UNIFICAÇÃO DE BANCO DE DADOS PARA ALIMENTAR MODELOS ONTOLÓGICOS

Caio Alexandre Soares Fabrão (caiofabrao@alunos.utfpr.edu.br) - Universidade Tecnológica Federal do Paraná

Rafael Voltolini (voltolini@alunos.utfpr.edu.br) - Universidade Tecnológica Federal do Paraná

Milton Borsato (borsato@utfpr.edu.br) - Universidade Tecnológica Federal do Paraná

RESUMO

Com os recorrentes avanços na Internet e na Internet das Coisas, está cada vez mais rápido adquirir zettabytes de dados de fontes e tipos diversos. A heterogeneidade de dados se torna um problema quando se deseja integrar tal montante de informação. A integração virtual de banco de dados é uma técnica versátil para solucionar este problema, focada no acesso dos dados, permitindo que eles estejam sempre atualizados sem a necessidade de realizar cópias. Este trabalho apresenta uma aplicação prática dessa técnica ao usar dois bancos de dados relacionais para alimentar uma ontologia com instâncias. Foram utilizados, principalmente, o software Stardog e a ontologia function_cost_18#, uma ontologia que trata da estimativa de custos nas fases iniciais do desenvolvimento do produto ponte rolante, levando em consideração as peças e componentes das pontes rolantes assim como as funções a serem realizadas. A ferramenta de mapeamento de banco de dados Stardog Mapping Syntaxe 2 apresentou grande rapidez e flexibilidade para unificar os dados com a ontologia, tendo a unificação validada com sucesso ao executar corretamente as questões de competência que nortearam a desenvolvimento da ontologia.

Palavras chave: *união de dados*; *ontologia; integração virtual; Stardog; SMS2;*

1. INTRODUÇÃO

O mundo encontra-se na era dos dados digitais. Com os recorrentes avanços na Internet, na análise de dados e na IoT, está cada vez mais fácil adquirir zettabytes de dados em ritmo acelerado (AGRAWAL et al., 2007). Essa facilidade em aquisição de dados trouxe consigo uma grande heterogeneidade nos dados.

Os dados proveem de diversos meios, possuindo naturezas distintas umas das outras. Por conta disso, há diversas opções de soluções de armazenamento de dados, cada um com suas vantagens e desvantagens. Buscando melhor suprir as diferentes características dos dados, há, por exemplo, os métodos de sistemas de arquivos, armazenamentos de objetos, armazenamento de blobs, banco de dados NoSQL, entre tantos outros (LI, 2018). Em um primeiro momento, essa diversificação não apresenta grandes problemas. Porém, isso muda quando é preciso relacionar dados de fontes distintas.

Os silos de dados é a forma mais simples de solucionar esse problema. Por mais que possua vantagens, esse método demanda um alto custo de implementação, pois é preciso replicar os dados originais, centralizando os em um novo banco de dados (WIDOM, 1995). A cópia dos dados gera outros problemas, como espaço de armazenamento, cópias desatualizadas dos dados, etc.

Em contrapartida com os silos de dados, a integração virtual busca solucionar os problemas com a duplicação dos dados. A grande diferença está na utilização de um mediador para buscar os dados necessários de cada fonte, convertendo-os em tempo de execução da consulta (HAN; JIAO, 2018). Isso significa que quanto mais fontes de dados distintos são necessários e quanto mais volátil for a natureza desses dados, isto é, o quão suscetível a alterações eles estão, mais a integração virtual se torna a solução ideal.

Este documento visa apresentar uma aplicação prática da integração virtual de dados para alimentar uma ontologia. As instâncias formam parte do alicerce de uma ontologia. Poder ter acesso fácil a dados atualizados e de diversas fontes, torna a ontologia ainda mais efetiva na tomada de decisões.

As seções seguintes desse artigo estão estruturadas da seguinte forma: na Seção 2, é apresentado uma revisão dos conceitos básicos para a compreensão das atividades; a Seção 3 descreve a metodologia utilizada para realizar as atividades propostas; a análise dos resultados obtidos com o artigo são demonstrados na Seção 4; e a Seção 5 conclui com uma breve revisão e discussão sobre as atividades executadas.

2. REVISÃO TEÓRICA

2.1 Banco de Dados Relacionais

O modelo de dados relacionais não é um conceito novo, datando desde 1970. Neste modelo, os dados relacionais são representados como uma coleção de relações, o que significa que, tais dados e relações são representados na forma de uma tabela de valores. O nome da tabela diz sobre a qual domínio os dados está associado, as colunas, quais informações desse domínio possui relação e as linhas são as entidades do mundo real (ELMASRI; NAVATHE, 2016). A Figura 1 apresenta dados sobre motores elétricos.

MOTOR

ID	NOME	POTENCIA
1	24VB1-60	1.54
7	54VC-110	5.0

Figura 1: Tabela de valores simples para motores elétricos. Fonte: Do autor, 2019.

Como cada linha representa fatos sobre motores elétricos, o nome da tabela pode ser claramente identificado por Motor. Os nomes das colunas ajudam a como interpretar os valores associados. Assim: i) ID representa um identificador único para o dado motor, ii) NOME representa o nome do modelo, e iii) POTENCIA, a potência do motor.

2.2 Diagrama Entidade Relacionamento

O diagrama de entidade relacionamento (DER) é um padrão para a representação conceitual do esquema de um banco de dados relacional. Sua função é a apresentação de forma clara e objetiva como as tabelas de valores de um banco de dados estão relacionadas entre si, evidenciando essas relações (TEOREY et al., 2002). É retratado no diagrama, pelo menos, o nome da tabela, as chaves primárias e secundárias e as relações entre as tabelas.

