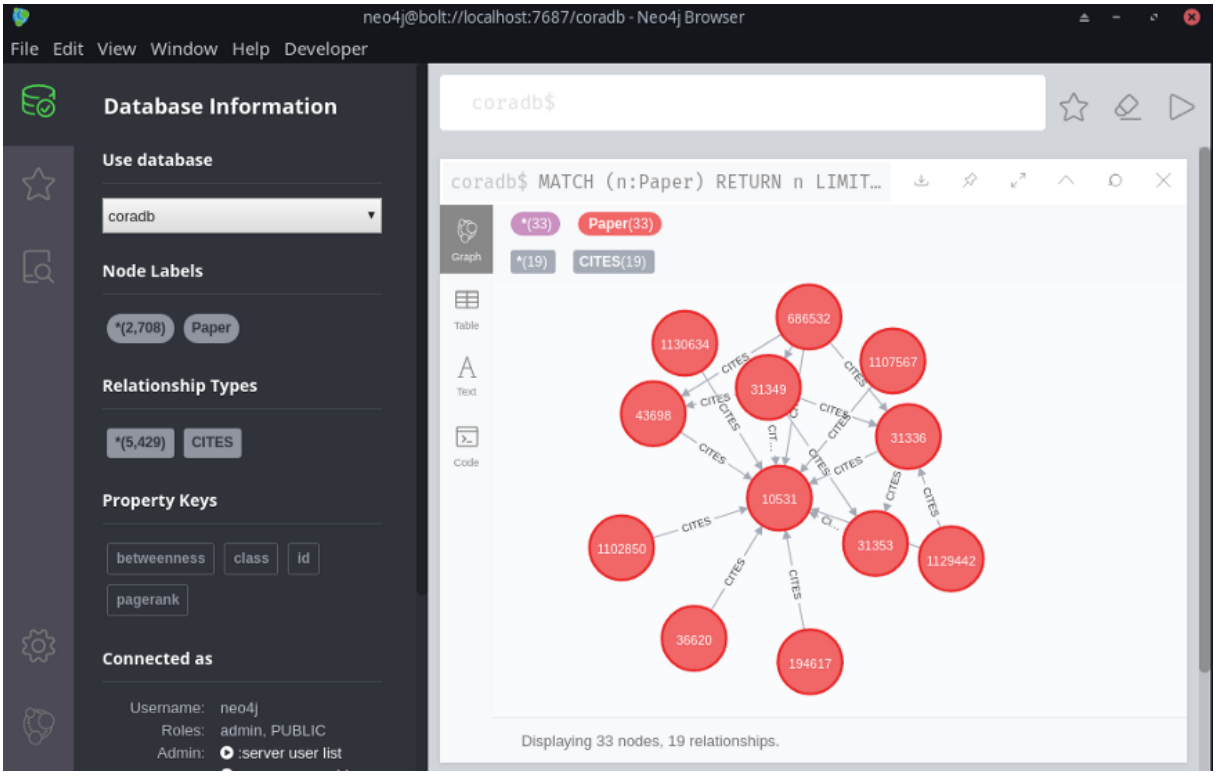

Figura 4.2: "Browser" do Neo4j [71]

é https://neo4j.com/download-neo4j-now/ e preencher com as informações necessárias o formulário presente no lado direito do ecrã.

De seguida, devemos clicar no botão "Download Neo4j Now". O sistema operativo e especificações do computador em causa são automaticamente preenchidas e o "download" é iniciado.

Uma vez terminado o "download", devemos carregar no executável, que abre uma janela (figura 4.3) onde é feita a configuração da instalação que pretendemos.

É escolhido o local pretendido para a instalação e aguarda-se pela conclusão da mesma. Uma vez concluída, o Neo4j pode ser executado (figura 4.4).

Quando a aplicação inicia, existe a possibilidade de iniciar um projeto novo (figura 4.5).

Uma vez criado o projeto, podem ser adicionados "plugins" (ou bibliotecas) (figura 4.6), tais como a APOC e a Graph Data Science (GDS) e criadas as "graph databases"

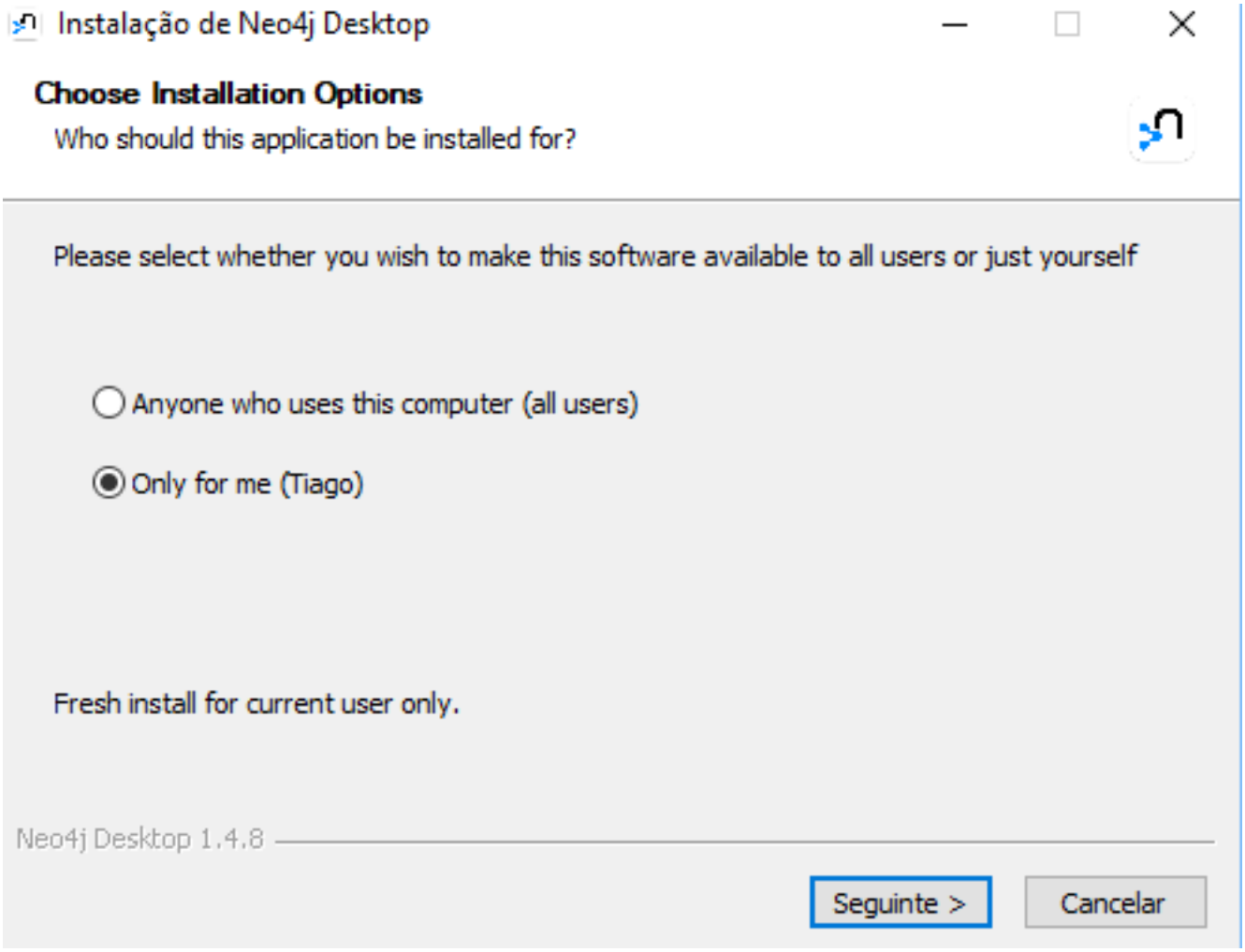

Figura 4.3: Janela de instalação do Neo4j

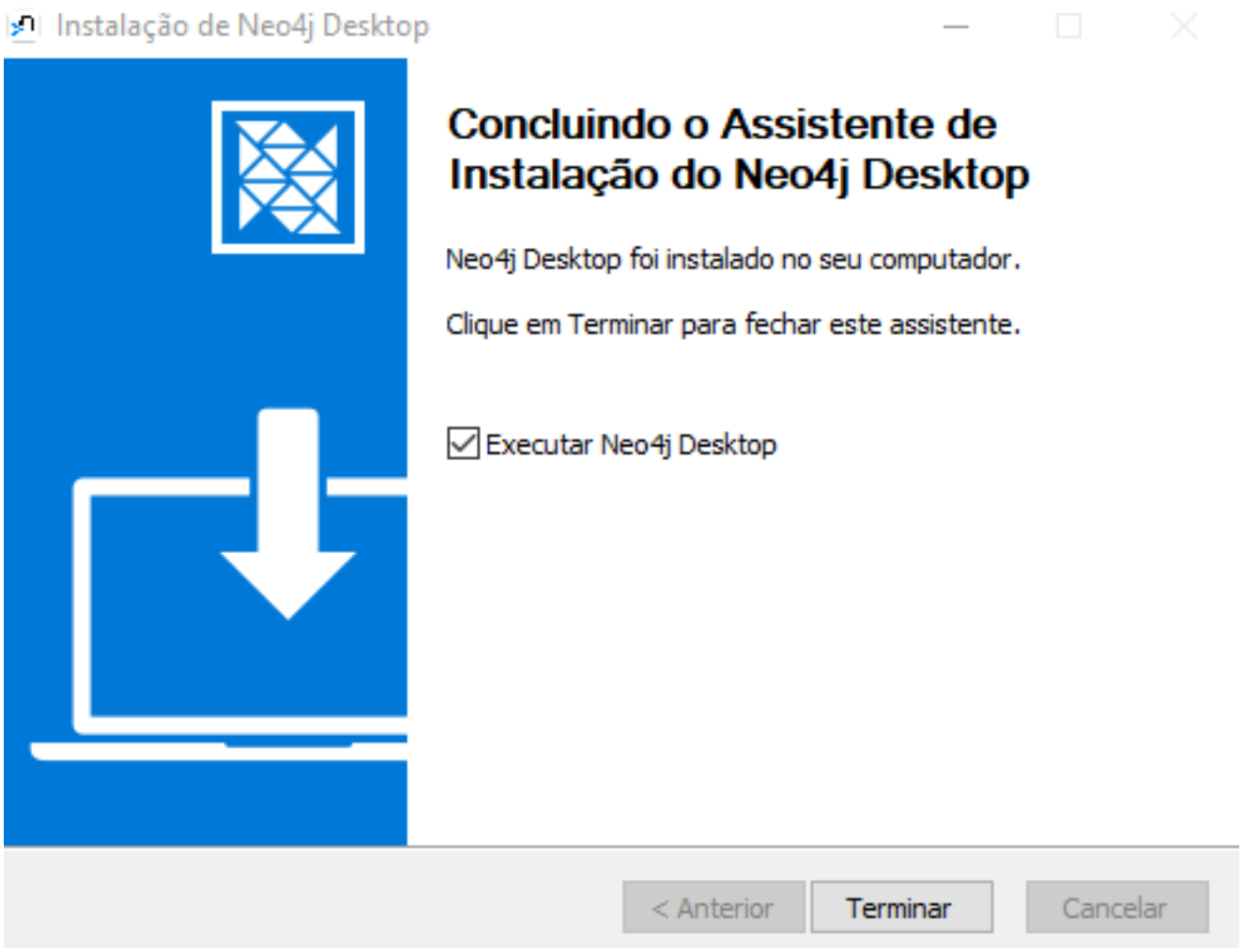

Figura 4.4: Terminada instalação do Neo4j

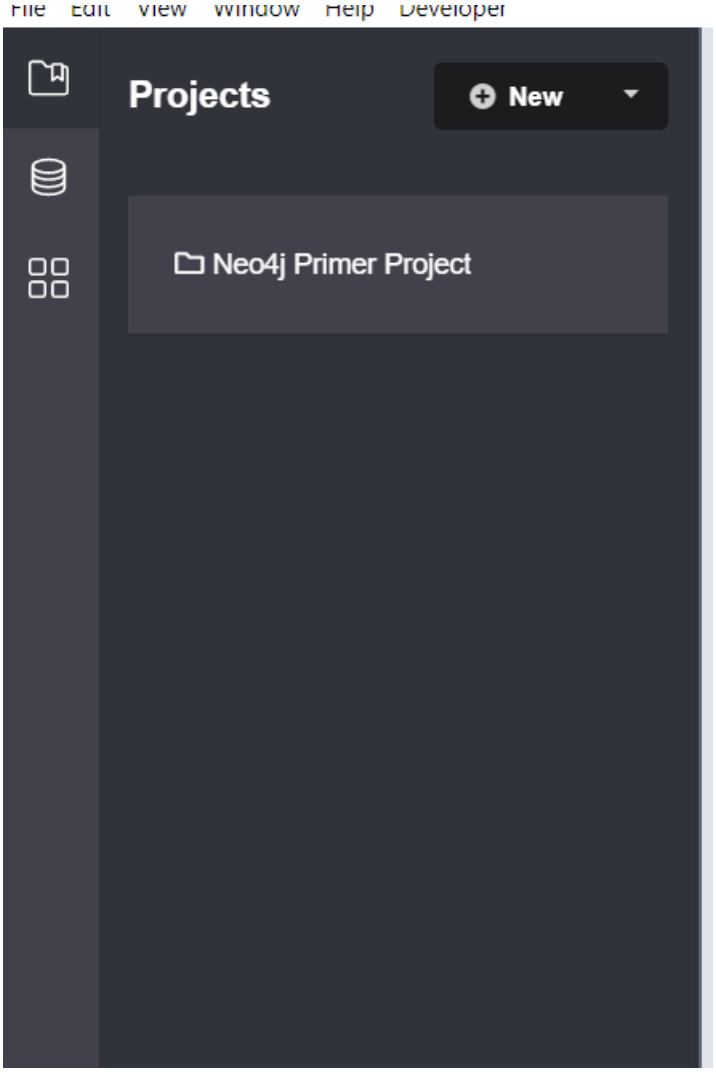

Figura 4.5: Criação de um novo projeto

pretendidas (figura 4.7).

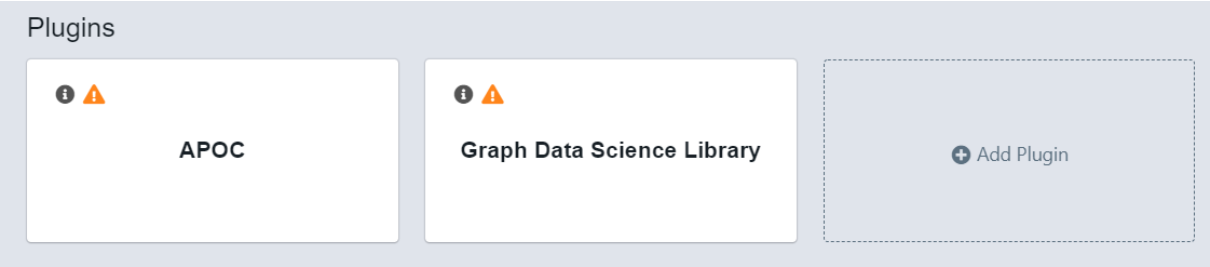

Figura 4.6: Adicionar Plugins

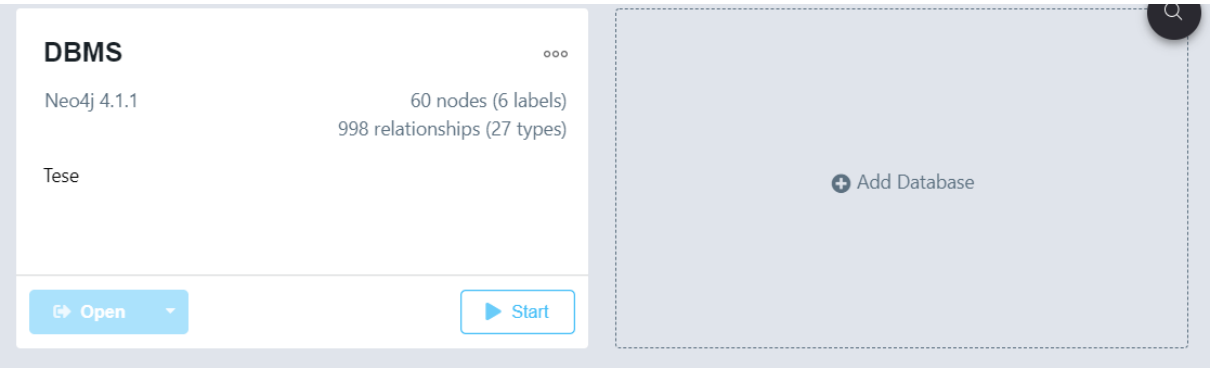

Figura 4.7: Adicionar uma “graph database“

Capítulo 5

Implementação

Nos últimos anos, as atividades fraudulentas estão a aumentar exponencialmente em todo o mundo [73], sendo que as escolas não são exceção a essa tendência [73].

O problema é emergente e passa a exigir soluções reais de inferência sobre como alguns comportamentos ou algum conjunto de variáveis, que quando conjugados entre si, podem ser um grande indicador de atividades fraudulentas como plágio, trapaça em exames, cópias de trabalhos e até a existência de alunos a dizer aos colegas as respostas corretas durante um teste de forma ilícita.

No presente capítulo, é apresentada uma contextualização, tanto histórica como das várias alternativas de necessária consideração tendo em conta o problema (formas possíveis de fraude), seguida da construção de uma ontologia com o objetivo de ser a base para a deteção de possíveis relações que sejam identificadas como de possível fraude. Será explicada a abordagem e as alterações à mesma, comparando resultados após a aplicação de fórmulas, sob a forma de algoritmos, à medida que a ontologia é melhorada. Não existe um conjunto de dados reais passíveis de serem utilizados, pelo que os dados foram criados manualmente e apenas como propósito de teste para ser possível a utilização dos referidos algoritmos.

No final do capítulo, são apresentados os resultados e discutida a sua importância, sendo isto um mecanismo importante para serem obtidos indicadores do facto de o caminho escolhido ter sido, ou não, o certo, chegando assim ao término a dissertação.

5.1 Contextualização

O conceito de fraude está presente ao longo de nossa história enquanto seres humanos desde a fundação das sociedades primordiais. A primeira fraude famosa ou bem conhecida foi registada na Grécia Antiga, no ano 300 A.C. [74]. Esta tentativa inicial foi protagonizada por um marinheiro de nome Hegestratos, que pretendia fazer um seguro para o seu navio e a carga contida nele.

Naquela época, a artimanha era bem conhecida e batizada com o nome (em inglês) "bottomry". A estratégia passava pelo comerciante pedir dinheiro emprestado no valor equivalente ao seu navio e carga. Se o navio chegasse seguro ao seu destino (com a carga intacta), o empréstimo era pago com juros. Se um empréstimo, nessas condições de segurança, não fosse pago, haveria a reintegração de posse do barco e a correspondente carga.

No caso de Hegestratos, ele tentou, naquela que possivelmente é a primeira tentativa de fraude registada, afundar o seu navio vazio e vender a sua carga, enquanto simultaneamente receberia o empréstimo, para desta forma conseguir receber o dobro do dinheiro suposto pelo conteúdo que o navio transportava.

Esta primeira tentativa não teve sucesso porque Hegestratos foi apanhado enquanto afundava o seu barco pela sua própria tripulação, foi expulso do navio e acabou por se afogar na sua tentativa de escapar da sua própria tripulação.

A partir daqui foi aberto um precedente e as atividades fraudulentas foram crescendo e expandindo-se a todas as áreas, com os malfeitores a conseguirem ter cada vez mais formas originais de realizarem as atividades mal intencionadas.

Assim como acontece em qualquer tipo de atividade, a desonestidade humana começa desde cedo e neste caso, a fraude académica começou logo nas primeiras provas, nos exames civis que decorriam da China, através da prática de enganar e subornar os examinadores.

No início do século XX, foi estimado [75] que dois terços de todos os alunos cometeram ilegalidades a nível académico, em algum momento dos seus percursos. Normalmente, há grupos de alunos que criam laços fortes com o passar dos anos e integram cada dia mais

alunos ao seu grupo, o que pode criar uma rede de interesses, com tendência em alguns casos à formação de uma rede de interesses e de ajuda mútua.

5.2 Fraudes académicas

Um exemplo de entreajuda entre estudantes consiste em algo chamado "essay banks", ou em português "bancos de ensaio", que as fraternidades costumavam construir e que eram criados com o objetivo de alunos diferentes em anos diferentes, apresentarem trabalhos copiados uns dos outros, iguais, mudando apenas o nome no papel [76].

Existem diversos métodos de fraude relacionados às instituições de ensino, como subornar, que consiste no ato de "oferecer algo, geralmente dinheiro ou bens, para que se faça ou consiga uma coisa ilegal" [77], enganar, que consiste no ato de mentir aos professres/supervisores em relação a um exercício académico (por exemplo, alegar ter submetido um trabalho quando isso não é verdade), roubo de identidade, que é o acto de alguém fazer um exame no lugar de uma suposta pessoa que deveria estar a fazê-lo, plágio, que é o acto de copiar um trabalho já existente e pertencente a outro indivíduo (um dos tipos mais comuns de fraude académica) e também copiar, de diversas formas, em testes.

O acto de trapacear pode assumir várias formas, como a de tirar notas pessoais ilegais de provas, copiar de uma prova de alguém, com ou sem o consentimento da outra pessoa, ter acesso a informações proibidas através dos seus telemóveis pessoais ou ainda a de dois ou mais alunos compartilharem informações durante uma atividade de avaliação. Muitas técnicas de copiar foram desenvolvidas ao longo da história do ensino, fazendo uso muitas vezes de relações entre alunos/amigos. Outro exemplo é a sinalização das respostas certas através de sinais ou outro tipo de troca de informação, como a passagem de folhas de rascunho.

Existem casos de fraude em que as pessoas podem estar envolvidas nela sem serem beneficiadas pelo acto de participar no esquema. Alguns exemplos são alunos a passar informações a outros outros por meio de um diálogo durante um exame, ou a transmissão de informações pela via de um telemóvel, dando ao beneficiado as respostas corretas, ou

ainda trocas de informações quando os exames são agendados em horários diferentes, mas consecutivos, nos quais os alunos podem ter tempo de sair da sala e informar os outros sobre as questões presentes na atividade de avaliação.

5.3 Construção do modelo de representação do conhecimento

O foco principal deste trabalho é caminhar na direção de uma padronização em termos de comportamentos comuns que podem levar-nos a ter suspeitas de ações fraudulentas entre dois ou mais alunos. Pode ser importante para nós saber o nível de confiança entre cada um dos alunos. Por exemplo, se se conhecem circunstancialmente, se são amigos e o quão amigos são ou mesmo se são namorados/noivos. O facto de frequentarem as mesmas disciplinas ou em que grupos se encontravam durante as avaliações pode ser outro elemento importante a ter em conta.

Uma rede de influências pode crescer a partir de relacionamentos entre as pessoas e se um grande número de pessoas se conhece mutuamente, torna-se grande a probabilidade de um ou mais relacionamentos suspeitos nascerem e serem criados, aumentando cada vez mais a probabilidade de existirem indivíduos na comunidade que necessitarão de benefícios e por consequência serão beneficiados por um eventual esquema de fraude.

Com base nos conceitos apresentados, é obtida uma perspectiva geral do problema e foi construído um modelo de dados esquemático, com o objetivo de traduzir o domínio do problema, através de metadados. Isto resultou na construção de uma "graph database", ajustada ao longo do tempo, na qual são inseridos dados manualmente e fictícios, por forma a testar a aplicação de algoritmos e verificar a diferença em diferentes fases no que toca à aproximação de uma base para solucionar o problema proposto.

Estudantes

A primeira coisa a ser definida foram os tipos de entidade, definidos por um "label" e as relações que cada um deles tem, seja com os do próprio tipo ou com aqueles que possuem tipos diferentes. Logicamente, este problema gira em torno da deteção de possíveis casos de fraude académica. Por isso, o primeiro tipo de "nodes" é denominado "Student" (a nomenclatura é toda em inglês por motivos de trabalho futuro, porque a ontologia fica mais compatível para ser entendida por pessoas de todo o Mundo). Não existe fraude académica sem a intervenção de estudantes interessados em beneficiar de tal.

Cada "node" do tipo "Student" possui a si associadas as propriedades "age" e "number", bem como o nome, presente no "label" de cada "node" deste tipo. O número representa o número mecanográfico e é do tipo "string". Por exemplo, o estudante "TS" tem o número mecanográfico "a28248", que serve para o distinguir no contexto escolar dos outros todos.

As relações entre estudantes são fundamentais e parte integrante deste trabalho, sendo que há algumas formas como um estudante pode relacionar-se com outro:

- Através de relações de conhecimento/ amizade.
- Serem namorados/ parceiros um do outro.
- Através de relações apontadas como possíveis de fraude, conceptuais neste esquema.

