Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Arquitetura de Integração entre Bancos de Dados: Uma abordagem aplicada à Secretaria de Estado de Educação do Distrito Federal - SEEDF

Eduardo Pires Fernandes

Dissertação apresentada como requisito parcial para conclusão do
Mestrado Profissional em Computação Aplicada

Orientadora
Profª. Drª. Maristela Terto de Holanda

Coorientador
Prof. Dr. Marcio de Carvalho Victorino

Brasília
2017

Ficha catalográfica elaborada automaticamente,
com os dados fornecidos pelo(a) autor(a)

Fernandes, Eduardo
FF363a Arquitetura de Integração entre Bancos de Dados: Uma abordagem aplicada à Secretaria de Estado de Educação do Distrito Federal (SEEDF) / Eduardo Fernandes; orientador Maristela Holanda; co-orientador Marcio Victorino. -- Brasília, 2017.
57 p.

Dissertação (Mestrado - Mestrado Profissional em Computação Aplicada) -- Universidade de Brasília, 2017.

1. Banco de Dados. 2. Integração entre Bancos de Dados. 3. Ontologia. 4. Relacional. 5. NoSQL. I. Holanda, Maristela, orient. II. Victorino, Marcio, co-orient. III. Título.

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Arquitetura de Integração entre Bancos de Dados: Uma abordagem aplicada à Secretaria de Estado de Educação do Distrito Federal - SEEDF

Eduardo Pires Fernandes

Dissertação apresentada como requisito parcial para conclusão do
Mestrado Profissional em Computação Aplicada

Profª. Drª. Maristela Terto de Holanda (Orientadora)
CIC/UnB

Profª. Drª. Edna Dias Canedo
CIC/UnB

Profª. Drª. Ana Paula Bernardi da Silva
MGCTI/UCB

Prof. Dr. Marcelo Ladeira
Coordenador do Programa de Pós-Graduação em Computação Aplicada

Brasília, 12 de dezembro de 2017

Dedicatória

A Deus e aos bons espíritos, que nunca me abandonam estando comigo em qualquer escolha de luz que eu faça na vida.

A minha mãe Maria da Graça Silveira Pires, que com o seu amor incondicional, me criou e me apresentou a importância do estudo para o meu crescimento pessoal e profissional.

Ao meu pai Carlos Alberto Fernandes, que desde quando eu era pequeno acredita no meu potencial.

Com carinho, ao meu avô Joaquim Gonçalves Pires (*in memorian*), que me deu o exemplo de como ser um homem íntegro com a vida.

Ao meu irmão Felipe Pires Fernandes, que acredito no potencial de se tornar, além de um grande homem para a família, um grande matemático em sua vida profissional.

Ao meu cachorro Steve Jobs, que como o seu nome já diz, é um cão brilhante que sempre alegra os meus dias.

Agradecimentos

Agradeço à Universidade de Brasília que proporciona um ambiente de crescimento acadêmico com muitos desafios e aprendizados a serem conquistados.

À Secretaria de Estado de Educação do Distrito Federal que me proporciona um ambiente diário de crescimento profissional, colocando em prova todos aqueles meus conhecimentos adquiridos durante toda a minha vida acadêmica e profissional.

À minha orientadora Maristela Terto de Holanda, que desde a época da graduação acredita no meu potencial e me incentiva, com muita paciência e dedicação, a crescer academicamente e profissionalmente.

Ao meu coorientador Marcio de Carvalho Victorino, que é uma grande referência para mim como profissional.

Aos meus amigos pessoais, que desde quando me conheceram me alegram e me incentivam sempre a crescer na vida.

Aos meus colegas de trabalho, que acreditam no meu potencial como servidor público e desde o início me incentivam e me dão condições para concluir este mestrado.

Aos meus colegas de mestrado, que dia após dia, noite após noite, estivemos unidos, ajudando uns aos outros, para concluirmos com êxito mais essa etapa acadêmica.

Resumo

Com o crescimento no número de dados e de sistemas nos últimos anos, a integração entre banco de dados heterogêneos, algo não trivial, porém de suma importância, deve ser feito em empresas que apresentam diversos sistemas de *software* disponíveis e que queiram uma integração entre os mesmos. Neste contexto, esta dissertação propõe uma solução para o problema que consiste na definição de mecanismos que permitam integrar Sistemas Gerenciadores de Bancos de Dados Relacionais (SGBDR) com um NoSQL que aceite comandos SQL por meio de ontologia, com ênfase no ecossistema educacional da Secretaria de Estado de Educação do Distrito Federal (SEEDF). A arquitetura de integração entre bancos de dados heterogêneos proposta, integra SGBD a NoSQL, a partir de ontologias. Para permitir tal integração, foram desenvolvidas primitivas de mapeamento entre os termos ontológicos, além da sincronização de dados de acordo com regras estabelecidas. O mecanismo proposto foi validado por meio de um *middleware*, denominado Noach, que realizou a integração entre banco de dados heterogêneos do ecossistema computacional da SEEDF.

Palavras-chave: Bancos de Dados, Integração, Arquitetura híbrida, NoSQL, Relacional, Ontologia.

Abstract

With the growth in the number of data and systems in the last years, the integration between heterogeneous databases, which is not trivial but of great importance, should be done in companies that have several software systems available and wanting an integration between them. In this context, this dissertation proposes a solution to the problem that consists in the definition of mechanisms that allow integrating Relational Database Management Systems (RDBMS) with a NoSQL that accepts SQL commands through ontology, with emphasis on the educational ecosystem of the Secretariat of State of Education of the Federal District (SSEFD). The proposed heterogeneous database integration architecture integrates DBMS with NoSQL, based on ontologies. In order to allow such integration, mapping primitives were developed between the ontological terms, in addition to the synchronization of data according to established rules. The proposed mechanism was validated through a textit middleware, called Noach, which performed the integration between heterogeneous databases of the SSEFD computational ecosystem.

Keywords: Databases, Integration, Hybrid Architecture, NoSQL, Relational, Ontology.

Sumário

1 Introdução **1**
1.1 Contextualização 1
1.2 Problema de Pesquisa 2
1.3 Objetivos 2
1.3.1 Objetivos Específicos 2
1.4 Estrutura do Trabalho 3

2 Fundamentação Teórica **4**
2.1 Sistema de Banco de Dados 4
2.1.1 Modelos Relacionais 7
2.1.2 Modelos não relacionais 9
2.2 Integração entre Banco de Dados Heterogêneos 13
2.2.1 Técnicas de Integração entre Banco de Dados Heterogêneos 13
2.2.2 Arquiteturas e Métodos de Integração de Dados 16
2.3 Sincronização de Dados 21
2.3.1 Replicação de Dados 21
2.3.2 Tipos de Sincronização 21
2.4 Trabalhos Relacionados 22
2.4.1 Arquiteturas e Métodos de Integração de Dados 23
2.4.2 Técnicas de Integração entre Bancos de Dados Heterogêneos 23
2.5 Síntese do Capítulo 24

3 Arquitetura Proposta **26**
3.1 Ecossistema Computacional Antes da Integração 26
3.1.1 Retomada do Problema 27
3.2 Aspectos Necessários para a Arquitetura 28
3.3 Profissionais para a Integração 29
3.4 A Arquitetura de Integração Baseada em Ontologia 29
3.4.1 Arquitetura Lógica 29

3.4.2 Arquitetura Física 32
3.5 Ecossistema Computacional para Validação da Integração 36
3.6 Síntese do Capítulo 37

4 Análise da Arquitetura **39**
4.1 A escolha dos SGBD e do NoSQL 39
4.2 Criação da Ontologia 41
4.3 A Execução do Noach e Validação da Arquitetura 46
4.4 Síntese do Capítulo 50

5 Conclusão e Trabalhos Futuros **51**

Referências **53**

Apêndice **57**

A Ontologia de Aplicação da SEEDF **58**

Lista de Figuras

2.1 Diagrama simplificado de um ambiente de sistema de banco de dados. Adaptado de [1]. 6
2.2 Modelo Relacional - Tabela. 7
2.3 Teorema CAP. Adaptado de [2]. 12
2.4 Exemplo de *Schema Matching*. 14
2.5 Data Warehouse. 15
2.6 Banco de Dados Federado. 16
2.7 Arquitetura baseada em LINQ, adaptado de [3]. 17
2.8 Arquitetura baseada em *Web Service*, adaptado de [4]. 18
2.9 Arquitetura baseada em *Wrapper-Mediator*. 18
2.10 Arquitetura baseada em ontologia para bancos de dados. 20

3.1 Ecossistema Computacional Antes da Integração. 27
3.2 Arquitetura Lógica de Integração entre Bancos de Dados. 30
3.3 Equivalência Semântica entre Tabelas/Colunas com nomes diferentes. . . . 32
3.4 Arquitetura Física do Noach. 33
3.5 Cenário Após a Implantação da Arquitetura. 37

4.1 Esquema da Tabela Local - PostgreSQL. 39
4.2 Esquema da Tabela Local - MySQL. 40
4.3 Esquema da Tabela Global. 41
4.4 Protégé - OntoGraf - Classes. 41
4.5 Protégé - Classes. 42
4.6 Protégé - Equivalência Semântica entre Classes. 42
4.7 Protégé - Propriedades dos Objetos. 43
4.8 Protégé - Equivalência Semântica entre as Propriedades nome, nm_escola e nm_unidade_escolar. 44
4.9 Protégé - Equivalência Semântica entre as Propriedades cidade, telefone e type, respectivamente, com cid_unidade_escolar, tel_unidade_escolar e tp_unidade_escolar. 44

4.10 Protégé - Arquivo OWL. 45
4.11 Fluxo de Execução do *middleware* Noach. 46
4.12 Noach - Tela de Acesso 47
4.13 Noach - Página de Configuração. 47
4.14 Noach - Página de Status de Conexão. 48
4.15 Noach - Página de Sincronização 49

Lista de Abreviaturas e Siglas

ACID Atomicidade, Consistência, Isolamento e Durabilidade.

API *Application Programming Interface*.

BSON *Binary JSON*.

CAP *Consistency, Availability and Partition*.

CSS *Cascading Style Sheets*.

CSV *Comma-Separated Values*.

CT Consistência e Tolerância a Partições.

DC Disponibilidade e Consistência.

DDL *Data Definition Language*.

DFTrans Transporte Urbano do Distrito Federal.

DISIS Diretoria de Desenvolvimento de Sistemas.

DML *Data Manipulation Language*.

DT Disponibilidade e Tolerância a Partições.

DW *Data Warehouse*.

GDF Governo do Distrito Federal.

HTML *HyperText Markup Language*.

JSON *JavaScript Object Notation*.

LINQ *Language-Integrated Query*.

NoSQL *Not Only SQL.*

OLAP *On-line Analytical Processing.*

OLTP *On-line Transaction Processing.*

OWL *Web Ontology Language.*

PDO *PHP Data Objects.*

PHP *PHP: Hypertext Preprocessor*.

PLE Passe Livre Estudantil.

RAP *RDF API for PHP V0.9.6*.

RDF *Resource Description Framework.*

SBDF Sistema de Banco de Dados Federado.

SEEDF Secretaria de Estado de Educação do Distrito Federal.

SEQUEL *Structured English Query Language.*

SGBD Sistema Gerenciador de Banco de Dados.

SQL *Structured Query Language.*

SUMTEC Subsecretaria de Modernização e Tecnologia.

TI Tecnologia da Informação.

VM *Virtual Machines.*

XML *eXtensible Markup Language.*

Capítulo 1

Introdução

1.1 Contextualização

Para um gestor, na realização da tomada de decisão, é necessário ter conhecimento das informações que são geradas por seus sistemas computacionais. Para adquirir essas informações, nem sempre é uma tarefa fácil, visto que os dados podem estar dispostos em diversos sistemas. Para se obter estas informações é necessário que haja um relacionamento entre estes dados, ou seja, que eles estejam integrados. Este relacionamento pode ser feito por diversas técnicas, arquiteturas e métodos.

A Secretaria de Estado de Educação do Distrito Federal (SEEDF) é a maior Secretaria do Governo do Distrito Federal (GDF), no qual o público de atendimento é estimado aproximadamente em 460 mil estudantes. Para o funcionamento desta Secretaria, o seu corpo de servidores está em torno de 47 mil servidores efetivos e temporários. Com esse quantitativo de pessoas, são gerados por dia milhares de dados dentro dos mais diversos sistemas de informação da SEEDF/GDF.

Atualmente, a SEEDF está trabalhando intensivamente para a informatização de seus procedimentos internos para que haja uma melhora na velocidade, qualidade e eficiência na prestação de seus serviços ao público. Em especial, a Subsecretaria de Modernização e Tecnologia (SUMTEC), unidade interna da SEEDF responsável pela modernização e tecnologia, tem como missão prover recursos de Tecnologia da Informação (TI), visando apoiar os processos de ensino-aprendizagem e de gestão educacional.

Os dados da SEEDF são armazenados em diferentes tipos de Sistema Gerenciador de Banco de Dados (SGBD), ou seja, em um ambiente heterogêneo. Os principais SGBD utilizados na SEEDF são o MySQL, o PostgreSQL, o Oracle e o SQL Server. Todos estes SGBD são relacionais e, atualmente, a integração entre eles é realizada minimamente, isto é, apenas alguns sistemas contém algum tipo de integração com outros sistemas.

1.2 Problema de Pesquisa

Nos dias de hoje é um desafio integrar dados que se encontram em SGBD diferentes mantendo a interoperabilidade dos sistemas e a consistência da informação. Este tipo de integração é uma necessidade elementar para qualquer sistema de informação que almeje um ciclo de vida duradouro e para que uma organização tenha uma gestão de qualidade.

Coulouris em [5] e Tanembaum em [6] destacam que sistemas abertos são aqueles que se caracterizam pela interoperabilidade dos mesmos, ou seja, com uma comunicação de dados entre eles. Nesse sentido, é fundamental que interfaces estejam publicadas e disponíveis para que sistemas heterogêneos possam se comunicar.

A Diretoria de Desenvolvimento de Sistemas (DISIS), unidade orgânica de direção interna a SUMTEC, compete gerenciar as ações relativas a todos os sistemas e ferramentas utilizadas pela SEEDF, desenvolver os sistemas de informações necessários ao atendimento das necessidades, executar ações para estabelecer metodologia de desenvolvimento de sistemas, coordenar a prospecção de novas tecnologias da informação e comunicação no âmbito da SEEDF e manter um processo contínuo de planejamento, desenvolvimento, implantação e manutenção de sistemas. Por esses motivos, ela tem como necessidade primordial que as bases de dados de seus sistemas se integrem de forma consistente para que haja uma gestão de qualidade e melhoria na educação pública do Distrito Federal, provendo relatórios gerenciais, analisando e minerando esses dados gerados que são de suma importância para a organização.

Desta maneira, este trabalho vem com o intuito de diminuir a falta de integração, identificando e integrando os bancos de dados existentes dentro da SEEDF, de tal modo que os seus dados estejam consistentes e integrados de forma autônoma, isto é, automática entre os diversos sistemas de informação.

1.3 Objetivos

A SEEDF, por possuir um enorme volume e variedade de dados que cresce com uma velocidade surpreendente, tem a necessidade de mantê-los e integrá-los entre as estruturas heterogêneas. O objetivo deste trabalho é o desenvolvimento de uma arquitetura que permita a integração automática de dados, por meio da utilização de ontologia, entre os SGBD Relacionais com esta base de dados global.

1.3.1 Objetivos Específicos

No intuito de atingir o objetivo geral, foram definidos alguns objetivos específicos:

- Definir os aspectos necessários que uma arquitetura de integração de banco de dados deve ter para o ambiente computacional da SEEDF, e também para um ambiente genérico, de forma que funcione entre bancos de dados heterogêneos;
- Especificar e implementar uma arquitetura de integração entre bancos de dados;
- Validar a arquitetura no ambiente computacional da SEEDF, identificando e analisando os resultados.

1.4 Estrutura do Trabalho

Este documento está dividido nos capítulos apresentados a seguir:

- Capítulo 2 apresenta a fundamentação teórica necessária para o desenvolvimento da pesquisa. Os principais conceitos, características e desafios de integração de bancos de dados em um ambiente heterogêneo são descritos e discutidos. Os temas que são abordados neste capítulo envolve, sistemas de banco de dados relacionais e não relacionais, técnicas de integração entre banco de dados heterogêneos, arquiteturas e métodos de integração de dados e os tipos de sincronização. Além disso, são apresentados os trabalhos relacionados;
- Capítulo 3 mostra o ecossistema computacional da SEEDF antes da integração, a explicação lógica e física da arquitetura de integração baseada em ontologia, além do ecossistema computacional posterior a integração;
- Capítulo 4 traz uma análise sobre a arquitetura, demonstrando o motivo da escolha dos SGBD e do NoSQL, como foi realizada a criação da ontologia e dos SGBD Locais Integrados e a execução do *middleware* Noach, validando a arquitetura proposta.
- Capítulo 5 é realizada a conclusão e apresentado quais são os trabalhos futuros da pesquisa.

Capítulo 2

Fundamentação Teórica

Neste capítulo, é apresentado a fundamentação teórica necessária para o desenvolvimento deste trabalho, algumas técnicas de integração de banco de dados heterogêneos, arquiteturas e métodos de integração de dados existentes são descritos. Este capítulo é dividido nas seguintes seções: 2.1 Sistema de Banco de Dados, onde são apresentadas as principais características dos sistemas de banco de dados relacionais e não relacionais; 2.2 Técnicas de Integração entre Banco de Dados Heterogêneos, que aborda os conceitos e as características das principais técnicas existentes; 2.3 Arquiteturas e Métodos de Integração de Dados, que apresenta o conceito dos principais tipos de integração entre dados; 2.4 Tecnologias de Sincronização, que fala sobre a replicação de dados e os tipos de sincronização e 2.5 Síntese do Capítulo, que apresenta um breve resumo do capítulo.

2.1 Sistema de Banco de Dados

Um banco de dados, em Date [7], é definido como "uma coleção de dados persistentes, usada pelos sistemas de aplicação de uma determinada empresa", com dados definido como "fatos conhecidos que podem ser registrados e possuem significado implícito", como por exemplo, nomes, números de telefone e endereço de pessoas do seu ambiente de trabalho. Esses dados podem estar registrados em uma agenda do seu *smartphone*, em um sistema de gestão de pessoas da sua empresa ou até mesmo em uma agenda de papel. Essa coleção de dados relacionados, com um significado implícito, é um banco de dados.

É bastante genérica essa definição de banco de dados, pois o conjunto de palavras que compõem este documento, pode ser considerado dados relacionados, que apresentam um significado implícito, portanto, constitui um banco de dados. Porém, sabe-se que o termo banco de dados é comumente mais restrito e que apresenta as seguintes propriedades implícitas [1]:

- Um banco de dados é logicamente uma coleção de dados previamente pensada com algum significado pertinente.
- Representando algum aspecto do mundo real, um banco de dados, às vezes é chamado de universo de discurso (*UoD - Universe of Discourse*) ou de minimundo.
- Um banco de dados possui um grupo definido de usuários e algumas aplicações previamente geradas, além de serem projetados, construídos e populados com dados para essas finalidades.

Um banco de dados precisa representar verdadeiramente o reflexo de um minimundo, ou seja, pode-se dizer que ele apresenta alguma fonte da qual o dado é extraído, algum tipo de interação com eventos e um público que tem interesse real em seu conteúdo para extrair informações importantes.

Um banco de dados pode apresentar diversos níveis de complexidade e de tamanho, como por exemplo, a lista de nomes, número e endereços da agenda telefônica mencionados anteriormente pode ter uma estrutura simples e consistir em apenas algumas centenas de registros. Por outro lado, existem sistemas com uma complexidade bem maior, que organizam e gerenciam uma grande quantidade de informação, de modo que os usuários possam criar, recuperar, atualizar e excluir dados, como é o caso do sistema de gestão escolar da SEEDF que gere as mais de 700 escolas e a vida escolar dos mais de 460 mil estudantes.