A Figura 2 demonstra um diagrama simples do relacionamento entre as tabelas *Accessory*, *Motor*, *Trolley* e *TrolleyAcc*. *Motor* e *Trolley* possuem um relacionamento direto através da chave *idMotor*, sendo ela uma chave primária na primeira tabela e estrangeira na segunda. *Trolley* e *Accessory* possuem uma relação mais complexa. As tuplas de *Accessory* estão associadas a uma ou mais tuplas de *Trolley*, e vise-versa. Portanto essas duas tabelas estão

associados a uma tabela intermediária (*TrolleyAcc*), utilizando suas chaves primarias para diminuir a complexidade da relação.

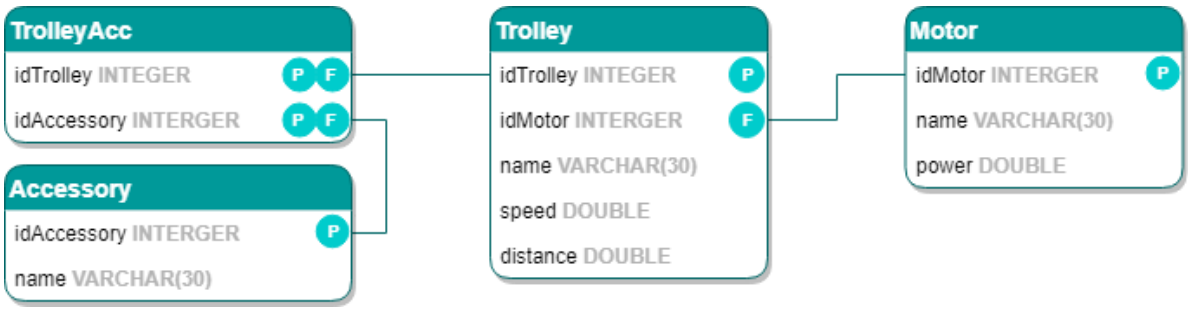

Figura 2: DER simplificado. Fonte: Do autor, 2019.

2.3 Stardog

Stardog é uma plataforma de unificação de dados que facilita a consulta a dados massivos e heterogêneos independentes das suas estruturas de armazenamento. São usados Grafos de Conhecimento para juntar duas tendências na área de dados corporativos: virtualização e grafos. A virtualização é utilizada para integrar os dados e os grafos, para estruturar e organizá-los (STARDOG UNION, 2019a). O primeiro soluciona o problema de acesso a dados heterogêneos, enquanto que o segundo facilita a implementação e a consulta a eles.

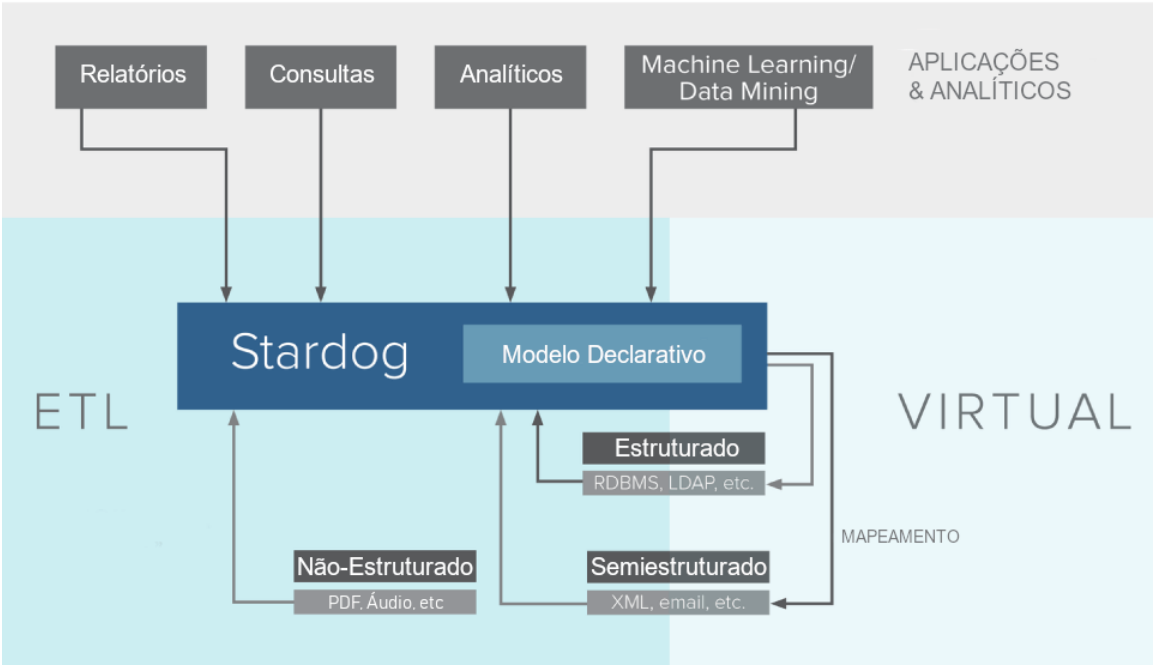

Figura 3: Visão geral do funcionamento da plataforma Stardog. Fonte: Traduzido de Stardog Union (2019a).

A Figura 3 apresenta as três camadas de funcionamento do Stardog. Na camada de aplicação, o usuário pode de forma transparente consultar, analisar, criar relatórios, etc, sobre os dados de forma rápida. Na camada de virtualização, são realizados os mapeamentos dos dados estruturados e semiestruturados. E na camada ETL, são mapeados os dados não estruturados.