Estas relações são denominadas na "graph database" como "friends", "engaged" e "fraud", respetivamente.

Uma relação de amizade tem características ("properties") como a data em que foi formada (esta propriedade tem o nome "since"), que representa o ano em que a amizade foi formada (se for possível saber) e o nível de importância que essa relação tem entre os dois alunos. Esta importância pode ser definida por um peso, "weight", que diferirá conforme o nível de afinidade de um estudante com o outro. O peso é representado por um inteiro e o valor vai de 0 a 3.

No caso da relação “engaged“, esse peso é o máximo, pois a afinidade e o nível de confiança entre eles também é grande.

Estas relações podem ser importantes para os alunos formarem grupos de trabalho, com pessoas de confiança. Uma das partes poderá usar a outra de maneira a obter resultados escolares, resultando numa relação de fraude.

Disciplinas

Os estudantes estão inscritos e frequentam disciplinas, o que significa que aqui teve de surgir outro “label“, que é referente a todas as disciplinas, de nome “Subject“. Como consequência, foi criada outra relação (com o nome da ação de frequentar em inglês, “Attends”), de forma a garantir que esses alunos estivessem sempre associados de forma objetiva às cadeiras nas quais estariam inscritos.

A relação entre estudantes e disciplinas, de maneira a ser relevante para o domínio do problema, precisava de um atributo que servisse de ferramenta para poder avaliar um pouco do interesse do aluno pela disciplina, “presences“, uma medida quantitativa para observar quantas aulas foram frequentadas por cada indivíduo.

Avaliações

Ao longo do curso das UC, existe o propósito de os alunos adquirirem conhecimentos que poderão ser resultantes na aprovação da disciplina no final do semestre. Para obter essa aprovação, os alunos são submetidos a avaliações. Essas avaliações geralmente consistem na realização de trabalhos práticos e testes. Os “labels“ para este tipo de node são, respetivamente, “Assignement“ e “Test“.

Devido à existência dos “labels“ referentes às avaliações, foram incluídas duas propriedades chave, pois é nestas atividades que o aluno pode realmente realizar a fraude académica. Cada trabalho prático, assim como cada teste, tem um peso definido (“weight“), que pode ser importante, pois quanto mais importante para a nota final for a avaliação, maior será a probabilidade de as pessoas tentarem obter resultados de forma ilícita.

Os testes podem ser feitos online ou presencialmente, o que também pode ser relevante porque é mais provável que um aluno copie num teste online, segundo acredita a grande maioria dos professores, como descrito em [78]. Outra característica que pode alterar a probabilidade de ocorrência de um acto fraudulento, é o facto de um teste ser de escolha múltipla ou teórico, por exemplo. Portanto, os elementos "labeled" "Test" possuem uma propriedade que avalia isso, denominada "type".

A relação "participated" surge pelo facto de um determinado estudante participar num teste e tem a si associada uma propriedade quantitativa, a nota ("mark"). Esta propriedade poderá ser importante quando comparada com a nota do grupo, pois se for muito inferior pode significar que o aluno não apresenta tantos conhecimentos como a média do seu grupo, o que pode eventualmente ser um indicador de que o aluno não participou ativamente nos trabalhos ou mesmo que precisou de utilizar métodos ilícitos por meio a passar no teste.

Muitos testes possuem nota mínima e como consequência disso é preciso conhecimentos de maneira a obter aprovação. Poderá ser interessante perceber se esse facto poderá ter uma influência. Neste modelo inicial, a nota mínima é um atributo da relação "testof", que liga um teste ("Test") a uma disciplina("Subject"). Tem o nome "minimummark".

Grupos

Como o espírito de equipa é tão importante hoje em dia, tanto nos nossos empregos como em tarefas quotidianas, bem como no trabalho colaborativo, academicamente existe a intenção de simular, desde as bases, o que se passa nos nossos empregos, sob a forma de trabalhos de grupo.

As atribuições destes trabalhos normalmente passam pela formação de grupos. Por esse motivo, foi criado um "label" referente a grupos formados para a realização deste tipo de tarefas no esquema proposto. Esses "nodes" são conectados através de uma relação com aqueles que possuem a "label" "Student". Sendo assim, cada aluno pertence a um determinado grupo dentro do contexto de uma disciplina.

A relação "Grouped in" refere-se a um aluno que participa num grupo, ao passo

que a relação denominada “belongs to“ refere-se a um grupo que está relacionado a um “Assignement“. Nesta última relação, foi criado um atributo (“mark“) representativo de uma nota global para o grupo, sendo relevante numa possível comparação entre a nota do aluno no exame/ teste e a nota do grupo.

Outra propriedade considerada está relacionada à apresentação dos trabalhos e foi colocada no sistema com a intenção de representar um valor booleano, denominado “showed interest“. Se o comportamento de um aluno revelar que ele não demonstrou muito interesse em explicar o trabalho realizado [79], como por exemplo o acto de fugir a explicações por meio de deflação ou o de mudar o seu discurso padrão para algo mecanizado, isso pode indicar que ele não participou tanto quanto deveria ou que não participou de todo nessa atividade de avaliação.

Com estas entidades, atributos e relações, foi construído o modelo de metadados inicial para a construção desta ontologia, que pode ser visto na figura 5.1.

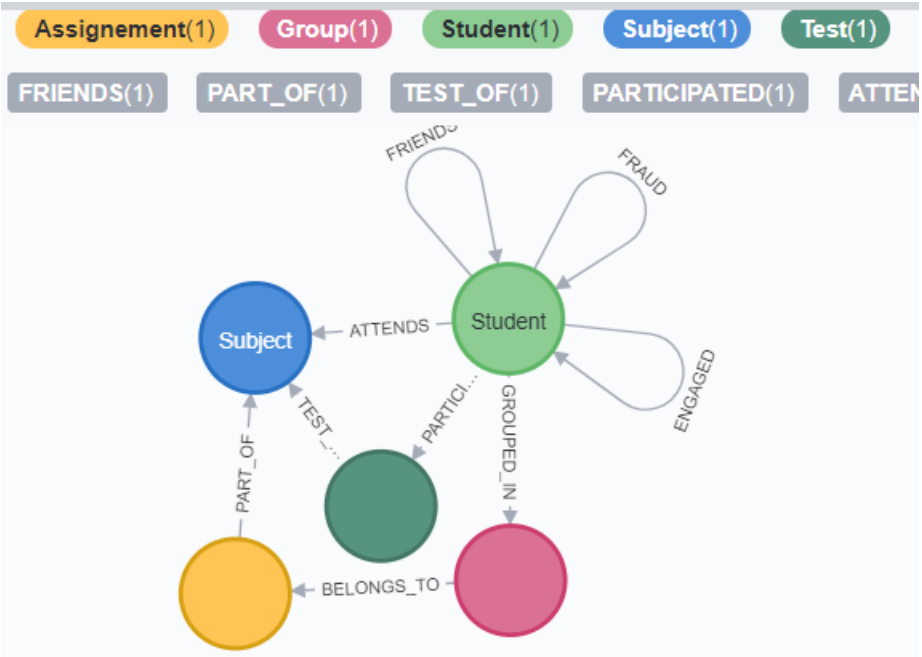

Figura 5.1: Modelo de metadados inicial

5.3.1 Data-set

Os “labels“ definidos em 5.3, assim como as relações entre eles resultam num conceito generalista de como os dados serão colocados e comunicarão, via instâncias, entre si. Assim, para o “label“ “Student“, foram criados vinte e um “nodes“, resultantes em noventa e seis relações, sendo oitenta e oito do tipo “Friends“ e duas do tipo “Engaged”, além da inserção de seis relações suspeitas de fraude, conceptuais, por forma a serem aplicados algoritmos que nos possam, aproximar de saber quais as relações mais significativas na deteção de uma fraude.

O “label“ “Subject“ contém três “nodes“ ou instâncias. Cada uma dessas instâncias, relacionadas a si, possui dois “nodes“ com o rótulo “Assignement“ e um do tipo “Test“ (ou seja, neste conjunto de dados existe, para cada disciplina, um teste e dois trabalhos práticos). Isto resulta, no final em seis trabalhos e três testes, dois teóricos e um de escolha múltipla.

Logicamente, cada trabalho contém grupos relacionados a ele e, neste contexto, foram inseridos quatro grupos por trabalho, o que resulta em 24 “nodes“ desse tipo.

Explicados os tipos de “node“ inseridos, bem como suas quantidades, na figura 5.2 o conjunto de dados inicial resultante é mostrado.

5.3.2 Aplicação de algoritmos ao conjunto inicial

Neste trabalho, também foi feito um esforço para obter valores mais objetivos tendo como recurso à biblioteca GDS do Neo4J.

GDS, é uma biblioteca que contém uma vasta gama de algoritmos para aplicar em GDB, divididos em três categorias: “quality production“, que são algoritmos robustos que “foram testados em relação à estabilidade e escalabilidade“ [80], “beta“ e “alfa“, que são menos robustos e foram menos testados, com alfa a corresponder ao nível mais experimental.

Esses algoritmos são divididos em várias categorias, incluindo centralidade, detecção

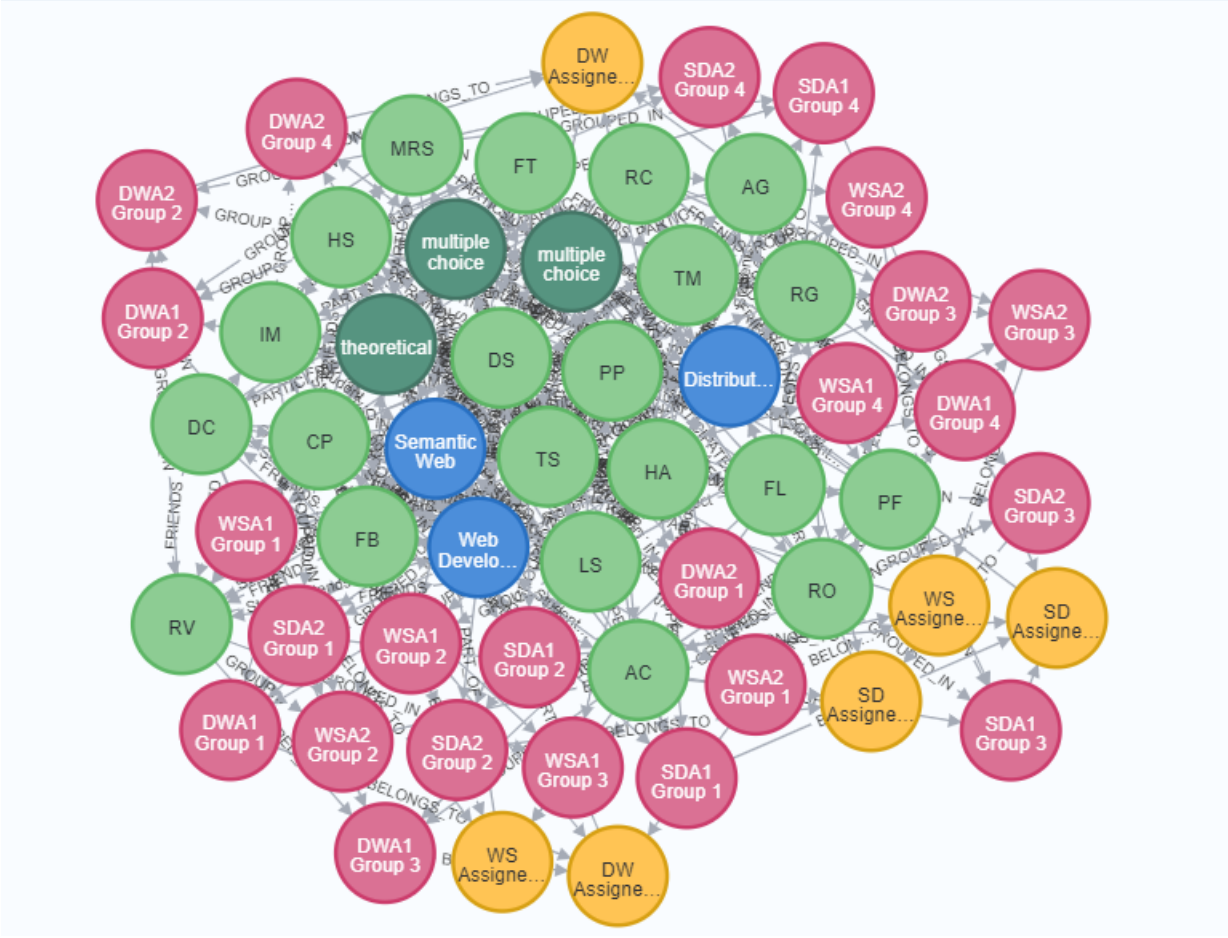

Figura 5.2: Conjunto de dados inicial

de comunidades, similaridade, “path location“ e “link prediction“. De seguida, são apresentados alguns exemplos de algoritmos utilizados, bem como a sua aplicação no presente problema.

Label Propagation

O primeiro algoritmo apresentado, foi utilizado com a intenção de encontrar comunidades dentro do conjunto de dados do problema, pois trata-se de um algoritmo de detecção das mesmas, que “detecta essas comunidades usando apenas uma estrutura de rede como guia, e não requer uma pré-função objetivo definida ou informação prévia sobre as comunidades“ [81], denominado “Label Propagation“.

O algoritmo tem a seguinte lógica:

- Cada “node“ é inicializado com um identificador exclusivo, sob a forma de um “label“ de comunidade.
- Estes “labels“ propagam-se pela rede.
- A cada iteração, cada “node“ atualiza o seu “label“ para um comum identificador da sua vizinhança.
- O algoritmo atinge a convergência no caso de um “node“ atingir o rótulo dos seus vizinhos ou caso exista uma instrução específica para essa paragem ao fim de um número determinado pelo utilizador de iterações.
- Os “nodes“ com o mesmo “label“, nesta fase, são considerados da mesma comunidade.

Esta lógica está explicada em [81] e é aprofundada em [82].

A primeira etapa para utilizar qualquer algoritmo que tenha qualidade de produção é chamar a instrução “gds.graph.create“, a fim de criar um sub-grafo que represente o que se deseja obter. Neste caso, pretendia-se obter todos os alunos que tivessem uma relação de “Friends“ com outros da mesma natureza, de uma forma não-orientada (ou seja, funciona simetricamente). Esse sub-grafo foi obtido da seguinte forma:

```
CALL gds.graph.create('weights', 'Student',
        {
        FRIENDS: {
        orientation: 'UNDIRECTED',
        properties: 'weight'
        }
        }
)
```

O próximo passo foi saber o número de comunidades e o número de alunos em cada uma delas. Para tal, foi necessária a instanciação do próprio algoritmo. Com base no subgrafo construído anteriormente, de nome "weights", criado anteriormente e na propriedade "weight", foi criada uma propriedade denominada "community", relacionada com cada "node" do tipo "Student".

```
CALL gds.labelPropagation.write(
    'weights',
    {
        relationshipWeightProperty: 'weight',
        maxIterations: 10,
        writeProperty: 'community'
    })
```

A última etapa referente ao algoritmo "Label Propagation", relativamente ao conjunto de dados inicial, foi realizar uma consulta à BD, a fim de saber quantos alunos ("s") faziam parte de cada comunidade, obtendo-se um "number of nodes".

```
match (s:Student)
with s.community as community, count(s)
as number_of_nodes
return community, number_of_nodes order
by number_of_nodes desc limit 50
```

Este algoritmo não produziu, nesta fase, resultados relevantes, sendo que apenas foram obtidas duas comunidades (tabela 5.1).

Propriedade	**Nº de comunidades**	**Elementos por comunidade**
weight	2	19, 2

Tabela 5.1: Resultado inicial da aplicação do algoritmo "Label Propagation"

Betweenness Centrality

O algoritmo tem a intenção de "detectar a quantidade de influência que um "node" tem sobre o fluxo de informações num grafo. É frequentemente usado para encontrar "nodes" que servem como ponte de uma parte de um grafo para outro"[83].

O algoritmo possui uma lógica baseada no cálculo dos caminhos mais curtos não ponderados entre os "nodes", em relação a todos os pares de "nodes". Isso dá a cada "node" uma pontuação calculada, com o mais alto sendo o "node" mais influente neste problema especificamente.

O significado disto é que aqueles alunos que possuam uma pontuação mais alta, poderão ser considerados membros com bastante influência na comunidade e, como tal, elementos fundamentais nas cadeias de transmissão de conteúdos.

Conforme indicado em 5.3.2, o primeiro passo foi criar um sub-grafo, considerando a relação de cada aluno com os outros e posteriormente fazer um cálculo através da chamada do algoritmo, anexando cada "Id" de cada "node" a uma pontuação, e associando essas duas variáveis a cada estudante. O último passo foi associar uma nova propriedade a cada instância com o "label" "Student", levando em consideração o sub-grafo criado com o nome "between1".

Isso foi conseguido desta forma:

```
CALL gds.graph.create('between1', 'Student',
    {FRIENDS: {orientation: 'UNDIRECTED'}})
```

```
CALL gds.betweenness.stream('between1')
YIELD nodeId, score
RETURN gds.util.asNode(nodeId).name
AS name, score
ORDER BY score DESC
```

```
CALL gds.betweenness.write('between1',
{writeProperty: 'friendsBetweenness'})
```

Os resultados podem ser consultados na tabela 5.2.

Nome	**Valor**
PP	25.177822177822176
HA	15.921220446220444
FB	15.035447885447885
RC	14.111263736263737
CP	10.226273726273728

Tabela 5.2: "Betweenness Centrality" aplicado ao modelo inicial

Page Rank

O segundo algoritmo de centralidade utilizado neste trabalho. Consiste em medir "a importância de cada "node" dentro do grafo, com base no número de relações de entrada e na importância dos "nodes" de origem correspondentes. A suposição subjacente a grosso modo é que uma página é tão importante quanto as páginas que ligam a ela." [84] e resolve a equação 5.3, onde é assumido que uma página A possui as páginas $T1$ a Tn que apontam para ela, d é um fator de amortecimento que pode ser definido entre 0 (inclusivo) e 1 (exclusivo), sendo que geralmente é definido como 0,85 e $C(A)$ é definido como o número de links que saem a partir da página A.

O objetivo neste domínio era obter os "nodes" mais influentes que possuem o "label" "Student", o que significaria que haveria uma grande possibilidade de os "nodes" com a

$$PR(A) = (1-d) + d(\frac{PR(T_1)}{C(T_1)} + ... + \frac{PR(T_n)}{C(T_n)})$$

Figura 5.3: Fórmula do algoritmo “page rank“ [84]

pontuação mais alta estarem vinculados a relações suspeitas.

A lógica foi semelhante àquela aplicada em 5.3.2, com a criação de um sub-grafo denominado “friends“ e a seguir efetivando a chamada dele, utilizando este algoritmo sobre o subconjunto de dados inseridos no sub-grafo. Mais uma vez, os “ids“ dos nodes foram levados em consideração, ordenando os registos pela sua pontuação e associando a si os nomes de cada aluno.

O código utilizado para a aolicação deste algoritmo neste caso em específico foi:

```
call gds.graph.create('friends', 'Student',
{
FRIENDS: {
    orientation: 'UNDIRECTED'
}
})

CALL gds.pageRank.stream('friends')
YIELD nodeId, score
RETURN gds.util.asNode(nodeId).name
AS name, score
ORDER BY score DESC LIMIT 10

CALL gds.pageRank.stream('friends')
YIELD nodeId, score AS pageRank
```

```
with gds.util.asNode(nodeId) AS n, pageRank
match (n)-[i:FRIENDS]-()
Return n.name AS name, pageRank, count(1)
AS degree, sum(i.weight) AS weightedDegree
ORDER BY pageRank DESC LIMIT 10

CALL gds.pageRank.write('friends',
{writeProperty: 'friendsPageRank'})
```

Os resultados podem ser vistos na tabela 5.3.

Nome	**Valor**
FB	1.4979643339291218
PP	1.4338711076881734
DS	1.2872454083990308
CP	1.2126396907959134
TS	1.1934854719787833

Tabela 5.3: "Page Rank" aplicado ao modelo inical

5.4 Ajustes ao modelo

Com o objetivo de obter algumas noções importantes, várias "queries" à BD foram realizadas. O primeiro objetivo foi perceber quais partes do conjunto de dados eram as mais interessantes de serem exploradas futuramente e como elas poderiam estar relacionadas entre si. Esta secção serve para explicar uma reformulação de partes do modelo, de maneira a conseguir obter uma aproximação de resultados que poderiam ser de interessante análise, inclusive para acrescentar mais propriedades ou relações à ontologia, reformulando-a e voltando a aplicar os algoritmos utilizados em 5.3.

Friend of a Friend

A primeira "query" relaciona quais alunos são amigos de amigos de um determinado aluno. Para isso, foi realizada uma "query" de teste.

Para obter os amigos dos amigos do aluno de nome "TS", a "query" construída (em "Cypher") foi:

```
MATCH (t:Student)-[:FRIENDS*2]->(fof)
WHERE t.name = "TS"
AND NOT((t)-[:FRIENDS]-(fof))
RETURN DISTINCT fof.name;
```

Isto resultou na criação da relação "FriendOfFriend" entre os "labels" "Student", de maneira a melhor detectar comunidades entre estudantes.

Para o aluno "TS":

```
MATCH (t:Student)-[:FRIENDS*2]->(fof)
WHERE t.name = "TS"
AND NOT((t)-[:FRIENDS]-(fof))
Merge (t)-[:FriendOfFriend]-(fof);
```

Para os restantes:

```
MATCH (nott:Student)-[:FRIENDS*2]->(fof)
WHERE NOT nott.name = "TS"
AND NOT((nott)-[:FRIENDS]-(fof))
Merge (nott)-[:FriendOfFriend]-(fof);
```

A instrução "merge" é utilizada para a criação de uma nova relação entre instâncias. Sessenta e duas relações novas foram criadas.

Grupos

Outra coisa que foi identificada como interessante para conhecer foram os grupos em que dois alunos estão juntos. Quanto maior o número, possivelmente maior será a probabilidade de um ato fraudulento. Observe-se que "a" é a variável com o "label" "Student", relacionada a "m", um grupo, que está relacionado a "d" (outro aluno). O resultado final é o retorno de um número descendente ("coun") de alunos que participaram no mesmo grupo (ver figura 5.4). A "query" construída foi:

```
MATCH (a:Student)-[:GROUPED_IN]-
(m:Group)-[:GROUPED_IN]-(d)
RETURN a.name, d.name, count(m) as coun
order by coun desc;
```

	a.name	d.name	coun
1	"TS"	"CP"	6
2	"TS"	"DS"	6
3	"TS"	"HA"	6
4	"HA"	"CP"	6
5	"HA"	"TS"	6

Figura 5.4: Resultado da "query" entre estudantes e grupos

Como consequência desta "query", achei que seria importante ter no modelo mais uma relação entre "Students", de maneira a ganhar robustez quanto aos seus relacionamentos uns com os outros. A relação nova inserida foi "InTheSameGroup" e possui a propriedade "numberOfTimes", que assume precisamente o valor de "coun".

Grupos relacionados a fraude

A terceira "query" é uma variante da anterior, agregando à "query" existente o conceito de fraude, de forma a associar o número de grupos que dois indivíduos tinham em comum e as relações declaradas manualmente como fraudes. De realçar que as variáveis "a", "m", "d" e "coun" têm o mesmo significado que tinham em 5.4, com o acréscimo da variável "f" que representa a relação "Fraud" e "frauds" que representa o número de fraudes. A "query" escrita foi:

```
MATCH (a:Student)-[:GROUPED_IN]->(m:Group)
-[:GROUPED_IN]-(d),
(a:Student)-[f:FRAUD]->(d)
RETURN a.name, d.name, count(m) as coun,
count(f) as frauds
order by frauds desc;
```

O resultado desta "query" pode ser consultado na figura 5.5. No entanto, não é conclusivo.

a.name	d.name	coun	frauds
"FT"	"IM"	6	6
"AG"	"PP"	3	3
"RC"	"PP"	1	1

Figura 5.5: Resultado da "query" entre "frauds" e "groups"

5.4.1 Reformular relações e propriedades

A ontologia inicial foi identificada como insuficiente para descrever o problema de uma forma clara, devido aos nomes de algumas relações serem muito ambíguos e também ao

facto de a quantidade de relações poder ser alargado e melhorado, podendo isso inclusive alterar os resultados quando os algoritmos são aplicados e ainda permitir utilizar os algoritmos aplicando-os às novas relações, dando uma ideia melhor e objetiva de estudantes que possam ser influentes nas suas comunidades.