Um Sistema Gerenciador de Banco de Dados (SGBD) é definido como sendo um tipo específico de software usado para criar, armazena, organizar e acessar dados a partir de um banco de dados, ou seja, um software de uso geral, que permite aos usuários criar e manter um banco de dados. O SGBD facilita o processo de definição, que especifica os tipos, as estruturas e restrições dos dados a serem armazenados, de construção, que armazenam os dados em algum meio controlado pelo SGBD, de manipulação, que por exemplo inclui funções como a inclusão de consulta ao banco de dados para a atualização de algum dado específico, de compartilhamento, que permite que diversos usuários e programas acessem-no simultaneamente [8].

Um Sistema de Banco de Dados é definido por Elmasri e Navathe em [1] como sendo a união do banco de dados com o software SGBD, isto é, a união dos dados e metadados em um banco de dados, um acesso direto de um SGBD para processar as consultas, além de uma aplicação/sistema e um usuário/programador para realizar isso. Isso é demonstrado na Figura 2.1.

Um **Modelo de Dados**, ou modelo de banco de dados, é definido por Carlos Heuser em [9] como sendo uma "descrição formal da estrutura de um banco de dados", por Elmasri e Navathe em [1] que modelo de dados é "uma coleção de conceitos que podem ser

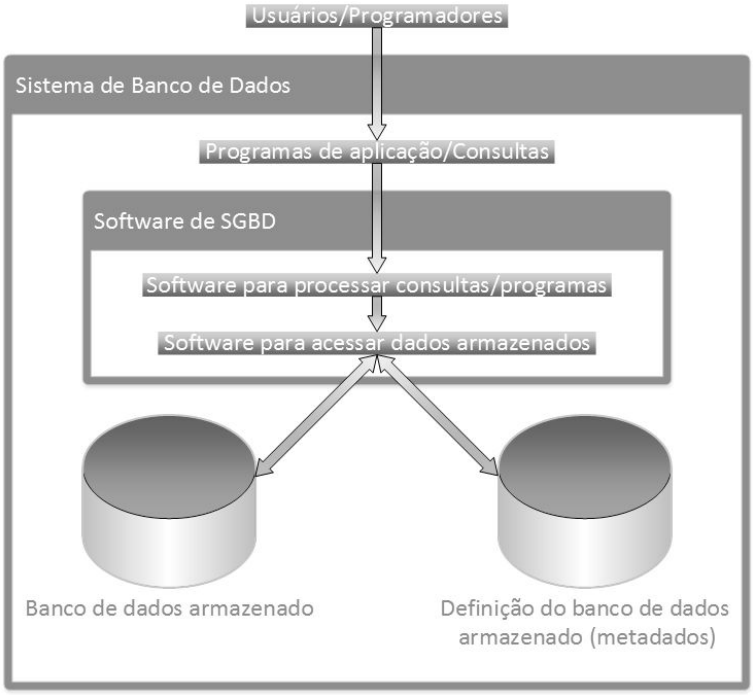
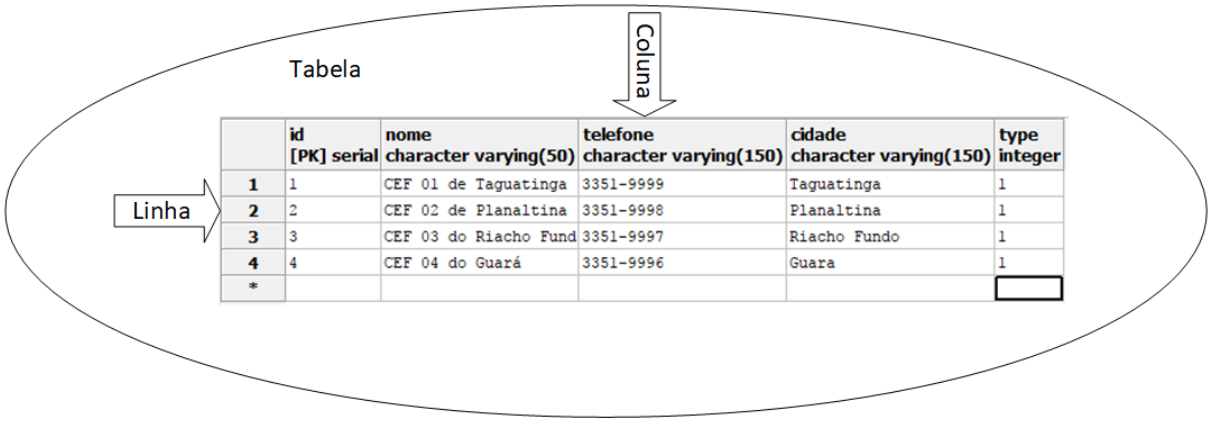


Figura 2.1: Diagrama simplificado de um ambiente de sistema de banco de dados. Adaptado de [1].

usados para descrever a estrutura de um banco de dados"e por Date em [7] como sendo "definição abstrata, autônoma e lógica dos objetos, operadores e outros elementos que, juntos, constituem a máquina abstrata com a qual os usuários interagem, isto é, em todas as definições pode-ser dizer que modelo de dados é uma descrição dos tipos de dados que estão armazenadas em um banco de dados. Essa definição oferece os meios necessários para alcançar a abstração de dados, que é uma característica fundamental da abordagem de banco de dados. A abstração de dados é definida por Elmasri e Navathe em [1] como sendo à "supressão de detalhes da organização e armazenamento de dados, destacando recursos essenciais para um melhor conhecimento desses dados".

A tecnologia nos dias de hoje evolui e nas grandes aplicações web é cada vez mais comum ter um grande crescimento na quantidade de dados. Para lidar com esse grande conjunto de dados de forma eficiente é necessário escolher o tipo correto de modelo de banco de dados a ser utilizado de acordo com cada aplicação conforme [10]. Neste contexto, é explicado nas próximas subseções os conceitos e as principais características dos modelos de bancos de dados relacionais e dos não-relacionais, também conhecidos como NoSQL.



	id [PK] serial	nome character varying(50)	telefone character varying(150)	cidade character varying(150)	type integer
1	1	CEF 01 de Taguatinga	3351-9999	Taguatinga	1
2	2	CEF 02 de Planaltina	3351-9998	Planaltina	1
3	3	CEF 03 do Riacho Fund	3351-9997	Riacho Fundo	1
4	4	CEF 04 do Guará	3351-9996	Guara	1
*					

Figura 2.2: Modelo Relacional - Tabela.

2.1.1 Modelos Relacionais

Surgido como sucessor dos modelos hierárquico e de redes, o modelo relacional se tornou padrão para a grande maioria dos SGBD e teve origem no início dos anos 1970 no artigo de Edgar F. Codd intitulado *A Relational Model of Data for Large Shared Data Banks* [11], que traz como elementos básicos as tabelas (ou relações), as quais são compostas por linhas (ou tuplas) e colunas (ou atributos), conforme pode ser visto na Figura 2.2.

Uma das características do modelo relacional é a garantia que a restrição de integridade dos dados seja mantida [7], ou seja, que os dados não sofram alterações impróprias ou sem permissão. Essa garantia é realizada mais comumente pelas chaves, mais especificamente, a chave primária, que foi criada com o objetivo de identificar unicamente as linhas da tabela e ainda de determinar a ordem física dessas linhas, e a chave estrangeira, que permite uma relação de dependência entre atributos de tabelas diferentes, de tal forma que os valores permitidos em uma coluna dependam dos valores de outra coluna de uma tabela distinta. Essas definições de chaves servem como base para a definição de índice, que são elementos que proporcionam um significativo ganho de desempenho no processamento de consultas [7].

Outra característica importante do modelo relacional é o processo conhecido como normalização, que pode ser definido em Elsmari e Navathe em [1], como a aplicação de certas regras sobre as tabelas de forma que garantem às tabelas, o projeto adequado, ou seja, a separação de dados referentes a elementos distintos em tabelas distintas, associadas por meio da utilização de chaves. Carlos Heuser em [9], define normalização como sendo o processo que se baseia no conceito da forma normal, isto é, regras que devem ser obedecidas pelas tabelas para que sejam consideradas bem projetadas. Heuser, em [9], considera quatro formas normais utilizadas na normalização, denominadas, primeira, segunda, terceira e quarta forma normal, abreviadamente 1FN, 2FN, 3FN e 4FN. A primeira forma normal diz que uma tabela não deve conter tabelas aninhadas, isto é, não podem ter

valores repetidos e nem atributos possuindo mais de um valor. A segunda forma normal mostra que além da tabela estar na 1FN, não deve conter dependências parciais, ou seja, quando uma coluna depende apenas de uma chave primária composta. Além de estar na 2FN, a terceira forma normal, não pode conter dependências transitivas, isto é, toda coluna não chave depende diretamente apenas da chave primária. Para a quarta forma normal, além de estar na 3FN, não deve conter dependências multivaloradas, ou seja, não deve ocorrer a presença de uma ou mais linhas em uma tabela que implicam a presença de uma ou mais outras linhas na mesma tabela.

Para a realização de operações nos elementos da tabela de um modelo relacional, foi desenvolvido por Donald D. Chamberlin e Raymond F. Boyce, ambos pesquisadores da IBM no ano de 1974 a *Structured English Query Language (SEQUEL)* [12], posteriormente chamada de Linguagem Estruturada de Consulta em [13], que é conhecida como *Structured Query Language (SQL)*, que é uma linguagem de definição, do inglês *Data Definition Language (DDL)* e de manipulação, do inglês *Data Manipulation Language (DML)* de dados. Essa linguagem foi inspirada na álgebra relacional, e por ser simples e com um alto poder de expressão, tornou-se uma das linguagens de consulta mais utilizadas no mundo, ajudando a consolidar o modelo relacional.

Além dessas características, os SGBD Relacionais apresentam a possibilidade que vários usuários possam manipular e acessar os dados simultaneamente de forma eficiente.

Caso ocorra algum tipo de falha, os SGBD Relacionais apresentam outra característica importante, que é a de recuperação a falha [7], isto é, a capacidade de retornar ao ponto anterior à falha ocorrida de forma adequada, garantindo que o banco permaneça sempre em um estado consistente.

Devido a essas características, os SGBD relacionais se mantiveram em destaque entre os mais diversos tipos de ambientes computacionais, porém não coibiram o surgimento de certos problemas quando fala-se do fenômeno do *Big Data* presentes nos bancos de certas organizações, principalmente aquelas que trabalham com sistemas web de ambientes colaborativos, como por exemplo, as redes sociais.

Limitações

Com a popularização das redes sociais, principal precursor, porém não único, houve um grande desafio a ser realizado pelos pesquisadores de banco de dados, pois com o crescimento expressivo do número de usuários e um fenômeno conhecido como *Big Data*, começou a ter um ritmo de aumento acelerado no número de dados a serem armazenados, caindo o desempenho dos SGBD relacionais, como mencionado em [14], porém existem outras pesquisas que questionam esse pouco desempenho em ambientes com grande vo-

lume de dados, como em [15]. Esse questionamento dos SGBD Relacionais não serem performáticos, deve-se ao fato deles apresentarem regras e estruturas rígidas.

De uma maneira geral, o modelo relacional apresenta dificuldades quando a demanda por escalabilidade horizontal passa a ser cada vez mais frequente, como por exemplo, quando determinada aplicação web passa a ter um crescimento substancial no número de usuários.

Esse tipo de aplicação começa a ter uma queda em seu desempenho e para solucionar esse problema, tem-se duas alternativas. Fazer um aprimoramento no servidor, isto é, aumentar a sua capacidade de processamento e memória ou aumentar o número de servidores para abarcar essa quantidade de usuários, porém em ambas situações não resolveriam o problema, caso esse número de usuários crescesse em ritmo cada vez mais intenso.

Para casos assim, o problema passaria a ser o próprio acesso ao banco de dados. A abordagem mais plausível para solucionar este problema seria a de escalar o banco de dados, distribuindo-o em diversos outros servidores, porém mesmo sendo possível fazê-la em um SGBD relacional, ela não é realizada de maneira simples, devido a sua natureza estruturada.

Os desenvolvedores passaram a perceber que em sistemas distribuídos, trabalhando com particionamento de dados, o modelo relacional apresentava dificuldades em se organizar os dados [16], e este é um dos pontos em que percebe-se que os modelos não relacionais podem conter o foco para essa solução.

2.1.2 Modelos não relacionais

Ao se propor alternativas ao uso do modelo relacional em ambientes que precisariam de um maior grau de escalabilidade horizontal, os pesquisadores de banco de dados passaram a pensar de uma maneira diferente, que flexibilizaria a estrutura fixa do modelo relacional em uma estrutura livre de certas características presentes nele.

Para ganhar o desempenho proposto em ambientes distribuídos, tais soluções perdiam diversas regras de consistência presentes no modelo relacional, flexibilizando os sistemas de banco de dados para as características particulares desses ambientes.

O termo NoSQL apareceu no final da década de 1990 em [17], a partir de uma solução de banco de dados relacional de código aberto que não oferecia uma interface SQL. Futuramente, o termo passou a representar soluções alternativas ao modelo relacional, ou seja, os não relacionais, tornando-se uma abreviação de *Not Only SQL* (não apenas SQL). Essas soluções apareceram com o propósito de não substituir os modelos relacionais como um todo, mas de substituí-los apenas nos casos que seja necessária uma maior flexibilidade na estruturação do banco de dados.

Em 2003, o Google lançou o Big Table, que é um banco de dados proprietário de alto desempenho que tem como objetivo promover uma maior disponibilidade e escalabilidade dos dados [18], tendo como ideia principal a flexibilização da forte estrutura utilizada do modelo relacional, sendo assim, uma das primeiras implementações de um NoSQL, ou também conhecido como modelo não relacional.

Em 2007, a Amazon lançou o Dynamo, que é um banco de dados NoSQL que tem como característica básica ser um banco de dados que utiliza-se do modelo não relacional, apresentando uma alta disponibilidade e sendo utilizado pelos serviços web da própria empresa [19].

Em 2008, foi a vez do Facebook lançar o Cassandra, que também foi desenvolvido para lidar com grande volume de dados distribuídos e com uma alta disponibilidade [20]. Demonstrando sua importância cada vez mais crescente, em 2010, o Cassandra substituiu o MySQL como banco de dados do Twitter [21].

Em 2009, a fundação Apache lançou o CouchDB, que é um banco de dados de código aberto orientado a documentos, o qual armazena os dados sem necessidade de esquema pré-definido [22], e que utiliza o conceito do modelo de programação *MapReduce*, especialmente projetada para suportar computação distribuída em larga escala [23].

No mesmo ano, foi lançado o MongoDB, que é um banco de dados livre de esquemas, de alto desempenho, escalável e orientado a objetos, com várias outras características semelhantes ao CouchDB [24].

Todas essas soluções até então apresentadas possuem características em comum e outras específicas que as diferenciam. As cinco principais características comuns a um NoSQL, de acordo com [25] são:

1. A habilidade de replicar e distribuir dados através de vários servidores;
2. Uma simples interface de chamada de nível ou protocolo (em contraste com uma chamada SQL);
3. Um modelo de concorrência mais fraco do que as transações de Atomicidade, Consistência, Isolamento e Durabilidade (ACID), que apresentam na maioria dos SGBD relacionais;
4. Uso eficiente de índices e memória RAM para o armazenamento de dados;
5. A capacidade de adicionar dinamicamente novos atributos aos dados já registrados.

Em relação à classificação do modelo de dados dos NoSQL, não existe um consenso na comunidade acadêmica, porém neste trabalho se utilizará das quatros categorias citadas em [26]:

- Chave-valor: Estas bases de dados permitem armazenar dados arbitrários em uma chave. Eles funcionam de modo semelhante a uma tabela *hash* convencional, mas com a distribuição de chaves (e valores) entre um conjunto de nós físicos;
- Orientado a Documentos: Um documento é uma série de campos com atributos, como por exemplo: nome="Maria"e sobrenome="Silva"é um documento com dois atributos. Muitos bancos de dados deste tipo armazenam documentos em formatos semi-estruturados como o *eXtensible Markup Language* (XML), *JavaScript Object Notation* (JSON) ou *Binary JSON* (BSON). Eles trabalham parecidamente como os bancos de dados chave-valor, porém a chave é sempre um identificador do documento e o valor é um documento com um pré-definido tipo (JSON ou XML, por exemplo) que permite consultar um campo do documento.
- Família de Colunas: Em vez de armazenar os dados em linhas (como em bancos de dados relacionais), este tipo de banco de dados armazena-os em colunas. Assim, algumas linhas podem não conter dados em algumas colunas, oferecendo flexibilidade na definição de dados e possibilitando a aplicação de algoritmos de compressão de dados por coluna. Além disso, as colunas que não são frequentemente consultadas juntas podem ser distribuídas através de diferentes nós.
- Orientado a Grafos: Estes bancos de dados destinam-se em armazenar dados em estruturas de grafos. O dado é representado por arestas e vértices, com seus atributos particulares. A maioria destes bancos de dados permitem uma eficiente travessia de grafo, mesmo quando os vértices estão em nós físicos separados. Além disso, este tipo de banco de dados tem recebido muita atenção devido à sua aplicabilidade em dados sociais.

Teorema CAP e a Classificação dos NoSQL

De acordo com Brewer [27], no ano 2000, foi apresentado o Teorema CAP (acrônimo de *Consistency, Availability and Partition*), que traz como ideia central a premissa que nenhum sistema distribuído consegue satisfazer simultaneamente as três características a seguir, podendo apenas cumprir duas delas:

- Forte Consistência: todos os clientes veem a mesma versão dos dados, mesmo em atualizações para um determinado conjunto de dados;
- Alta Disponibilidade: todos os clientes podem sempre encontrar pelo menos uma cópia dos dados solicitados, mesmo que algumas das máquinas em um *cluster* esteja *off-line*;

Figura 2.3: Teorema CAP. Adaptado de [2].

- Tolerância a Partição: propriedade de um sistema continuar funcionando mesmo quando um problema ocorre na rede dividindo o sistema em duas ou mais partições, o que faz com que os nós de uma partição não consigam se comunicar com as outras partições.

De acordo com o Teorema CAP e os diferentes interesses dos NoSQL, pode-se fazer uma classificação preliminar dos bancos de dados NoSQL conforme descrição a seguir e Figura 2.3:

- Interesse na **Disponibilidade e Consistência (DC)**: Parte da base de dados usa principalmente a abordagem de replicação para garantir a disponibilidade e a consistência dos dados. Os sistemas que adotam essa abordagem são: o banco de dados relacional tradicional, o Vertica, Aster, o Greenplum, dentre outros;
- Interesse na **Consistência e Tolerância a Partições (CT)**: Estes bancos armazenam o dado nos nós distribuídos, garantindo a consistência desses dados. Os sistemas que adotam essa abordagem são: BigTable, Hypertable, HBase, MongoDB, Terrastore, Redis, Scalaris, MemcacheDB e o Berkeley DB.
- Interesse na **Disponibilidade e Tolerância a Partições (DT)**: Esses bancos de dados resistem a falhas de hardware, software e energia, cujo objetivo é manter os serviços disponibilizados o máximo de tempo possível. Os sistemas que adotam essa abordagem são: Voldemort, Tokyo Cabinet, KAI, CouchDB, SimpleDB, Riak, dentre outros.