2.4 Ontologia

Uma RDF (do inglês, estrutura de descrição de recurso) é uma tipo de modelagem de dados para descrever recursos em forma de grafos. Os dados são descritos como triplas do tipo sujeito, propriedade e objeto (GENG et al., 2018). A Figura 4 apresenta um exemplo de RDF descrevendo como os integrantes da banda Beatles se relacionam entre si e com a música *Love Me Do*. Paul McCartney e John Lenon são artistas solos, membros da banda The Beatles e compositores da música *"Love Me Do"*, que é uma faixa do álbum *"Please Please Me"*.

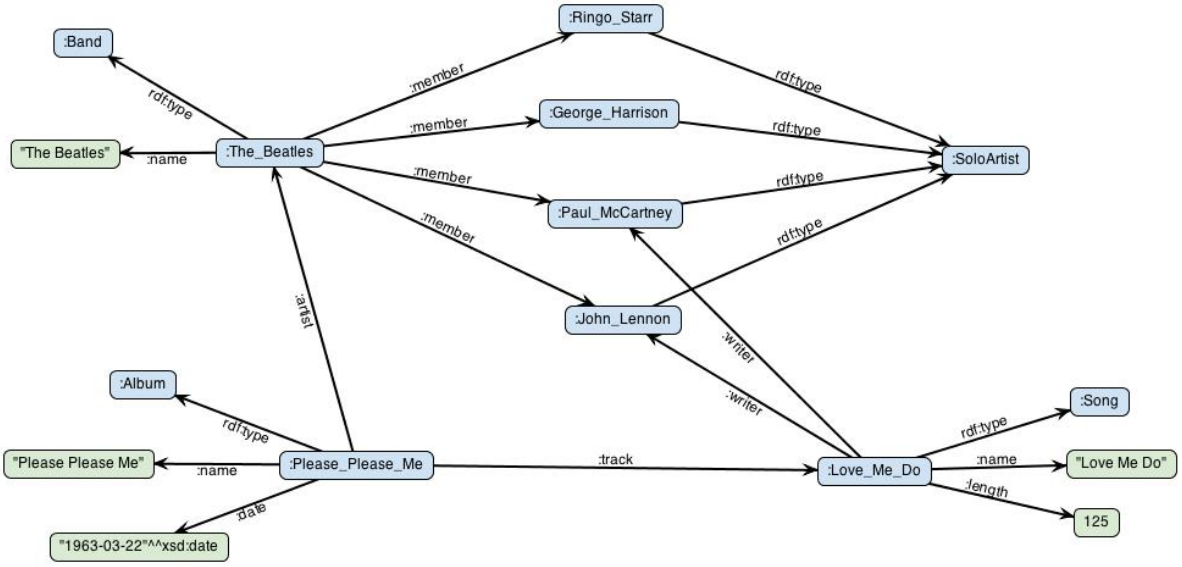

Figura 4: Exemplo de RDF descrevendo a banda Beatles. Fonte: Stardog Union (2019b)

Ontologias são tipos especiais de RDFs que, além de descrever um recurso, descrevem um conhecimento do mundo real bem estabelecido sobre um determinado domínio, de forma que seja compreendido tanto por computadores quanto por pessoas (STUDER et al., 1998). A OWL, padronizada pela W3C, é a linguagem utilizada para implementar as ontologias. Isso

permite que programas possam verificar a consistência do conhecimento armazenado e também inferir novas relações, ou seja, novos conhecimentos (W3C, 2012a).

2.5 Stardog Mapping Syntax

Stardog *Mapping Syntax* (SMS) é uma sintaxe de mapeamento de dados relacionais para RDF desenvolvida pela Stardog Union. A SMS é baseada em R2RML (*RDB to RDF Mapping Language*), mapeando as relações do banco de dados relacional (do inglês, RDB), de acordo com a sua estrutura, e retornando um grafo RDF especifico (W3C, 2012b). Caso alguma estrutura mude, seja o esquema do RDB ou a definição do grafo de destino, é possível adaptar o mapeamento de acordo com as novas necessidades.

O código da Figura 5, escrito em R2RML, realiza o mapeamento de funcionários e departamentos de uma empresa. É descrita a relação entre o funcionário e seu departamento com base no seu cargo. Funcionários que possuem o cargo de guarda-noturno são ligados ao departamento de segurança; enquanto os engenheiros ao setor de engenharia.

```
1.  @prefix rr: <http://www.w3.org/ns/r2rml#> .
2.  @prefix emp: <http://example.com/emp#> .
3.  @prefix dept: <http://example.com/dept#> .
4.  @prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
5.  @base <http://example.com/base/> .
6.
7.  <DeptTriplesMap>
8.  a rr:TriplesMap;
9.  rr:logicalTable [ rr:tableName "DEPT" ];
10. rr:subjectMap [rr:template "http://data.example.com/dept/{\"deptno\"}";
11. rr:class dept:Department ];
12. rr:predicateObjectMap [
13. rr:predicate
     dept:deptno ;
14. rr:objectMap [ rr:column "\"deptno\""; rr:datatype xsd:positiveInteger]
15. ];
16. rr:predicateObjectMap [
17. rr:predicate
                                                                    d
     ept:location ;
18. rr:objectMap                                                    [
     rr:column "\"loc\"" ]
19. ].
20.
21. <EmpTriplesMap>
22.  a rr:TriplesMap;
23. rr:logicalTable [ rr:sqlQuery """
24. SELECT "EMP".*, (CASE "job"
25. WHEN 'CLERK' THEN 'general-office'
26. WHEN 'NIGHTGUARD' THEN 'security'
27. WHEN 'ENGINEER' THEN 'engineering'
```

```
28. END) AS ROLE FROM "EMP"
29. """ ];
30. rr:subjectMap [
31. rr:template "http://data.example.com/employee/{\"empno\"}";
32. rr:class emp:Employee
33. ];
34. rr:predicateObjectMap [
35. rr:predicate emp:name ;
36. rr:objectMap    [ rr:column "\"ename\"" ];
37. ];
38. rr:predicateObjectMap [
39. rr:predicate emp:role;
40. rr:objectMap [ rr:template "http://data.example.com/roles/{ROLE}" ];
41. ];
42. rr:predicateObjectMap [
43. rr:predicate emp:department;
44. rr:objectMap [ rr:template "http://example.com/dept/{\"deptno\"}"; ];
45. ].
```

Figura 5: Exemplo de mapeamento em R2RML. Fonte: Stardog Union (2019b).