A presente sub-secção tem como intuito explicar as transformações realizadas ao modelo de dados, bem como aos nomes atribuídos às relações iniciais. Na tabela 5.4 é possível verificar quais as alterações realizadas a essas nomenclaturas. O objetivo desta mudança foi, principalmente, tornar os nomes mais intuitivos para quem ler a ontologia conseguir perceber exatamente do que se trata cada uma das relações.

Relação inicial	**Relação atualizada**
(Assignement)-[:PartOf]->(Subject)	(Assignement)-[:AssignmentOfSubject]->(Subject)
(Test)-[:TestOf]->(Subject)	(Test)-[:TestOfSubject]->(Subject)
(Group)-[:BelongsTo]->(Assignement)	(Group)-[:GroupAssignment]->(Assignement)
(Student)-[:Attends]->(Subject)	(Student)-[:StudentAttendsSubject]->(Subject)
(Student)-[:GroupedIn]->(Group)	(Student)-[:StudentGroup]->(Group)
(Student)-[:Participated]->(Test)	(Student)-[:TookTest]->(Test)
(Student)-[:Friends]->(Student)	(Student)-[:Knows]->(Student)

Tabela 5.4: Atualização da nomenclatura das relações existentes.

Por forma a navegar o grafo com mais versatilidade e em ambas as direções, era importante criar relações correspondentes às que já estavam presentes no grafo, mas que significam coisas diferentes relativamente às já existentes. Por exemplo, um estudante ao pertencer a um grupo vai automaticamente significar que o grupo contém o estudante, formando uma relação "GroupStudent".

Novas entidades e relações

Uma falha que o modelo inicial tinha era que, em momento algum, os estudantes estavam diretamente ligados aos trabalhos, apenas aos grupos correspondentes. Como tal, foi criada essa relação porque era importante estar presente. Isto evita que, por exemplo, num motor de inferência ("reasoner") contido numa tecnologia como "Protegé", esta relação necessite de ser inferida, caso o meta-modelo seja "migrado" para essa ferramenta.

A relação foi denominada “StudentAssignment“.

Foram acrescentados “nodes“ para representar a existência de professores (com o “label“ “Teacher“) de cada disciplina (relação “teaches“), com a taxa de aprovação a ser uma propriedade com importância porque, logicamente, se uma disciplina com um determinado professor for mais difícil, poderá emergir uma necessidade maior de copiar, da parte dos estudantes. Os anos de experiência do professor também foram adicionados, sob a forma de propriedade nos “nodes“, com o nome “yearsOfExperience“.

Relativamente aos “nodes“ representativos das entidades referentes aos professores, existe uma ligação às avaliações (“evaluatedBy“) e aos estudantes, sendo que cada estudante vai ter vários professores e vice-versa, denominada “teacherOf“ (ou a inversa “StudentOf“).

Na figura 5.6 é possível ter uma perspectiva visual do modelo de dados final, enquanto que na tabela 5.5 se pode ver uma representação esquemática disso. Já na tabela 5.6 é possível ver os tipos de “nodes“ e quais os seus atributos.

5.4.2 Aplicação de algoritmos ao novo modelo de dados

Similarmente àquilo que foi feito em 5.3.2, o objetivo passou por aplicar os algoritmos “PageRank“, “Label Propagation“ e “Betweenness Centrality“, por forma a detectar comunidades e “nodes“ centrais no conjunto de dados, desta vez baseados em diferentes tipos de relações, que facultariam a possibilidade de chegar a conclusões mais objetivas sobre aqueles que teriam transversalmente mais influência no problema.

Como a lógica do código em “Cypher“ e a sua aplicação já foi demonstrada anteriormente no presente capítulo, no anexo E encontra-se o código para a criação dos sub-grafos e aplicação dos algoritmos.

O primeiro passo foi verificar se existia influência dos ajustes ao modelo naquilo que seriam as aplicações dos algoritmos às relações onde tinham sido anteriormente aplicados.

De seguida, foram aplicados os algoritmos a novas relações, para calcular a transversalidade da influência de um determinado “node“ no sistema. São mostrados, na forma de

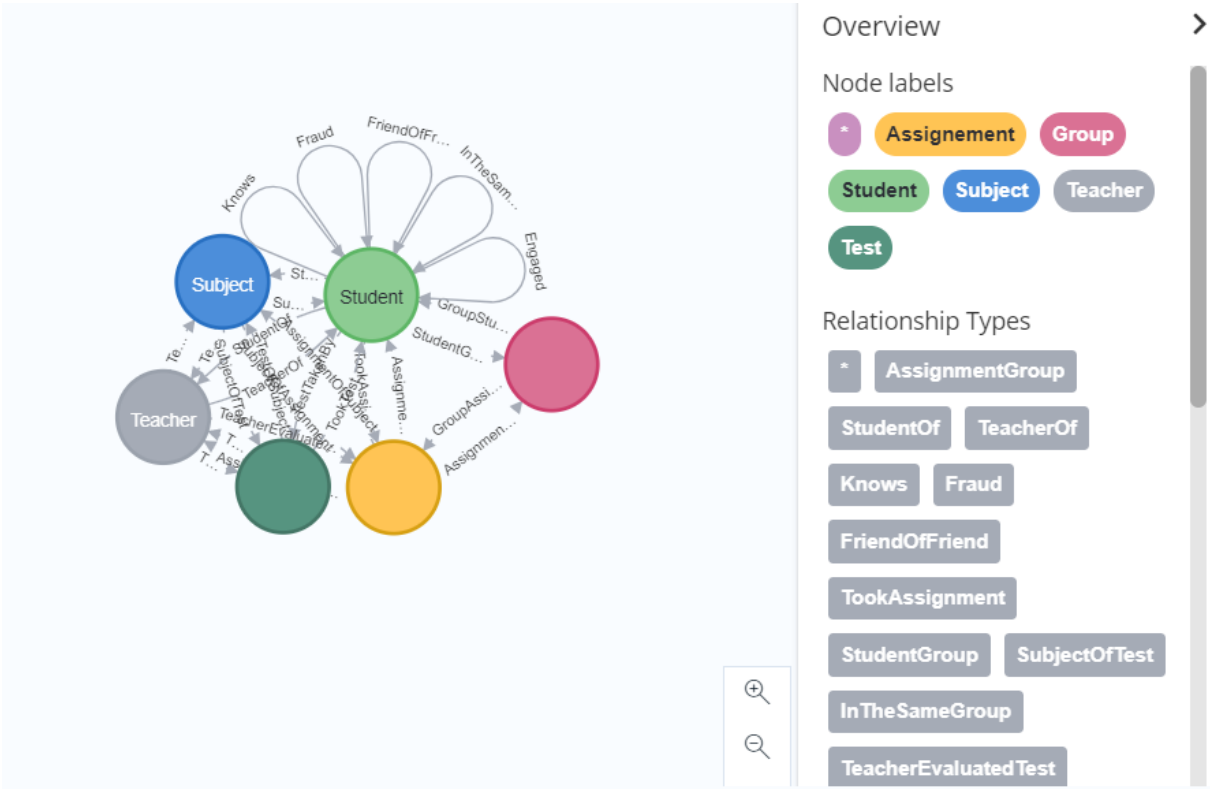

Figura 5.6: “Metamodel“ final

Node 1	Relação	Propriedades	Node 2	Direção
Assignement	AssignmentOfSubject	weight	Subject	uni-direcional
Subject	SubjectOfAssignment	-	Assignement	uni-direcional
Test	TestOfSubject	weight, minimumMark	Subject	uni-direcional
Subject	SubjectOfTest	-	Test	uni-direcional
Group	GroupAssignment	mark	Assignement	uni-direcional
Assignement	AssignmentGroup	-	Group	uni-direcional
Student	StudentAttendsSubject	presences	Subject	uni-direcional
Subject	SubjectAttendedByStudent	-	Student	uni-direcional
Student	StudentGroup	showedinterest	Group	uni-direcional
Group	GroupStudent	-	Student	uni-direcional
Student	TookTest	mark	Test	uni-direcional
Test	TestTakenBy	-	Student	uni-direcional
Student	Knows	since, weight	Student	bi-direcional
Student	InTheSameGroup	numberOfTimes	Student	bi-direcional
Student	Fraud	suspicion	Student	bi-direcional
Student	Engaged	since	Student	bi-direcional
Student	FriendOfFriend	-	Student	bi-direcional
Student	StudentOf	-	Teacher	uni-direcional
Teacher	TeacherOf	-	Student	uni-direcional
Student	TookAssignment	-	Assignement	uni-direcional
Assignement	AssignementTakenBy	-	Student	uni-direcional
Teacher	TeacherEvaluatedAssignment	-	Assignement	uni-direcional
Assignement	AssigementEvaluatedBy	-	Teacher	uni-direcional
Teacher	TeacherEvaluatedTest	-	Test	uni-direcional
Test	TestEvaluatedBy	-	Teacher	uni-direcional
Teacher	Teaches	approvalRate	Subject	uni-direcional
Subject	TeachedBy	-	Teacher	uni-direcional

Tabela 5.5: Relações resultantes das mudanças no modelo.

Label	Propriedades
Student	name, age, number
Teacher	name, yearsOfExperience
Test	designation, type
Group	designation
Assignement	designation
Subject	name

Tabela 5.6: Labels e as suas propriedades

uma tabela, os cinco registos melhor classificados, no caso dos algoritmos de centralidade.

Label Propagation - relação 'Knows'

Este algoritmo de deteção de comunidades foi testado para as várias propriedades dos "nodes" com o "label" "Student", de maneira a verificar se havia diferenças no número de comunidades detectadas e re-testado para a propriedade "weight" com foco na relação "Knows" para verificar se existiam diferenças depois da modificação do meta-modelo (tabela 5.7).

Propriedade	**Nº de comunidades**	**Elementos por comunidade**
weight	2	19, 2
since	1	21

Tabela 5.7: "Label Propagation" após ajustes ao modelo

PageRank - relação 'Knows'

Ver tabela 5.8.

Nome	**Valor**
FB	1.4979643339291218
PP	1.4338711076881734
DS	1.2872454083990308
CP	1.2126396907959134
TS	1.1934854719787833

Tabela 5.8: "Page Rank" após ajustes ao modelo

Beetweenness Centrality - relação 'Knows'

Ver tabela 5.9.

Label Propagation - relação 'InTheSameGroup'

Ver tabela 5.10.

Nome	Valor
PP	25.177822177822176
HA	15.921220446220444
FB	15.035447885447885
RC	14.111263736263737
CP	10.226273726273728

Tabela 5.9: “Betweenness Centrality“ após ajustes ao modelo

ID Comunidade	Nº Membros
17	11
9	6
1	3
12	1

Tabela 5.10: “Label Propagation“ aplicado à relação “InTheSameGroup“

- Membros da comunidade com “id“ 17: FL, PF, RG, AC, DC, RV, IM, FT, MRS, HS, LS.
- Membros da comunidade com “id“ 9: RO, RC, PP, FB, AG, TM.
- Membros da comunidade com “id“ 3: TS, HA, DS.
- Membros da comunidade com “id“ 1: CP.

PageRank - relação 'InTheSameGroup'

Neste caso, aquilo que era pretendido eram os “nodes“ com menos ligações, significando que a sua versatibilidade nos grupos escolhidos é pouca (menos “nodes“ a quem estão conectados, logo menos variação nos grupos escolhidos). Por isso, a tabela foi ordenada de forma descendente (tabela 5.11).

Beetweenness Centrality - relação 'InTheSameGroup'

Neste caso, tal como no anterior, aquilo que era pretendido eram os “nodes“ com menos ligações, pelos mesmos motivos (tabela 5.12).

Nome	Valor
RO	0.6289410125464201
PP	0.7064514878438783
HA	0.7231739214621483
RC	0.7430613334989173
FB	0.7589776011882351

Tabela 5.11: “Page Rank“ “InTheSameGroup“

Nome	Valor
CP	0.0
TS	0.0
HA	0.0
RG	1.0607142857142857
RO	1.3

Tabela 5.12: “Betweenness Centrality“ “InTheSameGroup“

Label Propagation - relação 'FriendOfFriend'

Apenas foi detectada uma comunidade no que toca a relações do tipo “FriendOfFriend“, que corresponde à totalidade dos dados.

PageRank - relação 'FriendOfFriend'

Ver tabela 5.13.

Nome	Valor
FT	1.7754185068886728
AC	1.3800086796749382
RO	1.2604468120262025
RC	1.2588831903878597
FL	1.179649500036612

Tabela 5.13: “Page Rank“ “FriendOfFriend“

Beetweenness Centrality - relação 'FriendOfFriend'

Ver tabela 5.14.

Nome	Valor
FT	24.101298701298703
RC	14.196969696969697
AC	13.447727272727272
FL	13.03690476190476
RO	12.567207792207792

Tabela 5.14: “Betweenness Centrality“ “FriendOfFriend“

5.5 Resultados

Esta secção tem como intenção avaliar os resultados obtidos aquando da aplicação dos algoritmos sobre o modelo de dados construído. Serão comparados os resultados entre a aplicação dos mesmos no modelo inicial e no final, verificando se existe uma diferença significativa naquelas relações onde já existia a possibilidade de os aplicar. Posteriormente será avaliada a influência e interpretados os resultados dos algoritmos de centralidade e comunidade em cada uma das relações, percebendo o seu significado.

Inicialmente, foi construído um modelo que apenas permitia avaliar relações sob a forma de “amizade“ ou de conhecimento mútuo entre determinados estudantes. Posteriormente e com a adição de novas relações ao modelo, foi possível ter uma noção mais transversal do relacionamento entre os estudantes (fictícios).

As diferenças ao aplicar os algoritmos “Label Propagation“, “Page Rank“ e “Betweenness Centrality“ entre o modelo inicial (tabelas 5.3, 5.1 e 5.2, respetivamente) e o final (tabelas 5.7, 5.8 e 5.9, respetivamente) no caso da relação “Knows“, não são significativas. Isso é explicado pelo facto de o conjunto de dados e as suas relações, em termos de quantidade e forma, não terem sofrido alterações. Num contexto real e com alterações dinâmicas na dimensão do problema, isso muito provavelmente não se verificaria.

No entanto, com o aumento do número de relações entre os estudantes, relacionando-os em mais variáveis devido ao ajuste do modelo, podem ser retiradas algumas conclusões, sintetizadas e traduzidas em significado prático em todas as relações sobre as quais os três algoritmos foram aplicados.

Passo então a sintetizar e dar significado aos resultados, por cada relação:

Relação "Knows"

O algoritmo "Label Propagation" não demonstrou ser útil na tarefa da deteção de diversas comunidades no "data-set". Apenas detectou duas, o que é pouco e não transmite qualquer tipo de informação relevante neste caso. No entanto, futuramente e com um conjunto de dados real, de maior dimensão, é algo a ter em conta para a deteção de diversas comunidades de estudantes que se conhecem. Factores como o ano em que tiveram certas disciplinas (o conceito de tempo) podem ser também relevantes, porque o problema passaria para outra dimensão, sem os estudantes estarem todos na mesma linha temporal.

Já os algoritmos de deteção daquilo que são os "nodes" centrais de uma comunidade, embora com poucos dados, demonstraram resultados interessantes. Ambos demonstraram resultados semelhantes, avaliados mediante a seguinte classificação, determinada pela posição (de primeiro a quinto) em que ficaram ao nível da centralidade (que será convenção nesta secção, a partir desta definição):

- 1º lugar - 5 pontos
- 2º lugar - 4 pontos
- 3º lugar - 3 pontos
- 4º lugar - 2 pontos
- 5º lugar - 1 pontos

Por isso, caso um "node" esteja em terceiro lugar nos resultados (tabela 5.8) do algoritmo "Page Rank" e em segundo nos resultados (tabela 5.9) do algoritmo "Betweenness Centrality", a sua classificação ao nível da centralidade será de $3 + 4 = 7$ pontos. Apenas são considerados os "nodes" que apareçam no "top-5" de ambos os algoritmos.

Na tabela 5.15, está presente a classificação de centralidade dos "nodes" para a relação "Knows".

Aquilo que se pode interpretar destes resultados é que ambos os algoritmos de centralidade consideraram estes três "nodes" como extremamente influentes, no que toca a

Posição	Nome	Total de pontos	Pos. Page Rank	Pos. Betweenness
1º	PP	9	2º	1º
2º	FB	8	1º	3º
3º	CP	3	4º	5º

Tabela 5.15: "Nodes" centrais nas relações de conhecimento entre os estudantes

ter relações de conhecimento com outros alunos. Isto, transportado para um caso real, poderia significar que estes estudantes seriam potenciais fontes de fraude académica.

Relação "InTheSameGroup"

Neste caso, o algoritmo de deteção de comunidades já foi mais útil, detectando (tabela 5.10) quatro comunidades diferentes. Denota-se que os elementos com baixa centralidade relativamente ao número de estudantes com o qual tinham relações estabelecidas (relação "Knows"), pertencem todos ao grupo com maior número de alunos, o que pode indicar que são elementos que não são núcleo duro dos grupos de trabalho, frequentando muitos grupos diferentes.

Outro dado interessante é que os "nodes" do tipo "Estudante" com os nomes "PP" e "FB" pertencem à mesma comunidade, sendo que já tinham sido aqueles com maior centralidade anteriormente.

A classificação relativa à centralidade (neste caso, apenas pertencer sempre aos mesmos grupos) está presente na tabela 5.16, com base nos resultados das tabelas 5.12 e 5.11.

Posição	Nome	Total de pontos	Pos. Page Rank	Pos. Betweenness
1º	RO	6	1º	5º

Tabela 5.16: Estudantes que tendem a permanecer no mesmo grupo de outros

O resultado, no caso dos algoritmos de centralidade, não foi conclusivo. O algoritmo "Page Rank" não deu os resultados (tabela 5.11) esperados. Já o "Betweenness Centrality" obteve os resultados (tabela 5.12) esperados, dado que, os "nodes" com classificação de zero são realmente, no caso deste "data-set" aqueles que não mudam de grupo em nenhum trabalho.

Relação "FriendOfFriend"

O objetivo de aplicar os algoritmos nesta relação é conseguir dar um passo no sentido de prever que relações futuras se podem formar, de confiança entre alunos, mediante serem conhecidos de conhecidos de outro. Aqueles estudantes que obtiverem a classificação mais alta, são potenciais futuros pontos centrais de redes de conhecimento (relação "Knows") numa rede dinâmica de alunos.

Por este motivo e pela dimensão dos dados do problema, o algoritmo de comunidade voltou a não se mostrar útil, pelo que não vai ser foco.

Já os algoritmos "Betweenness Centrality" e "Page Rank" deram exatamente o mesmo "top-5" (tabelas 5.14 e 5.13, respetivamente) em termos de resultados, embora com ordem trocada, o que num caso real poderia efetivamente ser indicativo daqueles alunos que poderiam ser centrais na comunidade futuramente e quem sabe, cometer fraude. A classificação está presente na tabela 5.17.

Posição	**Nome**	**Total de pontos**	**Pos. Page Rank**	**Pos. Betweenness**
1º	FT	10	1º	1º
2º	AC	7	2º	3º
3º	RC	6	4º	2º
4º	RO	4	3º	5º
5º	FL	3	5º	4ª

Tabela 5.17: Estudantes que tendem a permanecer no mesmo grupo de outros

Com estes testes concluídos, conclui-se que há "nodes" que são centrais neste momento na rede de conhecimentos e que num mundo dinâmico como o académico, rapidamente estes "nodes" teriam tendência a sobrepor-se nesta função, tomando novos alunos os seus lugares e assim sucessivamente. Esta centralidade poderá indicar, de facto, a existência de fraude académica nos "nodes" centrais de um eventual conjunto de dados não-fictício e de muita maior complexidade.

Capítulo 6

Conclusões e Trabalho Futuro

Penso que em geral, os objetivos propostos inicialmente foram cumpridos. Embora não tenha sido desenvolvida forma de encontrar a fraude propriamente dita, foram dados passos importantes para a resolução do problema. Foi construído um meta-modelo útil e robusto, uma base sólida para no futuro se chegar à resolução do problema em questão, com base em mecanismos de inferência.

O conjunto de dados fictício que foi criado manualmente, embora pequeno, pode ser visto como uma representação relativamente fidedigna de uma situação académica num determinado momento do tempo, em que alunos frequentam unidades curriculares e participam em avaliações, supervisionadas por professores. De resto, a dificuldade para encontrar um conjunto de dados já preparado para ser estudado, foi o maior entrave a uma maior profundidade na investigação e interpretação dos resultados, bem como a um desenvolvimento mais profundo dessas técnicas de inferência. O conceito de tempo será algo importante, por forma a dar dinamismo aos dados, que se alteram ao longo dos semestres e dos anos letivos, gerando novas redes de conhecimento, com "nodes" a poder ganhar preponderância.

Os resultados obtidos foram animadores e revelam que pode realmente existir relação entre o número, tipo e quantidade de relações interpessoais entre determinados alunos e a existência de redes de fraude académica.

Futuramente, seria interessante converter de alguma forma este meta-modelo numa

notação mais poderosa (OWL), utilizando por exemplo a “framework“ “Protégé“, que faz uso desta notação e que possui mecanismos de inferência para relações subentendidas e ainda uma validação para a lógica da ontologia desenvolvida [85]. A notação OWL poderia levar o modelo a outro nível em termos de robustez.

Para a previsão de relações consideradas de fraude (estão presentes no modelo, mas são algo conceptual), seria interessante o uso da tecnologia “Neptune ML“, propriedade da “Amazon“, que tem a funcionalidades como a previsão de ligações entre “nodes“ ou o valor de propriedades associadas aos mesmos, através de métodos de inteligência artificial, tais como redes neuronais [86].

Em suma, foi uma experiência enriquecedora a nível pessoal, que contribui para um início prometedor relativamente a uma possível solução para o problema.

Bibliografia

[1] J. P. Cordeiro e P. Inácio. (jul. de 2021). "Detectamos regularmente Fraude Académica?" URL: https://www.ubi.pt/Sites/forumfraudeacademica/pt/Pagina/cap2.

[2] "The use of ontologies for effective knowledge modelling and information retrieval," *Applied Computing and Informatics*, vol. 14, n.º 2, pp. 116–126, 2018, ISSN: 2210-8327. DOI: https://doi.org/10.1016/j.aci.2017.07.003. URL: https://www.sciencedirect.com/science/article/pii/S2210832717300649.

[3] "Novel graph based anomaly detection using background knowledge," *FLAIRS 2017 - Proceedings of the 30th International Florida Artificial Intelligence Research Society*, pp. 538–543, 2017. URL: https://www.aaai.org/ocs/index.php/FLAIRS/FLAIRS17/paper/viewFile/15403/14998.

[4] (Jun. de 2018). "Vantagens e desvantagens das redes sociais – Mídias Sociais," URL: https://news.comschool.com.br/vantagens-e-desvantagens-das-redes-sociais-news-comschool/.

[5] D. P. Silveira. (ago. de 2018). "Vantagens e desvantagens das redes sociais," URL: https://www.oficinadanet.com.br/post/18285-vantagens-e-desvantagens-das-redes-sociais.

[6] "Friendships and Social Networks," *Encyclopedia of Creativity*, vol. 3, pp. 521–525, 2020. DOI: https://doi.org/10.1016/B978-0-12-809324-5.23837-6. URL: https://www.sciencedirect.com/science/article/pii/B9780128093245238376.

[7] "Graph-Based Fraud Detection in the Face of Camouflage," *ACM Transactions on Knowledge Discovery from Data*, vol. 11, pp. 1–26, 2017. DOI: https://doi.org/10.1145/3056563. URL: https://dl.acm.org/doi/10.1145/3056563.

[8] "Anomaly detection in online social networks," *Social Networks*, vol. 39, pp. 62–70, 2014. DOI: https://doi.org/10.1016/j.socnet.2014.05.002. URL: https://www.sciencedirect.com/science/article/pii/S0378873314000331.

[9] R. Alario. (jul. de 2009). "O homem é um animal social – Aristóteles," URL: https://projetophronesis.wordpress.com/2009/01/10/o-homem-e-um-animal-social-aristoteles.

[10] (). "Socialização," URL: https://pt.wikipedia.org/wiki/Socializa%C3%A7%C3%A3o.

[11] (). "Timeline of Computer History," URL: https://www.computerhistory.org/timeline/computers.

[12] (). "Socialização," URL: https://pt.wikipedia.org/wiki/Big_data.

[13] "Perspective of anomaly detection in big data for data quality improvement," *Elsevier*, 2021. DOI: https://doi.org/10.1016/j.matpr.2021.05.597. URL: https://www.sciencedirect.com/science/article/pii/S2214785321042243.

[14] L. Husamaldin e N. Saeed, "Big Data Analytics Correlation Taxonomy," *Information*, vol. 11, p. 17, dez. de 2019. DOI: 10.3390/info11010017.