2.2 Integração entre Banco de Dados Heterogêneos

Softwares são produtos caros de tal maneira que para receber este retorno investido, o software deve ser utilizado por vários anos. Grandes empresas e organizações, como é o caso da SEEDF, dependem de sistemas que tem mais de décadas de funcionamento. Muitos desses antigos sistemas ainda são fundamentais para o funcionamento dessas empresas e órgãos, e qualquer falha no serviço prestado por eles, pode acarretar um sério efeito no dia a dia de trabalho nessas organizações. A esses sistemas antigos, fundamentais para uma organização, foi dado o nome de sistemas legados [28].

Para as organizações é de fundamental importância uma arquitetura de dados que integre esses bancos de dados legados aos novos. No momento da definição do projeto de integração de dados de sistemas de software, devem ser levadas em consideração, as vantagens oferecidas por cada uma das técnicas de persistências disponíveis e a necessidade da aplicação. É de suma importância também reparar as tendências, pois, ao passar do tempo, o volume de informação deve ter um aumento considerável e, associado a este fato, os usuários passam a ser mais exigentes, especialmente em relação à velocidade do tráfego dos dados [29].

Atualmente, a integração de bancos de dados não tem apenas como vantagem ou característica não funcional que deva ser observada, é uma necessidade para sistemas de informação que almeje um ciclo de vida duradouro e para uma gestão de qualidade. Coulouris em [5] e Tanembaum em [6] destacam que sistemas abertos se caracterizam pela interoperabilidade dos mesmos. Nesse sentido é fundamental que interfaces estejam publicadas e disponíveis para que sistemas heterogêneos possam se comunicar.

2.2.1 Técnicas de Integração entre Banco de Dados Heterogêneos

A integração entre banco de dados heterogêneos é uma das soluções para a interoperabilidade dos sistemas de informação e será discutido a seguir quais as técnicas que são mais utilizadas, de acordo com [30].

Correspondência de Esquemas *(Schema Matching)*

Uma das técnicas de integração de dados é a *Schema Matching* (Correspondência de Esquemas). Essa técnica é definida como sendo um processo de identificação de dois atributos semanticamente equivalentes entre esquemas diferentes, ou seja, identifica-se dois atributos em esquemas distintos que apresentam em seu conteúdo o mesmo domínio semântico. Quando essa identificação é realizada, pode-se dizer que foi realizado um

match, que é diferente de um *mapping*. O *mapping* ocorre quando é referenciada uma transformação entre estes atributos, ou seja, que para serem sematicamente equivalentes, necessitam de uma transformação [31].

Para facilitar o entendimento desses conceitos, tem-se como exemplo os dois esquemas a seguir:

- $BD1.aluno(cod, nome\_completo, data\_nascimento, escola\_cod)$;
- $BD2.estudante(id, nome, idade, turma\_id)$.

Como pode ser visto na Figura 2.4, o *match* ocorre quando a possível correspondência entre $BD1.aluno - nome\_completo$ e $BD2.estudante - nome$ é realizada. Já o *mapping* ocorre quando para ter uma equivalência semântica entre $BD1.aluno - data\_nascimento$ e $BD2.estudante - idade$ é necessários fazer o cálculo a partir da data de nascimento para saber qual é a idade do aluno/estudante.

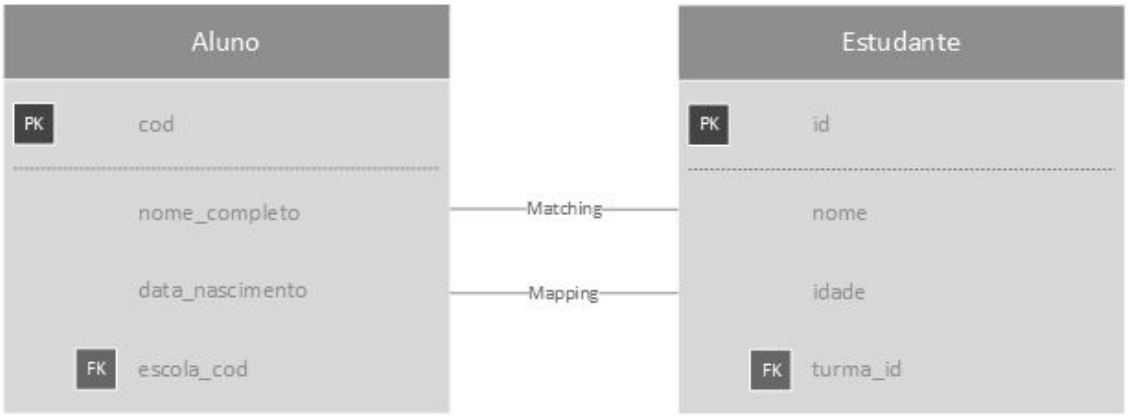


Figura 2.4: Exemplo de *Schema Matching*.

O *Schema Matching* é muitas vezes realizado manualmente, normalmente apoiado por uma ferramenta gráfica. Obviamente, quando esse processo é feito manualmente ele fica lento, além de ser propenso a erros humanos. Este problema cresce quando se tem diversas fontes de dados, e foi pensando nisso que a comunidade de pesquisadores de banco de dados estão constantemente tentando automatizar este processo, como pode ser visto em [31].

Armazém de Dados (*Data Warehouse*)

Segundo Inmon em [32], um *Data Warehouse (DW)* seria definido como "um conjunto de banco de dados integrados baseados em assuntos, onde cada instância do dado está relacionada a um momento". Nota-se que de acordo com essa definição, um ambiente de *DW* visa a integração de várias fontes de dados, permitindo acesso rápido a um grande

número de dados consolidados por meio de um conjunto de ferramentas de consulta, análise e apresentação das informações disponíveis, não consistindo apenas em dados [33].

O *DW* é considerado uma integração física de dados, pois os mesmos são materializados em um esquema global [34], ou seja, eles são extraídos de diversas fontes de dados, manipulados e armazenados em um esquema, conforme pode ser visto na Figura 2.5.

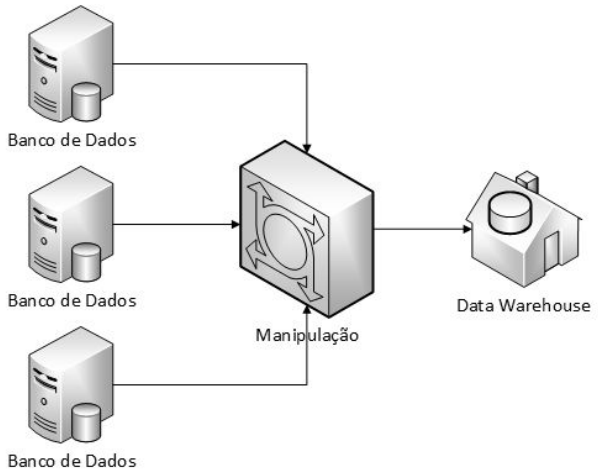


Figura 2.5: Data Warehouse.

Em um *DW* geralmente existe um banco de dados especializado, ou seja, um resultado da consolidação e do gerenciamento do fluxo de informação provenientes dos bancos de dados corporativos, até mesmo, de fontes externas às organizações. O foco de um DW é diferente daqueles contidos em bancos de dados operacionais, pois como principal objetivo, eles apresentam a análise de dados e a tomada de decisão e não a execução operacional do negócio. Isso acontece porque um DW foi criado com o objetivo de *On-line Analytical Processing* (OLAP), ou seja, de analisar grandes volumes de dados nas mais diversas perspectivas, e não de *On-line Transaction Processing* (OLTP), que visa automatizar tarefas transacionais de dados [35].

Banco de Dados Federado (*Federated Databases*)

Um Sistema de Banco de Dados Federado (SBDF) é um conjunto de sistemas de bancos de dados que são autônomos e possivelmente heterogêneos que trabalham cooperativamente para um fim único [36]. Cada banco de dados de um sistema de banco de dados federado é completamente auto-sustentável e funcional. Quando é enviada alguma consulta, o sistema descobre em qual banco de dados os dados requerentes dessa consulta estão e envia essa consulta para ele. Um banco de dados federado pode ser pensado como sendo uma virtualização de banco de dados, conforme é visto na Figura 2.6.

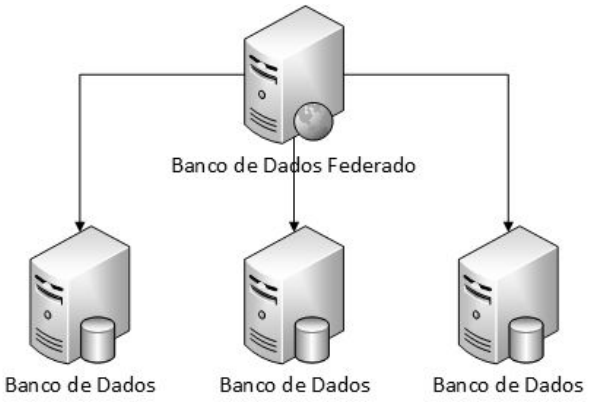


Figura 2.6: Banco de Dados Federado.

Diferentemente de um DW, que pega os dados de diversas fontes e os colocam fisicamente em um único lugar (esquema global), os bancos de dados federados, fornecem um armazém virtual, sem mover os dados, cujo principal papel é a tradução de consultas que vem do cliente para as diversas fontes de dados [34].

2.2.2 Arquiteturas e Métodos de Integração de Dados

Por meio das técnicas de integração de bancos de dados foram desenvolvidas algumas arquiteturas e métodos de integração de dados que facilitam a interoperabilidade de sistemas. Estas arquiteturas e métodos são apresentados a seguir.

Language-Integrated Query - LINQ

A arquitetura baseada em *Language-Integrated Query* (LINQ) apresenta um projeto de integração de dados heterogêneos de múltiplas fontes. O LINQ inclui quatro camadas principais, conforme pode ser visto na Figura 2.7 [3]:

- A **camada de dados heterogêneos** consiste em uma variedade de fontes de dados suportadas pelo LINQ, podendo ser dados estruturados, *web service*, sistema de arquivos, SQL, entre qualquer outro formato em que o LINQ oferece disponibilidade;
- A **camada de acesso aos dados** é uma camada computacional que permite um acesso rápido e simples aos dados armazenados. Essa camada é a base para todo o sistema de integração de dados, pois ela consiste em um conversor de dados baseado em LINQ;
- A **camada lógica de negócio**, que também é conhecida como camada de domínio, separa a lógica de negócio dos outros módulos, como a camada de acesso aos dados e a interface de dados e consulta. Esta camada tem como responsabilidade analisar

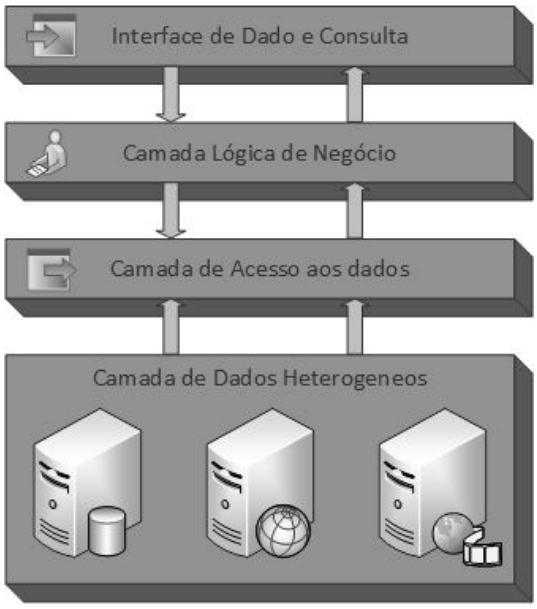


Figura 2.7: Arquitetura baseada em LINQ, adaptado de [3].

e decompor as consultas apresentadas pela interface de dados para a camada de acesso aos dados;

- A **camada de dado e consulta** é a camada mais externa, responsável por exibir os dados e receber a entrada do usuário.

Web Service

O método de integração de dados conhecido como *Web Service* é considerado como auto-suficiente, auto-descritivo, como aplicações modulares que podem ser publicadas, localizadas e invocadas por meio da *internet*, isto é, são componentes que permitem aos produtos de *softwares* enviarem e receberem os dados através de estruturas como o formato XML, permitindo que cada aplicação trabalhe com sua própria linguagem desde que haja uma tradução para alguma linguagem universal, como o XML, JSON, CSV, etc [4].

Como pode ser visto na Figura 2.8, a arquitetura tem dois tipos de participantes, o provedor de serviço, que prover o *web service* para o usuário, e o solicitante de serviço, que consome as informações ou os serviços oferecidos pelo provedor de serviço. A arquitetura também contém os seguintes componentes [4]:

- O tradutor, que traduz entre linguagens externas usadas pelos participantes e linguagens internas usadas pelo gerador de processos;
- O gerador de processos, que tenta gerar um plano que compõe a disponibilidade de serviços no repositório de serviço que preencha a requisição;

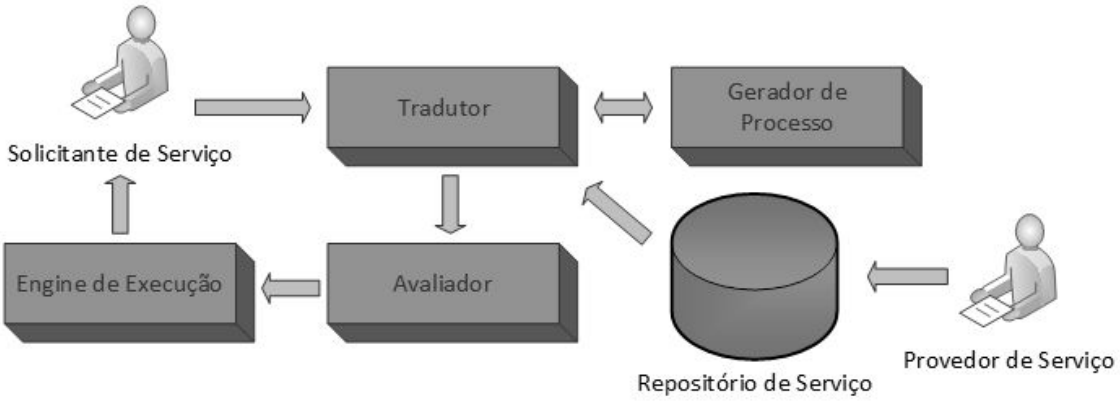


Figura 2.8: Arquitetura baseada em *Web Service*, adaptado de [4].

- O avaliador, que avalia todo tipo de plano e propõe o melhor para a execução, a *engine* de execução, que executa o plano e retorna o resultado para o provedor de serviço;
- O repositório de serviço, que armazena todos os tipos de serviços provenientes do provedor de serviço.

Wrapper-Mediator

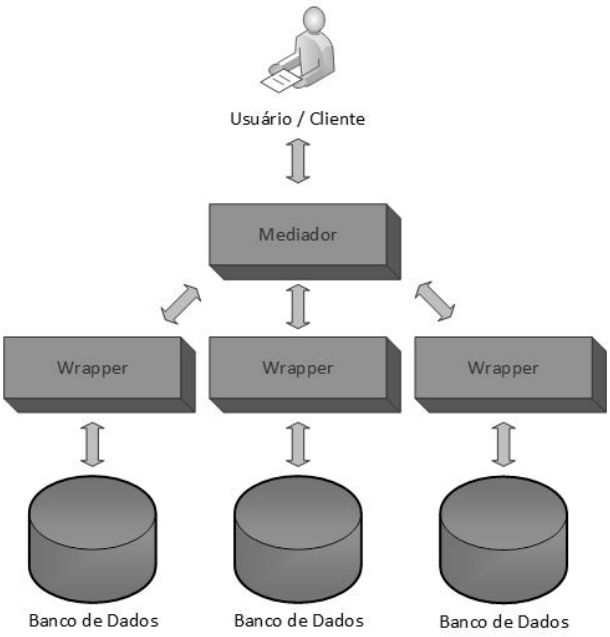


Figura 2.9: Arquitetura baseada em *Wrapper-Mediator*.

O *Wrapper-Mediator* é o método utilizado para melhorar a eficácia da integração em que uma fonte de dados é associada com um *wrapper* (invólucro), e o mediador interage com as fontes de dados por meio dos *wrappers* [37].

Em comparação com a técnica de DW, ela não materializa os dados dentro de um local persistente, pois ao invés disso, os dados são diretamente obtidos de uma fonte individual de dados durante o tempo de execução, isto é, ela fornece informações mais atualizadas e é mais capaz de integrar fontes de informações em ambientes altamente dinâmicos como a Web.

O *Wrapper-Mediator* tipicamente consiste de duas partes, conforme pode ser visto na Figura 2.9:

- um *wrapper* que fornece acesso a dados para a fonte de dados individual;
- um mediador que fornece tradução de dados entre diferentes formatos.

Na maioria dos casos, o mediador está ligado a um grande número de *wrappers* para permitir o acesso centralizado e unificada para os *wrapper* de ligação [38]. No entanto, em alguns casos, um *wrapper* também atua como um mediador e oferece a mediação entre formatos de dados heterogêneos [39].

Ontologia

A definição mais comumente citada de uma ontologia é a de Gruber em [40] que diz ser "uma especificação formal, explícita de uma conceituação compartilhada", ou seja, é a essência do significado de algum termo.

De Freitas, em seu trabalho [41], cita que existem diferentes tipos de ontologias. Elas são classificadas, de acordo com seu grau de genericidade:

- Ontologias de Representação: são aquelas que definem as primitivas de representação de forma declarativa, como *frames*, axiomas, atributos e outros.
- Ontologias Gerais (ou de topo): são definidas como abstrações necessárias para o entendimento de características gerais, como por exemplo, tempo, processos, papéis, seres, coisas, espaço, dentre outros.
- Ontologias Centrais (ou genéricas de domínio): definem as subáreas de estudo de uma área e/ou conceitos mais genéricos e abstratos desta área.
- Ontologias de domínio: São conhecidas por tratarem de um domínio mais específico de uma área genérica de conhecimento, como patologia clínica, ensino-aprendizagem, bancos de dados, dentre outros.

- Ontologias de aplicação: São responsáveis por procurar solucionar um problema específico de um domínio, como identificar estudantes com problemas de aprendizagem, a partir de sua frequência.

Como percebido, os tipos de ontologias estão descritos de acordo com o seu grau de genericidade. O tipo de ontologia necessária para resolver algum problema, vai depender do nível em que se deseja alcançar por meio da aplicação desenvolvida.

A ontologia é utilizada em várias áreas do conhecimento, mas especificamente em banco de dados, significa ser um método que provê a conversão do esquema de um banco de dados para uma ontologia de aplicação, e após essa ontologia será convertida para uma ontologia de domínio baseada na classe, propriedade e tipo de propriedade, juntamente com a semântica [42], conforme pode ser visto da Figura 2.10.

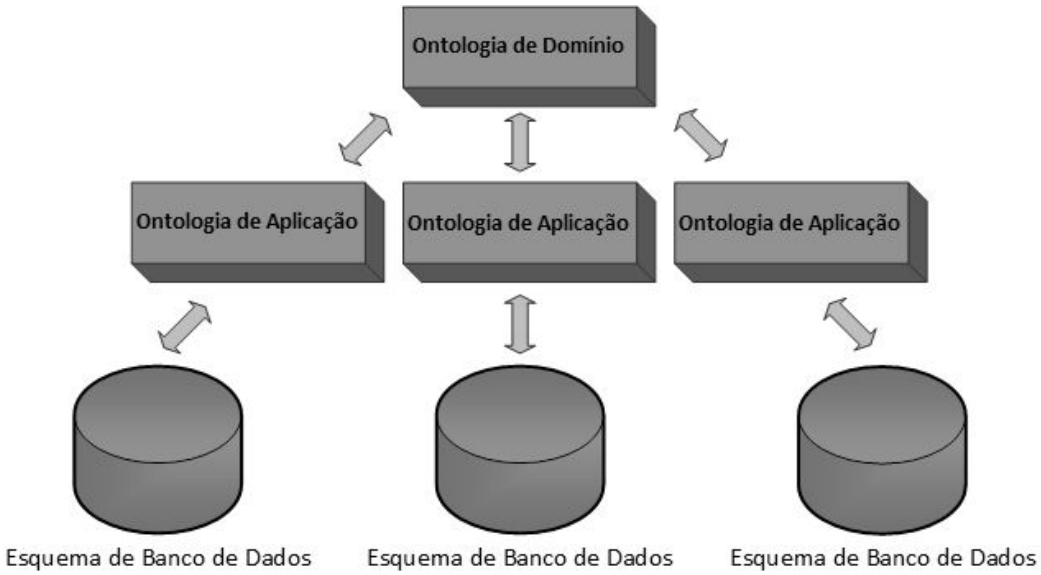


Figura 2.10: Arquitetura baseada em ontologia para bancos de dados.