A SMS é uma linguagem mais simples que R2RML, facilitando a implementação, leitura e compreensão dos mapeamentos realizados (STARDOG UNION, 2019b). O código da Figura 6 requer menos linhas para se obter o mapeamento das relações entre o funcionário e o seu departamento, indicando uma melhora na implementação.

```
1.  @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
2.  @prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
3.  @prefix emp: <http://example.com/emp/> .
4.  @prefix dept: <http://example.com/dept/> .
5.  @prefix sm: <tag:stardog:api:mapping:> .
6.
7.  dept:{"deptno"} a dept:Department ;
8.  dept:location "{\"loc\"}" ;
9.  dept:deptno "{\"deptno\"}"^^xsd:integer ;
10. sm:map [
11. sm:table "DEPT" ;
12. ] .
13.
14. emp:{"empno"} a emp:Employee ;
15. emp:name "{\"ename\"}" ;
16. emp:role emp:{ROLE} ;
17. emp:department dept:{"deptno"} ;
18. sm:map [
19. sm:query """
20. SELECT \"empno\", \"ename\", \"deptno\", (CASE \"job\"
21. WHEN 'CLERK' THEN 'general-office'
22. WHEN 'NIGHTGUARD' THEN 'security'
23. WHEN 'ENGINEER' THEN 'engineering'
24. END) AS ROLE FROM \"EMP\"
25. """ ;
26. ] .
```

Figura 6: Exemplo de mapeamento em SMS. Fonte: Stardog Union (2019b).

SMS utiliza a cláusula *sm:table* para agilizar o mapeamento de tabelas, além de permitir que consultas SQL sejam realizadas no mapeamento utilizando *sm:query*. Com isso, os dados das linhas da tabela são mapeados para uma ou mais triplas do RDF.

2.5.1 Stardog Mapping Syntax 2

Stardog *Mapping Syntax* 2 (SMS2) é uma evolução para a SMS, com uma estrutura similar à SPARQL. Um código em SMS2 é dividido em cinco cláusulas: *prólogo*, *mapping*, *from*, *to*, e *where* (STARDOG UNION, 2019c). No prólogo são declarados os prefixos necessários. A cláusula *mapping* inicia a declaração do mapeamento, podendo, opcionalmente, designar também o nome deste. Em f*rom* tem-se a definição da origem dos dados, com sua estrutura e os nomes das variáveis. Em *to*, a definição do RDF, utilizando a mesma estrutura de um construtor em SPARQL, tirando vantagem da utilização de variáveis para definir as triplas. E por fim, em *where* são realizadas as conversões de variáveis utilizando a cláusula *bind*. A Figura 7 apresenta o código de exemplo da seção 2.5 em SMS2.

```
1.  PREFIX emp: <http://example.com/emp/>
2.  PREFIX dept: <http://example.com/dept/>
3.
4.  MAPPING <urn:departments>
5.  FROM SQL {
6.  SELECT * FROM "DEPT"
7.  }
8.  TO {
9.  ?deptIri a dept:Department ;
10. dept:location ?loc ;
11. dept:deptno "{?deptno}"^^xsd:integer .
12. }
13. WHERE {
14. BIND (template("http://example.com/dept/{deptno}") AS ?deptIri)
15. }
16.
17. ;
18.
19. MAPPING <urn:employees>
20. FROM SQL {
21. SELECT \"empno\", \"ename\", \"deptno\", (CASE \"job\"
22. WHEN 'CLERK' THEN 'general-office'
23. WHEN 'NIGHTGUARD' THEN 'security'
24. WHEN 'ENGINEER' THEN 'engineering'
25. END) AS ROLE FROM \"EMP\"
26. }
27. TO {
28. ?empIri a emp:Employee ;
29. emp:name ?ename ;
```

```
30. emp:role ?roleIri ;
31. emp:department ?deptIri .
32. }
33. WHERE {
34. BIND (template("http://example.com/emp/{empno}") AS ?empIri)
35. BIND (template("http://example.com/dept/{deptno}") AS ?deptIri)
36. BIND (template("http://example.com/emp/{ROLE}") AS ?roleIri)
37. }
```

Figura 7: Exemplo de mapeamento em SMS2. Fonte: Stardog Union (2019c).

3. METODOLOGIA

Para a realização deste projeto, foi utilizado o software Stardog com o intuito de integrar dois RDBs e alimentar a ontologia function_cost_18# com os dados. A ontologia trata da estimativa de custos nas fases iniciais do produto ponte rolante, considerando as partes que compõe o produto (viga, motor, gancho, etc) e suas funções de operação (içar, deslocar em x, deslocar em y, etc) (VOLTOLINI, 2019).

Também foram utilizadas a VOWL e a IDE MySQL, para obter uma visualização gráfica da ontologia, assim realizando as primeiras análises, e para desenvolver os bancos de dados em SQL, respectivamente.