[15] T. Naeem. (nov. de 2020). "Noções básicas sobre dados estruturados, semiestruturados e não estruturados," URL: https://www.astera.com/pt/type/blog/structured-semi-structured-and-unstructured-data.

[16] "MODELOS DE REPRESENTAÇÃO SEMÂNTICA NA ERA DO BIG DATA," *Brazilian Journal of Information Studies*, 2018. URL: https://brapci.inf.br/index.php/res/download/99420.

[17] (). "Wolfram|Alpha," URL: https://pt.wikipedia.org/wiki/WolframAlpha.

[18] R. A. S. (set. de 2020). "How does facebook handle the 4+ petabyte of data generated per day? Cambridge Analytica - facebook data scandal.," URL: https://medium.com/@srank2000/how-facebook-handles-the-4-petabyte-of-data-generated-per-day-ab86877956f4.

[19] H. P. (fev. de 2021). "A Review of the Semantic Web Field," URL: https://cacm.acm.org/magazines/2021/2/250085-a-review-of-the-semantic-web-field/fulltext.

[20] (). "Google Knowledge Graph," URL: https://pt.wikipedia.org/wiki/Google_Knowledge_Graph.

[21] "Towards a Definition of Knowledge Graphs," 2016. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.1054.8298&rep=rep1&type=pdf.

[22] P. H., "Knowledge Graph Refinement: A Survey of Approaches and Evaluation Methods," *Semantic Web Journal, (Preprint):1–20*, 2016. URL: http://www.semantic-web-journal.net/system/files/swj1167.pdf.

[23] "Special Issue on Knowledge Graphs," *Semantic Web Journal*, 2016.

[24] B. A., "From Taxonomies over Ontologies to Knowledge Graphs," 2014. URL: https://semantic-web.com/from-taxonomies-over-ontologies-to-knowledge-graphs/.

[25] "Linked Data Quality of DBpedia, Freebase, OpenCyc, Wikidata, and YAGO," *Semantic Web Journal*, 2016. URL: http://www.semantic-web-journal.net/system/files/swj1366.pdf.

[26] "Knowledge Graph Identification," *In Proceedings of the 12th International Semantic Web Conference*, 2013. URL: https://link.springer.com/chapter/10.1007/978-3-642-41335-3_34.

[27] (). "What is a Knowledge Graph?" URL: https://www.ontotext.com/knowledgehub/fundamentals/what-is-a-knowledge-graph/.

[28] (). "GeoNames," URL: https://pt.wikipedia.org/wiki/GeoNames.

[29] (). "FactForge – Open Data and News about People, Organizations and Locations," URL: https://www.ontotext.com/knowledgehub/demoservices/factforge-explore-linked-open-data/.

[30] Y. Seth. (out. de 2019). "Introduction to Question Answering over Knowledge Graphs," URL: https://yashuseth.blog/2019/10/08/introduction-question-answering-knowledge-graphs-kgqa/.

[31] "Graph based anomaly detection and description: a survey," *Data Mining and Knowledge Discovery*, vol. 29, pp. 626–688, 2015. URL: https://link.springer.com/article/10.1007%2Fs10618-014-0365-y.

[32] K. Anand, J. Kumar e K. Anand, "Anomaly detection in online social network: A survey," em *2017 International Conference on Inventive Communication and Computational Technologies (ICICCT)*, 2017, pp. 456–459. DOI: 10.1109/ICICCT.2017.7975239.

[33] "Anomaly detection in dynamic networks: a survey," *WIREs Comput Stat*, vol. 7, pp. 223–247, 2015. DOI: https://doi.org/10.1002/wics.1347. URL: https://wires.onlinelibrary.wiley.com/doi/10.1002/wics.1347.

[34] S. Bhattacharyya, S. Jha, K. Tharakunnel e J. C. Westland, "Data mining for credit card fraud: A comparative study," *Decision Support Systems*, vol. 50, n.º 3, pp. 602–613, 2011, On quantitative methods for detection of financial fraud, ISSN: 0167-9236. DOI: https://doi.org/10.1016/j.dss.2010.08.008. URL: https://www.sciencedirect.com/science/article/pii/S0167923610001326.

[35] A. Abdallah, M. A. Maarof e A. Zainal, "Fraud detection system: A survey," *Journal of Network and Computer Applications*, vol. 68, pp. 90–113, 2016, ISSN: 1084-8045. DOI: https://doi.org/10.1016/j.jnca.2016.04.007. URL: https://www.sciencedirect.com/science/article/pii/S1084804516300571.

[36] E. Ngai, Y. Hu, Y. Wong, Y. Chen e X. Sun, "The application of data mining techniques in financial fraud detection: A classification framework and an academic

review of literature," *Decision Support Systems*, vol. 50, n.º 3, pp. 559–569, 2011, On quantitative methods for detection of financial fraud, ISSN: 0167-9236. DOI: https://doi.org/10.1016/j.dss.2010.08.006. URL: https://www.sciencedirect.com/science/article/pii/S0167923610001302.

[37] "Fraud detection: A systematic literature review of graph-based anomaly detection approaches," 2020. DOI: https://doi.org/10.1016/j.dss.2020.113303. URL: https://www.sciencedirect.com/science/article/pii/S0167923620300580.

[38] L. Becker. (jul. de 2019). "O que é Web Semântica?" URL: https://www.organicadigital.com/blog/o-que-e-web-semantica.

[39] O. L. Tim Berners-Lee James Hendler. (2001). "The Semantic Web," URL: https://www.scientificamerican.com/article/the-semantic-web/.

[40] (). "Web semântica," URL: https://pt.wikipedia.org/wiki/Web_sem%C3%A2ntica.

[41] H. Holarmide. (mar. de 2018). "Semantic Web," URL: https://www.seminarstopics.com/seminar/7553/semantic-web.

[42] A. Morrison. (set. de 2020). "What is the difference between a knowledge graph and a graph database?" URL: https://www.quora.com/What-is-the-difference-between-a-knowledge-graph-and-a-graph-database.

[43] (). "Graph database," URL: https://en.wikipedia.org/wiki/Graph_database.

[44] (Jul. de 2017). "Visualización de información: ideas e innovación," URL: https://www2.utp.edu.co/cidt/listar-noticias/visualizacion-de-informacion-ideas-e-innovacion.html.

[45] N. Singh. (abr. de 2019). "What are Graph databases and different types of Graph databases," URL: https://singhnaveen.medium.com/what-are-graph-databases-and-different-types-of-graph-databases-369e5040a9d0.

[46] J. Hoppa. (set. de 2019). "15 Rules of a Native Graph Database," URL: https://dzone.com/articles/15-rules-of-a-native-graph-database.

[47] Y. Xu. (nov. de 2018). "Building a Graph Database on a Key-Value Store?" URL: https://dzone.com/articles/building-a-graph-database-on-a-key-value-store.

[48] M. Santana. (mai. de 2016). "Introdução ao Spark GraphX e GraphFrames," URL: https://medium.com/data-hackers/introdu%C3%A7%C3%A3o-ao-spark-graphx-e-graphframes-9b10089f2e7f.

[49] (). "Significado de Ontologia," URL: https://www.significados.com.br/ontologia/.

[50] "An overview about ontologies: survey about definitions, types, applications, evaluation and building methods," *Revista Ciência da Informação*, 2003. DOI: https://doi.org/10.1590/S0100-19652003000300002. URL: https://www.scielo.br/j/ci/a/LR68syZsPSSmwvPHrNXmC8N/?lang=pt.

[51] W. N. Borst, "Construction of Engineering Ontologies for Knowledge Sharing and Reuse," 1997.

[52] N. Guarino, "Formal Ontologies and Information Systems," jun. de 1998.

[53] "Ontologias: conceitos, usos, tipos, metodologias, ferramentas e linguagens," rel. téc.

[54] (). "Ontology components," URL: https://en.wikipedia.org/wiki/Ontology_components.

[55] (). "Introduction to Ontologies," URL: https://www.geeksforgeeks.org/introduction-to-ontologies/.

[56] "Ontology-Based Integration of Information - A Survey of Existing Approaches," *Proceedings of the IJCAI'01 Workshop on Ontologies and Information Sharing, Seattle, Washington, USA, Aug 4-5*, ago. de 2002.

[57] "Ontology Based Information Integration: A Survey," set. de 2019. URL: https://arxiv.org/abs/1909.13762.

[58] "Ontology Issues and Applications - Guest Editors' Introduction.," *J. Braz. Comp. Soc.*, vol. 11, pp. 5–16, nov. de 2005. DOI: 10.1590/S0104-65002005000300001.

[59] "A SURVEY ON ONTOLOGY CONSTRUCTION METHODOLOGIES," 2011.

[60] "Supporting Ontology Development with ODEd.," vol. 9, jan. de 2002, pp. 1–11. DOI: 10.1590/S0104-65002003000300005.

[61] (). "Sistema de gerenciamento de banco de dados," URL: https://pt.wikipedia.org/wiki/Sistema_de_gerenciamento_de_banco_de_dados.

[62] (). "Redis," URL: https://pt.wikipedia.org/wiki/Redis.

[63] (). "API Blueprint," URL: https://apiblueprint.org/.

[64] (). "AllegroGraph Overview," URL: https://allegrograph.com/products/allegrograph/.

[65] (). "Trinity," URL: https://www.microsoft.com/en-us/research/project/trinity/.

[66] (). "A versioning data store for time-variant graph data.," URL: https://recallgraph.tech/.

[67] (Out. de 2021). "neo4j Alternatives," URL: https://alternativeto.net/software/neo4j/.

[68] (). "Neo4j," URL: https://en.wikipedia.org/wiki/Neo4j.

[69] (). "Neo4j - Overview," URL: https://www.tutorialspoint.com/neo4j/neo4j_overview.htm.

[70] (Jan. de 2021). "Detailed explanation of Python using py2neo operation diagram database neo4j," URL: https://developpaper.com/detailed-explanation-of-python-using-py2neo-operation-diagram-database-neo4j/.

[71] S. Yang. (jul. de 2020). "How to query Neo4j from Python," URL: https://towardsdatascience.com/neo4j-cypher-python-7a919a372be7.

[72] (). "Neo4j APOC Library," URL: https://neo4j.com/developer/neo4j-apoc/.

[73] (). “9 reasons digital fraud is on the rise,” URL: https://www.securitymagazine.com/gdpr-policy?url=https%3A%2F%2Fwww.securitymagazine.com%2Farticles%2F93912-reasons-digital-fraud-is-on-the-rise.

[74] A. Beattie. (set. de 2021). “The Pioneers of Financial Fraud,” URL: https://www.investopedia.com/articles/financial-theory/09/history-of-fraud.asp.

[75] S. Zheng. (abr. de 2017). “How students cheated in exams to get into China’s imperial civil service,” URL: https://www.scmp.com/news/china/society/article/2088423/how-students-cheated-exams-get-chinas-imperial-civil-service.

[76] (). “Essay Mill,” URL: https://en.wikipedia.org/wiki/Essay_mill.

[77] (). “Suborno,” URL: https://dicionario.priberam.org/suborno.

[78] D. Newton. (ago. de 2020). “Another problem with shifting education online: cheating,” URL: https://hechingerreport.org/another-problem-with-shifting-education-online-cheating/.

[79] (Set. de 2020). “Did My Student Copy and Paste?” Here’s How to Find Out,” URL: https://completeliterature.com/did-my-student-copy-and-paste-heres-how-to-find-out/.

[80] (). “Algorithms,” URL: https://neo4j.com/docs/graph-data-science/current/algorithms/.

[81] (). “Label Propagation,” URL: https://neo4j.com/docs/graph-data-science/current/algorithms/label-propagation/.

[82] V. Mallawaarachchi. (mar. de 2020). “Label Propagation Demystified,” URL: https://towardsdatascience.com/label-propagation-demystified-cd5390f27472.

[83] (). “Betweenness Centrality,” URL: https://neo4j.com/docs/graph-data-science/current/algorithms/betweenness-centrality/.

[84] (). “Page Rank,” URL: https://neo4j.com/docs/graph-data-science/current/algorithms/page-rank/.

[85] J. A. Khan e S. Kumar, "Deep analysis for development of RDF, RDFS and OWL ontologies with protege," em *Proceedings of 3rd International Conference on Reliability, Infocom Technologies and Optimization*, 2014, pp. 1–6. DOI: 10.1109/ICRITO.2014.7014747.

[86] (). "Amazon Neptune ML for machine learning on graphs," URL: https://docs.aws.amazon.com/neptune/latest/userguide/machine-learning.html#machine-learning-capabilities.

Apêndice A

Proposta Original do Projeto

INSTITUTO POLITÉCNICO DE BRAGANÇA
Escola Superior de Tecnologia e Gestão

Curso de Mestrado em Informática
Dissertação - Ano letivo de 2020/2021

Ontologias para deteção de fraude em meio académico

Aluno: Tiago Filipe Fernandes dos Santos (n. 28248)

Orientador: Paulo Jorge Teixeira Matos

1 Objetivo

Esta dissertação visa analisar a viabilidade e adequação do uso de ontologias, base de dados orientadas a grafos e outras metodologias e ferramentas do âmbito da web semântica, na representação das diversas dimensões de contexto académico que possam influenciar ou denunciar a existência de fraude nos processos de avaliação.

A adequação será aferida desenvolvendo soluções que através de padrões comportamentais ou de relacionamento, ou através do cálculo de métricas aritméticas, permitam evidenciar situações de potencial fraude. As abordagens a seguir poderão incidir sobre a identificação de contextos que tendem a originar fraude, permitindo assim agir de forma preventiva para evitar estas situações; ou poderão incidir numa perspetiva de identificar situações fraudulentas já ocorridas.

2 Detalhes

As ontologias são a forma natural de descrever relacionamentos entre entidades abstratas ou concretas. Não obstante a simplicidade dos conceitos, o potencial é enorme e em grande parte inexplorado no que diz respeito a problemas concretos. É o caso do contexto em que decorrem as avaliações de índole académica. Há alunos, professores, cursos, disciplinas, diversas formas de avaliação, grupos de trabalhos, amizades, e imensas outras relações que devidamente modeladas e exploradas, podem facultar condições excecionais de identificação de fraude ou eminência de fraude.

O problema pode ser explorado unicamente na vertente lógica, mas há várias outras alternativas, que complementarmente podem produzir resultados muito interessantes. Desde logo a aplicação de métodos estatísticos sobre as estruturas de grafos ou a implementação de sistemas de aprendizagem supervisionada que permitam a identificação por si só de novos padrões.

Esta dissertação é assim de investigação e desenvolvimento aplicado, explorando tecnologias e a sua articulação na construção de soluções, que permitam por um lado uma representação útil do conhecimento, e por outro a implementação de mecanismos que explorem essa representação no sentido de obter os resultados pretendidos.

3 Metodologia de trabalho

O desenvolvimento desta dissertação envolve o levantamento do estado da arte, quer na perspetiva do domínio do problema, quer na perspetiva do uso das tecnologias que se pretende utilizar.

A fase seguinte será analisar o domínio do problema obtendo uma descrição alargada e detalhada dos procedimentos e metodologias de avaliação, na perspetiva dos alunos e dos docentes.

Posteriormente dar-se-á início à construção dos modelos de representação do conhecimento e, em simultâneo, à implementação de soluções experimentais de cálculo de métricas e de identificação de padrões. Solução esta que terá que ser construída de forma incremental, em que cada experiencia, permitirá validar ideias e, eventualmente expandir a solução no sentido de alcançar os objetivos.

As soluções obtidas terão que ser posteriormente validadas em contexto próximo do real e quantificados

os resultados. Esperando assim haver material para produção científica.

Por último, será redigido a dissertação, tendo como tese "Adequação das tecnologias em causa na identificação de situações de fraude académica".

Apêndice B

Artigo Proposto sobre o trabalho

Academic Fraud Detection

1st Tiago Santos
Polytechnic Institute of Bragança
Bragança, Portugal
a28248@alunos.ipb.pt

2nd Paulo Matos
Polytechnic Institute of Bragança
Bragança, Portugal
pmatos@ipb.pt

***Abstract*—Fraudulent activity is increasing in the academical world and solutions start to be needed to stop it and define a standardization of suspicious behaviours or relationships between students. In this present article, the authors present an exploratory work, where a graph database was designed with the goal of supporting potential solutions that can assess this problem. The work includes a meta-model, a test data-set based on that the queries and algorithms performed and, the achieved results. The software used was Neo4J and Cypher.**

***Index Terms*—Graph database, plagiarism, meta-model, fraud, student**

I. Indroduction

As fraudulent activities are exponentially increasing everywhere in the world, schools are not exceptions to that trend. This problem is emergent and starts to require real solutions of inference about how some behaviours or some set of variables, when conjugated with each other, can be a major indicator of fraudulent activities like plagiarism, cheating on exams, copying works and even teaching colleagues some answers during a test in an illicit way.

The core part of this article starts with an historical contextualization in section II about the concept of fraud, evolving to a more specific domain, which is the main topic of the problem approached in this paper: the domain of academic fraud.

After that, in section IV-A a more incisive overview of the problem is given, explaining everything about a model that makes a standardization of the approached problem, being the start of a solution to identify relationships and behaviour patterns related with people trying to overcome the academic system via illicit activities.

Section V approaches, the problem is approached in a more specific way, by populating the database that was obtained in section IV-A and explaining the performed queries and algorithms over the data-set, as well as its results.

II. Contextualization

The concept of fraud, or deceiving is present throughout our history as human beings, since the primordial societies were founded. The first famous, or well known fraud was recorded back in Ancient Greece, in the year 300 BC. This initial attempt was performed by a sea merchant named Hegestratos, that sought to insure his ship and cargo, so he took an insurance policy against the insurance people. At that moment of time, the policy was well known and named "bottomry", and the strategy was for the merchant to borrow money to the value equivalent to his ship and cargo. If the ship arrived safe to its destination (with its cargo intact), the loan was paid back with interest. If a loan, under this conditions of safeness was not paid, there would be a repossession of the boat and its cargo. In the case of Hegestratos, he tried, in what is possibly the first recorded attempt of fraud, to sink his empty ship and sell its cargo (that was corn) and also receive the loan, so he would get twice the money of his cargo. This first attempt didn't succeed because Hegestratos was caught while sinking his boat by his own crew and was chased off the ship and drowned trying to escape his own crew.

After this recorded case, frauds have grown exponentially and its performers and shapes have without any doubt grown and changed, as badly malicious people reinvent themselves, creating a lot of fraudulent acts and ways of performing it.

As stated before, the fraudulent acts are very different and can vary a lot. For example, falsification of documents (forgery) an counterfeiting are acts of physical duplication or fabrication. Another type of fraud is the theft of personal information/identity, like for instance someone getting another person's credit card and then using its number and CVV in order to get the personal benefit of spending the victim's money. Another type of fraud are computer crimes, like phishing, the distribution of hoax emails, accessing unauthorized computers, data mining through spyware and malware, hacking computer systems in order to get access to sensible personal information, as well as the act of sending computer viruses or worms to affect other party's system.

Frauds are the attempt of deceiving a victim with the objective of personal benefits, and the fraudulent party could be one or more persons (acting as a group), being the victim one or more individuals, or a system via the act of exploiting it and its vulnerabilities.

The main focus of this article is to write and investigate about academic frauds, an example of group deceiving. The concept can be split in various sub-concepts, such as academic misconduct, as well as academic dishonesty. All of those refer to a set of actions that go against the norms of a learning institution. Academic frauds start from the elementary school all the way up to universities, throughout history.

As well as any kind of fraud, human dishonesty starts from the very beginning and so academic fraud started on the first tests, on the Chinese civil service exams, through cheating and bribery of examiners.

At the turn of the 20th century, it has been estimated [1]

that two-thirds of all students cheated at some stage. Usually, there are groups of students that create strong bonds as the years advance, and integrate each day more students to their group, which can create a net of interests, with tendencies to in some cases perform the act of cheating.

For example, fraternities usually operated something called "essay banks", that were created for equal assignments to be submitted by different students in different years, only changing the name on the paper. Over the last years since the latter half of the twentieth century, the problem of academic fraud has grown a lot.

There are several acts of fraud related to learning institutions, like bribing, that consists of "an act of giving money or gift giving that alters the behavior of the recipient" [1], deception, which consists in the act of lying to the supervisor concerning an academic exercise (e.g. claiming to have submitted a work while it's not true), impersonation, that is the act of someone taking an exam for the supposed person that should be taking it, plagiarism, which is the act of copying someone's work (one of the most common types of academic fraud) and the main subject of this article, which is cheating.

Cheating can take a lot of forms, like taking personal illegal notes to tests, copying from someone's exam, with or without the other person's consent, have access to forbidden information with their cellphones or the share of information inside an evaluation activity between two students. A lot of cheating techniques have been developed throughout the years. Between students/ friends some methods emerged, like for example signaling the right answer.

In the case of cheating, it differs from other forms of academic fraud/dishonesty, because for instance people can be involved in it without being benefited themselves for the act of participating in the schema. Some examples are students benefiting others through a dialog before or during an exam, via cellphone, and giving the benefited party the correct answers, as well as information exchanges when exams are scheduled in different but consecutive hours, in which students can be leaving the room and informing others.

III. The domain of the problem

The main focus of this article is to walk in a direction of standardization in terms of common behaviors that can lead us to have suspicions of fraudulent actions between two or more students. It can be important for us to know, for instance, if the students are friends, or are engaged, or even if they were taking the same subjects or in which groups they were in. A net develops from strong relationships between people, and if a big number of persons know each other, with that number growing, the highest is the probability of one or more suspicious relationships to be born and raise, appending more and more relationships that benefit some individuals.

A. Graph Databases

In order to achieve some initial perspective of the problem, it was used a concept that are graph databases that consist on nodes, properties and edges. Those nodes represent actors of the system. The edges represent relationships between them.

Both of the previous two concepts have properties. For instance, Tiago (node) can be married with Cristina (node). Tiago is 27 years old and lives in Bragança, while Cristina is 30 and lives in Lisbon. Those characteristics are properties of the nodes. Now, let's imagine that they are married since 06/06/2020. This is also a characteristic, but in this case of their relationship (between two nodes). So, "since" is named a relationship property.

IV. Structuring the problem

In this first stage, the objective was to create a graph database, representing a embryonic idea about how could an academic fraud problem be in fact standardized through relations and characteristics (properties) of the relations and actors.

A. Meta-model

So, the first thing that people have a tendency to think in the case of this problem is about students. Here we have our first kind of nodes (label). Each node of this type has an age and number. Of course, students have interpersonal relationships. So, how can students relate between them?

- By friendship relationships
- Being engaged to each other
- Forming groups with each other
- If some conditions are met, via fraud relationships

A friendship relationship has characteristics like when it was formed and the level of importance that relationship has between the two students. If they are engaged, the weight of the friendship is the maximum. This friendship can be important for the students to form groups. We can have a subjective idea that tells us if a student was sure about what he was presenting in that assignment.

Those students attend subjects, which means that another label needed to emerge here, that referred to all the subjects. As a consequence, another relationship was created (with the name of the action, "Attends"), in order to make sure that those students were always associated with the disciplines that they attended.

That relationship, in order to be relevant for the problem domain needed an attribute to assess the interest of the student in the subject, like "Presences", a quantitative measure to observe how many classes were attended by each individual.

As students attend classes, they do it for the purpose of achieving knowledge and that can result in approval to the discipline at the end of the semester. In order to get that approval, students are submitted to evaluations. Those evaluations, usually consist in assignments and tests.

For this reason, two key labels were included, because it's here that the students can really perform academic fraud. Each assignment, as well as each test, has a defined weight, which can be important to the probability of someone cheating on it. The more important to the final score is the evaluation, the more likely is for people to try to get marks in an illicit way.

Tests can be taken online or in person, which can be also relevant. It's more likely that a student cheats in an online test. Another characteristic that can change the probability of a fraudulent act, is if a test is of multiple choice or a theoretical one, for example. So, "Test" has a property assessing that, named "Type".

As team-spirit is so important nowadays in our everyday jobs and tasks, as well as collaborative work, academically the assignments typically need groups to be formed. For this reason, group is another label of the proposed schema. This nodes are connected via relationship to the ones with the label "Student", so a student belongs to a certain group related to an assignment.

The relationship "Grouped In" refers to a student participating in a group, and the relationship named "Belongs To" refers to a group making an assignment. In this last relationship, a global mark to the group is assessed, being relevant in a later stage, in which we compare the student's mark to the group mark.

Another attribute considered is related to the work presentation, with the intention of having a boolean value, named "showed interest". If a student's behaviour reveals that he/her didn't show a lot of interest in explaining the work done, it could indicate that he/her was fraudulent in that work, letting other members of the group doing the job for him/her.

Complementary to assignments in the set that composes an evaluation, tests are taken by students, which means another relationship between nodes. Here, the individual mark is considered and is a property of the relationship that can be used, as already stated, in the comparison between the individual mark and the group mark. It can be relevant for us to have the information of either the test has a minimal score to be achieved or not. If it has, the probability of cheating is higher than if it hasn't.