Com outro ponto de vista, Mizoguchi em seu trabalho [43], define as ontologias em duas categorias:

- Ontologia Leve: Esta categoria de ontologia é comumente utilizada para motores de busca na web, como a ontologia do Yahoo!, que consiste em uma hierarquia de tópicos com pouca rigorosidade na conceitualização de um termo, assim como no princípio da organização dos conceitos, distinção entre palavras e conceitos, dentre outros. O principal objetivo desta categoria é o poder no motor de busca e, portanto, ele é muito dependente da utilização.
- Ontologia Pesada: Ela é reconhecida pela atenção rigorosa ao desenvolvimento do significado de cada conceito, organizando princípios desenvolvidos na filosofia, rigo-

rosa relação semântica entre os conceitos, etc. Essa categoria de ontologia geralmente é construída para modelar um mundo-alvo, que requer uma conceitualização cuidadosa para garantir a consistência e a fidelidade.

A popularidade da ontologia, basicamente surgiu do campo da inteligência artificial, cuja utilidade se deu por meio da construção de um vocábulo formal, de maneira explícita para compartilhar dados entre aplicativos. Mesmo estando claro que o objetivo da ontologia não seja integrar a heterogeneidade [44], ela tem a capacidade de encontrar o grau de semelhança entre duas ou mais palavras diferentes, consequentemente, ela é muito utilizada em integração.

2.3 Sincronização de Dados

Quando após a ocorrência de alteração de algum registro armazenado em alguma coleção de dados é propagada a outra coleção - cliente/servidor - as transformando em equivalentes, ocorreu uma sincronização de dados. Essa sincronização é resultado da integração de esquemas iguais entre dois bancos de dados [45]. Para que essa sincronização de dados ocorra é necessário que haja uma replicação de dados.

2.3.1 Replicação de Dados

Quando as transações executadas, assim como as operações, são totalmente reproduzidas na mesma ordem a qual foram solicitadas de um banco de dados para outro banco de dados, sendo possível até mesmo para vários bancos de dados, ocorre um processo de replicação. Caso isso não ocorra, pode acontecer inconsistências nos dados entre os bancos de dados.

Existem dois tipos diferentes de replicação, a parcial, a qual apenas parte dos dados de um banco de dados são replicados para outros bancos, e a total, em que todos os dados são replicados de um banco para outro [1].

Se a disponibilidade de ao menos uma das cópias estiver acessível, na replicação total, a confiabilidade dos dados aumenta, visto que o usuário não depende somente dos dados disponibilizados em uma única base de dados. Caso ocorra algum problema no sistema, o usuário poderá solicitar uma réplica desses dados [46].

2.3.2 Tipos de Sincronização

Os tipos de sincronização de dados são classificados de duas maneiras diferentes, segundo Palazzo em [47]. Em *one-way* e *two-way*, conforme é apresentado a seguir:

- *One-Way*: Quando o processo de transferência das atualizações de dados é realizado em sentido único, ocorre a *one-way*. Esse tipo de sincronização pode ocorrer unilateralmente tanto de um banco de dados cliente para o banco de dados servidor, como o inverso. O processo ocorre da seguinte maneira: é estabelecida uma conexão entre os bancos de dados e, de forma sincronizada, submete-se todas as alterações efetuadas de um banco de dados de origem para o banco de dados de destino. Não deixando, ao final do processo, ambos bancos de dados idênticos;
- *Two-Way*: Quando o processo de transferência das atualizações de dados é realizado nos dois sentidos, ocorre a *two-way*. A iniciativa neste caso é por parte daquele banco de dados em que a alteração ocorreu primeiro, submetendo os dados para o outro banco de dados, que por sua vez identifica e trata os conflitos, retornando ao cliente os dados corretos. No final deste processo, ambos bancos de dados possuem os mesmos dados.

2.4 Trabalhos Relacionados

A integração de dados é um método essencial para a fusão de dados que se encontram em diferentes fontes. Os dados são armazenados em vários bancos de dados e arquivos que precisam ser combinados por diversas razões, como é o caso entre a SEEDF e o Transporte Urbano do Distrito Federal (DFTrans), que compartilham dados entre si para a liberação do benefício do Passe Livre Estudantil (PLE) aos estudantes da rede pública de ensino. Além desse tipo de compartilhamento, existem outros, como por exemplo, a montagem de um DW, a junção de duas ou mais empresas, a interoperabilidade de sistemas, etc.

A integração entre bancos de dados não é um processo simples, tendo em vista os diferentes tipos de heterogeneidades nas estruturas, no processamento de consultas, na semântica e nos modelos de dados dos mais diversos SGBD. Esse tipo de integração geralmente envolve a combinação de dados que residem em diferentes fontes, fornecendo uma visualização unificada destes dados [3][48].

Devido a heterogeneidade maciça que existe entre as diversas fontes de dados, fazer a correspondência entre os diversos esquemas é um processo complexo, demorado e propenso a erros [30], uma vez que as fontes de dados normalmente são projetadas de forma independente sem levar em consideração padrões ou metologias existentes.

A integração entre bancos de dados heterogêneos é uma área bastante explorada por pesquisadores, pois existem diversos métodos, técnicas e arquiteturas, conforme foram apresentados no capítulo anterior. Embora esta área já seja explorada, a estas arquiteturas, métodos e técnicas são semi-automáticas, isto é, precisam da intervenção de usuários com conhecimento semântico de todos os esquemas e domínio a ser integrado.

2.4.1 Arquiteturas e Métodos de Integração de Dados

Diversas arquiteturas e métodos já foram propostos, como é o caso da integração de dados baseada em LINQ [3], *Web Services* [49], *Wrapper-Mediator* [37] e Ontologia [42].

A arquitetura baseada em LINQ [3] apresenta um projeto de sistema de integração de banco de dados heterogêneos multi-fonte baseado na tecnologia LINQ que inclui uma camada de lógica de negócio que decompõe a consulta lógica e integra os resultados das consultas, uma camada de acesso ao dados que se conecta a todos os tipos de fontes de dados pegando aqueles necessários para o formato unificado, e uma camada de dados heterogêneos que é aquela que se encontram todas essas bases de dados.

O método baseado em *Web Service* [49] emprega o sistema de correspondência de modelos que usa computação granular para otimizar a composição dos serviços da Web para melhorar a eficiência e a qualidade dos sistemas de integração heterogêneos.

O método baseado em *Wrapper-Mediator* descreve um trabalho para melhorar a eficácia da integração em que uma fonte de dados está associada a um invólucro (*wrapper*) e um mediador (*mediator*) que interage com as fontes de dados.

Já a abordagem baseada em ontologia [42] fornece o método para a conversão do esquema de banco de dados para uma ontologia local e posteriormente a conversão dela para uma ontologia de domínio global com base nas classes, propriedades e tipos de propriedades, juntamente com a semântica.

2.4.2 Técnicas de Integração entre Bancos de Dados Heterogêneos

Diversos pesquisadores tentaram integrar bancos de dados por meio de técnicas com base em *Schema Matching*, como pode ser visto em *Relational Schemas Matching* [48], CUPID [50], *Similarity Yield Matcher (SYM)* [51], HDSM [52], Sistema de Correspondência Complexa [53], abordagem baseada em XML [54], iMAP [55], *Similarity Flooding (SF)* [56] e COMA [57].

O *Relational Schemas Matching* [48] fornece uma técnica para integrar esquemas relacionais ao empregar o combinador de esquemas relacionais, o combinador de nomes de atributos, o combinador de tipos de dados, o combinador de restrições e o combinador de instâncias de dados.

O CUPID [50] tem apresentado uma abordagem que ajuda a transformar o esquema original em uma arvore e depois é computada uma estrutura de similaridade baseada em um combinador de pares de folhas em um esquema.

O *Schema Matching* Híbrido [51] introduziu o SYM, que consiste em um novo combinador linguístico projetado, o combinador de estrutura e o combinador de estrutura refinado baseado na semelhança ponderada que trabalha bem com o tipo de correspondência 1:1.

A Combinação de Esquema de Mineração [52] que propõe a técnica HDSM que pode encontrar mapeamentos entre atributos que são muito difíceis de interpretar. Essa técnica é utilizada por meio da mineração de dados.

O Sistema de Correspondência Complexa [53] propôs uma técnica que emprega um processador de *cluster* para reduzir o espaço de correspondência dos candidatos que melhoram a eficiência da correspondência, além de usar um conjunto de pesquisadores com o propósito especial de explorar uma parte especializada do espaço da pesquisa para descobrir as correspondências 1:1 e combinações complexas.

A abordagem baseada em XML [54] descreve o mapeamento de esquemas de bancos de dados heterogêneos para um documento XML e o procedimento para integração entre estes bancos por meio da resolução de conflitos.

o iMAP [55] introduz um sistema que descobre semi-automaticamente as combinações complexas e correspondências 1:1. Ele emprega um conjunto de pesquisadores que reformulam combinações de esquemas como uma busca em um grande ou infinito tipo de correspondência de espaço.

A *SF* [56] provê uma técnica na qual os esquemas são convertidos primeiramente em grafos nomeadamente direcionados. Esta técnica usa o cálculo do ponto fixo para determinar a semelhança entre os nós. O mapeamento de nível do elemento é alimentado pelo combinador *SF*. Os filtros são aplicados para selecionar subconjuntos relevantes de correspondências.

O COMA [57] descreve um sistema que escolhe múltiplas combinações de uma biblioteca de combinações para determinar a combinação de resultados contidas em valores de similaridade entre cada elemento de dois esquemas.

2.5 Síntese do Capítulo

Hoje em dia, no entanto, há a necessidade de uma automática integração entre bancos de dados heterogêneos e que nesses métodos, técnicas e arquiteturas apresentados são considerados manuais ou semi-automáticos, não sendo soluções ideais para a implantação da SEEDF, em razão de seu domínio específico. Ainda há pesquisas em andamento para a construção de métodos, técnicas e arquiteturas que disporão de uma abordagem mais efetiva de integração entre bancos de dados heterogêneos. Esta dissertação traz como desafio a criação de uma arquitetura mais eficiente de integração entre bancos de dados heterogêneos, cuja a interferência de um analista da área de TI seja considerada

mínima para achar as correspondências entre os esquemas, isto é, quem determinará essa integração será um analista da área de negócio, sendo ela realizada automaticamente por meio de uma ontologia. A próxima seção apresenta esta arquitetura proposta.

Capítulo 3

Arquitetura Proposta

Neste capítulo, é apresentado a Arquitetura de Integração entre Bancos de Dados, mostrando o ecossistema computacional da SEEDF antes e depois da integração. Este capítulo é dividido nas seguintes seções: 3.1 Ecossistema Computacional Antes da Integração, onde é exposto o cenário de sistemas, SGBD e VM da SEEDF, assim como a retomada do problema de pesquisa; 3.2 Aspectos Necessários para a Arquitetura, onde informa os requisitos imprescindíveis para o funcionamento da arquitetura; 3.3 Profissionais para a Integração, que aborda os perfis dos profissionais necessários para a integração; 3.4 A Arquitetura de Integração Baseada em Ontologia, que apresenta a arquitetura lógica com as suas três camadas de integração (camada de dados local, camada de integração e camada de dados global) e a arquitetura física, explicando o desenvolvimento do *middleware* Noach e o fluxo dos dados durante a sincronização; 3.5 Ecossistema Computacional para Validação da Integração, cujo objetivo é demonstrar o cenário completo após a integração ser realizada; e 3.6 A Síntese do Capítulo.

3.1 Ecossistema Computacional Antes da Integração

Para um melhor entendimento a respeito da arquitetura de integração que é apresentada neste trabalho, é importante conhecer o ecossistema computacional, ou seja, quais as estruturas fundamentais dos sistemas que estão em produção.

A Diretoria de Desenvolvimento de Sistemas (DISIS), local onde são desenvolvidos e mantidos grande parte dos sistemas da SEEDF, conta com aproximadamente 18 aplicações, sendo que nove delas são desenvolvidas em PHP, quatro em ASP, três em Java, uma em DELPHI e uma em Excel. Essas aplicações estão divididas em 15 *Virtual Machines* (VM) diferentes e utilizavam cerca de 12 bancos de dados diferentes, sendo eles, seis em MySQL, três em SQL Server, um em PostgreSQL, um em Oracle e um em FireBird, conforme pode ser visto na Figura 3.1.

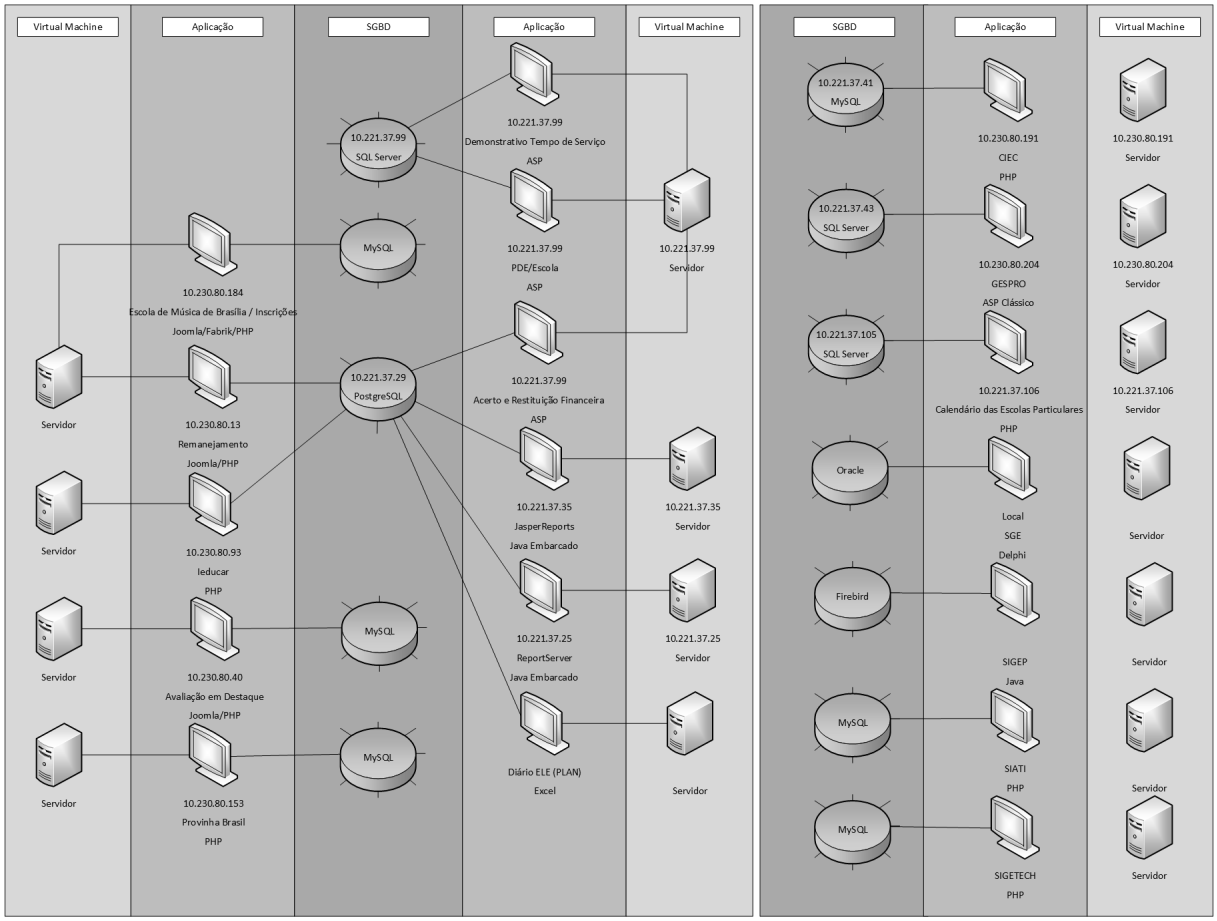


Figura 3.1: Ecossistema Computacional Antes da Integração.

A SEEDF, por meio da DISIS, apresentava um cenário com aplicações bastante diversificadas. Cada uma foi desenvolvida com um tipo de arquitetura interna de sistema de acordo com os conhecimentos de cada programador, além delas serem feitas com tecnologias e SGBD diferentes. Essa variedade na forma de desenvolvimento, na escolha do SGBD e da linguagem de programação utilizada, dificultava na integração entre os bancos de dados dos sistemas, havendo o mínimo de integração possível somente dentro daqueles sistemas que utilizam o mesmo tipo de SGBD na mesma VM.

3.1.1 Retomada do Problema

Dentro do GDF, a SEEDF é a maior das secretarias, apresentando cerca de 460 mil estudantes ativos. Estes estudantes tem aulas, nos três turnos, em cerca de 700 escolas que estão localizadas em 14 regionais de ensino. Para o funcionamento desta secretaria, o corpo de servidores efetivos e temporários contam com mais de 47 mil pessoas.

Para gerir todo esse órgão é necessário a utilização e desenvolvimento de diversos sistemas que são especializados em diferentes áreas da educação e da gestão pública,

como é o caso da escrituração escolar, da avaliação, da gestão de pessoas e da gestão de recursos financeiros, por exemplo. Estes sistemas contam com uma enorme quantidade de transações de dados, gerando muitas atualizações, inserções e exclusões em tipos de dados variados em uma velocidade grande.

Para que haja um maior controle sobre as informações geradas, faz-se necessário que todos esses bancos de dados estejam integrados para que não ocorra uma redundância descontrolada de dados, o que pode acarretar diversos problemas, como por exemplo, a má tomada de decisão ou a publicidade de informações erradas para a população.

3.2 Aspectos Necessários para a Arquitetura

A criação de uma proposta de arquitetura entre bancos de dados não é uma tarefa fácil. Por mais que existam diversas técnicas, métodos e arquiteturas, nem sempre elas fazem exatamente o que se quer de forma simples e automática. Visando isso, para a criação da arquitetura proposta por esta dissertação, foram definidos alguns aspectos necessários que ela deva conter:

- Heterogeneidade de SGBD: Precisa-se que o ecossistema computacional a que se queira integrar, contenha diversos tipos de SGBD;
- Automatização na integração: É de grande relevância que os envolvidos pelos sistemas tenham a menor interação possível com o processo de integração, isto é, que a integração funcione de maneira automática;
- Suporte a grande quantidade de dados: Faz-se necessário que seja escolhido um banco de dados global que seja possível armazenar uma grande quantidade de dados e suporte um alto número de transações;
- Profissional com conhecimento na construção de ontologia: É necessário o conhecimento da construção de ontologia, pois é por meio delas que a arquitetura fará a integração de forma automática. Quanto mais completa for a ontologia utilizada, maior será a abrangência da integração;
- Profissional com conhecimento técnico em TI: Este profissional será o responsável pela implantação do *middleware*, que utiliza a Arquitetura de Integração proposta, no ambiente computacional a que se deseja integrar.