Por fim, a validação desta unificação foi realizada ao implementar as questões de competência, desenvolvidas para validar a ontologia, sobre o mapeamento realizado no Stardog.

3.1 Construção dos Bancos de Dados

O primeiro passo para a construção dos bancos de dados foi a identificação das classes cuja as instâncias deveriam ser armazenadas em bancos de dados. O processo de identificação foi realizado através de uma análise sobre a ontologia utilizando o visualizador gráfico VOWL, o que agilizou o processo de identificação. As classes que caracterizam a estrutura física de uma ponte rolante e as que caracterizam os custos de montagem e operação, deveriam ter suas instâncias armazenadas em bancos de dados. Essa decisão foi tomada uma vez que a natureza desses dados é massiva e/ou sujeitas a constantes atualizações.

O DER descrito na Figura 8, demonstra como as classes anteriores se relacionam. As classes da estrutura física se agrupam em torno de três montagens: a ponte (*Brigde*), o carro (*Trolley*) e o guindaste (*Hoist*). As peças e as montagens têm um relacionamento direto, uma montagem possui uma peça de cada tipo, exceto com relação aos acessórios. Uma montagem

pode possuir zero ou mais acessórios, portanto foram criadas as tabelas de junção *AccBrigde*, *AccTrolley* e *AccHoist* para juntar a tabela *Accessory* as tabelas *Brigde*, *Trolley* e *Hoist*, respectivamente.

As classes referentes ao custo se agruparam em três categorias: peças compradas, peças produzidas na própria empresa e depreciação por desgaste, sendo respectivamente as tabelas *Purchase*, *Production* e *Extra*.

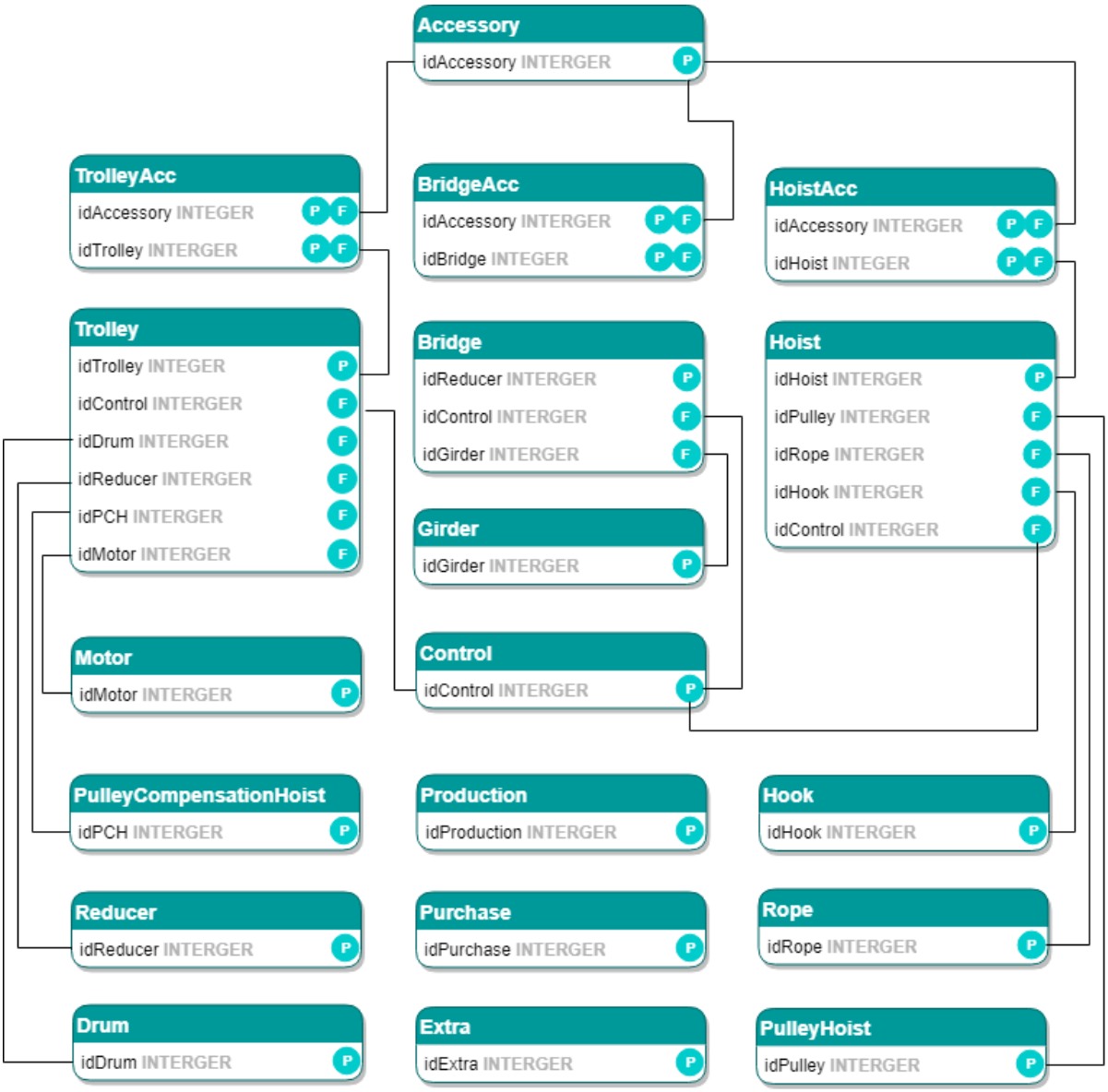

Figura 8: DER das classes selecionadas da ontologia. Fonte: Do autor, 2019.