For the relation between students and subjects, as already stated, an important factor could be the number of classes that are attended by him/her. For instance, if the student doesn't attend regularly to classes but works in a group with other students, that showed more interest along the course of the classes but have better grades, this pattern can signal a fraudulent or suspicious relationship.

For the purpose of design and test this embryonic metadata structure, it was used Neo4j, a graph database management system developed by Neo4j, Inc. that is a transactional database with graph storage and processing, implemented in Java and "accessible from software written in other languages using the Cypher query language through a transactional HTTP endpoint, or through the binary "bolt" protocol." [2].

Being that explained, and considering the labels and relationships previously stated, a meta-model was resultant, using Neo4j, graphically presented in 1.

B. Creating the data-set

The labels referred in IV-A, as well as the relationships between them result in a generalist concept, of how data will be placed and communicate, via instances, between itself. So,

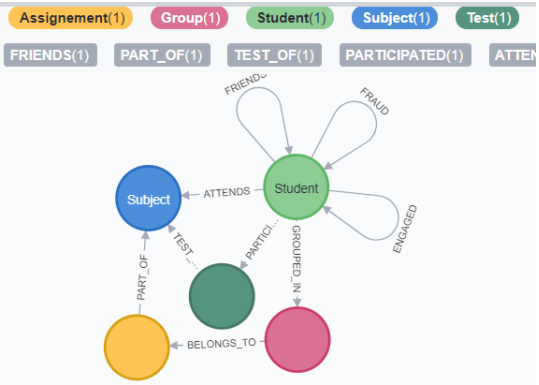

Fig. 1. Meta-model of the problem

for the label "Student", 21 nodes were created, resulting in 96 relationships, with 88 being of friendship 2 of the type "Engaged", as well as 6 of them being labeled as Fraud.

The label "Subject" contains three nodes, or instances. Each of this instances, related to itself has two nodes with the "Assignment" label and one of the kind "Test". For this reason, the database was filled with six nodes of the "Assignment" type and three of the kind "Test". Logically, each assignment contains groups related to it, and in this problem I've inserted four groups per assignment, which results in 24 nodes of that kind.

Being stated the types of the nodes inserted, as well as their quantities, in 2 the resulting data-set is shown.

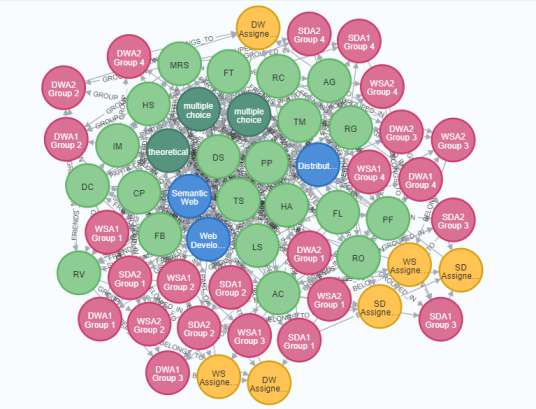

Fig. 2. Data-set

V. Solution and Results

With the objective of obtaining some embryonic results, several queries to the database were implemented. The first goal was to perceive which parts of the data-set were the most

interesting to be exploited and how they could be related with each other.

1) Initial Queries: The first query relates which students are friends of other student's friends, as a friend of a friend relationship. Note that "tiago" is the variable to represent the student, "fof" is the resulting set of results and "FRIENDS" is the relationship used to obtain the sub-set of data. A test query was created in order to achieve that. For example, to get the Tiago Santos' friends of his friends, the constructed query was:

```
MATCH (tiago:Student)-[:FRIENDS*2]->(fof)
WHERE tiago.name = "Tiago Santos"
AND NOT((tiago)-[:FRIENDS]-(fof))
RETURN DISTINCT fof.name;
```

Another thing that could be interesting to have knowledge about were the groups in which two students were together. The higher the number, the higher is the probability of a fraudulent act. Note that "a" is the variable with the label "Student", related with "m", a group, that is also related with "d" (another student). The final result is the return of a descendent number ("coun") of students that participated in the same group 3. The constructed query was:

```
MATCH (a:Student)-[:GROUPED_IN]->
(m:Group)-[:GROUPED_IN]-(d)
RETURN a.name, d.name, count(m) as coun
order by coun desc;
```

	a.name	d.name	coun
1	"TS"	"CP"	6
2	"TS"	"DS"	6
3	"TS"	"HA"	6
4	"HA"	"CP"	6
5	"HA"	"TS"	6

Fig. 3. Result of the query between persons and groups

The third query evolves the previous one, adding to the existing query the concept of fraud, in order to associate the number of groups in each two individuals had in common, and the relationships stated as frauds. Note that the variables "a", "m", "d" and "coun" have the same meaning as in the previous query, with the addition of the variable "f" that represents the relationship "Fraud" and "frauds" that represents the number of frauds. The resulting query was:

```
MATCH (a:Student)-[:GROUPED_IN]->(m:Group)
-[:GROUPED_IN]-(d),
(a:Student)-[f:FRAUD]->(d)
RETURN a.name, d.name, count(m) as coun,
count(f) as frauds
order by frauds desc;
```

The results of this query are stated in 4.

a.name	d.name	coun	frauds
"FT"	"IM"	6	6
"AG"	"PP"	3	3
"RC"	"PP"	1	1

Fig. 4. Result of the query between frauds and groups

On this research, was also done an effort to get more objective values using the libraries of Neo4J. GDS, or Graph Data Science, is a library that contains a lot of graph algorithms, divided into three categories: production-quality, that are robust algorithms that have "been tested with regards to stability and scalability" [**?**], beta and alpha-quality, that are less robust and have been less tested, with alpha being in the lowest level.

Esses algoritmos são divididos em várias categorias, incluindo centralidade, detecção de comunidades, similaridade, "path location" e "'link prediction"..

2) Label Propagation: The first algorithm that I used had the intention of finding communities within the data-set of the problem, as it is a community-detection algorithm, that "detects these communities using network structure alone as its guide, and doesn't require a pre-defined objective function or prior information about the communities" [3], named Label Propagation.

The first stage to use any algorithm that has production quality is to call the instruction "gds.graph.create", in order to create a graph representing whichever we want to obtain. In this case, the goal was to obtain every student that had a "Friends" relationship with others of the same kind, with an undirected orientation (meaning that it works symmetrically).

```
CALL gds.graph.create('weights', 'Student',
        {
        FRIENDS: {
        orientation: 'UNDIRECTED',
        properties: 'weight'
        }
        }
)
```

The next step was to get the number of communities, and the number of students in each one. For this purpose, the instantiation of the algorithm itself was needed. Based on

the graph named "weights", created previously, and on the property "weight", a property called "community" was created, related with each node of the label "Student".

```
CALL gds.labelPropagation.write(
        'weights',
        {
        relationshipWeightProperty: 'weight',
        maxIterations: 10,
        writeProperty: 'community'
        })
```

The last step regarding the Label Propagation algorithm was to query the database, in order to get how many students ("s") were part of each community, obtaining a "number of nodes".

```
match (s:Student)
with s.community as community, count(s)
as number_of_nodes
return community, number_of_nodes order
by number_of_nodes desc limit 50
```

This algorithm didn't have the expected results, finding only two communities5.

community	number_of_nodes
9	19
12	2

Fig. 5. Result of performing Label Propagation

3) Betweenness Centrality: The first of two centrality algorithms used in this problem, has the intention of "detecting the amount of influence a node has over the flow of information in a graph. It is often used to find nodes that serve as a bridge from one part of a graph to another" [4].

The algorithm has a logic based upon the calculation of the unweighted shortest paths between nodes, regarding all the pairs of nodes. This gives each node a calculated score, with the highest being the most influential node in this problem specifically.

As section V-2, the first step was to create a graph, considering each student's friendship with others and subsequently make a calculation via the call of the algorithm, appending each node Id to a score, and associating this two variables to each student's name. The last step was to associate a new property to each instance with the label "Student", taking in account the created graph with the name "between1".

```
CALL gds.graph.create('between1', 'Student',
    {FRIENDS: {orientation: 'UNDIRECTED'}})

CALL gds.betweenness.stream('between1')
YIELD nodeId, score
RETURN gds.util.asNode(nodeId).name
AS name, score
ORDER BY score DESC

CALL gds.betweenness.write('between1',
{writeProperty: 'friendsBetweenness'})
```

The results 6 were satisfactory, showing the most influential students as it should.

Fig. 6. Result of performing Betweenness

4) Page Rank: The last algorithm used to deal with this problem is the second centrality algorithm, named "Page Rank". It consists of measuring "the importance of each node within the graph, based on the number incoming relationships and the importance of the corresponding source nodes. The underlying assumption roughly speaking is that a page is only as important as the pages that link to it." [5] and resolves the following equation 7:

$$PR(A) = (1-d) + d(\frac{PR(T_1)}{C(T_1)} + ... + \frac{PR(T_n)}{C(T_n)})$$

Fig. 7. Page Rank equation [5]

The objective in our domain was to obtain the most influential nodes with the label "Student", meaning that there was a big possibility of the nodes with the highest score being linked with suspicious relationships.

The logic was similar to section V-3, with the creation of a graph named "friends" and then perform the call of it, using Page Rank over the sub-set of data retrieved by the graph. Once again, the node ids were taken in account, ordering the records by their scores and showing us the names of each student.

After that, and as shown in 8, were retrieved the same results with weighted degrees attached to it and, similarly to the actions taken in V-3 and **??**, a property referring to that score was injected into each node.

```
call gds.graph.create('friends', 'Student',
{
FRIENDS: {
        orientation: 'UNDIRECTED'
}
})

CALL gds.pageRank.stream('friends')
YIELD nodeId, score
RETURN gds.util.asNode(nodeId).name
AS name, score
ORDER BY score DESC LIMIT 10

CALL gds.pageRank.stream('friends')
YIELD nodeId, score AS pageRank
with gds.util.asNode(nodeId) AS n, pageRank
match (n)-[i:FRIENDS]-()
Return n.name AS name, pageRank, count(1)
AS degree, sum(i.weight) AS weightedDegree
ORDER BY pageRank DESC LIMIT 10

CALL gds.pageRank.write('friends',
{writeProperty: 'friendsPageRank'})
```

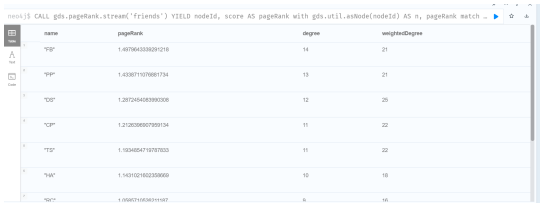

Fig. 8. Page Rank with weighted degree

VI. Conclusion

In this work, an embryonic idea on how some relationships between students, as well as their behaviour can be used to infer academic fraud was built and tested successfully, with satisfactory results.

It was designed a meta-model and data was inserted to test its robustness, and after that some queries and algorithms were performed to prove that a standardization of such a problem can be done.

It was concluded that behaviours like not attending to classes, show low interest during an assignment presentation, have much less score in the test than the medium mark of the group, as well as being part of a lot of groups with the same people over and over again, having less score than them, can indicate fraudulent activity.

Nevertheless, this is only an initial idea of a very complex problem and, as future work, the inclusion of a concept like time could improve the present results, as students knowing each other throughout years could be crucial to the perform of acts like the impersonating fraud.

The notion of teacher and how much easier it is to copy with some teachers than with others, or even sharing subjects that are teached by the same person could be, among others, some really valuable variables added to the system.

References

[1] Your Money. 2021. A History Of Fraud Through The Ages And How To Avoid Being A Victim - Your Money. [online] Available at: <https://www.yourmoney.com/credit-cards-loans/a-history-of-fraud-through-the-ages-and-how-to-avoid-being-a-victim/> [Accessed 12 November 2020].

[2] Neo4j.com. 2021. Chapter 6. Algorithms - The Neo4j Graph Data Science Library Manual V1.4. [online] Available at: <https://neo4j.com/docs/graph-data-science/current/algorithms/> [Accessed 20 December 2020].

[3] Neo4j.com. 2021. 6.3.2. Label Propagation - 6.3. Community Detection Algorithms. [online] Available at: <https://neo4j.com/docs/graph-data-science/current/algorithms/label-propagation/> [Accessed 10 January 2021].

[4] Neo4j.com. 2021. 6.2.2. Betweenness Centrality - 6.2. Centrality Algorithms. [online] Available at: <https://neo4j.com/docs/graph-data-science/current/algorithms/betweenness-centrality/> [Accessed 20 January 2021].

[5] Neo4j.com. 2021. 6.2.1. Pagerank - 6.2. Centrality Algorithms. [online] Available at: <https://neo4j.com/docs/graph-data-science/current/algorithms/page-rank/> [Accessed 23 January 2021].