3.3 Profissionais para a Integração

Para o entendimento do funcionamento da Arquitetura de Integração entre Bancos de Dados proposta, é fundamental conhecer os papeis dos profissionais necessários para o funcionamento da integração. Para isso, é apresentado os dois perfis a seguir:

- Analista da área de TI: Profissional formado em computação ou área correlacionada de TI que tenha conhecimento em desenvolvimento de sistemas, para atuar na configuração, implantação e manutenção da arquitetura proposta;
- Analista da área de Negócio: Profissional com conhecimento técnico e de gestão relacionado a área de negócio na qual a arquitetura de integração proposta pretende atuar. Este profissional tem conhecimento amplo sobre a área de domínio do negócio. Para este trabalho se faz necessário que este profissional tenha conhecimento na criação de ontologias. Quanto melhor for o conhecimento em relação a área de negócio e a criação de ontologias, maior será a abrangência da integração.

3.4 A Arquitetura de Integração Baseada em Ontologia

Para solucionar o problema mencionado na Subseção 3.1.1, esta dissertação apresenta uma arquitetura de integração entre bancos de dados, que tem por finalidade a automatização do processo de integração, proporcionando aos analistas da área de negócio a autonomia da integração das bases de dados apenas utilizando o conhecimento do domínio e de ontologias. Para um melhor entendimento, a explicação da arquitetura foi dividida em Lógica e Física, que são apresentadas nas próximas duas subseções.

3.4.1 Arquitetura Lógica

A Arquitetura Lógica proposta apresenta três camadas: Camada de Dados Local, Camada de Integração e Camada de Dados Global, conforme pode ser visualizado na Figura 3.2.

Camada de Dados Local

A Camada de Dados Local é responsável pelo armazenamento de dados dos sistemas. Esta camada é composta por bancos de dados locais que se encontram integrados dentro de SGBD por suas respectivas tecnologias. Além destes SGBD, esta camada contém os conectores para o acesso aos mesmos, como por exemplo, conectores para MySQL, PostgreSQL e SQL Server.

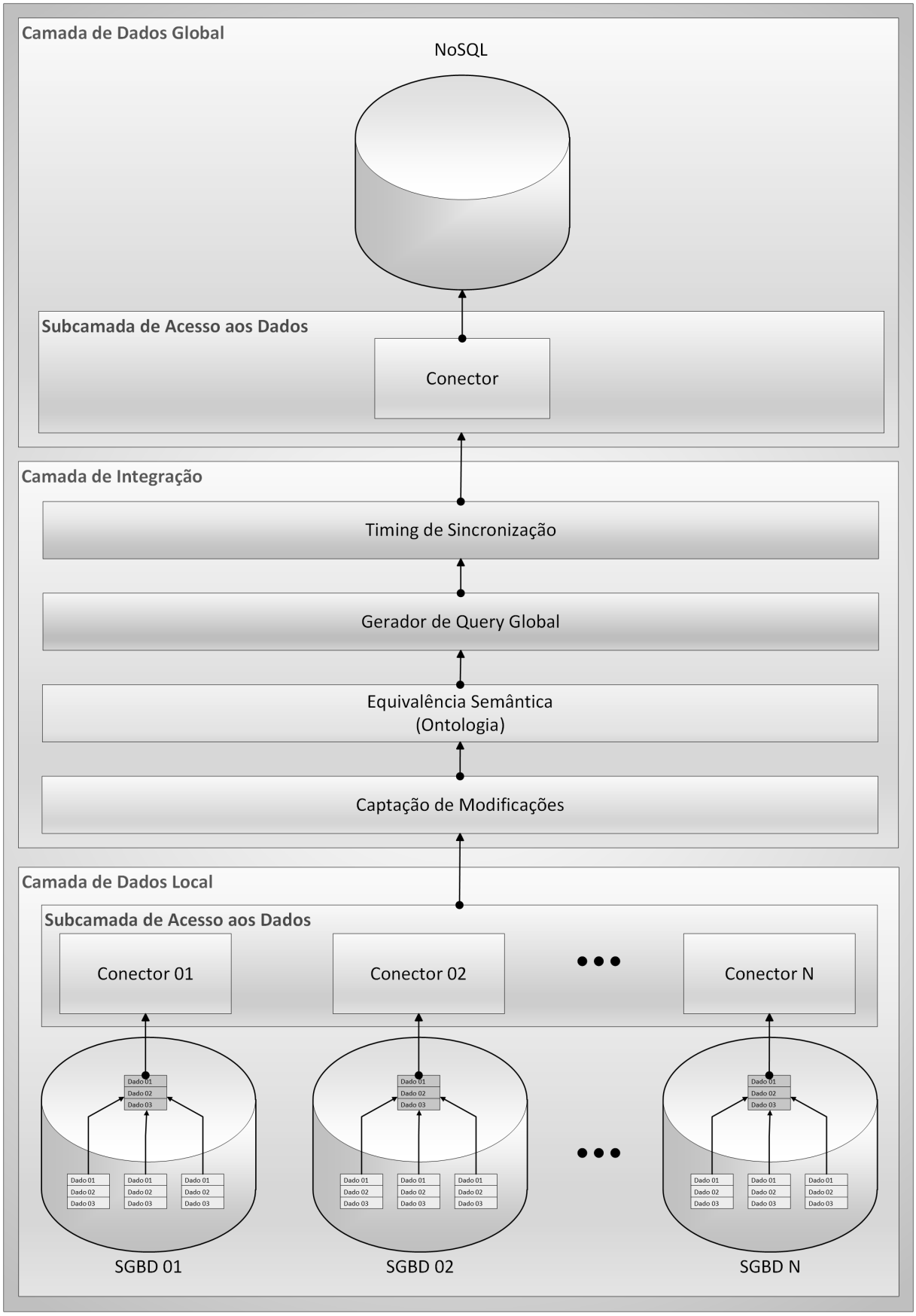


Figura 3.2: Arquitetura Lógica de Integração entre Bancos de Dados.

Camada de Integração

A Camada de Integração é composta por quatro subcamadas. A subcamada de Captação de Modificações, a subcamada de Equivalência Semântica (Ontologia), a subcamada Geração de *Query* Global e, por último, a subcamada *Timing* de Sincronização.

A Captação de Modificações é responsável por captar as modificações das tabelas de cada SGBD que se deseja integrar com a base de dados global. Esta captação é realizada por meio de uma tabela denominada tabela de sincronização, ou também conhecida como tabela de *log* ou de auditoria. Esta tabela armazena por meio de *Stored Procedures* e *Triggers*, que deverão estar presentes em todos os SGBD relacionais que queiram se integrar com a arquitetura, todas as modificações realizadas dentro das tabelas que são alvos da integração.

A subcamada de Equivalência Semântica, ou também conhecida como subcamada de Ontologia, pode ser considerada como o núcleo deste trabalho, ou seja, esta é a subcamada responsável por acessar as equivalências semânticas de classes e propriedades de objetos da Ontologia de Domínio aplicada ao processo de integração, e além desse acesso, esta subcamada é a responsável pela diminuição da dependência de um analista de TI, ou seja, será necessário um analista da área de negócio, ou equipe da área de negócio, que tenha conhecimento em Ontologia para realizar tal integração.

A equivalência semântica será vital para o relacionamento semântico automático entre as tabelas e colunas que tenham nomenclaturas diferentes nos diversos tipos de bancos de dados dentro do domínio de integração. Como exemplo deste tipo de equivalência, observando a Figura 3.3, dentro de um SGBD Local Integrado que tenha uma tabela com o nome de 'escola' e coluna 'nome' e outra com o nome de 'colegio' e coluna 'nm_colegio' dentro de outro tipo de SGBD Local Integrado poderão ser integradas com a tabela 'unidade_escolar' e coluna 'nm_unidade_escolar' que se encontra dentro do banco de dados global. Para que esse tipo de integração funcione de maneira automática, a equivalência deve estar presente dentro de um glossário da ontologia de domínio por meio de classes, que representam as tabelas, e propriedades de objetos, as colunas, para que a integração entre as tabelas e colunas dos bancos de dados sejam realizadas com sucesso.

A subcamada de Gerador de *Query* Global é a responsável por montar as *queries* que irão transmitir os dados da Camada de Dados Local para a Camada de Dados Global. Essas *queries* recebem como parâmetros os dados que serão transmitidos, o tipo de ação (*update*, *insert* ou *delete*), o nome da tabela local, o nome da coluna desta tabela e o ID daquele registro. Todos estes dados são adquiridos por meio da tabela de sincronização de cada SGBD Local Integrado. Com essas informações, esta subcamada acessa a subcamada de equivalência semântica, converte os termos (nomes das tabelas e colunas) para os que

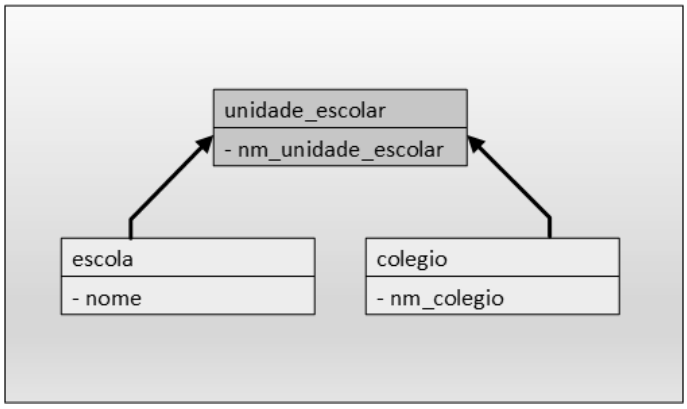


Figura 3.3: Equivalência Semântica entre Tabelas/Colunas com nomes diferentes.

serão utilizados dentro da Camada de Dados Global e gera as *queries* que serão persistidas nesta última camada.

A subcamada de *Timing* de Sincronização é a responsável por ativar toda a integração de acordo com o que foi configurado pelo analista da área de negócio, ou seja, esta é a subcamada que configura o intervalo de tempo que se deseja que esta integração/sincronização se realize.

Camada de Dados Global

Por último, tem-se a Camada de Dados Global que conta com uma subcamada de acesso aos dados e um banco de dados *NoSQL*. A escolha deste tipo de banco de dados para compor esta camada se deu não apenas ao fato de sua característica de escalonamento horizontal, que suporta um grande volume de dados, mas também pela grande quantidade de transações que serão realizadas (*update*, *insert* e *delete*), além da variedade de dados que ele suporta.

3.4.2 Arquitetura Física

Para a validação da Arquitetura Lógica, com todas as suas camadas, foi desenvolvido um *middleware* e testado em um ambiente de dados controlado da SEEDF. Para um melhor entendimento da criação e funcionamento deste *middleware*, esta subseção é divida em *Middleware* Noach, onde será exposta a motivação do nome da aplicação, a Arquitetura Física de Integração, onde será apresentado os módulos da aplicação com suas respectivas funcionalidades, a Implementação e Codificação, onde é exibido detalhes do desenvolvimento do Noach, e por último a Criação de um SGBD Local Integrado, onde será apresentado o processo de criação e configuração de um SGBD a Arquitetura.

Middleware Noach

Nesta dissertação foi criado um *middleware* para integração de bancos de dados, denominado Noach, que do Hebraico, é o nome do herói bíblico que recebeu ordens de Deus para a construção de uma arca, para salvar a Criação do Dilúvio [58]. Inspirado nesta história, deu-se o nome deste *middleware*, tendo em vista que o seu papel é o de integrar os dados de bases locais (Criação) em uma base de dados global (Arca), consolidando as informações, evitando a redundância descontrolada de dados que os sistemas geram (Dilúvio) e sendo realizada de maneira automática sem o intermédio de um analista de TI (Poder Divino).

Arquitetura Física do Noach

O Noach é composto por quatro módulos: O módulo de Validação de Acesso, de Configuração, de Verificação de Conexão com os Bancos de Dados e o de Sincronização, conforme pode ser visualizado na Figura 3.4.

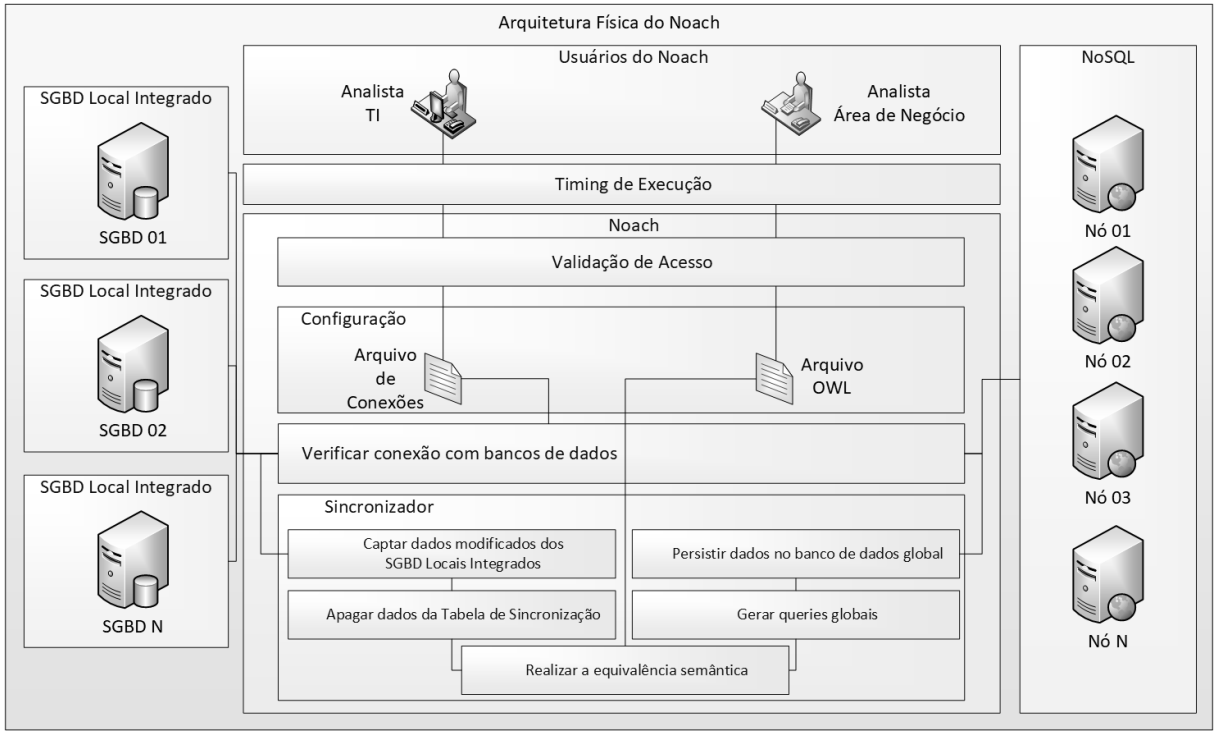


Figura 3.4: Arquitetura Física do Noach.

O Módulo de Validação de Acesso é o responsável por validar o acesso ao *middleware* apenas para usuários previamente cadastrados, ou seja, de realizar o *login* do usuários que irão executar a integração.

O Módulo de Configuração tem como incumbência ter os dados de acesso necessários aos SGBD e ao NoSQL. Para o funcionamento deste módulo, é de grande valia que o

Analista de TI crie um arquivo de conexões e o insira no Noach. O arquivo de conexões, é onde será definido os parâmetros básicos para conectar aos SGBD Integrados Locais e ao NoSQL Global. Além deste arquivo, outro importante a ser inserido é o OWL, que é o responsável por conter o código da ontologia de aplicação proposta para a integração automática entre as bases de dados. Um exemplo deste código será dado e explicado na subseção 4.2.

O Módulo de Verificação de Conexão com os Bancos de Dados tem como funcionalidade acessar os bancos de dados a serem integrados e verificar se eles estão disponíveis, ou seja, *on-line* para que seja possível a integração.

O Módulo de Sincronizador, é o encarregado de executar a integração entre os bancos de dados. Este módulo, conta com cinco funcionalidades. A primeira é a de captar dados modificados dos SGBD Locais Integrados, isto é, ele acessa a tabela de sincronização e realiza um *select* nela, trazendo os dados necessários para a integração. A segunda é responsável por apagar os dados da tabela de sincronização, permitindo que em uma próxima execução, tenha-se apenas os dados que ainda não foram integrados. A terceira é responsável por acessar o arquivo OLW e realizar a inferência para encontrar a equivalência semântica entre os nomes das tabelas e colunas. A quarta recebe todos os dados necessários para a geração das *queries* para o NoSQL. A última camada é aquela responsável por persistir as *queries* geradas pela última funcionalidade dentro do NoSQL.

Além dos módulos do Noach, tem-se uma camada de *Timing* de Execução, onde será definido pelos usuários do sistema se a integração será executada periodicamente ou se a ela será realizada manualmente, por meio da utilização da própria aplicação.

Implementação e Codificação

O Noach foi desenvolvido utilizando a linguagem de programação *PHP* [59], no paradigma Orientado a Objetos, e sua extensão *PHP Data Objects* (PDO) [60], que fornece uma interface padronizada para trabalhar com bancos de dados, cuja finalidade é abstrair a conexão e interações com os mais diversos tipos de SGBD. Para acessar a ontologia e fazer o mapeamento proposto, foi utilizada a *RDF API for PHP V0.9.6* (RAP) [61], que é uma API para PHP criada para analisar, consultar, manipular, serializar e servir modelos RDF. Por meio da RAP consegue-se acessar a ontologia que está em um arquivo OWL, gerada neste caso pelo Protégé [62], que é um *framework* e editor de ontologia *open-source*.

O Noach, em seu desenvolvimento conta com duas classes, a classe *ConnectDB* e a classe *MapOntology*. A primeira classe foi criada para realizar a conexão com todas as bases de dados locais e a base de dados global. Além dessas conexões, ela tem como

responsabilidade a preparação de cada *query* que será executada nos bancos, tão como a limpeza das tabelas de sincronização de cada base local.

A *ConnectDB* conta com quatro métodos. O primeiro é o *connect_test*, que é um método privativo a classe, que recebe como entrada a conexão com o banco de dados a ser testada e é responsável por analisar se ela foi realizada com sucesso. O segundo é o *connect_db*, que é um método público que realiza a conexão com o banco de dados, de acordo com as configurações escritas no arquivo de conexão a ser inserido no Noach. O terceiro é o *q_prepare*, um método público que recebe como parâmetros de entrada uma *query* e a conexão com o banco em que ela será executada. O quarto e último método desta classe é também público e recebe como parâmetro de entrada o nome do SGBD e a conexão para limpar a tabela de sincronização.

A segunda classe é a *MapOntology*. Esta é a responsável por realizar a captura de todos os dados das bases locais a serem sincronizados com a base global, acessar a ontologia e realizar a inferência para descobrir as equivalências semânticas das tabelas e colunas e, por último, gerar as *queries* globais para serem persistidas na base de dados global. Para a realização física da subcamada de *Timing* de Sincronização, foi colocado no próprio código, um contador de tempo, que pode ser configurado de acordo com a necessidade de integração/sincronização que negócio exige.

Esta classe conta com seis métodos, o primeiro, denominado *hstoreToJsonArray*, recebe como parâmetro de entrada um dado HSTORE, isto é, chave-valor, e o converte para JSON. O segundo método, *getDatas*, recebe como entrada o nome do SGBD e captura todos os dados da tabela de sincronização e a trunca ao final, isto é, excluindo todos os dados. Os dois próximos métodos, *mapNameColumnsTableOntologyRDF* e *mapNameTableOntologyRDF*, recebem respectivamente, o nome da coluna e da tabela do banco de dados local e os convertem para o banco de dados global. Essa conversão é realizada acessando a ontologia que foi configurada no Noach. O penúltimo método, chamado *makeGlobalQuery* é o responsável por recepcionar o nome da tabela da banco de dados global, a ação que deverá ser realizada (*update, insert ou delete*), o ID do dado a ser alterado e os dados que foram alterados por meio de um *array*, e gerar uma *query* que será persistida dentro do banco de dados global. Por último, o método *syncMapIteraction* é o responsável por ativar toda a integração de dados. Estes dois métodos estão escritos em um arquivo chamado *sync.class.php*.