No MySQL, o diagrama originou dois bancos de dados, *overheadcrane* e *financial*. O primeiro ficou responsável por armazenar os dados referentes as peças e montagens, enquanto

que o segundo ficou responsável pelos dados de custo, assim caracterizando dois setores em uma empresa: montagem e financeiro. Por mais que pudesse ser desenvolvido um único banco de dados com os dois setores, foi deliberadamente escolhido separá-los para evidenciar a integração de bancos distintos.

A população de dados do setor de montagem foi retirada dos catálogos em CIMAF (2014) e em CSM (2019). Para o setor financeiro, *os* custos dos componentes foram adquiridos realizando os cálculos das médias dos preços praticados no mercado brasileiro (MERCADO LIVRE, 2019).

3.2 Mapeamento dos Dados Relacionais

O mapeamento dos dados relacionais em SQL para RDF foi realizado utilizando a SMS2. A estruturação do mapeamento em SMS2 começa com a definição do cabeçalho de prefixos e da consulta que extrairá os dados (linguagem alvo, colunas a serem selecionadas e a tabela de origem). A Figura 9 apresenta o início do mapeamento da tabela *motor*.

```
PREFIX : <ontologies/function_cost_18#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

MAPPING
FROM SQL {
  SELECT *
  FROM `motor`
}
```

Figura 9: Prefixos e início do mapeamento em SMS2. Fonte: Do autor, 2019.

Em seguida foi criado o RDF dos dados selecionados, associando as IRIs e os valores das colunas pertinentes por meio das propriedades definidas na ontologia (Figura 10).

```
to{
    ?subject rdfs:label ?name .
    ?subject :hasMotorPower ?hasMotorPower .
    ?subject rdf:type :Motor
}
```

Figura 10: Definição do RDF dos dados mapeados em SMS2.Fonte: Do autor, 2019.

Por fim, tem-se a definição dos identificadores individuais (IRIs) e as conversões de tipos de dados (Figura 11).

```
WHERE {
    BIND(template(
        "ontologies/function_cost_18#Motor{idMotor}"
        ) AS ?subject)
    BIND(xsd:double(?power) AS ?hasMotorPower)
}
```

Figura 11: Definição de IRIs e transformação de tipos de dados em SMS2.Fonte: Do autor, 2019.

O processo de mapeamento das tabelas foi repetido para todas as tabelas do *overheadcrane* e do *financial*, agrupando-as em dois blocos de mapeamento, um para cada banco.

3.2 Validação do Mapeamento

Para finalizar, foi realizada a conversão das questões de competência propostas em Voltoline (2019) em consultas SPARQL. As questões de competência visam validar a ontologia function_cost_18# como um todo, implicando em elas serem bons parâmetros para validar também o mapeamento e integração dos bancos de dados.

Como o Stardog apresenta os mapeamentos na forma de grafos virtuais, as consultas em SPARQL podem ser facilmente escritas como se cada mapeamento representasse um grafo nomeado da ontologia. Assim, a Figura 12 apresenta a implementação da pergunta “qual o custo de um motor com potência maior que 50 cavalos” em SPARQL.

```
SELECT ?motorName ?cost WHERE{
    graph <virtual://overheadcrane>{
        ?motor a :Motor;
            ?motor rdfs:label ?motorName;
            :hasMotorPower ?power.
    }
    graph <virtual://financial>{
        ?f rdfs:label ?motorName;
           :hasCost ?cost.
    }
    FILTER (?power > 50)
}
```

Figura 12: Exemplo de consulta SPARQL utilizando grafos virtuais. Fonte: Do autor, 2019.

4. RESULTADOS E DISCUSSÃO

Foram obtidos com esse trabalho dois grafos virtuais que unem os bancos de dados com a ontologia, sendo um grafo para cada banco. Os grafos virtuais proporcionam duas facilidades ao usuário da ontologia: trazer os dados mais atuais dos bancos de dados e requerer do usuário apenas o conhecimento de SPARQL.

A validação da integração da ontologia com os bancos de dados foi assegurada ao receber corretamente a resposta das questões de competência. A Figura 13 apresenta a resposta para a consulta “qual o custo de um motor com potência maior que 50 cavalos”. A coluna *motorName* tem a identificação do modelo do motor e está armazenada em *overheadcrane*. A coluna *cost* tem o valor de compra em reais do motor e está armazenada em *financial*.

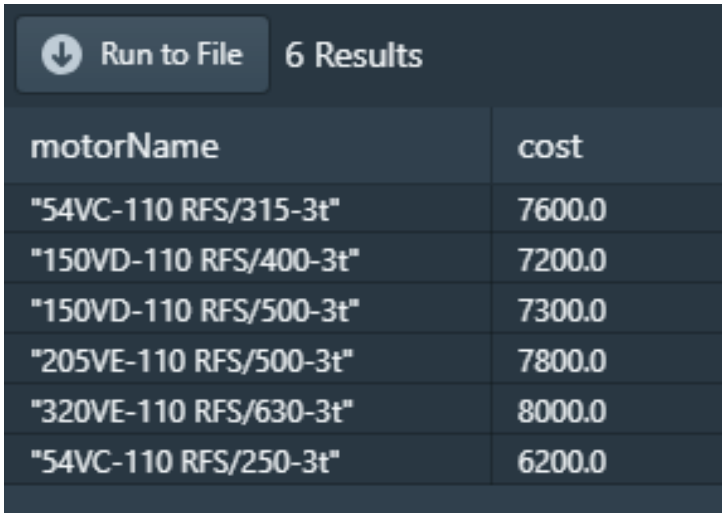

Figura 13: Resultado da consulta “Qual o custo de um motor com potência maior que 50 cavalos”. Fonte: Do autor, 2019.