Apêndice C

Código fonte de criação do modelo inicial

```
// DISCIPLINAS
match (a:Student {name: 'FL'}),
(b:Student {name: 'RC'}),
(c:Student {name: 'RG'}),
(d:Student {name: 'FB'}),
(e:Student {name: 'PP'}),
(f:Student {name: 'RO'}),
(g:Student {name: 'HA'}),
(h:Student {name: 'AG'}),
(i:Student {name: 'AC'}),
(j:Student {name: 'DC'}),
(k:Student {name: 'DS'}),
(l:Student {name: 'CP'}),
(m:Student {name: 'TM'}),
(n:Student {name: 'RV'}),
(o:Student {name: 'IM'}),
```

```
(p:Student {name: 'MRS'}),
(q:Student {name: 'HS'}),
(r:Student {name: 'FT'}),
(s:Student {name: 'PF'}),
(t:Student {name: 'LS'}),
(u:Student {name: 'TS'}),
(as:Subject {name: 'Distributed Systems'}),
(bs:Subject {name: 'Web Development'}),
(cs:Subject {name: 'Semantic Web'})
merge (a)-[:ATTENDS {presences: 5}]->(as)
merge (a)-[:ATTENDS {presences: 7}]->(bs)
merge (b)-[:ATTENDS {presences: 10}]->(bs)
merge (b)-[:ATTENDS {presences: 11}]->(cs)
merge (c)-[:ATTENDS {presences: 25}]->(as)
merge (c)-[:ATTENDS {presences: 20}]->(cs)
merge (d)-[:ATTENDS {presences: 15}]->(bs)
merge (d)-[:ATTENDS {presences: 20}]->(cs)
merge (e)-[:ATTENDS {presences: 17}]->(as)
merge (e)-[:ATTENDS {presences: 15}]->(cs)
merge (f)-[:ATTENDS {presences: 27}]->(as)
merge (f)-[:ATTENDS {presences: 30}]->(bs)
merge (g)-[:ATTENDS {presences: 28}]->(as)
merge (g)-[:ATTENDS {presences: 30}]->(bs)
merge (g)-[:ATTENDS {presences: 25}]->(cs)
merge (h)-[:ATTENDS {presences: 8}]->(as)
merge (h)-[:ATTENDS {presences: 10}]->(cs)
merge (i)-[:ATTENDS {presences: 21}]->(as)
merge (i)-[:ATTENDS {presences: 24}]->(bs)
merge (j)-[:ATTENDS {presences: 21}]->(bs)
```

```
merge (j)-[:ATTENDS {presences: 24}]->(cs)
merge (k)-[:ATTENDS {presences: 23}]->(as)
merge (k)-[:ATTENDS {presences: 25}]->(bs)
merge (k)-[:ATTENDS {presences: 24}]->(cs)
merge (l)-[:ATTENDS {presences: 23}]->(as)
merge (l)-[:ATTENDS {presences: 25}]->(bs)
merge (l)-[:ATTENDS {presences: 24}]->(cs)
merge (m)-[:ATTENDS {presences: 1}]->(bs)
merge (m)-[:ATTENDS {presences: 2}]->(cs)
merge (n)-[:ATTENDS {presences: 21}]->(bs)
merge (n)-[:ATTENDS {presences: 22}]->(cs)
merge (o)-[:ATTENDS {presences: 21}]->(as)
merge (o)-[:ATTENDS {presences: 22}]->(bs)
merge (o)-[:ATTENDS {presences: 27}]->(cs)
merge (p)-[:ATTENDS {presences: 30}]->(as)
merge (p)-[:ATTENDS {presences: 30}]->(bs)
merge (q)-[:ATTENDS {presences: 30}]->(as)
merge (q)-[:ATTENDS {presences: 30}]->(bs)
merge (r)-[:ATTENDS {presences: 10}]->(as)
merge (r)-[:ATTENDS {presences: 9}]->(bs)
merge (r)-[:ATTENDS {presences: 9}]->(cs)
merge (s)-[:ATTENDS {presences: 27}]->(as)
merge (s)-[:ATTENDS {presences: 21}]->(cs)
merge (t)-[:ATTENDS {presences: 30}]->(bs)
merge (t)-[:ATTENDS {presences: 30}]->(cs)
merge (u)-[:ATTENDS {presences: 30}]->(cs)
merge (u)-[:ATTENDS {presences: 25}]->(as)
merge (u)-[:ATTENDS {presences: 27}]->(bs)
```

```
match (a:Assignement {designation: 'SD Assignement 1'}),
(b:Assignement {designation: 'SD Assignement 2'}),
(c:Assignement {designation: 'DW Assignement 1'}),
(d:Assignement {designation: 'DW Assignement 2'}),
(e:Assignement {designation: 'WS Assignement 1'}),
(f:Assignement {designation: 'WS Assignement 2'}),
(as:Subject {name: 'Distributed Systems'}),
(bs:Subject {name: 'Web Development'}),
(cs:Subject {name: 'Semantic Web'})
merge (a)-[:PART_OF {weight: 0.3}]->(as)
merge (b)-[:PART_OF {weight: 0.4}]->(as)
merge (c)-[:PART_OF {weight: 0.4}]->(bs)
merge (d)-[:PART_OF {weight: 0.4}]->(bs)
merge (e)-[:PART_OF {weight: 0.3}]->(cs)
merge (f)-[:PART_OF {weight: 0.3}]->(cs)
```

```
//GRUPOS/DISCIPLINAS
match (a:Group {designation: 'SDA1 Group 1'}),
(b:Group {designation: 'SDA1 Group 2'}),
(c:Group {designation: 'SDA1 Group 3'}),
(d:Group {designation: 'SDA1 Group 4'}),

(e:Group {designation: 'SDA2 Group 1'}),
(f:Group {designation: 'SDA2 Group 2'}),
(g:Group {designation: 'SDA2 Group 3'}),
(h:Group {designation: 'SDA2 Group 4'}),
```

```
(i:Group {designation: 'DWA1 Group 1'}),
(j:Group {designation: 'DWA1 Group 2'}),
(k:Group {designation: 'DWA1 Group 3'}),
(l:Group {designation: 'DWA1 Group 4'}),

(m:Group {designation: 'DWA2 Group 1'}),
(n:Group {designation: 'DWA2 Group 2'}),
(o:Group {designation: 'DWA2 Group 3'}),
(p:Group {designation: 'DWA2 Group 4'}),

(q:Group {designation: 'WSA1 Group 1'}),
(r:Group {designation: 'WSA1 Group 2'}),
(s:Group {designation: 'WSA1 Group 3'}),
(t:Group {designation: 'WSA1 Group 4'}),

(u:Group {designation: 'WSA2 Group 1'}),
(v:Group {designation: 'WSA2 Group 2'}),
(w:Group {designation: 'WSA2 Group 3'}),
(x:Group {designation: 'WSA2 Group 4'}),

(sda1:Assignement {designation: 'SD Assignement 1'}),
(sda2:Assignement {designation: 'SD Assignement 2'}),
(dwa1:Assignement {designation: 'DW Assignement 1'}),
(dwa2:Assignement {designation: 'DW Assignement 2'}),
(wsa1:Assignement {designation: 'WS Assignement 1'}),
(wsa2:Assignement {designation: 'WS Assignement 2'})
merge (a)-[:BELONGS_TO {mark: 14.3}]->(sda1)
merge (b)-[:BELONGS_TO {mark: 12.4}]->(sda1)
```

```
merge (c)-[:BELONGS_TO {mark: 17.4}]->(sda1)
merge (d)-[:BELONGS_TO {mark: 15}]->(sda1)

merge (e)-[:BELONGS_TO {mark: 12.9}]->(sda2)
merge (f)-[:BELONGS_TO {mark: 14}]->(sda2)
merge (g)-[:BELONGS_TO {mark: 19}]->(sda2)
merge (h)-[:BELONGS_TO {mark: 14.6}]->(sda2)

merge (i)-[:BELONGS_TO {mark: 8.9}]->(dwa1)
merge (j)-[:BELONGS_TO {mark: 17}]->(dwa1)
merge (k)-[:BELONGS_TO {mark: 18.9}]->(dwa1)
merge (l)-[:BELONGS_TO {mark: 15}]->(dwa1)

merge (m)-[:BELONGS_TO {mark: 14.1}]->(dwa2)
merge (n)-[:BELONGS_TO {mark: 18.4}]->(dwa2)
merge (o)-[:BELONGS_TO {mark: 17.4}]->(dwa2)
merge (p)-[:BELONGS_TO {mark: 12.5}]->(dwa2)

merge (q)-[:BELONGS_TO {mark: 11.7}]->(wsa1)
merge (r)-[:BELONGS_TO {mark: 12.5}]->(wsa1)
merge (s)-[:BELONGS_TO {mark: 12.9}]->(wsa1)
merge (t)-[:BELONGS_TO {mark: 14.7}]->(wsa1)

merge (u)-[:BELONGS_TO {mark: 15.8}]->(wsa2)
merge (v)-[:BELONGS_TO {mark: 11.5}]->(wsa2)
merge (w)-[:BELONGS_TO {mark: 11}]->(wsa2)
merge (x)-[:BELONGS_TO {mark: 17.4}]->(wsa2)
```

```
// TESTES/ DISCIPLINAS

match (a:Test {designation: 'Multiple Choice SD'}),
(b:Test {designation: 'Theoretical DW'}),
(c:Test {designation: 'Multiple Choice WS'}),
(as:Subject {name: 'Distributed Systems'}),
(bs:Subject {name: 'Web Development'}),
(cs:Subject {name: 'Semantic Web'})

merge (a)-[:TEST_OF {minimum_mark: 7.0, weight: 0.3}]->(as)
merge (b)-[:TEST_OF {weight: 0.2}]->(bs)
merge (c)-[:TEST_OF {minimum_mark: 7.0, weight: 0.4}]->(cs)



// TESTES/ ALUNOS
match (a:Student {name: 'FL'}),
(b:Student {name: 'RC'}),
(c:Student {name: 'RG'}),
(d:Student {name: 'FB'}),
(e:Student {name: 'PP'}),
(f:Student {name: 'RO'}),
(g:Student {name: 'HA'}),
(h:Student {name: 'AG'}),
```

```
(i:Student {name: 'AC'}),
(j:Student {name: 'DC'}),
(k:Student {name: 'DS'}),
(l:Student {name: 'CP'}),
(m:Student {name: 'TM'}),
(n:Student {name: 'RV'}),
(o:Student {name: 'IM'}),
(p:Student {name: 'MRS'}),
(q:Student {name: 'HS'}),
(r:Student {name: 'FT'}),
(s:Student {name: 'PF'}),
(t:Student {name: 'LS'}),
(u:Student {name: 'TS'}),
(as:Test {designation: 'Multiple Choice SD'}),
(bs:Test {designation: 'Theoretical DW'}),
(cs:Test {designation: 'Multiple Choice WS'})
merge (a)-[:PARTICIPATED {mark: 7.0}]->(as)
merge (a)-[:PARTICIPATED {mark: 7.1}]->(bs)
merge (b)-[:PARTICIPATED {mark: 7.0}]->(bs)
merge (b)-[:PARTICIPATED {mark: 10.5}]->(cs)
merge (c)-[:PARTICIPATED {mark: 15.1}]->(as)
merge (c)-[:PARTICIPATED {mark: 12.8}]->(cs)
merge (d)-[:PARTICIPATED {mark: 12.1}]->(bs)
merge (d)-[:PARTICIPATED {mark: 12.3}]->(cs)
merge (e)-[:PARTICIPATED {mark: 18.9}]->(as)
merge (e)-[:PARTICIPATED {mark: 18.1}]->(cs)
merge (f)-[:PARTICIPATED {mark: 17.3}]->(as)
merge (f)-[:PARTICIPATED {mark: 15.3}]->(bs)
merge (g)-[:PARTICIPATED {mark: 13.1}]->(as)
```

```
merge (g)-[:PARTICIPATED {mark: 10.1}]->(bs)
merge (g)-[:PARTICIPATED {mark: 11.2}]->(cs)
merge (h)-[:PARTICIPATED {mark: 7.0}]->(as)
merge (h)-[:PARTICIPATED {mark: 10.0}]->(cs)
merge (i)-[:PARTICIPATED {mark: 7.2}]->(as)
merge (i)-[:PARTICIPATED {mark: 5.9}]->(bs)
merge (j)-[:PARTICIPATED {mark: 10.8}]->(bs)
merge (j)-[:PARTICIPATED {mark: 15.1}]->(cs)
merge (k)-[:PARTICIPATED {mark: 16.1}]->(as)
merge (k)-[:PARTICIPATED {mark: 17.2}]->(bs)
merge (k)-[:PARTICIPATED {mark: 15.8}]->(cs)
merge (l)-[:PARTICIPATED {mark: 12.1}]->(as)
merge (l)-[:PARTICIPATED {mark: 9.5}]->(bs)
merge (l)-[:PARTICIPATED {mark: 13.1}]->(cs)
merge (m)-[:PARTICIPATED {mark: 7.0}]->(bs)
merge (m)-[:PARTICIPATED {mark: 11.0}]->(cs)
merge (n)-[:PARTICIPATED {mark: 8.9}]->(bs)
merge (n)-[:PARTICIPATED {mark: 5.6}]->(cs)
merge (o)-[:PARTICIPATED {mark: 17.1}]->(as)
merge (o)-[:PARTICIPATED {mark: 11.0}]->(bs)
merge (o)-[:PARTICIPATED {mark: 7.1}]->(cs)
merge (p)-[:PARTICIPATED {mark: 19.1}]->(as)
merge (p)-[:PARTICIPATED {mark: 18.5}]->(bs)
merge (q)-[:PARTICIPATED {mark: 18.2}]->(as)
merge (q)-[:PARTICIPATED {mark: 17.1}]->(bs)
merge (r)-[:PARTICIPATED {mark: 6.1}]->(as)
merge (r)-[:PARTICIPATED {mark: 17.1}]->(bs)
merge (r)-[:PARTICIPATED {mark: 12.1}]->(cs)
merge (s)-[:PARTICIPATED {mark: 12.8}]->(as)
```

```
merge (s)-[:PARTICIPATED {mark: 12.5}]->(cs)
merge (t)-[:PARTICIPATED {mark: 15.5}]->(as)
merge (t)-[:PARTICIPATED {mark: 15.0}]->(cs)
merge (u)-[:PARTICIPATED {mark: 15.1}]->(as)
merge (u)-[:PARTICIPATED {mark: 17.1}]->(bs)
merge (u)-[:PARTICIPATED {mark: 17.7}]->(cs)


match (a:Student {name: 'FL'}),
(b:Student {name: 'RC'}),
(c:Student {name: 'RG'}),
(d:Student {name: 'FB'}),

(e:Student {name: 'PP'}),
(f:Student {name: 'RO'}),
(g:Student {name: 'HA'}),
(h:Student {name: 'AG'}),

(i:Student {name: 'AC'}),
(j:Student {name: 'DC'}),
(k:Student {name: 'DS'}),
(l:Student {name: 'CP'}),

(m:Student {name: 'TM'}),
(n:Student {name: 'RV'}),
(o:Student {name: 'IM'}),
(p:Student {name: 'MRS'}),

(q:Student {name: 'HS'}),
```

```
(r:Student {name: 'FT'}),
(s:Student {name: 'PF'}),
(t:Student {name: 'LS'}),

(u:Student {name: 'TS'}),

(ga:Group {designation: 'SDA1 Group 1'}),
(gb:Group {designation: 'SDA1 Group 2'}),
(gc:Group {designation: 'SDA1 Group 3'}),
(gd:Group {designation: 'SDA1 Group 4'}),

(ge:Group {designation: 'SDA2 Group 1'}),
(gf:Group {designation: 'SDA2 Group 2'}),
(gg:Group {designation: 'SDA2 Group 3'}),
(gh:Group {designation: 'SDA2 Group 4'}),

(gi:Group {designation: 'DWA1 Group 1'}),
(gj:Group {designation: 'DWA1 Group 2'}),
(gk:Group {designation: 'DWA1 Group 3'}),
(gl:Group {designation: 'DWA1 Group 4'}),

(gm:Group {designation: 'DWA2 Group 1'}),
(gn:Group {designation: 'DWA2 Group 2'}),
(go:Group {designation: 'DWA2 Group 3'}),
(gp:Group {designation: 'DWA2 Group 4'}),

(gq:Group {designation: 'WSA1 Group 1'}),
(gr:Group {designation: 'WSA1 Group 2'}),
(gs:Group {designation: 'WSA1 Group 3'}),
```

```
(gt:Group {designation: 'WSA1 Group 4'}),

(gu:Group {designation: 'WSA2 Group 1'}),
(gv:Group {designation: 'WSA2 Group 2'}),
(gw:Group {designation: 'WSA2 Group 3'}),
(gx:Group {designation: 'WSA2 Group 4'})

merge (u)-[:GROUPED_IN {showed_interest: true}]->(ga)
merge (u)-[:GROUPED_IN {showed_interest: true}]->(ge)
merge (u)-[:GROUPED_IN {showed_interest: true}]->(gi)
merge (u)-[:GROUPED_IN {showed_interest: true}]->(gm)
merge (u)-[:GROUPED_IN {showed_interest: true}]->(gq)
merge (u)-[:GROUPED_IN {showed_interest: true}]->(gu)

merge (a)-[:GROUPED_IN {showed_interest: true}]->(gd)
merge (a)-[:GROUPED_IN {showed_interest: true}]->(gg)
merge (a)-[:GROUPED_IN {showed_interest: false}]->(gk)
merge (a)-[:GROUPED_IN {showed_interest: false}]->(gp)

merge (c)-[:GROUPED_IN {showed_interest: false}]->(gc)
merge (c)-[:GROUPED_IN {showed_interest: false}]->(gh)
merge (c)-[:GROUPED_IN {showed_interest: false}]->(gr)
merge (c)-[:GROUPED_IN {showed_interest: false}]->(gw)

merge (e)-[:GROUPED_IN {showed_interest: true}]->(gb)
merge (e)-[:GROUPED_IN {showed_interest: true}]->(gf)
merge (e)-[:GROUPED_IN {showed_interest: true}]->(gt)
merge (e)-[:GROUPED_IN {showed_interest: true}]->(gx)
```

```
merge (f)-[:GROUPED_IN {showed_interest: true}]->(gc)
merge (f)-[:GROUPED_IN {showed_interest: true}]->(gg)
merge (f)-[:GROUPED_IN {showed_interest: true}]->(gl)
merge (f)-[:GROUPED_IN {showed_interest: true}]->(go)

merge (g)-[:GROUPED_IN {showed_interest: true}]->(ga)
merge (g)-[:GROUPED_IN {showed_interest: true}]->(ge)
merge (g)-[:GROUPED_IN {showed_interest: true}]->(gi)
merge (g)-[:GROUPED_IN {showed_interest: true}]->(gm)
merge (g)-[:GROUPED_IN {showed_interest: true}]->(gq)
merge (g)-[:GROUPED_IN {showed_interest: true}]->(gu)

merge (h)-[:GROUPED_IN {showed_interest: false}]->(gb)
merge (h)-[:GROUPED_IN {showed_interest: false}]->(gg)
merge (h)-[:GROUPED_IN {showed_interest: false}]->(gt)
merge (h)-[:GROUPED_IN {showed_interest: false}]->(gx)

merge (i)-[:GROUPED_IN {showed_interest: true}]->(gc)
merge (i)-[:GROUPED_IN {showed_interest: true}]->(gg)
merge (i)-[:GROUPED_IN {showed_interest: true}]->(gk)
merge (i)-[:GROUPED_IN {showed_interest: false}]->(gp)

merge (k)-[:GROUPED_IN {showed_interest: true}]->(ga)
merge (k)-[:GROUPED_IN {showed_interest: true}]->(ge)
merge (k)-[:GROUPED_IN {showed_interest: true}]->(gi)
merge (k)-[:GROUPED_IN {showed_interest: true}]->(gm)
merge (k)-[:GROUPED_IN {showed_interest: true}]->(gq)
merge (k)-[:GROUPED_IN {showed_interest: true}]->(gu)
```

```
merge (l)-[:GROUPED_IN {showed_interest: true}]->(ga)
merge (l)-[:GROUPED_IN {showed_interest: true}]->(ge)
merge (l)-[:GROUPED_IN {showed_interest: true}]->(gi)
merge (l)-[:GROUPED_IN {showed_interest: true}]->(gm)
merge (l)-[:GROUPED_IN {showed_interest: true}]->(gq)
merge (l)-[:GROUPED_IN {showed_interest: true}]->(gu)

merge (o)-[:GROUPED_IN {showed_interest: true}]->(gb)
merge (o)-[:GROUPED_IN {showed_interest: true}]->(gf)
merge (o)-[:GROUPED_IN {showed_interest: true}]->(gj)
merge (o)-[:GROUPED_IN {showed_interest: true}]->(gp)
merge (o)-[:GROUPED_IN {showed_interest: true}]->(gr)
merge (o)-[:GROUPED_IN {showed_interest: true}]->(gv)

merge (p)-[:GROUPED_IN {showed_interest: true}]->(gd)
merge (p)-[:GROUPED_IN {showed_interest: true}]->(gh)
merge (p)-[:GROUPED_IN {showed_interest: true}]->(gj)
merge (p)-[:GROUPED_IN {showed_interest: true}]->(gn)

merge (q)-[:GROUPED_IN {showed_interest: true}]->(gd)
merge (q)-[:GROUPED_IN {showed_interest: true}]->(gh)
merge (q)-[:GROUPED_IN {showed_interest: true}]->(gj)
merge (q)-[:GROUPED_IN {showed_interest: true}]->(gn)

merge (r)-[:GROUPED_IN {showed_interest: false}]->(gb)
merge (r)-[:GROUPED_IN {showed_interest: false}]->(gf)
merge (r)-[:GROUPED_IN {showed_interest: false}]->(gj)
merge (r)-[:GROUPED_IN {showed_interest: false}]->(gp)
merge (r)-[:GROUPED_IN {showed_interest: false}]->(gr)
```

```
merge (r)-[:GROUPED_IN {showed_interest: false}]->(gv)

merge (s)-[:GROUPED_IN {showed_interest: true}]->(gc)
merge (s)-[:GROUPED_IN {showed_interest: true}]->(gh)
merge (s)-[:GROUPED_IN {showed_interest: true}]->(gs)
merge (s)-[:GROUPED_IN {showed_interest: true}]->(gw)

merge (t)-[:GROUPED_IN {showed_interest: true}]->(gd)
merge (t)-[:GROUPED_IN {showed_interest: true}]->(gf)
merge (t)-[:GROUPED_IN {showed_interest: true}]->(gr)
merge (t)-[:GROUPED_IN {showed_interest: true}]->(gw)

merge (b)-[:GROUPED_IN {showed_interest: false}]->(gl)
merge (b)-[:GROUPED_IN {showed_interest: false}]->(go)
merge (b)-[:GROUPED_IN {showed_interest: true}]->(gt)
merge (b)-[:GROUPED_IN {showed_interest: true}]->(gw)

merge (d)-[:GROUPED_IN {showed_interest: false}]->(gl)
merge (d)-[:GROUPED_IN {showed_interest: false}]->(go)
merge (d)-[:GROUPED_IN {showed_interest: true}]->(gt)
merge (d)-[:GROUPED_IN {showed_interest: true}]->(gx)

merge (j)-[:GROUPED_IN {showed_interest: true}]->(gk)
merge (j)-[:GROUPED_IN {showed_interest: false}]->(gn)
merge (j)-[:GROUPED_IN {showed_interest: true}]->(gs)
merge (j)-[:GROUPED_IN {showed_interest: false}]->(gv)

merge (m)-[:GROUPED_IN {showed_interest: false}]->(gl)
merge (m)-[:GROUPED_IN {showed_interest: false}]->(go)
```

```
merge (m)-[:GROUPED_IN {showed_interest: false}]->(gs)
merge (m)-[:GROUPED_IN {showed_interest: false}]->(gx)

merge (n)-[:GROUPED_IN {showed_interest: true}]->(gk)
merge (n)-[:GROUPED_IN {showed_interest: true}]->(gn)
merge (n)-[:GROUPED_IN {showed_interest: true}]->(gs)
merge (n)-[:GROUPED_IN {showed_interest: false}]->(gv)
```

Relação (conceptual) de fraude

```
match (a:Student {name: 'FL'}),
(b:Student {name: 'TS'}),
(c:Student {name: 'RC'}),
(d:Student {name: 'AG'}),
(e:Student {name: 'DC'}),
(f:Student {name: 'TM'}),
(g:Student {name: 'FT'}),
(h:Student {name: 'PP'}),
(i:Student {name: 'IM'}),
(j:Student {name: 'DS'})
merge (a)-[:FRAUD {suspicion: 'strong'}]->(b)
merge (c)-[:FRAUD {suspicion: 'strong'}]->(h)
merge (d)-[:FRAUD {suspicion: 'strong'}]->(h)
merge (e)-[:FRAUD {suspicion: 'weak'}]->(j)
merge (f)-[:FRAUD {suspicion: 'strong'}]->(j)
merge (g)-[:FRAUD {suspicion: 'strong'}]->(i)
```

Relação "Friend"

```
match (a:Student {name: 'TS'}),
```

```
(b:Student {name: 'RG'})
merge (a)-[:FRIENDS {since: 2017}]-(b)

match (a:Student {name: 'TS'}),
(b:Student {name: 'FB'})
merge (a)-[:FRIENDS {since: 2015}]-(b)

match (a:Student {name: 'TS'}),
(b:Student {name: 'PP'})
merge (a)-[:FRIENDS {since: 2015}]-(b)

match (a:Student {name: 'TS'}),
(b:Student {name: 'HA'})
merge (a)-[:FRIENDS {since: 2017}]-(b)

match (a:Student {name: 'TS'}),
(b:Student {name: 'DS'})
merge (a)-[:FRIENDS {since: 2010}]-(b)

match (a:Student {name: 'TS'}),
(b:Student {name: 'CP'})
merge (a)-[:ENGAGED {since: 2020}]-(b)

match (a:Student {name: 'TS'}),
(b:Student {name: 'IM'})
merge (a)-[:FRIENDS {since: 2009}]-(b)

match (a:Student {name: 'TS'}),
(b:Student {name: 'MRS'})
```

```
merge (a)-[:FRIENDS {since: 1993}]-(b)

match (a:Student {name: 'TS'}),
(b:Student {name: 'HS'})
merge (a)-[:FRIENDS {since: 1993}]-(b)

match (a:Student {name: 'FL'}),
(b:Student {name: 'RG'})
merge (a)-[:FRIENDS {since: 2018}]-(b)

match (a:Student {name: 'FL'}),
(b:Student {name: 'HA'})
merge (a)-[:FRIENDS {since: 2019}]-(b)

match (a:Student {name: 'FL'}),
(b:Student {name: 'AG'})
merge (a)-[:FRIENDS {since: 2015}]-(b)

match (a:Student {name: 'FL'}),
(b:Student {name: 'PF'})
merge (a)-[:FRIENDS {since: 2017}]-(b)

match (a:Student {name: 'FL'}),
(b:Student {name: 'RC'})
merge (a)-[:FRIENDS {since: 2019}]-(b)

match (a:Student {name: 'RC'}),
(b:Student {name: 'FB'})
merge (a)-[:FRIENDS {since: 2015}]-(b)
```

```
match (a:Student {name: 'RC'}),
(b:Student {name: 'PP'})
merge (a)-[:FRIENDS {since: 2015}]-(b)

match (a:Student {name: 'RC'}),
(b:Student {name: 'AG'})
merge (a)-[:FRIENDS {since: 2018}]-(b)

match (a:Student {name: 'RC'}),
(b:Student {name: 'HA'})
merge (a)-[:FRIENDS {since: 2017}]-(b)

match (a:Student {name: 'RC'}),
(b:Student {name: 'DC'})
merge (a)-[:FRIENDS {since: 2016}]-(b)

match (a:Student {name: 'RC'}),
(b:Student {name: 'DS'})
merge (a)-[:FRIENDS {since: 2017}]-(b)

match (a:Student {name: 'RC'}),
(b:Student {name: 'PF'})
merge (a)-[:FRIENDS {since: 2019}]-(b)

match (a:Student {name: 'RC'}),
(b:Student {name: 'RG'})
merge (a)-[:FRIENDS {since: 2019}]-(b)
```

```
match (a:Student {name: 'RG'}),
(b:Student {name: 'RO'})
merge (a)-[:FRIENDS {since: 2019}]-(b)

match (a:Student {name: 'RG'}),
(b:Student {name: 'HA'})
merge (a)-[:FRIENDS {since: 2018}]-(b)

match (a:Student {name: 'RG'}),
(b:Student {name: 'AG'})
merge (a)-[:FRIENDS {since: 2016}]-(b)

match (a:Student {name: 'RG'}),
(b:Student {name: 'PF'})
merge (a)-[:FRIENDS {since: 2010}]-(b)

match (a:Student {name: 'TS'}),
(b:Student {name: 'CP'})
merge (a)-[:FRIENDS {since: 2014}]-(b)

match (a:Student {name: 'FB'}),
(b:Student {name: 'PP'})
merge (a)-[:FRIENDS {since: 2014}]-(b)

match (a:Student {name: 'FB'}),
(b:Student {name: 'HA'})
merge (a)-[:FRIENDS {since: 2017}]-(b)

match (a:Student {name: 'FB'}),
```

```
(b:Student {name: 'AC'})
merge (a)-[:FRIENDS {since: 2017}]-(b)

match (a:Student {name: 'FB'}),
(b:Student {name: 'DC'})
merge (a)-[:FRIENDS {since: 2015}]-(b)

match (a:Student {name: 'FB'}),
(b:Student {name: 'DS'})
merge (a)-[:FRIENDS {since: 2014}]-(b)

match (a:Student {name: 'FB'}),
(b:Student {name: 'TM'})
merge (a)-[:FRIENDS {since: 2019}]-(b)

match (a:Student {name: 'FB'}),
(b:Student {name: 'RV'})
merge (a)-[:FRIENDS {since: 2017}]-(b)

match (a:Student {name: 'FB'}),
(b:Student {name: 'CP'})
merge (a)-[:FRIENDS {since: 2019}]-(b)

match (a:Student {name: 'PP'}),
(b:Student {name: 'RO'})
merge (a)-[:FRIENDS {since: 2019}]-(b)

match (a:Student {name: 'PP'}),
(b:Student {name: 'AC'})
```

```
merge (a)-[:FRIENDS {since: 2018}]-(b)

match (a:Student {name: 'PP'}),
(b:Student {name: 'AG'})
merge (a)-[:FRIENDS {since: 2013}]-(b)

match (a:Student {name: 'PP'}),
(b:Student {name: 'TM'})
merge (a)-[:FRIENDS {since: 2019}]-(b)

match (a:Student {name: 'PP'}),
(b:Student {name: 'DS'})
merge (a)-[:FRIENDS {since: 2016}]-(b)

match (a:Student {name: 'PP'}),
(b:Student {name: 'RV'})
merge (a)-[:FRIENDS {since: 2019}]-(b)

match (a:Student {name: 'PP'}),
(b:Student {name: 'MRS'})
merge (a)-[:FRIENDS {since: 2017}]-(b)

match (a:Student {name: 'PP'}),
(b:Student {name: 'HS'})
merge (a)-[:FRIENDS {since: 2017}]-(b)

match (a:Student {name: 'FB'}),
(b:Student {name: 'MRS'})
merge (a)-[:FRIENDS {since: 2017}]-(b)
```

```
match (a:Student {name: 'FB'}),
(b:Student {name: 'HS'})
merge (a)-[:FRIENDS {since: 2017}]-(b)

match (a:Student {name: 'RO'}),
(b:Student {name: 'AC'})
merge (a)-[:FRIENDS {since: 2012}]-(b)

match (a:Student {name: 'RO'}),
(b:Student {name: 'FT'})
merge (a)-[:FRIENDS {since: 2015}]-(b)

match (a:Student {name: 'RO'}),
(b:Student {name: 'PF'})
merge (a)-[:FRIENDS {since: 2019}]-(b)

match (a:Student {name: 'HA'}),
(b:Student {name: 'CP'})
merge (a)-[:FRIENDS {since: 2017}]-(b)

match (a:Student {name: 'HA'}),
(b:Student {name: 'AG'})
merge (a)-[:FRIENDS {since: 2018}]-(b)

match (a:Student {name: 'HA'}),
(b:Student {name: 'DS'})
merge (a)-[:FRIENDS {since: 2017}]-(b)
```

```
match (a:Student {name: 'HA'}),
(b:Student {name: 'TM'})
merge (a)-[:FRIENDS {since: 2020}]-(b)

match (a:Student {name: 'HA'}),
(b:Student {name: 'IM'})
merge (a)-[:FRIENDS {since: 2020}]-(b)

match (a:Student {name: 'AG'}),
(b:Student {name: 'PF'})
merge (a)-[:FRIENDS {since: 2018}]-(b)

match (a:Student {name: 'AC'}),
(b:Student {name: 'PF'})
merge (a)-[:FRIENDS {since: 2019}]-(b)

match (a:Student {name: 'AC'}),
(b:Student {name: 'CP'})
merge (a)-[:FRIENDS {since: 2019}]-(b)

match (a:Student {name: 'DC'}),
(b:Student {name: 'CP'})
merge (a)-[:FRIENDS {since: 2015}]-(b)

match (a:Student {name: 'DC'}),
(b:Student {name: 'DS'})
merge (a)-[:FRIENDS {since: 2016}]-(b)

match (a:Student {name: 'DC'}),
```

```
(b:Student {name: 'RV'})
merge (a)-[:FRIENDS {since: 2010}]-(b)

match (a:Student {name: 'DC'}),
(b:Student {name: 'RV'})
merge (a)-[:ENGAGED {since: 2014}]-(b)

match (a:Student {name: 'DC'}),
(b:Student {name: 'IM'})
merge (a)-[:FRIENDS {since: 2017}]-(b)

match (a:Student {name: 'DC'}),
(b:Student {name: 'FT'})
merge (a)-[:FRIENDS {since: 2017}]-(b)

match (a:Student {name: 'DS'}),
(b:Student {name: 'HS'})
merge (a)-[:FRIENDS {since: 2014}]-(b)

match (a:Student {name: 'DS'}),
(b:Student {name: 'MRS'})
merge (a)-[:FRIENDS {since: 2014}]-(b)

match (a:Student {name: 'DC'}),
(b:Student {name: 'HS'})
merge (a)-[:FRIENDS {since: 2014}]-(b)

match (a:Student {name: 'DC'}),
(b:Student {name: 'MRS'})
```

```
merge (a)-[:FRIENDS {since: 2014}]-(b)

match (a:Student {name: 'DS'}),
(b:Student {name: 'FT'})
merge (a)-[:FRIENDS {since: 2017}]-(b)

match (a:Student {name: 'DS'}),
(b:Student {name: 'TM'})
merge (a)-[:FRIENDS {since: 2007}]-(b)

match (a:Student {name: 'CP'}),
(b:Student {name: 'HS'})
merge (a)-[:FRIENDS {since: 2019}]-(b)

match (a:Student {name: 'CP'}),
(b:Student {name: 'MRS'})
merge (a)-[:FRIENDS {since: 2019}]-(b)

match (a:Student {name: 'CP'}),
(b:Student {name: 'RV'})
merge (a)-[:FRIENDS {since: 2018}]-(b)

match (a:Student {name: 'CP'}),
(b:Student {name: 'TM'})
merge (a)-[:FRIENDS {since: 2018}]-(b)

match (a:Student {name: 'CP'}),
(b:Student {name: 'FT'})
merge (a)-[:FRIENDS {since: 2017}]-(b)
```

```
match (a:Student {name: 'CP'}),
(b:Student {name: 'IM'})
merge (a)-[:FRIENDS {since: 2016}]-(b)

match (a:Student {name: 'IM'}),
(b:Student {name: 'MRS'})
merge (a)-[:FRIENDS {since: 2011}]-(b)

match (a:Student {name: 'IM'}),
(b:Student {name: 'HS'})
merge (a)-[:FRIENDS {since: 2012}]-(b)

match (a:Student {name: 'IM'}),
(b:Student {name: 'FT'})
merge (a)-[:FRIENDS {since: 2011}]-(b)

match (a:Student {name: 'MRS'}),
(b:Student {name: 'HS'})
merge (a)-[:FRIENDS {since: 1982}]-(b)
```