Criação de um SGBD Local Integrado

Para a criação de SGBD Local Integrado o primeiro passo é realizar o levantamento do ambiente computacional em que todos os sistemas de informação estão inseridos, isto é, modelar o cenário verificando quantos/quais são os sistemas e em qual estrutura fí-

sica ou virtualizada eles se encontram, assim como os seus SGBD. Como apresentado anteriormente (Seção 3.1), o ambiente da DISIS é composto por doze bancos de dados implementados em cinco SGBD diferentes.

Esse levantamento se faz necessário para reunir todos os bancos de dados de cada SGBD dentro de um único ambiente. Para realizar essa ação será necessário em algum momento, deixar o sistema off-line, para que não haja perda de dados, fazer um *backup* da base de dados e realizar um *restore* dentro desse novo ambiente do SGBD. Quando isso é realizado, a utilização de comandos de *stored procedures* e *triggers* entre as tabelas das bases já podem ser realizados, facilitando a integração entre estes bancos.

Dentro de cada SGBD Local Integrado, é necessário realizar a construção da tabela de sincronização de dados, isto é, uma tabela que irá captar por meio das *stored procedures* e *triggers*, os principais dados que são necessários para a sincronização. Esses dados foram apresentados na Subcamada de Gerador de Query da Arquitetura na seção 3.4.1, sendo eles: nome da tabela, nome da coluna, ID do dado, o dado alvo e a ação executada (*update*, *insert* ou *delete*).

Após a criação dos SGBD locais integrados, precisa-se inserir os dados de acesso a cada um deles no arquivo de conexão que será subido no Noach para que seja possível acessá-los e realizar a integração.

3.5 Ecossistema Computacional para Validação da Integração

O ambiente computacional da SEEDF, após a implementação da arquitetura proposta está demonstrado na Figura 3.5. Com esse novo ambiente computacional pretende-se adquirir uma maior integração de dados entre as aplicações, garantindo assim a consistência entre as informações geradas, e a realização automática da integração sem a necessidade de intervenção de um analista da área de TI sempre que houver a inserção/remoção de um banco de dados.

Neste cenário, as aplicações com o mesmo tipo de SGBD, ou seja, de mesma tecnologia, se encontram conectadas em apenas um SGBD, facilitando assim, a comunicação interna entre essas bases de dados. Como pode ser observado na Figura 3.5, os seis bancos de dados PostgreSQL estão integrados em apenas uma VM com o PostgreSQL, os seis MySQL estão em apenas um SGBD MySQL e assim por diante. Para cada SGBD Local Integrado de acordo com a sua tecnologia, deve-se comunicar de forma consistente com uma base de dados global, que é a responsável por armazenar os dados consolidados de todos os SGBD e, por meio de um *middleware*, sincroniza seus dados constantemente com os principais dados de todos os outros SGBD. A lacuna de tempo para que ocorra

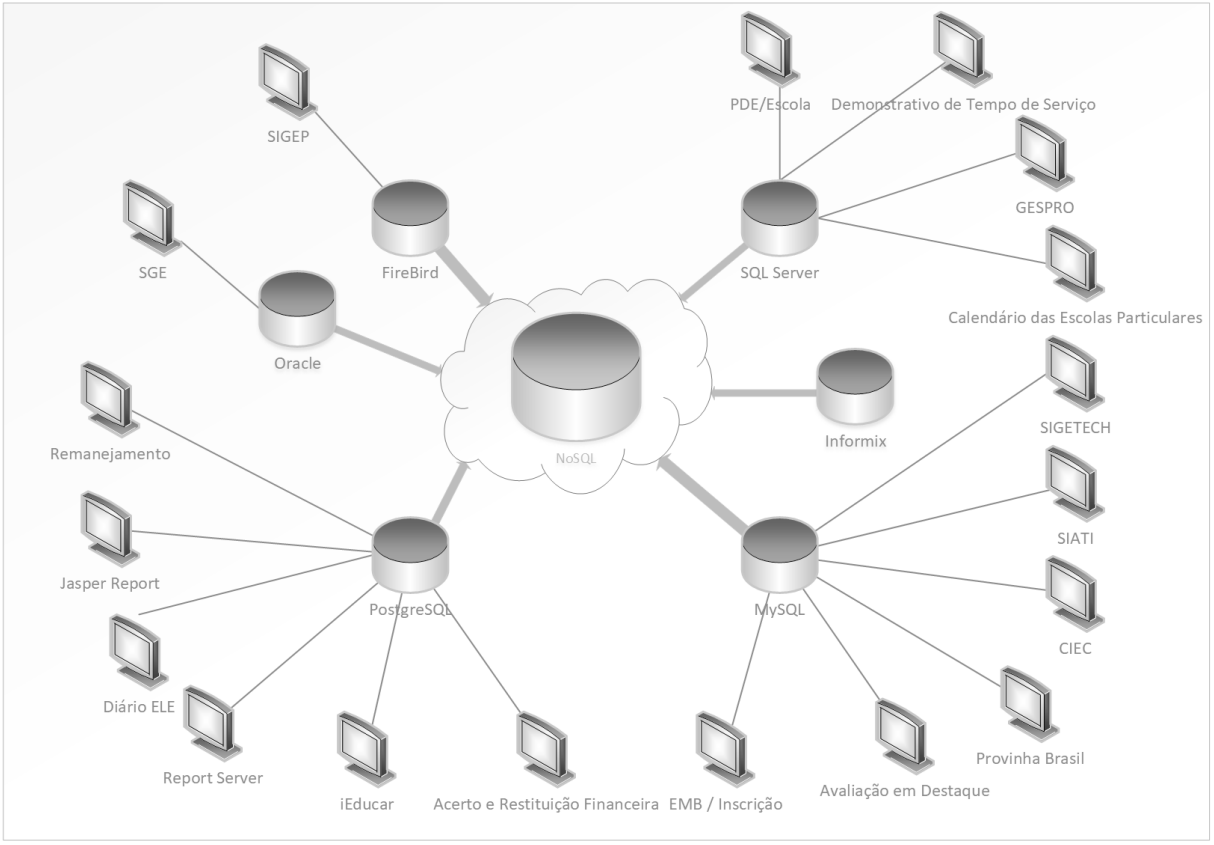


Figura 3.5: Cenário Após a Implantação da Arquitetura.

essa integração será calculada em trabalho posterior, devendo ser levado em conta a quantidade de transações realizadas pelas aplicações no ambiente computacional. Para a SEEDF, faremos a integração destes dados uma vez por dia.

3.6 Síntese do Capítulo

Para a criação da arquitetura de integração entre bancos de dados proposta neste trabalho, foi necessária uma análise detalhada do ecossistema computacional da SEEDF, com seus respectivos problemas. Após este entendimento e o conhecimento das arquiteturas, métodos e técnicas apresentadas no capítulo anterior, foi possível o desenvolvimento de uma nova arquitetura, dividida logicamente em 3 grandes camadas, a camada de dados local, a camada de integração e a camada de dados global. Esta arquitetura foi desenvolvida levando em consideração a técnica de *Schema Matching* e principalmente o conceito de ontologia, que é núcleo dessa arquitetura, cujo propósito é o de automatizar o processo de integração de dados, com a intervenção mínima de um analista de TI para isso, tendo como papel principal o de um analista da área de negócio que será responsável por montar essa ontologia de acordo com a necessidade da integração. Além do desenho e explicação

da arquitetura lógica, foi desenvolvido um *middleware*, denominado Noach, cuja função é fazer fisicamente a integração proposta. Após o desenvolvimento do Noach, o fluxo dos dados é explicado, levando em conta como é realizado dentro de SGBD Local Integrado e de um SGBD Local Integrado para o NoSQL.

Capítulo 4

Análise da Arquitetura

Neste capítulo é apresentado o processo realizado para a análise e validação da Arquitetura entre Bancos de Dados proposta nesta dissertação de mestrado. Este capítulo é divido nas seguintes seções: 4.1 A escolha dos SGBD e do NoSQL, 4.2 Criação da Ontologia, 4.3 A Execução do Noach e Validação da Arquitetura, e por fim, 4.4 A Síntese do Capítulo.

4.1 A escolha dos SGBD e do NoSQL

Para realizar o teste da integração por meio do Noach, foram escolhidos dois SGBD, o PostgreSQL e o MySQL como bases de dados locais. Isso se deu, além do fato deles serem *open-source*, deles serem os dois mais utilizados dentro do ecossistema computacional da SEEDF.

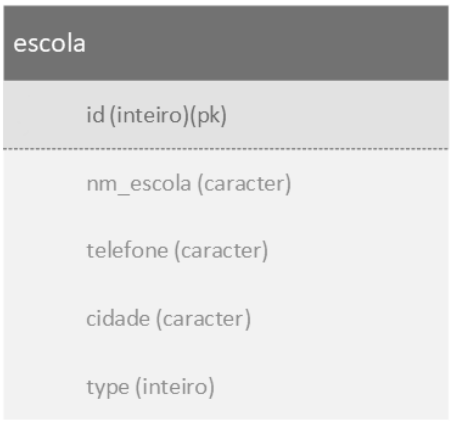


Figura 4.1: Esquema da Tabela Local - PostgreSQL.

Foi criada dentro do PostgreSQL, uma tabela denominada escola com as seguintes colunas: id, nome, telefone, cidade e type, conforme pode ser visto na Figura 4.1. O termo escola foi escolhido, por ser um termo bastante utilizado nas tabelas dos sistemas

da SEEDF, como é o caso do Sistema de Gestão Escolar, i-Educar, e o sistema de Gestão de Pessoas, SIGEP.

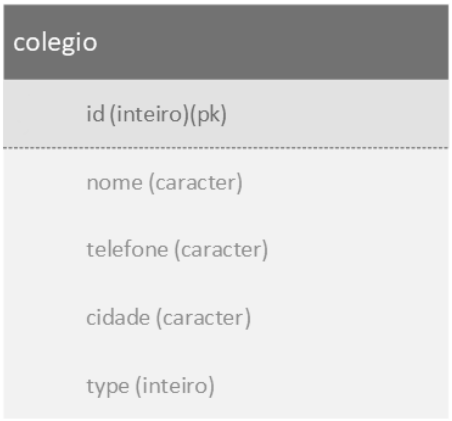


Figura 4.2: Esquema da Tabela Local - MySQL.

Dentro do MySQL, foi criada uma tabela denominada colegio com as seguintes colunas: id, nome, telefone, cidade e type, conforme pode ser visto na Figura 4.2. O termo colegio foi escolhido, por ser um termo utilizado na tabela do Sistema de Avaliação em Destaque, sistema responsável por captar e emitir relatórios das avaliações em rede da SEEDF.

Para a base de dados global foi escolhido o Splice Machine. A escolha dele se deu, além de ser baseado em tecnologias *open-source* como o Hadoop, HBase, Apache Spark e Apache Derby, ele contém uma camada relacional, criando assim, um banco de dados que oferece benefícios quando comparados aos SGBD Relacionais Tradicionais, como Oracle, IBM DB2 ou MySQL. Ele apresenta como características [63]:

- Suporte a SQL - Aproveita as existentes análises, relatório e aplicações baseadas em SQL sem reescrita, além de possibilitar a utilização de *queries* para persistência dos dados;
- Mais rápido que os SGBD relacionais tradicionais - Aproveita o HBase, um banco NoSQL distribuído, bem como o Spark, um potente *cluster* computacional em memória;
- Flexível - Provê um bom desempenho em simultâneas consultas OLTP e OLAP.

É importante enfatizar que a arquitetura é genérica suficiente para ser utilizada com outros SGBD relacionais e outros NoSQL que suportem SQL, como é o caso da combinação do Hive/HBase, Presto, Pig e outros.

Dentro da base de dados global, foi criada uma tabela denominada unidade_escolar com as colunas: id, nm_unidade_escolar, tel_unidade_escolar, cid_unidade_escolar e tp_unidade_escolar, como visto na Figura 4.3.

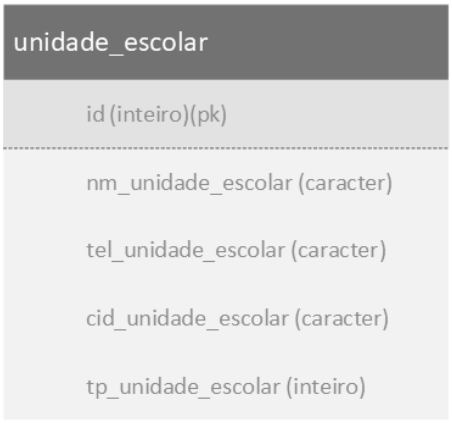


Figura 4.3: Esquema da Tabela Global.

4.2 Criação da Ontologia

Como o foco desta dissertação não é tratar sobre a criação de uma ontologia, mas a utilização de uma para a automatização da integração entre bancos de dados, foi elaborada uma ontologia de aplicação leve, por meio do software Protégé [62] para validar a arquitetura proposta.

Dentro do Protégé, como pode ser visto na Figura 4.4 e 4.5, foram criadas três classes de teste, a classe colegio, escola e unidade_escolar, que apresentam a mesma equivalência semântica, ou seja, significam ser um estabelecimento público ou privado destinado ao ensino coletivo. Essa equivalência é definida na tela de descrição da classe no campo *Equivalente To*, como pode ser visto na Figura 4.6.

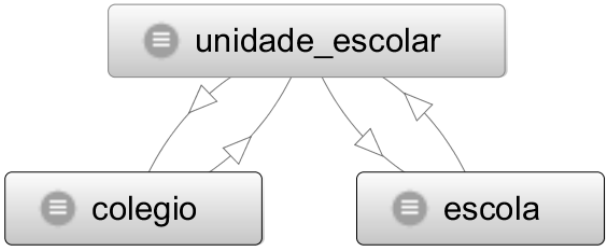


Figura 4.4: Protégé - OntoGraf - Classes.

A mesma Figura 4.6 mostra que as classes colegio e escola são equivalente a unidade_escolar e que a unidade_escolar tem equivalência semântica com escola e colegio. Isso significa dizer que quando as tabelas dos SGBD locais integrados tiveram o nome de colegio ou escola, elas serão integradas com a tabela do banco de dados global unidade_escolar.

A Figura 4.7 mostra as propriedades dos objetos que foram criadas para validar a arquitetura proposta. Para a classe colegio foram criadas as propriedades nome, cidade,

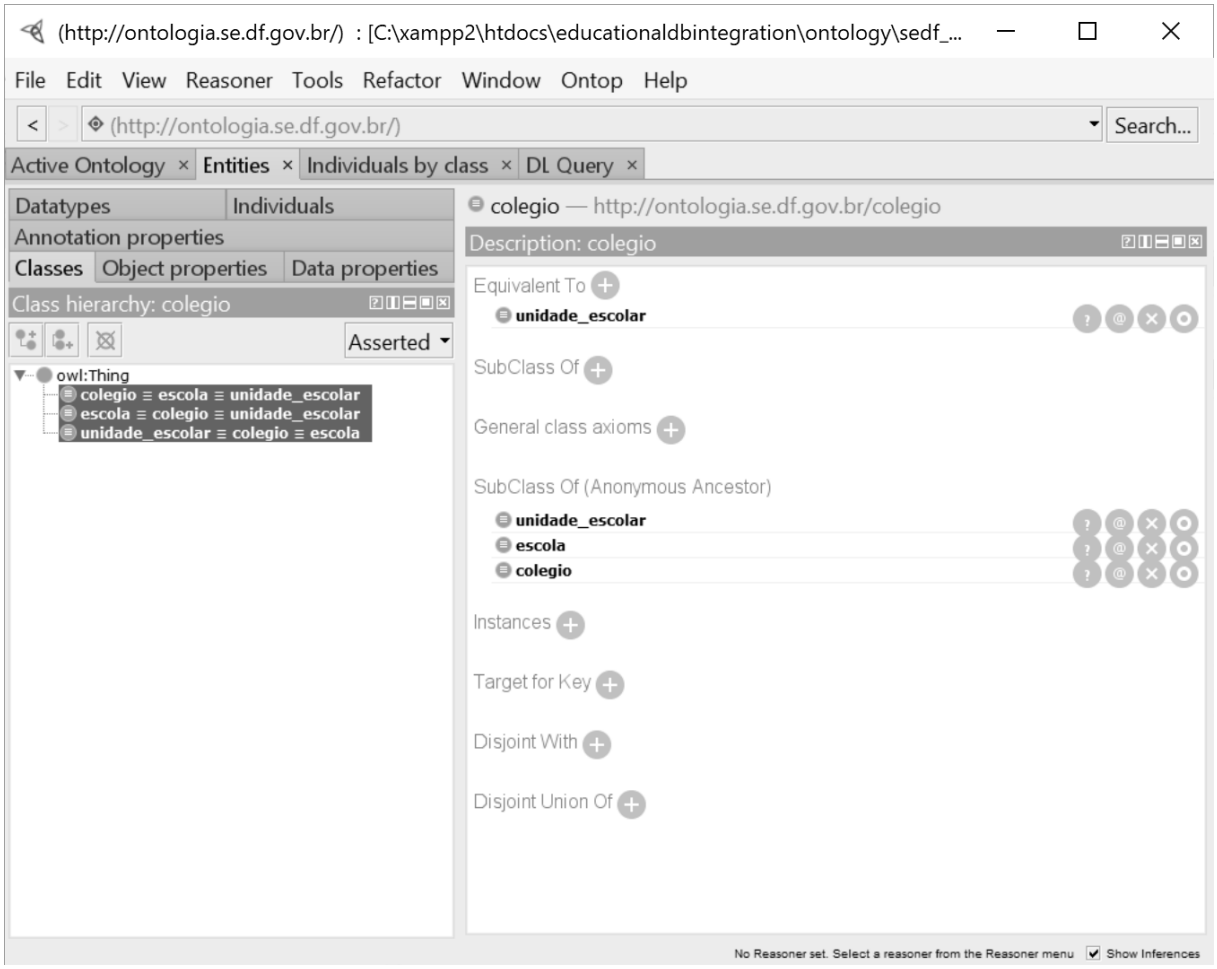


Figura 4.5: Protégé - Classes.

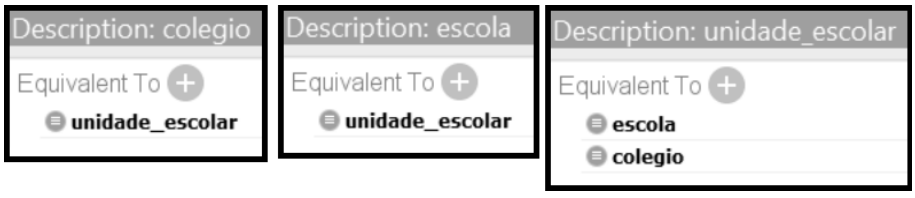


Figura 4.6: Protégé - Equivalência Semântica entre Classes.

telefone e *type*, este último dizendo se a escola é rural ou urbana. Estas propriedades foram criadas para que a classe escola tenha as propriedades nm_escola, cidade, telefone e *type*, e para a classe unidade_escolar, tem-se nm_unidade_escolar, cid_unidade_escolar, tel_unidade_escolar e tp_unidade_escolar.