A consulta apresenta a unificação direta entre os dados contidos no banco de dados do setor de montagem com os dados do setor financeiro.

Com relação as questões de competência, o código da Figura 14 apresenta a questão de competência 1. Ela busca saber o custo total do produto que atende uma dada especificação de operação. O processo de unificação é mais complexo, pois além de relacionar múltiplas tabelas do setor de montagem com o setor financeiro, também são utilizas as funções de custo descritas pela ontologia.

```
1.  SELECT (sum (?cost) AS ?total) WHERE{
```

```
2.  {
3.  graph<virtual://overheadcrane>{
4.  ?crane :hasDistanceX ?x;
5.  :hasTrolleyLoad ?xl;
6.  :hasSpeedX1 ?sx;
7.  :hasPart ?parts.
8.  ?parts rdfs:label ?name.
9.  }
10. graph<virtual://financial>{
11. ?xp rdfs:label ?name;
12. :hasCost ?cost.
13. }
14. FILTER(?x = 20 && ?xl>50 && ?sx=20)
15.
16. }UNION{
17. graph<virtual://overheadcrane>{
18. ?crane :hasDistanceY ?y;
19. :hasBeamLoad ?yl;
20. :hasSpeedY1 ?sy;
21. :hasPart ?parts.
22. ?parts rdfs:label ?name.
23. }
24. graph<virtual://financial>{
25. ?yp rdfs:label ?name;
26. :hasCost ?cost.
27. }
28. FILTER(?y=120 && ?yl=50 && ?sy=20)
29. }UNION{
30. graph<virtual://overheadcrane>{
31. ?crane :hasDistanceZ ?z;
32. :hasHoistLoad ?zl;
33. :hasSpeedZ1 ?sz;
34. :hasPart ?parts.
35. ?parts rdfs:label ?name;
36. }
37. graph<virtual://financial>{
38. ?zp rdfs:label ?name;
39. :hasCost ?cost.
40. }
41. FILTER(?z=100 && ?zl=50 && ?sz=4)
42. }
43. }
```

Figura 14: Questão de competência 1 em SPARQL. Fonte: Do autor, 2019.

A Figura 15 apresenta a resposta para a questão de competência 1, com a somatória dos custos das peças que melhor atendem as necessidades descritas, obtendo o valor de 54.700,00 reais.

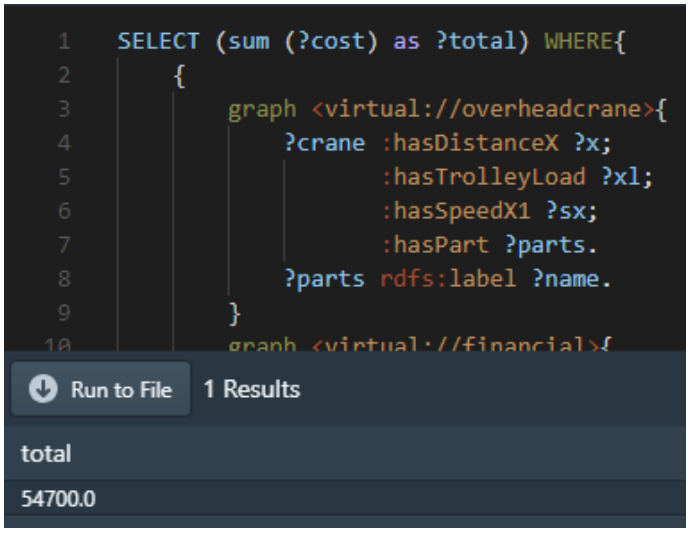

Figura 15: Resultado para a questão de competência 1. Fonte: Do autor, 2019.

Ao alterar a consulta para retornar os valores das variáveis utilizadas, a Tabela 1 é obtida. As colunas *crane* e *parts* são nativas do banco *overheadcrane*; as colunas *cost*, xp, yp e zp, do banco *financial*; e a coluna *name* é comum a ambos. Os valores da Tabela 1 são os componentes da montagem desejada e, assim como descritos em Voltolini (2019), são apresentados 16 itens. Ao somar os valores da coluna *cost* para identificar o custo total do produto, obtém-se o valor de 54.700,00 reais, assim validando o processo de unificação dos dados.

Tabela 1: Resultado da questão de competência 1 adaptada. Fonte: Do autor, 2019.

crane	parts	name	cost	xp	yp	zp
:Trolley3	:Accessory1	anticollisionDetector	2.000,00	:Purchase1		
:Trolley3	:Motor2	motor02	6.000,00	:Purchase10		
:Trolley3	:Reducer2	reducer02	2.200,00	:Purchase12		
:Trolley3	:Accessory2	severeUse	2.500,00	:Extra1		
:Trolley3	:Drum2	drum02	2.000,00	:Production2		
:Trolley3	:PulleyCompensationHoist2	puleyCompensation02	1.000,00	:Production8		
:Bridge3	:Accessory1	anticollisionDetector	2.000,00	:Purchase1		

:Bridge3	:Accessory2	severeUse	2.500,00	:Extra1	
:Bridge3	:Girder2	beam02	8.200,00	:Production5	
:Hoist3	:Accessory1	anticollisionDetector	2.000,00		:Purchase1
:Hoist3	:Accessory2	severeUse	2.500,00		:Extra1
:Hoist3	:Accessory5	handleTube	7.000,00		:Purchase3
:Hoist3	:Control3	control03	5.000,00		:Purchase8
:Hoist3	:Hook1	hook01	6.000,00		:Production6
:Hoist3	:PulleyHoist2	pulleyHoist02	1.300,00		:Production10
:Hoist3	:Rope2	rope02	2.500,00		:Purchase14

A plataforma Stardog é a responsável por integrar os dados provenientes dos bancos de dados implementados e deixá-los aptos a serem utilizados pela ontologia. A ontologia é a responsável por analisar as relações entre os dados e inferir novas informações.