Apêndice D

Ajustes ao modelo de dados

Grupos e trabalhos

```
match (a:Group {designation: 'SDA1 Group 1'}),
(b:Group {designation: 'SDA1 Group 2'}),
(c:Group {designation: 'SDA1 Group 3'}),
(d:Group {designation: 'SDA1 Group 4'}),

(e:Group {designation: 'SDA2 Group 1'}),
(f:Group {designation: 'SDA2 Group 2'}),
(g:Group {designation: 'SDA2 Group 3'}),
(h:Group {designation: 'SDA2 Group 4'}),

(i:Group {designation: 'DWA1 Group 1'}),
(j:Group {designation: 'DWA1 Group 2'}),
(k:Group {designation: 'DWA1 Group 3'}),
(l:Group {designation: 'DWA1 Group 4'}),

(m:Group {designation: 'DWA2 Group 1'}),
(n:Group {designation: 'DWA2 Group 2'}),
```

```
(o:Group {designation: 'DWA2 Group 3'}),
(p:Group {designation: 'DWA2 Group 4'}),

(q:Group {designation: 'WSA1 Group 1'}),
(r:Group {designation: 'WSA1 Group 2'}),
(s:Group {designation: 'WSA1 Group 3'}),
(t:Group {designation: 'WSA1 Group 4'}),

(u:Group {designation: 'WSA2 Group 1'}),
(v:Group {designation: 'WSA2 Group 2'}),
(w:Group {designation: 'WSA2 Group 3'}),
(x:Group {designation: 'WSA2 Group 4'}),

(sda1:Assignement {designation: 'SD Assignement 1'}),
(sda2:Assignement {designation: 'SD Assignement 2'}),
(dwa1:Assignement {designation: 'DW Assignement 1'}),
(dwa2:Assignement {designation: 'DW Assignement 2'}),
(wsa1:Assignement {designation: 'WS Assignement 1'}),
(wsa2:Assignement {designation: 'WS Assignement 2'})
merge (sda1)-[:AssignmentGroup]->(a)
merge (sda1)-[:AssignmentGroup]->(b)
merge (sda1)-[:AssignmentGroup]->(c)
merge (sda1)-[:AssignmentGroup]->(d)

merge (sda2)-[:AssignmentGroup]->(e)
merge (sda2)-[:AssignmentGroup]->(f)
merge (sda2)-[:AssignmentGroup]->(g)
merge (sda2)-[:AssignmentGroup]->(h)
```

```
merge (dwa1)-[:AssignmentGroup]->(i)
merge (dwa1)-[:AssignmentGroup]->(j)
merge (dwa1)-[:AssignmentGroup]->(k)
merge (dwa1)-[:AssignmentGroup]->(l)

merge (dwa2)-[:AssignmentGroup]->(m)
merge (dwa2)-[:AssignmentGroup]->(n)
merge (dwa2)-[:AssignmentGroup]->(o)
merge (dwa2)-[:AssignmentGroup]->(p)

merge (wsa1)-[:AssignmentGroup]->(q)
merge (wsa1)-[:AssignmentGroup]->(r)
merge (wsa1)-[:AssignmentGroup]->(s)
merge (wsa1)-[:AssignmentGroup]->(t)

merge (wsa2)-[:AssignmentGroup]->(u)
merge (wsa2)-[:AssignmentGroup]->(v)
merge (wsa2)-[:AssignmentGroup]->(w)
merge (wsa2)-[:AssignmentGroup]->(x)
```

Grupos e estudantes

```
match (a:Student {name: 'FL'}),
(b:Student {name: 'RC'}),
(c:Student {name: 'RG'}),
(d:Student {name: 'FB'}),

(e:Student {name: 'PP'}),
(f:Student {name: 'RO'}),
```

```
(g:Student {name: 'HA'}),
(h:Student {name: 'AG'}),

(i:Student {name: 'AC'}),
(j:Student {name: 'DC'}),
(k:Student {name: 'DS'}),
(l:Student {name: 'CP'}),

(m:Student {name: 'TM'}),
(n:Student {name: 'RV'}),
(o:Student {name: 'IM'}),
(p:Student {name: 'MRS'}),

(q:Student {name: 'HS'}),
(r:Student {name: 'FT'}),
(s:Student {name: 'PF'}),
(t:Student {name: 'LS'}),

(u:Student {name: 'TS'}),

(ga:Group {designation: 'SDA1 Group 1'}),
(gb:Group {designation: 'SDA1 Group 2'}),
(gc:Group {designation: 'SDA1 Group 3'}),
(gd:Group {designation: 'SDA1 Group 4'}),

(ge:Group {designation: 'SDA2 Group 1'}),
(gf:Group {designation: 'SDA2 Group 2'}),
(gg:Group {designation: 'SDA2 Group 3'}),
(gh:Group {designation: 'SDA2 Group 4'}),
```

```
(gi:Group {designation: 'DWA1 Group 1'}),
(gj:Group {designation: 'DWA1 Group 2'}),
(gk:Group {designation: 'DWA1 Group 3'}),
(gl:Group {designation: 'DWA1 Group 4'}),

(gm:Group {designation: 'DWA2 Group 1'}),
(gn:Group {designation: 'DWA2 Group 2'}),
(go:Group {designation: 'DWA2 Group 3'}),
(gp:Group {designation: 'DWA2 Group 4'}),

(gq:Group {designation: 'WSA1 Group 1'}),
(gr:Group {designation: 'WSA1 Group 2'}),
(gs:Group {designation: 'WSA1 Group 3'}),
(gt:Group {designation: 'WSA1 Group 4'}),

(gu:Group {designation: 'WSA2 Group 1'}),
(gv:Group {designation: 'WSA2 Group 2'}),
(gw:Group {designation: 'WSA2 Group 3'}),
(gx:Group {designation: 'WSA2 Group 4'})

merge (ga)-[:GroupStudent]->(u)
merge (ge)-[:GroupStudent]->(u)
merge (gi)-[:GroupStudent]->(u)
merge (gm)-[:GroupStudent]->(u)
merge (gq)-[:GroupStudent]->(u)
merge (gu)-[:GroupStudent]->(u)

merge (gd)-[:GroupStudent]->(a)
```

```
merge (gg)-[:GroupStudent]->(a)
merge (gk)-[:GroupStudent]->(a)
merge (gp)-[:GroupStudent]->(a)

merge (gc)-[:GroupStudent]->(c)
merge (gh)-[:GroupStudent]->(c)
merge (gr)-[:GroupStudent]->(c)
merge (gw)-[:GroupStudent]->(c)

merge (gb)-[:GroupStudent]->(e)
merge (gf)-[:GroupStudent]->(e)
merge (gt)-[:GroupStudent]->(e)
merge (gx)-[:GroupStudent]->(e)

merge (gc)-[:GroupStudent]->(f)
merge (gg)-[:GroupStudent]->(f)
merge (gl)-[:GroupStudent]->(f)
merge (go)-[:GroupStudent]->(f)

merge (ga)-[:GroupStudent]->(g)
merge (ge)-[:GroupStudent]->(g)
merge (gi)-[:GroupStudent]->(g)
merge (gm)-[:GroupStudent]->(g)
merge (gq)-[:GroupStudent]->(g)
merge (gu)-[:GroupStudent]->(g)

merge (gb)-[:GroupStudent]->(h)
merge (gg)-[:GroupStudent]->(h)
merge (gt)-[:GroupStudent]->(h)
```

```
merge (gx)-[:GroupStudent]->(h)

merge (gc)-[:GroupStudent]->(i)
merge (gg)-[:GroupStudent]->(i)
merge (gk)-[:GroupStudent]->(i)
merge (gp)-[:GroupStudent]->(i)

merge (ga)-[:GroupStudent]->(k)
merge (ge)-[:GroupStudent]->(k)
merge (gi)-[:GroupStudent]->(k)
merge (gm)-[:GroupStudent]->(k)
merge (gq)-[:GroupStudent]->(k)
merge (gu)-[:GroupStudent]->(k)

merge (ga)-[:GroupStudent]->(l)
merge (ge)-[:GroupStudent]->(l)
merge (gi)-[:GroupStudent]->(l)
merge (gm)-[:GroupStudent]->(l)
merge (gq)-[:GroupStudent]->(l)
merge (gu)-[:GroupStudent]->(l)

merge (gb)-[:GroupStudent]->(o)
merge (gf)-[:GroupStudent]->(o)
merge (gj)-[:GroupStudent]->(o)
merge (gp)-[:GroupStudent]->(o)
merge (gr)-[:GroupStudent]->(o)
merge (gv)-[:GroupStudent]->(o)

merge (gd)-[:GroupStudent]->(p)
```

```
merge (gh)-[:GroupStudent]->(p)
merge (gj)-[:GroupStudent]->(p)
merge (gn)-[:GroupStudent]->(p)

merge (gd)-[:GroupStudent]->(q)
merge (gh)-[:GroupStudent]->(q)
merge (gj)-[:GroupStudent]->(q)
merge (gn)-[:GroupStudent]->(q)

merge (gb)-[:GroupStudent]->(r)
merge (gf)-[:GroupStudent]->(r)
merge (gj)-[:GroupStudent]->(r)
merge (gp)-[:GroupStudent]->(r)
merge (gr)-[:GroupStudent]->(r)
merge (gv)-[:GroupStudent]->(r)

merge (gc)-[:GroupStudent]->(s)
merge (gh)-[:GroupStudent]->(s)
merge (gs)-[:GroupStudent]->(s)
merge (gw)-[:GroupStudent]->(s)

merge (gd)-[:GroupStudent]->(t)
merge (gf)-[:GroupStudent]->(t)
merge (gr)-[:GroupStudent]->(t)
merge (gw)-[:GroupStudent]->(t)

merge (gl)-[:GroupStudent]->(b)
merge (go)-[:GroupStudent]->(b)
merge (gt)-[:GroupStudent]->(b)
```

```
merge (gw)-[:GroupStudent]->(b)

merge (gl)-[:GroupStudent]->(d)
merge (go)-[:GroupStudent]->(d)
merge (gt)-[:GroupStudent]->(d)
merge (gx)-[:GroupStudent]->(d)

merge (gk)-[:GroupStudent]->(j)
merge (gn)-[:GroupStudent]->(j)
merge (gs)-[:GroupStudent]->(j)
merge (gv)-[:GroupStudent]->(j)

merge (gl)-[:GroupStudent]->(m)
merge (go)-[:GroupStudent]->(m)
merge (gs)-[:GroupStudent]->(m)
merge (gx)-[:GroupStudent]->(m)

merge (gk)-[:GroupStudent]->(n)
merge (gn)-[:GroupStudent]->(n)
merge (gs)-[:GroupStudent]->(n)
merge (gv)-[:GroupStudent]->(n)
```

Relação 'InTheSameGroup'

```
match (a:Student {name: 'FL'}),
(b:Student {name: 'RC'}),
(c:Student {name: 'RG'}),
(d:Student {name: 'FB'}),
```

```
(e:Student {name: 'PP'}),
(f:Student {name: 'RO'}),
(g:Student {name: 'HA'}),
(h:Student {name: 'AG'}),
(i:Student {name: 'AC'}),
(j:Student {name: 'DC'}),
(k:Student {name: 'DS'}),
(l:Student {name: 'CP'}),
(m:Student {name: 'TM'}),
(n:Student {name: 'RV'}),
(o:Student {name: 'IM'}),
(p:Student {name: 'MRS'}),
(q:Student {name: 'HS'}),
(r:Student {name: 'FT'}),
(s:Student {name: 'PF'}),
(t:Student {name: 'LS'}),
(u:Student {name: 'TS'})
```

```
merge (u)-[:InTheSameGroup {number_of_times: 6}]-(k)
merge (u)-[:InTheSameGroup {number_of_times: 6}]-(g)
merge (g)-[:InTheSameGroup {number_of_times: 6}]-(k)
merge (g)-[:InTheSameGroup {number_of_times: 6}]-(l)
merge (u)-[:InTheSameGroup {number_of_times: 6}]-(k)
merge (k)-[:InTheSameGroup {number_of_times: 6}]-(l)
merge (o)-[:InTheSameGroup {number_of_times: 6}]-(r)
merge (j)-[:InTheSameGroup {number_of_times: 4}]-(n)
merge (p)-[:InTheSameGroup {number_of_times: 4}]-(q)
merge (a)-[:InTheSameGroup {number_of_times: 3}]-(i)
```

```
merge (b)-[:InTheSameGroup {number_of_times: 3}]-(d)
merge (s)-[:InTheSameGroup {number_of_times: 3}]-(c)
merge (e)-[:InTheSameGroup {number_of_times: 3}]-(h)
merge (d)-[:InTheSameGroup {number_of_times: 3}]-(m)
merge (f)-[:InTheSameGroup {number_of_times: 2}]-(d)
merge (f)-[:InTheSameGroup {number_of_times: 2}]-(b)
merge (f)-[:InTheSameGroup {number_of_times: 2}]-(m)
merge (f)-[:InTheSameGroup {number_of_times: 2}]-(i)
merge (b)-[:InTheSameGroup {number_of_times: 2}]-(m)
merge (c)-[:InTheSameGroup {number_of_times: 2}]-(t)
merge (e)-[:InTheSameGroup {number_of_times: 2}]-(d)
merge (e)-[:InTheSameGroup {number_of_times: 2}]-(r)
merge (e)-[:InTheSameGroup {number_of_times: 2}]-(o)
merge (d)-[:InTheSameGroup {number_of_times: 2}]-(h)
merge (o)-[:InTheSameGroup {number_of_times: 2}]-(t)
merge (r)-[:InTheSameGroup {number_of_times: 2}]-(t)
merge (r)-[:InTheSameGroup {number_of_times: 1}]-(q)
merge (r)-[:InTheSameGroup {number_of_times: 1}]-(i)
merge (r)-[:InTheSameGroup {number_of_times: 1}]-(a)
merge (r)-[:InTheSameGroup {number_of_times: 1}]-(c)
merge (r)-[:InTheSameGroup {number_of_times: 1}]-(h)
merge (r)-[:InTheSameGroup {number_of_times: 1}]-(j)
merge (r)-[:InTheSameGroup {number_of_times: 1}]-(n)
merge (r)-[:InTheSameGroup {number_of_times: 1}]-(p)
merge (p)-[:InTheSameGroup {number_of_times: 1}]-(o)
merge (p)-[:InTheSameGroup {number_of_times: 1}]-(c)
merge (p)-[:InTheSameGroup {number_of_times: 1}]-(s)
merge (p)-[:InTheSameGroup {number_of_times: 1}]-(t)
merge (p)-[:InTheSameGroup {number_of_times: 1}]-(a)
```

```
merge (q)-[:InTheSameGroup {number_of_times: 1}]-(n)
merge (q)-[:InTheSameGroup {number_of_times: 1}]-(j)
merge (q)-[:InTheSameGroup {number_of_times: 1}]-(t)
merge (q)-[:InTheSameGroup {number_of_times: 1}]-(a)
merge (q)-[:InTheSameGroup {number_of_times: 1}]-(c)
merge (q)-[:InTheSameGroup {number_of_times: 1}]-(s)
merge (q)-[:InTheSameGroup {number_of_times: 1}]-(o)
merge (t)-[:InTheSameGroup {number_of_times: 1}]-(a)
merge (t)-[:InTheSameGroup {number_of_times: 1}]-(b)
merge (t)-[:InTheSameGroup {number_of_times: 1}]-(s)
merge (t)-[:InTheSameGroup {number_of_times: 1}]-(e)
merge (a)-[:InTheSameGroup {number_of_times: 1}]-(h)
merge (a)-[:InTheSameGroup {number_of_times: 1}]-(o)
merge (a)-[:InTheSameGroup {number_of_times: 1}]-(j)
merge (a)-[:InTheSameGroup {number_of_times: 1}]-(n)
merge (s)-[:InTheSameGroup {number_of_times: 1}]-(b)
merge (s)-[:InTheSameGroup {number_of_times: 1}]-(n)
merge (s)-[:InTheSameGroup {number_of_times: 1}]-(j)
merge (s)-[:InTheSameGroup {number_of_times: 1}]-(m)
merge (s)-[:InTheSameGroup {number_of_times: 1}]-(i)
merge (c)-[:InTheSameGroup {number_of_times: 1}]-(o)
merge (c)-[:InTheSameGroup {number_of_times: 1}]-(i)
merge (h)-[:InTheSameGroup {number_of_times: 1}]-(o)
merge (h)-[:InTheSameGroup {number_of_times: 1}]-(i)
merge (h)-[:InTheSameGroup {number_of_times: 1}]-(m)
merge (i)-[:InTheSameGroup {number_of_times: 1}]-(o)
merge (i)-[:InTheSameGroup {number_of_times: 1}]-(j)
merge (i)-[:InTheSameGroup {number_of_times: 1}]-(n)
merge (j)-[:InTheSameGroup {number_of_times: 1}]-(m)
```

```
merge (j)-[:InTheSameGroup {number_of_times: 1}]-(o)
merge (n)-[:InTheSameGroup {number_of_times: 1}]-(m)
```

Estudantes e trabalhos

```
match (a:Student {name: 'FL'}),
(b:Student {name: 'RC'}),
(c:Student {name: 'RG'}),
(d:Student {name: 'FB'}),
(e:Student {name: 'PP'}),
(f:Student {name: 'RO'}),
(g:Student {name: 'HA'}),
(h:Student {name: 'AG'}),
(i:Student {name: 'AC'}),
(j:Student {name: 'DC'}),
(k:Student {name: 'DS'}),
(l:Student {name: 'CP'}),
(m:Student {name: 'TM'}),
(n:Student {name: 'RV'}),
(o:Student {name: 'IM'}),
(p:Student {name: 'MRS'}),
(q:Student {name: 'HS'}),
(r:Student {name: 'FT'}),
(s:Student {name: 'PF'}),
(t:Student {name: 'LS'}),
(u:Student {name: 'TS'}),
(sda1:Assignement {designation: 'SD Assignement 1'}),
(sda2:Assignement {designation: 'SD Assignement 2'}),
(dwa1:Assignement {designation: 'DW Assignement 1'}),
```

```
(dwa2:Assignement {designation: 'DW Assignement 2'}),
(wsa1:Assignement {designation: 'WS Assignement 1'}),
(wsa2:Assignement {designation: 'WS Assignement 2'})


merge (sda1)-[:AssignmentTookBy]->(a)
merge (a)-[:TookAssignment]->(sda1)
merge (sda2)-[:AssignmentTookBy]->(a)
merge (a)-[:TookAssignment]->(sda2)
merge (dwa1)-[:AssignmentTookBy]->(a)
merge (a)-[:TookAssignment]->(dwa1)
merge (dwa2)-[:AssignmentTookBy]->(a)
merge (a)-[:TookAssignment]->(dwa2)
merge (dwa1)-[:AssignmentTookBy]->(b)
merge (b)-[:TookAssignment]->(dwa1)
merge (dwa2)-[:AssignmentTookBy]->(b)
merge (b)-[:TookAssignment]->(dwa2)
merge (wsa1)-[:AssignmentTookBy]->(b)
merge (b)-[:TookAssignment]->(wsa1)
merge (wsa2)-[:AssignmentTookBy]->(b)
merge (b)-[:TookAssignment]->(wsa2)
merge (sda1)-[:AssignmentTookBy]->(c)
merge (c)-[:TookAssignment]->(sda1)
merge (sda2)-[:AssignmentTookBy]->(c)
merge (c)-[:TookAssignment]->(sda2)
merge (wsa1)-[:AssignmentTookBy]->(c)
merge (c)-[:TookAssignment]->(wsa1)
merge (wsa2)-[:AssignmentTookBy]->(c)
merge (c)-[:TookAssignment]->(wsa2)
```

```
merge (sda1)-[:AssignmentTookBy]->(d)
merge (d)-[:TookAssignment]->(sda1)
merge (sda2)-[:AssignmentTookBy]->(d)
merge (d)-[:TookAssignment]->(sda2)
merge (wsa1)-[:AssignmentTookBy]->(d)
merge (d)-[:TookAssignment]->(wsa1)
merge (wsa2)-[:AssignmentTookBy]->(d)
merge (d)-[:TookAssignment]->(wsa2)
merge (sda1)-[:AssignmentTookBy]->(e)
merge (e)-[:TookAssignment]->(sda1)
merge (sda2)-[:AssignmentTookBy]->(e)
merge (e)-[:TookAssignment]->(sda2)
merge (wsa1)-[:AssignmentTookBy]->(e)
merge (e)-[:TookAssignment]->(wsa1)
merge (wsa2)-[:AssignmentTookBy]->(e)
merge (e)-[:TookAssignment]->(wsa2)
merge (sda1)-[:AssignmentTookBy]->(f)
merge (f)-[:TookAssignment]->(sda1)
merge (sda2)-[:AssignmentTookBy]->(f)
merge (f)-[:TookAssignment]->(sda2)
merge (dwa1)-[:AssignmentTookBy]->(f)
merge (f)-[:TookAssignment]->(dwa1)
merge (dwa2)-[:AssignmentTookBy]->(f)
merge (f)-[:TookAssignment]->(dwa2)
merge (sda1)-[:AssignmentTookBy]->(g)
merge (g)-[:TookAssignment]->(sda1)
merge (sda2)-[:AssignmentTookBy]->(g)
merge (g)-[:TookAssignment]->(sda2)
merge (dwa1)-[:AssignmentTookBy]->(g)
```

```
merge (g)-[:TookAssignment]->(dwa1)
merge (dwa2)-[:AssignmentTookBy]->(g)
merge (g)-[:TookAssignment]->(dwa2)
merge (wsa1)-[:AssignmentTookBy]->(g)
merge (g)-[:TookAssignment]->(wsa1)
merge (wsa2)-[:AssignmentTookBy]->(g)
merge (g)-[:TookAssignment]->(wsa2)
merge (sda1)-[:AssignmentTookBy]->(h)
merge (h)-[:TookAssignment]->(sda1)
merge (sda2)-[:AssignmentTookBy]->(h)
merge (h)-[:TookAssignment]->(sda2)
merge (wsa1)-[:AssignmentTookBy]->(h)
merge (h)-[:TookAssignment]->(wsa1)
merge (wsa2)-[:AssignmentTookBy]->(h)
merge (h)-[:TookAssignment]->(wsa2)
merge (sda1)-[:AssignmentTookBy]->(i)
merge (i)-[:TookAssignment]->(sda1)
merge (sda2)-[:AssignmentTookBy]->(i)
merge (i)-[:TookAssignment]->(sda2)
merge (dwa1)-[:AssignmentTookBy]->(i)
merge (i)-[:TookAssignment]->(dwa1)
merge (dwa2)-[:AssignmentTookBy]->(i)
merge (i)-[:TookAssignment]->(dwa2)
merge (dwa1)-[:AssignmentTookBy]->(j)
merge (j)-[:TookAssignment]->(dwa1)
merge (dwa2)-[:AssignmentTookBy]->(j)
merge (j)-[:TookAssignment]->(dwa2)
merge (wsa1)-[:AssignmentTookBy]->(j)
merge (j)-[:TookAssignment]->(wsa1)
```

```
merge (wsa2)-[:AssignmentTookBy]->(j)
merge (j)-[:TookAssignment]->(wsa2)
merge (sda1)-[:AssignmentTookBy]->(k)
merge (k)-[:TookAssignment]->(sda1)
merge (sda2)-[:AssignmentTookBy]->(k)
merge (k)-[:TookAssignment]->(sda2)
merge (dwa1)-[:AssignmentTookBy]->(k)
merge (k)-[:TookAssignment]->(dwa1)
merge (dwa2)-[:AssignmentTookBy]->(k)
merge (k)-[:TookAssignment]->(dwa2)
merge (wsa1)-[:AssignmentTookBy]->(k)
merge (k)-[:TookAssignment]->(wsa1)
merge (wsa2)-[:AssignmentTookBy]->(k)
merge (k)-[:TookAssignment]->(wsa2)
merge (sda1)-[:AssignmentTookBy]->(l)
merge (l)-[:TookAssignment]->(sda1)
merge (sda2)-[:AssignmentTookBy]->(l)
merge (l)-[:TookAssignment]->(sda2)
merge (dwa1)-[:AssignmentTookBy]->(l)
merge (l)-[:TookAssignment]->(dwa1)
merge (dwa2)-[:AssignmentTookBy]->(l)
merge (l)-[:TookAssignment]->(dwa2)
merge (wsa1)-[:AssignmentTookBy]->(l)
merge (l)-[:TookAssignment]->(wsa1)
merge (wsa2)-[:AssignmentTookBy]->(l)
merge (l)-[:TookAssignment]->(wsa2)
merge (dwa1)-[:AssignmentTookBy]->(m)
merge (m)-[:TookAssignment]->(dwa1)
merge (dwa2)-[:AssignmentTookBy]->(m)
```

```
merge (m)-[:TookAssignment]->(dwa2)
merge (wsa1)-[:AssignmentTookBy]->(m)
merge (m)-[:TookAssignment]->(wsa1)
merge (wsa2)-[:AssignmentTookBy]->(m)
merge (m)-[:TookAssignment]->(wsa2)
merge (dwa1)-[:AssignmentTookBy]->(n)
merge (n)-[:TookAssignment]->(dwa1)
merge (dwa2)-[:AssignmentTookBy]->(n)
merge (n)-[:TookAssignment]->(dwa2)
merge (wsa1)-[:AssignmentTookBy]->(n)
merge (n)-[:TookAssignment]->(wsa1)
merge (wsa2)-[:AssignmentTookBy]->(n)
merge (n)-[:TookAssignment]->(wsa2)
merge (sda1)-[:AssignmentTookBy]->(o)
merge (o)-[:TookAssignment]->(sda1)
merge (sda2)-[:AssignmentTookBy]->(o)
merge (o)-[:TookAssignment]->(sda2)
merge (dwa1)-[:AssignmentTookBy]->(o)
merge (o)-[:TookAssignment]->(dwa1)
merge (dwa2)-[:AssignmentTookBy]->(o)
merge (o)-[:TookAssignment]->(dwa2)
merge (wsa1)-[:AssignmentTookBy]->(o)
merge (o)-[:TookAssignment]->(wsa1)
merge (wsa2)-[:AssignmentTookBy]->(o)
merge (o)-[:TookAssignment]->(wsa2)
merge (sda1)-[:AssignmentTookBy]->(p)
merge (p)-[:TookAssignment]->(sda1)
merge (sda2)-[:AssignmentTookBy]->(p)
merge (p)-[:TookAssignment]->(sda2)
```

```
merge (dwa1)-[:AssignmentTookBy]->(p)
merge (p)-[:TookAssignment]->(dwa1)
merge (dwa2)-[:AssignmentTookBy]->(p)
merge (p)-[:TookAssignment]->(dwa1)
merge (sda1)-[:AssignmentTookBy]->(q)
merge (q)-[:TookAssignment]->(sda1)
merge (sda2)-[:AssignmentTookBy]->(q)
merge (q)-[:TookAssignment]->(sda2)
merge (dwa1)-[:AssignmentTookBy]->(q)
merge (q)-[:TookAssignment]->(dwa1)
merge (dwa2)-[:AssignmentTookBy]->(q)
merge (q)-[:TookAssignment]->(dwa2)
merge (sda1)-[:AssignmentTookBy]->(r)
merge (r)-[:TookAssignment]->(sda1)
merge (sda2)-[:AssignmentTookBy]->(r)
merge (r)-[:TookAssignment]->(sda2)
merge (dwa1)-[:AssignmentTookBy]->(r)
merge (r)-[:TookAssignment]->(dwa1)
merge (dwa2)-[:AssignmentTookBy]->(r)
merge (r)-[:TookAssignment]->(dwa2)
merge (wsa1)-[:AssignmentTookBy]->(r)
merge (r)-[:TookAssignment]->(wsa1)
merge (wsa2)-[:AssignmentTookBy]->(r)
merge (r)-[:TookAssignment]->(wsa2)
merge (sda1)-[:AssignmentTookBy]->(s)
merge (s)-[:TookAssignment]->(sda1)
merge (sda2)-[:AssignmentTookBy]->(s)
merge (s)-[:TookAssignment]->(sda2)
merge (wsa1)-[:AssignmentTookBy]->(s)
```

```
merge (s)-[:TookAssignment]->(wsa1)
merge (wsa2)-[:AssignmentTookBy]->(s)
merge (s)-[:TookAssignment]->(wsa2)
merge (dwa1)-[:AssignmentTookBy]->(t)
merge (t)-[:TookAssignment]->(dwa1)
merge (dwa2)-[:AssignmentTookBy]->(t)
merge (t)-[:TookAssignment]->(dwa2)
merge (wsa1)-[:AssignmentTookBy]->(t)
merge (t)-[:TookAssignment]->(wsa1)
merge (wsa2)-[:AssignmentTookBy]->(t)
merge (t)-[:TookAssignment]->(wsa2)
merge (sda1)-[:AssignmentTookBy]->(u)
merge (u)-[:TookAssignment]->(sda1)
merge (sda2)-[:AssignmentTookBy]->(u)
merge (u)-[:TookAssignment]->(sda2)
merge (dwa1)-[:AssignmentTookBy]->(u)
merge (u)-[:TookAssignment]->(dwa1)
merge (dwa2)-[:AssignmentTookBy]->(u)
merge (u)-[:TookAssignment]->(dwa2)
merge (wsa1)-[:AssignmentTookBy]->(u)
merge (u)-[:TookAssignment]->(wsa1)
merge (wsa2)-[:AssignmentTookBy]->(u)
merge (u)-[:TookAssignment]->(wsa2)
```