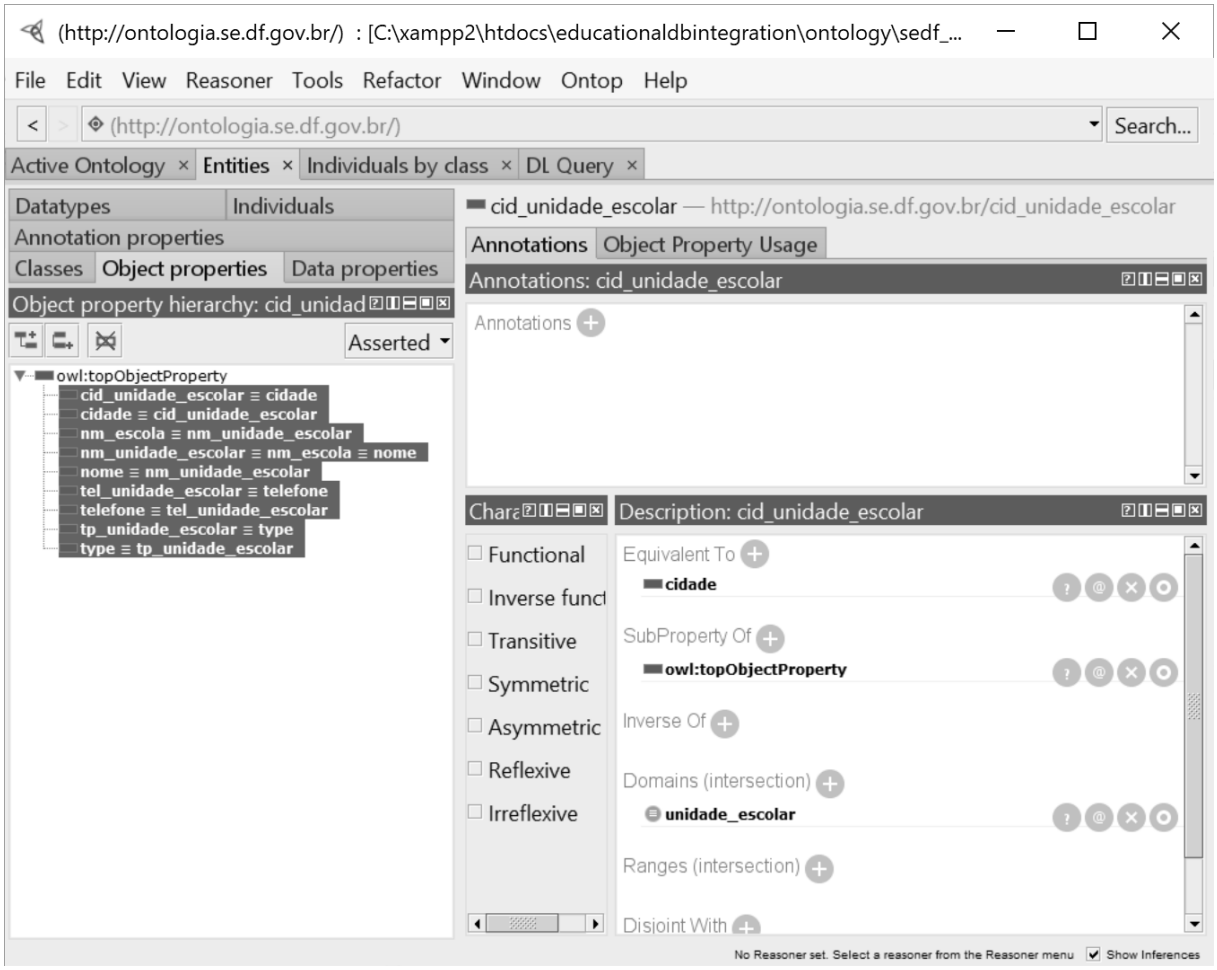


Figura 4.7: Protégé - Propriedades dos Objetos.

Assim como nas classes, foi necessária a realização da equivalência semântica entre os termos que compõe as propriedades de objetos. As propriedades nome da classe colegio e nm_escola da classe escola são equivalentes a nm_unidade_escolar da classe unidade_escolar, como pode ser visto na Figura 4.8. As propriedades cidade, telefone e *type* das classes colegio e escola são, respectivamente, equivalentes a cid_unidade_escolar, tel_unidade_escolar e tp_unidade_escolar da classe unidade_escolar, como vista na Figuras 4.9.

Após a criação desta ontologia leve de aplicação no Protégé, deve-se salvá-la no formato padrão *RDF/XML Syntax* para gerar o arquivo OWL, como visto na Figura 4.10 e na sua versão completa no Apêndice A, onde é mostrada a sintaxe dele, e é por meio deste arquivo que o *middleware* Noach integrará os bancos de dados.

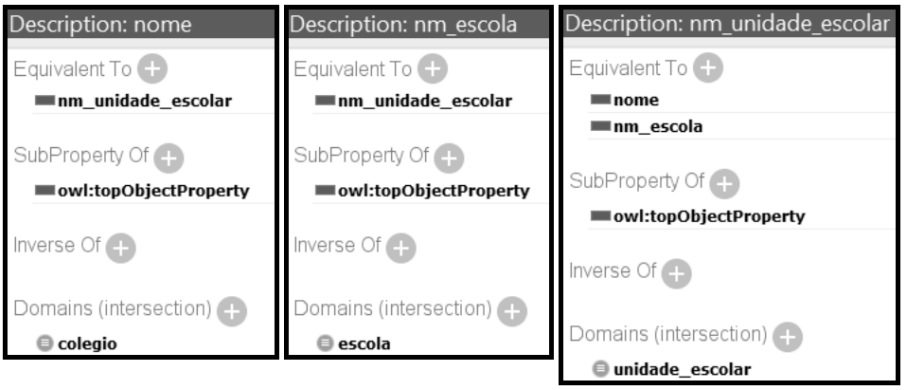


Figura 4.8: Protégé - Equivalência Semântica entre as Propriedades nome, nm_escola e nm_unidade_escolar.

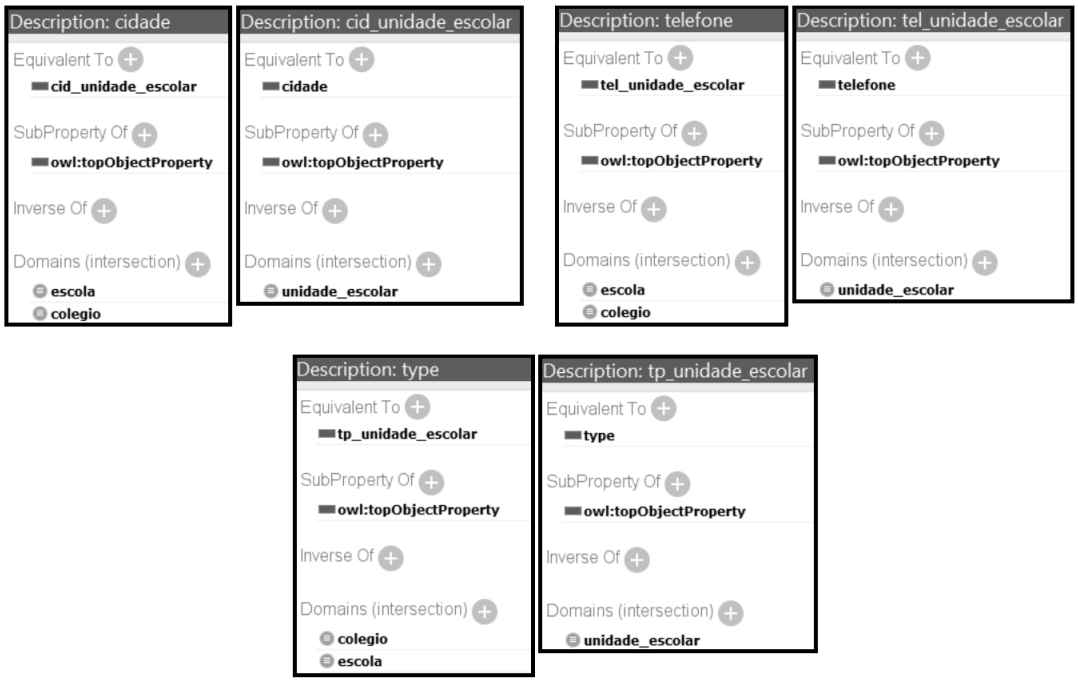


Figura 4.9: Protégé - Equivalência Semântica entre as Propriedades cidade, telefone e type, respectivamente, com cid_unidade_escolar, tel_unidade_escolar e tp_unidade_escolar.

sedf_version.owl

```
<?xml version="1.0"?>
<rdf:RDF xmlns="http://ontologia.se.df.gov.br/"
     xml:base="http://ontologia.se.df.gov.br/"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns:xml="http://www.w3.org/XML/1998/namespace"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
    <owl:Ontology rdf:about="http://ontologia.se.df.gov.br/"/>

    <!--
    ///////////////////////////////////////////////////////////////////////////////////////
    //
    // Object Properties
    //
    ///////////////////////////////////////////////////////////////////////////////////////
     -->

    <!-- http://ontologia.se.df.gov.br/cid_unidade_escolar -->

    <owl:ObjectProperty rdf:about="http://ontologia.se.df.gov.br/cid_unidade_escolar">
        <owl:equivalentProperty rdf:resource="http://ontologia.se.df.gov.br/cidade"/>
        <rdfs:subPropertyOf rdf:resource="http://www.w3.org/2002/07/owl#topObjectProperty"/>
        <rdfs:domain rdf:resource="http://ontologia.se.df.gov.br/unidade_escolar"/>
    </owl:ObjectProperty>

    <!-- http://ontologia.se.df.gov.br/cidade -->

    <owl:ObjectProperty rdf:about="http://ontologia.se.df.gov.br/cidade">
        <rdfs:subPropertyOf rdf:resource="http://www.w3.org/2002/07/owl#topObjectProperty"/>
        <rdfs:domain rdf:resource="http://ontologia.se.df.gov.br/colegio"/>
        <rdfs:domain rdf:resource="http://ontologia.se.df.gov.br/escola"/>
    </owl:ObjectProperty>
```

Figura 4.10: Protégé - Arquivo OWL.

4.3 A Execução do Noach e Validação da Arquitetura

O Noach é o *middleware* desenvolvido para validar a Arquitetura de Integração entre Bancos de Dados proposto nesta dissertação de mestrado. Este *middleware* foi desenvolvido levando em consideração todas as camadas descritas na arquitetura. Nesta seção é apresentado o passo a passo da execução do *middleware* Noach em um ambiente de teste controlado na DISIS/SEEDF.

O fluxo de execução do *middleware* Noach é demostrado na Figura 4.11. Ele se incia pela camada de interação com o usuário, cujo analista da área de negócio ou analista de TI acessa o *middleware* Noach. Na Camada de Dados Local da arquitetura, essa aplicação verifica se a conexão com os bancos de dados foram estabelecidas. Após esta verificação, inicia-se a integração, por meio da Camada de Integração, onde é realizada a captação de modificações, a equivalência semântica dos nomes das tabelas e colunas automaticamente por meio da leitura do arquivo OWL da ontologia. Com a equivalência semântica entre estes termos, as *queries* globais são geradas. Após a geração destas *queries*, na Camada de Dados Global, elas são persistidas no NoSQL realizando toda a integração com sucesso e voltando esta informação para o usuário de que a integração foi bem-sucedida.

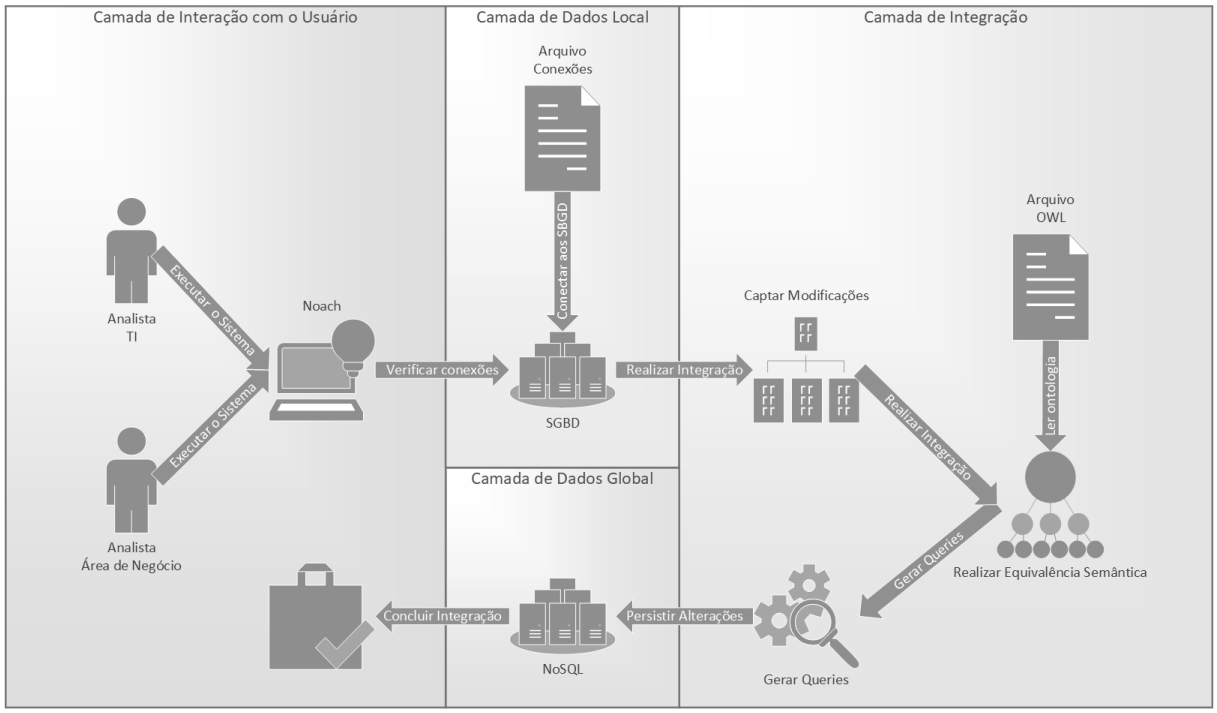


Figura 4.11: Fluxo de Execução do *middleware* Noach.

Para melhorar a explicação e visualização do funcionamento do Noach, foi criada uma aplicação web com as seguintes tecnologias: HTML5, Bootstrap, CSS3 e PHP. Essa

aplicação foi colocada dentro do XAMPP, ambiente de desenvolvimento para PHP. Na tela de acesso é solicitado o *username* e o *password*, conforme visto na Figura 4.12.

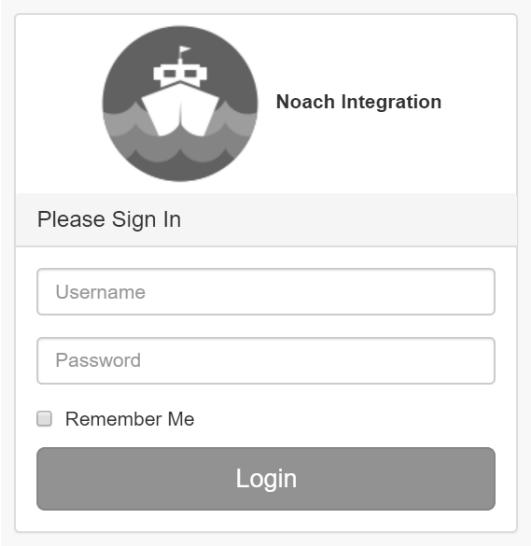


Figura 4.12: Noach - Tela de Acesso

Após realizar o acesso, a página de configuração do Noach é apresentada. Nesta tela é mostrado os *inputs* onde deverão ser submetidos os arquivos de conexão e da ontologia, conforme pode ser visto na Figura 4.13. Para verificar se as conexões foram realizadas com sucesso, acesse o item de menu *Status of Databases* e o item *Synchronizer* para verificar se a ontologia foi configurada corretamente.

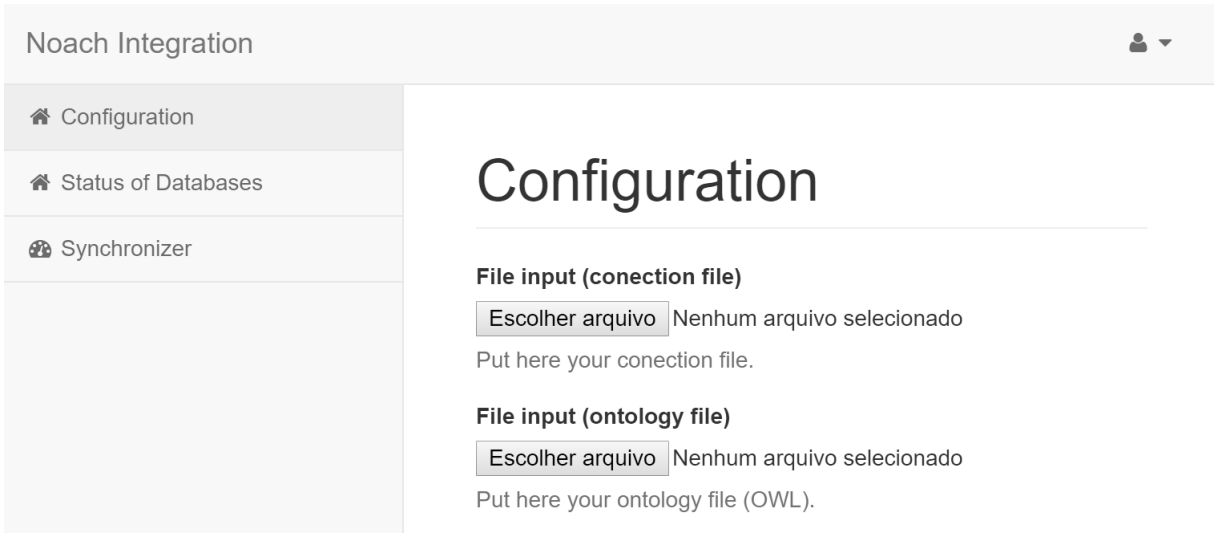


Figura 4.13: Noach - Página de Configuração.

Após realizar a configuração, basta clicar no *link Status of Databses* e a página com os *status* de conexão das bases de dados é apresentada. Nesta tela é mostrado o teste de

conexão com todas as bases de dados configuradas que serão realizadas as sincronizações, conforme pode ser visto na Figura 4.14. Caso alguma destas bases esteja desconectada, será possível a visualização desta informação nesta mesma tela. Como demonstração, é apresentada nesta tela se a conexão foi realizada com sucesso nos bancos MySQL, Splice Machine e PostgreSQL.