5. CONCLUSÃO

Esta pesquisa demonstrou a praticabilidade da utilização da integração virtual para unificar banco de dados e ontologias. A técnica de integração virtual possibilita que ontologistas levem menos tempo para adicionar grandes quantidades de instâncias na ontologia, podendo focar no desenvolvimento dela.

A virtualização permite também que a responsabilidade na manutenção da consistência das instâncias fique centralizada no administrador dos bancos de dados, descartando a necessidade de retrabalho ao verificar se as cópias também estão concisas.

A massiva quantidade de dados heterogênicos requer soluções de unificação de dados que técnicas empregadas em silos de dados não conseguem suprir. A integração virtual, por outro lado, permite uma maior facilidade no acesso a esses dados, dando a confiabilidade necessária para que plataformas como o Stardog possam utilizá-los para alimentar sistemas que requeiram dados consistentes e atualizados.

Foi demonstrado, de forma empírica, a versatilidade e facilidade de implementação da unificação de dados. A integração virtual dos bancos foi rapidamente executada devido à

flexibilidade no mapeamento proporcionada pela SMS2 e sua validação atestada ao executar corretamente as questões de competências que nortearam a concepção da ontologia.

Para pesquisas futuras, fica aberta a oportunidade de verificar os ganhos e perdas ao se empregar a integração virtual para alimentar ontologias. Pode-se determinar os impactos no tempo de processamento das consultas SPARQL, ao considerar o tempo de acesso aos bancos. Pode-se também avaliar o esforço de implementação do mapeamento em relação a adicionar manualmente as instâncias na ontologia.

6. AGRADECIMENTOS

Agradeço a todas as pessoas e instituições que contribuíram para a realização deste trabalho, através da dedicação na orientação, da oferta de infraestrutura física e, em especial, ao CNPq (conselho Nacional de Desenvolvimento Científico e Tecnológico), pela bolsa de estudos e auxílio financeiro que possibilitou a dedicação ao programa de iniciação científica.

7. REFERÊNCIAS

AGRAWAL, N. et al. A five-year study of file-system metadata. **ACM Transactions on Storage**, [S. l.], v. 3, p. 9-41, out. 2007.

CIMAF. **Manual Técnico de Cabos**. 2014. Disponível em: <https://www.aecweb.com.br/cls/catalogos/aricabos/catalogocimaf2014completo.pdf>. Acesso em: 12 abr. 2019.

CSM. **Catálogo de Produtos**. 2019. Disponível em: <http://www.csm.ind.br/catalogo_engenharia.pdf>. Acesso em: 12 abr. 2019.

ELMASRI, R.; NAVATHE, S. B. The Relational Data Model and Relational Database Constraints. In: _____. **Fundamentals of Database Systems**. 7th ed. [S. l.]: Pearson, 2016. p. 150–152.

GENG, D. et al. A Method for Information Management Based on RDF Model and Ontology Technology. **Advances in Intelligent Systems Research**, Paris, v. 146, p. 165–168, mar. 2018.

HAN, J.; JIAO, Y. Conception of QAR Data Application Mode Based on Virtual Integration. **2018 IEEE International Conference of Safety Produce Informatization (IICSPI)**. p.392–395, 2018. IEEE.

LI, Haoyuan. **Alluxio:** A Virtual Distributed File System. 2018. 93 f. Tese (Doutorado em Ciência da Computação) - Universidade da Califórnia, Berkeley, 2018.

Mercado Livre. **Mercado Livre Brasil**. 2019. Disponível em: <https://www.mercadolivre.com.br/>. Acesso em: 13 abr. 2019.

STARDOG UNION. **What is a Knowledge Graph? [Whitepaper]**, Disponível em: <https://fetch.stardog.com/web-enterprise-knowledge-graph/>. Acesso em: 16 maio 2019.

STARDOG UNION. **Stardog Mapping Syntax**. Disponível em: <https://www.stardog.com/docs/#_stardog_mapping_syntax>. Acesso em: 16 maio 2019.

STARDOG UNION. **Stardog Mapping Syntax 2**. Disponível em: <https://www.stardog.com/docs/#_stardog_mapping_syntax_2>. Acesso em: 16 maio 2019.

STUDER, R.; BENJAMINS, V. R.; FENSEL, D. Knowledge Engineering: Principles and methods. **Data and Knowledge Engineering**, v. 25, p. 161–197, mar. 1998.

TEOREY, T. J.; YANG, D.; FRY, J. P. A logical design methodology for relational databases using the extended entity-relationship model. **ACM Computing Surveys**, v. 18, p. 197–222, jun. 2002.

VOLTOLINI, Rafael. **Cost Estimation In Initial Development Stages Of Products** - An Ontological Approach. 2019. 109 f. Dissertação (Mestrado em Engenharia Mecânica) - Universidade Tecnológica Federal do Paraná, Curitiba, 2019.

W3C. **OWL**. 2012. Disponível em: <https://www.w3.org/OWL/>. Acesso em: 13 maio 2019.

W3C. **R2RML:** RDB to RDF Mapping Language. 2012. Disponível em: <http://www.w3.org/TR/2012/REC-r2rml-20120927/>. Acesso em: 16 maio 2019.

WIDOM, Jennifer. Research problems in data warehousing. **Conference on Information and Knowledge Management**, p. 25–30, nov. 1995.