Estudantes - testes

```
match (a:Student {name: 'FL'}),
(b:Student {name: 'RC'}),
(c:Student {name: 'RG'}),
```

```
(d:Student {name: 'FB'}),
(e:Student {name: 'PP'}),
(f:Student {name: 'RO'}),
(g:Student {name: 'HA'}),
(h:Student {name: 'AG'}),
(i:Student {name: 'AC'}),
(j:Student {name: 'DC'}),
(k:Student {name: 'DS'}),
(l:Student {name: 'CP'}),
(m:Student {name: 'TM'}),
(n:Student {name: 'RV'}),
(o:Student {name: 'IM'}),
(p:Student {name: 'MRS'}),
(q:Student {name: 'HS'}),
(r:Student {name: 'FT'}),
(s:Student {name: 'PF'}),
(t:Student {name: 'LS'}),
(u:Student {name: 'TS'}),
(as:Test {designation: 'Multiple Choice SD'}),
(bs:Test {designation: 'Theoretical DW'}),
(cs:Test {designation: 'Multiple Choice WS'})
merge (as)-[:TookTest]->(a)
merge (bs)-[:TookTest]->(a)
merge (bs)-[:TookTest]->(b)
merge (cs)-[:TookTest]->(b)
merge (as)-[:TookTest]->(c)
merge (cs)-[:TookTest]->(c)
merge (bs)-[:TookTest]->(d)
merge (cs)-[:TookTest]->(d)
```

```
merge (as)-[:TookTest]->(e)
merge (cs)-[:TookTest]->(e)
merge (as)-[:TookTest]->(f)
merge (bs)-[:TookTest]->(f)
merge (as)-[:TookTest]->(b)
merge (bs)-[:TookTest]->(g)
merge (cs)-[:TookTest]->(g)
merge (as)-[:TookTest]->(h)
merge (cs)-[:TookTest]->(h)
merge (as)-[:TookTest]->(i)
merge (bs)-[:TookTest]->(i)
merge (bs)-[:TookTest]->(j)
merge (cs)-[:TookTest]->(j)
merge (as)-[:TookTest]->(k)
merge (bs)-[:TookTest]->(k)
merge (cs)-[:TookTest]->(k)
merge (as)-[:TookTest]->(l)
merge (bs)-[:TookTest]->(l)
merge (cs)-[:TookTest]->(l)
merge (bs)-[:TookTest]->(m)
merge (cs)-[:TookTest]->(m)
merge (bs)-[:TookTest]->(n)
merge (cs)-[:TookTest]->(n)
merge (as)-[:TookTest]->(o)
merge (bs)-[:TookTest]->(o)
merge (cs)-[:TookTest]->(o)
merge (as)-[:TookTest]->(p)
merge (bs)-[:TookTest]->(p)
merge (as)-[:TookTest]->(q)
```

```
merge (bs)-[:TookTest]->(q)
merge (as)-[:TookTest]->(r)
merge (bs)-[:TookTest]->(r)
merge (cs)-[:TookTest]->(r)
merge (as)-[:TookTest]->(s)
merge (cs)-[:TookTest]->(s)
merge (as)-[:TookTest]->(t)
merge (cs)-[:TookTest]->(t)
merge (as)-[:TookTest]->(u)
merge (bs)-[:TookTest]->(u)
merge (cs)-[:TookTest]->(u)
```

SubjectAttendedByStudent

```
match (a:Student {name: 'FL'}),
(b:Student {name: 'RC'}),
(c:Student {name: 'RG'}),
(d:Student {name: 'FB'}),
(e:Student {name: 'PP'}),
(f:Student {name: 'RO'}),
(g:Student {name: 'HA'}),
(h:Student {name: 'AG'}),
(i:Student {name: 'AC'}),
(j:Student {name: 'DC'}),
(k:Student {name: 'DS'}),
(l:Student {name: 'CP'}),
(m:Student {name: 'TM'}),
(n:Student {name: 'RV'}),
(o:Student {name: 'IM'}),
```

```
(p:Student {name: 'MRS'}),
(q:Student {name: 'HS'}),
(r:Student {name: 'FT'}),
(s:Student {name: 'PF'}),
(t:Student {name: 'LS'}),
(u:Student {name: 'TS'}),
(as:Subject {name: 'Distributed Systems'}),
(bs:Subject {name: 'Web Development'}),
(cs:Subject {name: 'Semantic Web'})
merge (as)-[:SubjectAttendedByStudent]->(a)
merge (bs)-[:SubjectAttendedByStudent]->(a)
merge (bs)-[:SubjectAttendedByStudent]->(b)
merge (cs)-[:SubjectAttendedByStudent]->(b)
merge (as)-[:SubjectAttendedByStudent]->(c)
merge (cs)-[:SubjectAttendedByStudent]->(c)
merge (bs)-[:SubjectAttendedByStudent]->(d)
merge (cs)-[:SubjectAttendedByStudent]->(d)
merge (as)-[:SubjectAttendedByStudent]->(e)
merge (cs)-[:SubjectAttendedByStudent]->(e)
merge (as)-[:SubjectAttendedByStudent]->(f)
merge (bs)-[:SubjectAttendedByStudent]->(f)
merge (as)-[:SubjectAttendedByStudent]->(g)
merge (bs)-[:SubjectAttendedByStudent]->(g)
merge (cs)-[:SubjectAttendedByStudent]->(g)
merge (as)-[:SubjectAttendedByStudent]->(h)
merge (cs)-[:SubjectAttendedByStudent]->(h)
merge (as)-[:SubjectAttendedByStudent]->(i)
merge (bs)-[:SubjectAttendedByStudent]->(i)
merge (bs)-[:SubjectAttendedByStudent]->(j)
```

```
merge (cs)-[:SubjectAttendedByStudent]->(j)
merge (as)-[:SubjectAttendedByStudent]->(k)
merge (bs)-[:SubjectAttendedByStudent]->(k)
merge (cs)-[:SubjectAttendedByStudent]->(k)
merge (as)-[:SubjectAttendedByStudent]->(l)
merge (bs)-[:SubjectAttendedByStudent]->(l)
merge (cs)-[:SubjectAttendedByStudent]->(l)
merge (bs)-[:SubjectAttendedByStudent]->(m)
merge (cs)-[:SubjectAttendedByStudent]->(m)
merge (bs)-[:SubjectAttendedByStudent]->(n)
merge (cs)-[:SubjectAttendedByStudent]->(n)
merge (as)-[:SubjectAttendedByStudent]->(o)
merge (bs)-[:SubjectAttendedByStudent]->(o)
merge (cs)-[:SubjectAttendedByStudent]->(o)
merge (as)-[:SubjectAttendedByStudent]->(p)
merge (bs)-[:SubjectAttendedByStudent]->(p)
merge (as)-[:SubjectAttendedByStudent]->(q)
merge (bs)-[:SubjectAttendedByStudent]->(q)
merge (as)-[:SubjectAttendedByStudent]->(r)
merge (bs)-[:SubjectAttendedByStudent]->(r)
merge (cs)-[:SubjectAttendedByStudent]->(r)
merge (as)-[:SubjectAttendedByStudent]->(s)
merge (cs)-[:SubjectAttendedByStudent]->(s)
merge (bs)-[:SubjectAttendedByStudent]->(t)
merge (cs)-[:SubjectAttendedByStudent]->(t)
merge (cs)-[:SubjectAttendedByStudent]->(u)
merge (as)-[:SubjectAttendedByStudent]->(u)
merge (bs)-[:SubjectAttendedByStudent]->(u)
```

Introdução de professores

```
match (a:Student {name: 'FL'}),
(b:Student {name: 'RC'}),
(c:Student {name: 'RG'}),
(d:Student {name: 'FB'}),
(e:Student {name: 'PP'}),
(f:Student {name: 'RO'}),
(g:Student {name: 'HA'}),
(h:Student {name: 'AG'}),
(i:Student {name: 'AC'}),
(j:Student {name: 'DC'}),
(k:Student {name: 'DS'}),
(l:Student {name: 'CP'}),
(m:Student {name: 'TM'}),
(n:Student {name: 'RV'}),
(o:Student {name: 'IM'}),
(p:Student {name: 'MRS'}),
(q:Student {name: 'HS'}),
(r:Student {name: 'FT'}),
(s:Student {name: 'PF'}),
(t:Student {name: 'LS'}),
(u:Student {name: 'TS'}),
(t1:Teacher {name: 'PT'}),
(t2:Teacher {name: 'JR'}),
(t3:Teacher {name: 'JC'}),
(as:Subject {name: 'Distributed Systems'}),
(bs:Subject {name: 'Web Development'}),
(cs:Subject {name: 'Semantic Web'})
```

```
merge (a)-[:StudentOf]->(t1)
merge (b)-[:StudentOf]->(t2)
merge (b)-[:StudentOf]->(t3)
merge (c)-[:StudentOf]->(t1)
merge (c)-[:StudentOf]->(t3)
merge (d)-[:StudentOf]->(t2)
merge (d)-[:StudentOf]->(t3)
merge (e)-[:StudentOf]->(t1)
merge (e)-[:StudentOf]->(t3)
merge (f)-[:StudentOf]->(t1)
merge (f)-[:StudentOf]->(t2)
merge (g)-[:StudentOf]->(t1)
merge (g)-[:StudentOf]->(t2)
merge (g)-[:StudentOf]->(t3)
merge (h)-[:StudentOf]->(t1)
merge (h)-[:StudentOf]->(t3)
merge (i)-[:StudentOf]->(t1)
merge (i)-[:StudentOf]->(t2)
merge (j)-[:StudentOf]->(t2)
merge (j)-[:StudentOf]->(t3)
merge (k)-[:StudentOf]->(t1)
merge (k)-[:StudentOf]->(t2)
merge (k)-[:StudentOf]->(t3)
merge (l)-[:StudentOf]->(t1)
merge (l)-[:StudentOf]->(t2)
merge (l)-[:StudentOf]->(t3)
merge (m)-[:StudentOf]->(t2)
merge (m)-[:StudentOf]->(t3)
merge (n)-[:StudentOf]->(t2)
```

```
merge (n)-[:StudentOf]->(t3)
merge (o)-[:StudentOf]->(t1)
merge (o)-[:StudentOf]->(t2)
merge (o)-[:StudentOf]->(t3)
merge (p)-[:StudentOf]->(t1)
merge (p)-[:StudentOf]->(t2)
merge (q)-[:StudentOf]->(t1)
merge (q)-[:StudentOf]->(t2)
merge (r)-[:StudentOf]->(t1)
merge (r)-[:StudentOf]->(t2)
merge (r)-[:StudentOf]->(t3)
merge (s)-[:StudentOf]->(t1)
merge (s)-[:StudentOf]->(t3)
merge (t)-[:StudentOf]->(t2)
merge (t)-[:StudentOf]->(t3)
merge (u)-[:StudentOf]->(t3)
merge (u)-[:StudentOf]->(t1)
merge (u)-[:StudentOf]->(t2)
```

```
merge (t1)-[:TeacherOf]->(a)
merge (t2)-[:TeacherOf]->(a)
merge (t2)-[:TeacherOf]->(b)
merge (t3)-[:TeacherOf]->(b)
merge (t1)-[:TeacherOf]->(c)
merge (t3)-[:TeacherOf]->(c)
merge (t2)-[:TeacherOf]->(d)
```

```
merge (t3)-[:TeacherOf]->(d)
merge (t1)-[:TeacherOf]->(e)
merge (t3)-[:TeacherOf]->(e)
merge (t1)-[:TeacherOf]->(f)
merge (t2)-[:TeacherOf]->(f)
merge (t1)-[:TeacherOf]->(g)
merge (t2)-[:TeacherOf]->(g)
merge (t3)-[:TeacherOf]->(g)
merge (t1)-[:TeacherOf]->(h)
merge (t3)-[:TeacherOf]->(h)
merge (t1)-[:TeacherOf]->(i)
merge (t2)-[:TeacherOf]->(i)
merge (t2)-[:TeacherOf]->(j)
merge (t3)-[:TeacherOf]->(j)
merge (t1)-[:TeacherOf]->(k)
merge (t2)-[:TeacherOf]->(k)
merge (t3)-[:TeacherOf]->(k)
merge (t1)-[:TeacherOf]->(l)
merge (t2)-[:TeacherOf]->(l)
merge (t3)-[:TeacherOf]->(l)
merge (t2)-[:TeacherOf]->(m)
merge (t3)-[:TeacherOf]->(m)
merge (t2)-[:TeacherOf]->(n)
merge (t3)-[:TeacherOf]->(n)
merge (t1)-[:TeacherOf]->(o)
merge (t2)-[:TeacherOf]->(o)
merge (t3)-[:TeacherOf]->(o)
merge (t1)-[:TeacherOf]->(p)
merge (t2)-[:TeacherOf]->(p)
```

```
merge (t1)-[:TeacherOf]->(q)
merge (t2)-[:TeacherOf]->(q)
merge (t1)-[:TeacherOf]->(r)
merge (t2)-[:TeacherOf]->(r)
merge (t3)-[:TeacherOf]->(r)
merge (t1)-[:TeacherOf]->(s)
merge (t3)-[:TeacherOf]->(s)
merge (t2)-[:TeacherOf]->(t)
merge (t3)-[:TeacherOf]->(t)
merge (t3)-[:TeacherOf]->(u)
merge (t1)-[:TeacherOf]->(u)
merge (t2)-[:TeacherOf]->(u)

merge (t1)-[:Teaches]->(as)
merge (t2)-[:Teaches]->(bs)
merge (t3)-[:Teaches]->(cs)

merge (as)-[:TeachedBy {approval_average: 55}]->(t1)
merge (bs)-[:TeachedBy {approval_average: 65}]->(t2)
merge (cs)-[:TeachedBy {approval_average: 75}]->(t3)
```

Professores - avaliações

```
match (t1:Teacher {name: 'PT'}),
(t2:Teacher {name: 'JR'}),
(t3:Teacher {name: 'JC'}),
(sda1:Assignement {designation: 'SD Assignement 1'}),
(sda2:Assignement {designation: 'SD Assignement 2'}),
(dwa1:Assignement {designation: 'DW Assignement 1'}),
```

```
(dwa2:Assignement {designation: 'DW Assignement 2'}),
(wsa1:Assignement {designation: 'WS Assignement 1'}),
(wsa2:Assignement {designation: 'WS Assignement 2'}),
(as:Subject {name: 'Distributed Systems'}),
(bs:Subject {name: 'Web Development'}),
(cs:Subject {name: 'Semantic Web'}),
(sdt:Test {designation: 'Multiple Choice SD'}),
(dwt:Test {designation: 'Theoretical DW'}),
(wst:Test {designation: 'Multiple Choice WS'})

merge (t1)-[:TeacherEvaluatedAssignment]->(sda1)
merge (t1)-[:TeacherEvaluatedAssignment]->(sda2)
merge (t2)-[:TeacherEvaluatedAssignment]->(dwa1)
merge (t2)-[:TeacherEvaluatedAssignment]->(dwa2)
merge (t3)-[:TeacherEvaluatedAssignment]->(wsa1)
merge (t3)-[:TeacherEvaluatedAssignment]->(wsa2)

merge (sda1)-[:AssignmentEvaluatedBy]->(t1)
merge (sda2)-[:AssignmentEvaluatedBy]->(t1)
merge (dwa1)-[:AssignmentEvaluatedBy]->(t2)
merge (dwa2)-[:AssignmentEvaluatedBy]->(t2)
merge (wsa1)-[:AssignmentEvaluatedBy]->(t3)
merge (wsa2)-[:AssignmentEvaluatedBy]->(t3)

merge (t1)-[:TeacherEvaluatedTest]->(sdt)
merge (t2)-[:TeacherEvaluatedTest]->(dwt)
merge (t3)-[:TeacherEvaluatedTest]->(wst)
merge (sdt)-[:TestEvaluatedBy]->(t1)
merge (dwt)-[:TestEvaluatedBy]->(t2)
```

```
merge (wst)-[:TestEvaluatedBy]->(t3)


merge (as)-[:SubjectOfAssignment]->(sda1)
merge (as)-[:SubjectOfAssignment]->(sda2)
merge (bs)-[:SubjectOfAssignment]->(dwa1)
merge (bs)-[:SubjectOfAssignment]->(dwa2)
merge (cs)-[:SubjectOfAssignment]->(wsa1)
merge (cs)-[:SubjectOfAssignment]->(wsa2)

merge (as)-[:SubjectOfTest]->(sdt)
merge (bs)-[:SubjectOfTest]->(dwt)
merge (cs)-[:SubjectOfTest]->(wst)
```

Apêndice E

Aplicação dos Algoritmos

Label Propagation

```
//knows
CALL gds.graph.create('graphlp', 'Student',
        {
                Knows: {
                        orientation: 'UNDIRECTED',
                properties: 'since'
                }
        }
)

CALL gds.graph.create('weights', 'Student',
        {
                Knows: {
                        orientation: 'UNDIRECTED',
                        properties: 'weight'
                }
        }
```

```
)


CALL gds.labelPropagation.write(
        'weights',
        {
                relationshipWeightProperty: 'weight',
                maxIterations: 10,
                writeProperty: 'community'
        })

//IntheSameGroup

match (s:Student)
with s.community as community, count(s) as number_of_nodes
return community, number_of_nodes order by number_of_nodes desc limit 50

CALL gds.graph.create('graphsg', 'Student',
        {
                InTheSameGroup: {
                        orientation: 'UNDIRECTED',
                properties: 'number_of_times'
                }
        }
)

CALL gds.labelPropagation.write(
        'graphsg',
        {
```

```
                relationshipWeightProperty: 'number_of_times',
                maxIterations: 10,
                writeProperty: 'communitysamegroup'
        })

\\Friend of a Friend
CALL gds.graph.create('graphfof', 'Student',
        {
                FriendOfFriend: {
                        orientation: 'UNDIRECTED'
                }
        }
)

CALL gds.labelPropagation.write(
        'graphfof',
        {
                maxIterations: 10,
                writeProperty: 'communityFriendOfFriend'
        })
```

Page Rank

```
//Knows
call gds.graph.create('friends', 'Student',{
Knows: {
        orientation: 'UNDIRECTED'
}
})
```

```
CALL gds.pageRank.stream('friends') YIELD nodeId, score AS pageRank
with gds.util.asNode(nodeId) AS n, pageRank
match (n)-[i:Knows]-()
Return n.name AS name, pageRank, count(1) AS degree, sum(i.weight) AS weighte
ORDER BY pageRank DESC LIMIT 10


//InTheSameGroup
call gds.graph.create('samegroup', 'Student',{
InTheSameGroup: {
        orientation: 'UNDIRECTED'
}
})

CALL gds.pageRank.stream('samegroup') YIELD nodeId, score AS pageRank
with gds.util.asNode(nodeId) AS n, pageRank
match (n)-[i:InTheSameGroup]-()
Return n.name AS name, pageRank, count(1) AS degree, sum(i.weight) AS weighte
ORDER BY pageRank DESC LIMIT 10

//FriendOfFriend

call gds.graph.create('fof', 'Student',{
FriendOfFriend: {
        orientation: 'UNDIRECTED'
}
})
```

E4

```
CALL gds.pageRank.stream('fof') YIELD nodeId, score AS pageRank
with gds.util.asNode(nodeId) AS n, pageRank
match (n)-[i:FriendOfFriend]-()
Return n.name AS name, pageRank, count(1) AS degree, sum(i.weight) AS weighte
ORDER BY pageRank DESC LIMIT 10
```

Betweenness Centrality

```
//Betweenness
//Knows
CALL gds.graph.create('between', 'Student', {Knows: {orientation: 'UNDIRECTED

CALL gds.betweenness.stream('between')
YIELD nodeId, score
RETURN gds.util.asNode(nodeId).name AS name, score
ORDER BY score DESC

//InTheSameGroup

CALL gds.graph.create('samegroupbetween', 'Student', {IntheSameGroup: {orient



CALL gds.betweenness.stream('samegroupbetween')
YIELD nodeId, score
RETURN gds.util.asNode(nodeId).name AS name, score
ORDER BY score DESC



//FriendOfFriend
```

```
CALL gds.graph.create('fofbetween', 'Student', {FriendOfFriend: {orientation:
```

```
CALL gds.betweenness.stream('fofbetween')
YIELD nodeId, score
RETURN gds.util.asNode(nodeId).name AS name, score
ORDER BY score DESC
```