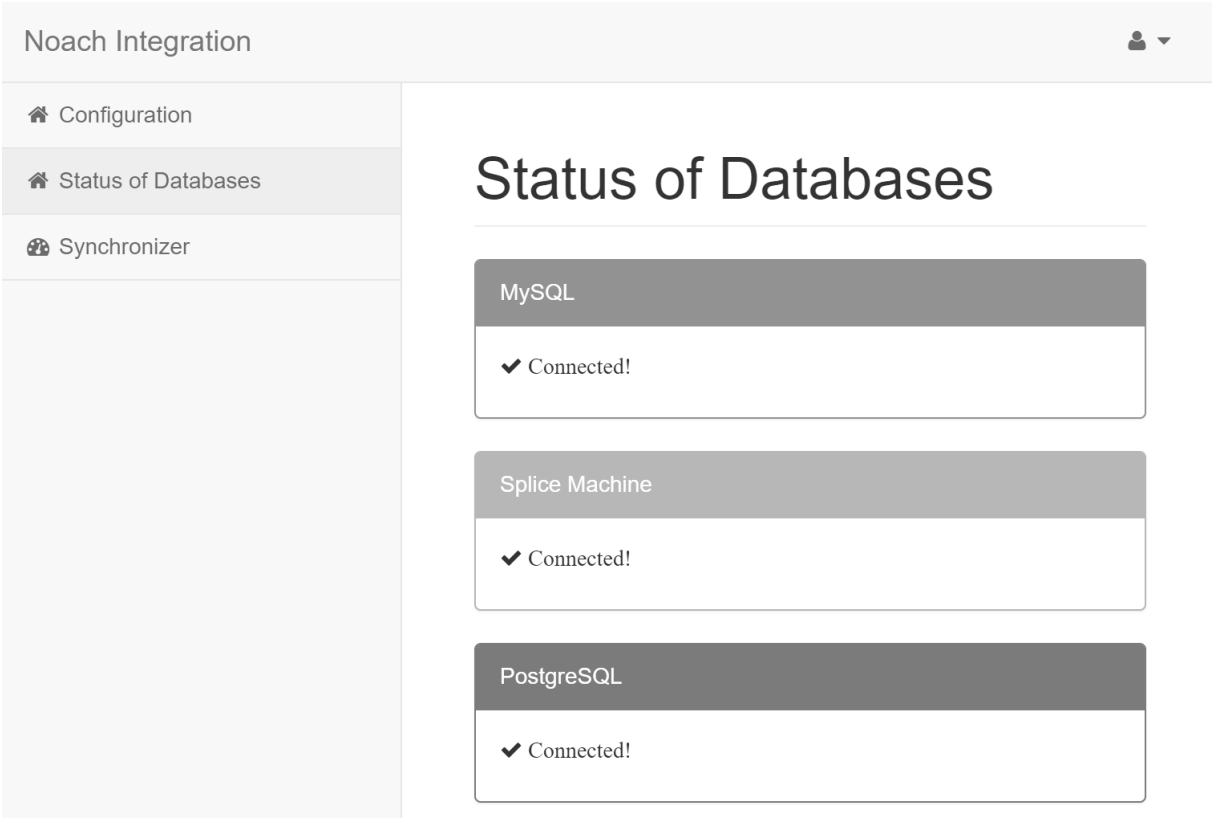


Figura 4.14: Noach - Página de Status de Conexão.

Para iniciar o processo de integração, basta entrar no link *Synchronizer*, no menu lateral esquerdo, que o *middleware* Noach será ativado. Durante este processo de integração ele acessa a tabela de sincronização de cada SGBD Local Integrado, de acordo com o explicado na Camada de Dados Local da Arquitetura. Em seguida ele captura todos estes dados, colocando-os em memória e *truncando* esta tabela, validando assim, a subcamada de Captação de Modificações da Camada de Integração. Após essa captação, para cada iteração, ele acessa a ontologia realizando a equivalência semântica entre os nomes das tabelas e colunas, validando a Subcamada de Equivalência Semântica. Imediatamente depois, com todos estes dados armazenados em memória, são montadas as *queries* globais, ratificando a subcamada de Gerador de *Query* Global. Após a geração delas, todas são persistidas no banco de dados global, validando assim, o acesso aos dados da Camada de Dados Global. O resultado final da integração pode ser visualizado na Figura 4.15.

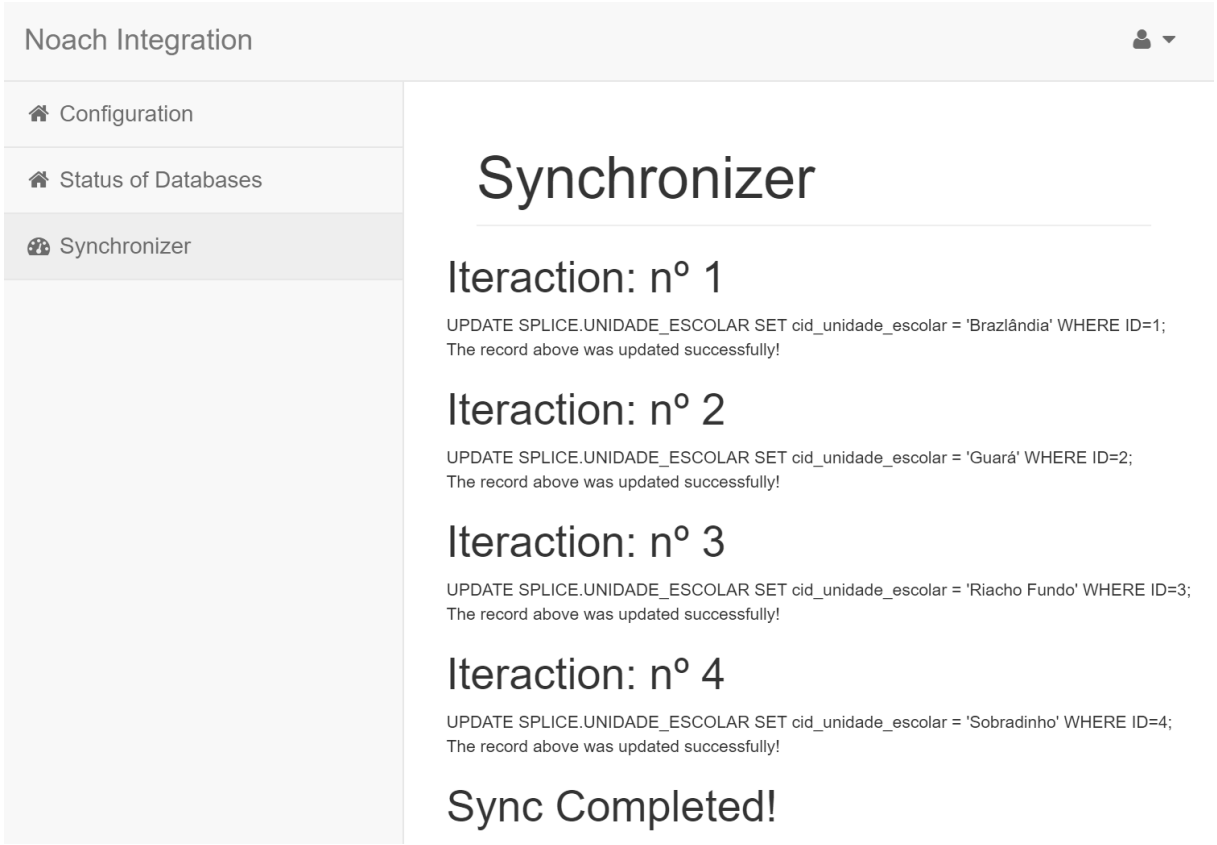


Figura 4.15: Noach - Página de Sincronização

Para um entendimento mais técnico e de visualização da fácil utilização do Noach, basta verificar o Código 4.1. O comando da segunda linha é responsável por retirar o limite da execução da aplicação. Esta ação se faz necessária devido ao grande volume de dados que será integrado, ou seja, pelo fato do *middleware* ter sido desenvolvido em *PHP*, que por padrão, a execução de um *script* dura apenas 30 segundos, precisa-se retirar este limite para que se possa integrar uma grande quantidade de dados. As linhas cinco, seis e sete, são as linhas necessárias para inclusão da RAP API e do arquivo de classes do Noach. A linha dez é responsável pela instanciação do objeto *sync* da classe *MapOntology* e a última linha executa o Noach.

Caso, em ambiente de produção, se deseje automatizar a execução do Noach, existem duas maneiras distintas. A primeira basta acessar o Agendador de Tarefas do *Windows* ou o Cron do Linux e realizar o agendamento de quando ele deverá ser executado, simplesmente criando um arquivo PHP e inserindo o Código 4.1. Para realizar a segunda maneira, deve-se alterar o código da aplicação, colocar uma estrutura de repetição com algum argumento em que ela entre em *loop* infinito, como por exemplo o caso do *while(1)*, uma linha antes do comando que executa o Noach. Essas duas ações estão ratificando a definição empregada na subcamada da arquitetura denominada *Timing* de Sincronização.

Conforme demonstrado nesta seção, o Noach, realiza uma integração automática entre os bancos de dados sem uma intervenção manual de algum analista da área de TI, necessitando apenas de uma ontologia, que deve ser feita por um analista da área de negócio. Dessa forma, a Arquitetura proposta por este projeto de pesquisa é validada.

```
1  //unlimited execution
2  set_time_limit(0);
3  //include classes of sync
4  define("RDFAPI_INCLUDE_DIR", "../vendor/rdfapi-php/api/");
5  include_once(RDFAPI_INCLUDE_DIR . "RdfAPI.php");
6  include_once('sync.class.php');
7  //instantiating the sync object
8  $sync = new MapOntology;
9  //start integration/synchronization
10 $sync->syncMapIteraction(false);
```

Código 4.1: Código em PHP que executa a integração

4.4 Síntese do Capítulo

Para a análise e validação da Arquitetura de Integração entre Bancos de Dados proposta neste trabalho de pesquisa, foi demonstrado neste capítulo o motivo das escolhas dos bancos de dados locais e global de teste, o processo de criação da ontologia leve de aplicação empregada para a validação, como é realizado a criação de um SGBD Local Integrado e, por fim, como é simples a execução do Noach e a validação da Arquitetura.

Capítulo 5

Conclusão e Trabalhos Futuros

A Secretaria de Estado de Educação do Distrito Federal (SEEDF) por ser a maior Secretaria do Governo do Distrito Federal (GDF), está dividia entre 14 coordenações regionais de ensino, três prédios sede e em torno de 700 unidades escolares. O público de atendimento educacional é estimado aproximadamente em 460 mil estudantes. A SEEDF conta com aproximadamente 47 mil servidores efetivos e temporários.

A Subsecretaria de Modernização e Tecnologia (SUMTEC), representada por meio da Diretoria de Desenvolvimento de Sistemas (DISIS), desenvolve diversos sistemas de informação para auxiliar e facilitar a execução do trabalho de todo o quadro de servidores da SEEDF. Nestes sistemas são gerados por dia milhares de dados. Para um gestor, na realização da tomada de decisão, se faz necessário ter as informações obtidas por meio destes dados.

Como contribuição profissional, esta dissertação de mestrado traz para a SEEDF um *middleware*, denominado Noach, desenvolvido seguindo as premissas da três camadas da Arquitetura aqui proposta. O Noach realiza o processo de integração de bancos de dados de forma automática por meio da utilização de uma ontologia.

A comunicação das bases de dados locais dos sistemas da SEEDF com uma base de dados global, proporciona uma maior facilidade para prover relatórios gerenciais, sendo possível uma melhor análise dos dados, gerando políticas públicas que são de suma importância para a comunidade escolar e para se ter uma melhora na gestão das escolas públicas do Distrito Federal.

Além de oferecer esta contribuição profissional a SEEDF, esta dissertação de mestrado disponibiliza uma nova arquitetura de integração entre banco de dados heterogêneos à comunidade acadêmica de Ciência da Computação e Ciência da Informação, com a finalidade de auxiliar na integração entre bancos de dados de uma forma mais simples e automática. Esta arquitetura apresenta regras bem definidas e replicação física de dados, sendo automatizada por meio da utilização de uma ontologia leve de aplicação, cujo papel é de

mapear as equivalências semântica entre os nomes das tabelas e colunas dos bancos de dados locais para um banco de dados global.

Como trabalhos futuros, faz-se necessária a criação de uma ontologia de domínio mais completa no âmbito da SEEDF, para que seja possível a integração completa de todas as bases de dados do ecossistema computacional da SUMTEC. Além desta criação, realizar uma manutenção evolutiva no Noach para que seja possível a leitura de mais de uma ontologia de domínio ao mesmo tempo, para que ele possa ser apresentado e levado para outros órgãos que necessitem do intercâmbio de informações entre sistemas dos mais diversos tipos de domínios de aplicação. Além desses trabalhos, pretende-se realizar uma pesquisa mais aprofundada em relação ao desempenho do *middleware* a fim de encontrar o tempo mínimo para que seja possível realizar a integração o mais próximo do tempo-real.

Com essa integração, será possível gerar relatórios gerenciais mais precisos, *dashboards* mais completos e realizar trabalhos de mineração de dados para trazer padrões que se encontram implícitos a todos estes dados, melhorando a cada dia a educação pública no Distrito Federal.

Referências

[1] Elmasri, Ramez e Navathe, Shamkant B: *Sistemas de banco de dados*. Pearson Addison Wesley, 2011. x, 4, 5, 6, 7, 21

[2] Han, Jing e Haihong, E e Le Guan e Du Jian: *Survey on nosql database*. Em *Pervasive computing e applications (ICPCA), 2011 6th international conference on*, páginas 363–366. IEEE, 2011. x, 12

[3] Wang, Yunxiao e Zhang, Xuecheng: *The research of multi-source heterogeneous data integration based on linq*. Em *Computer Science e Electronics Engineering (ICCSEE), 2012 International Conference on*, volume 1, páginas 147–150. IEEE, 2012. x, 16, 17, 22, 23

[4] Rao, Jinghai e Su, Xiaomeng: *A survey of automated web service composition methods*. Em *International Workshop on Semantic Web Services e Web Process Composition*, páginas 43–54. Springer, 2004. x, 17, 18

[5] Coulouris, George e Dollimore, Jean e Kindberg Tim e Blair Gordon: *Sistemas Distribuídos: Conceitos e Projeto*, volume 5. Bookman Editora, 2013. 2, 13

[6] Tanenbaum, Andrew S e Woodhull, Albert S: *Sistemas Operacionais: Projetos e Implementação*, volume 3. Bookman Editora, 2009. 2, 13

[7] Date, Christopher J: *Introdução a sistemas de bancos de dados*. Elsevier Brasil, 2004. 4, 6, 7, 8

[8] Laudon, Kenneth C e Jane P Laudon: *Sistema de Informação Gerencial*, volume 11. Pearson, 2014. 5

[9] Heuser, Carlos Alberto: *Projeto de banco de dados: Volume 4 da Série Livros didáticos informática UFRGS*. Bookman Editora, 2009. 5, 7

[10] Győrödi, Cornelia e Győrödi, Robert e Sotoc Roxana: *A comparative study of relational e non-relational database models in a web-based application*. International Journal of Advanced Computer Science & Applications, 1(6):78–83, 2015. 6

[11] Codd, Edgar F: *A relational model of data for large shared data banks*. Communications of the ACM, 13(6):377–387, 1970. 7

[12] Chamberlin, Donald D e Boyce, Raymond F: *Sequel: A structured english query language*. Em *Proceedings of the 1974 ACM SIGFIDET (now SIGMOD) workshop on Data description, access e control*, páginas 249–264. ACM, 1974. 8

[13] Chamberlin, Donald D e Astrahan, Morton M e Blasgen Michael W e Gray James N e King W Frank e Lindsay Bruce G e Lorie Raymond e Mehl James W e Price Thomas G e Putzolu Franco e outros: *A history e evaluation of system r*. Communications of the ACM, 24(10):632–646, 1981. 8

[14] Sagiroglu, Seref e Sinanc, Duygu: *Big data: A review*. Em *Collaboration Technologies and Systems (CTS), 2013 International Conference on*, páginas 42–47. IEEE, 2013. 8

[15] Floratou, Avrilia e Teletia, Nikhil e DeWitt David J e Patel Jignesh M e Zhang Donghui: *Can the elephants handle the nosql onslaught?* Proceedings of the VLDB Endowment, 5(12):1712–1723, 2012. 9

[16] Leavitt, Neal: *Will nosql databases live up to their promise?* Computer, 43(2):12–14, 2010. 9

[17] Strozzi, Carlo: *Nosql - a relational database management system*. http://bit.do/strozzi1998nosql, 1998. 9

[18] Chang, Fay e Dean, Jeffrey e Ghemawat Sanjay e Hsieh Wilson C e Wallach Deborah A e Burrows Mike e Chandra Tushar e Fikes Andrew e Gruber Robert E: *Bigtable: A distributed storage system for structured data*. ACM Transactions on Computer Systems (TOCS), 26(2):4, 2008. 10

[19] DeCandia, Giuseppe e Hastorun, Deniz e Jampani Madan e Kakulapati Gunavardhan e Lakshman Avinash e Pilchin Alex e Sivasubramanian Swaminathan e Vosshall Peter e Vogels Werner: *Dynamo: amazon's highly available key-value store*. ACM SIGOPS Operating Systems Review, 41(6):205–220, 2007. 10

[20] Lakshman, Avinash e Malik, Prashant: *Cassandra: a decentralized structured storage system*. ACM SIGOPS Operating Systems Review, 44(2):35–40, 2010. 10

[21] Lai, Eric: *Twitter growth prompts switch from mysql to 'nosql'database*. http://www.computerworld.com/article/2520084/database-administration/twitter-growth-prompts-switch-from-mysql-to–nosql–database.html, 2010. 10

[22] Anderson, J Chris e Lehnardt, Jan e Slater Noah: *CouchDB: the definitive guide*. " O'Reilly Media, Inc.", 2010. 10

[23] Dean, Jeffrey e Ghemawat, Sanjay: *Mapreduce: simplified data processing on large clusters*. Communications of the ACM, 51(1):107–113, 2008. 10

[24] Chodorow, Kristina: *MongoDB: the definitive guide*. " O'Reilly Media, Inc.", 2013. 10

[25] Cattell, Rick: *Scalable sql e nosql data stores*. Acm Sigmod Record, 39(4):12–27, 2011. 10

[26] Corbellini, Alejandro e Mateos, Cristian e Zunino Alejandro e Godoy Daniela e Schiaffino Silvia: *Persisting big-data: The nosql landscape*. Information Systems, 63:1–23, 2017. 10

[27] Brewer, Eric A: *Towards robust distributed systems.* Em *PODC*, volume 7, 2000. 11

[28] Sommerville, Ian: *Engenharia de software*, volume 9. Editora Pearson, 2011. 13

[29] Oliveira, Manoel M A e Carlos, Danilo G e Oliveira Antonio R do V e de Castro Angelica F e outros de: *Um estudo comparativo entre banco de dados orientado a objetos, banco de dados relacionais e framework para mapeamento objeto/relacional, no contexto de uma aplicação web.* Holos, 1:182–198, 2015. 13

[30] Shelake, Vijay Maruti e Varunakshi Sachin Bhojane: *A novel approach for multi-source heterogeneous database integration.* Em *Machine Intelligence and Research Advancement (ICMIRA), 2013 International Conference on*, páginas 184–190. IEEE, 2013. 13, 22

[31] Rahm, Erhard e Bernstein, Philip A: *A survey of approaches to automatic schema matching.* the VLDB Journal, 10(4):334–350, 2001. 14

[32] Inmon, William H: *Como construir o Data Warehouse*, volume 2. Editora Campus, 1997. 14

[33] Kimball, Ralph: *The data warehouse lifecycle toolkit: expert methods for designing, developing, e deploying data warehouses.* John Wiley & Sons, 1998. 15

[34] Almahy, Ibrahim e Salim, Naomie: *Database integration: Importance and approaches.* Journal of Theoretical & Applied Information Technology, 54(1), 2013. 15, 16

[35] Chaudhuri, Surajit e Dayal, Umeshwar: *An overview of data warehousing e olap technology.* ACM Sigmod record, 26(1):65–74, 1997. 15

[36] Sheth, Amit P e Larson, James A: *Federated database systems for managing distributed, heterogeneous, e autonomous databases.* ACM Computing Surveys (CSUR), 22(3):183–236, 1990. 15

[37] Yang, Kai e Steele, Robert: *A semantic integration solution for online accommodation information integration.* Em *2011 6th IEEE Conference on Industrial Electronics e Applications*, páginas 1105–1110. IEEE, 2011. 19, 23

[38] Chawathe, Sudarshan e Garcia-Molina, Hector e Hammer Joachim e Ireland Kelly e Papakonstantinou Yannis e Ullman Jeffrey e Widom Jennifer: *The tsimmis project: Integration of heterogenous information sources.* 1994. 19

[39] Ng, Wee Siong e Ooi, Beng Chin e Tan K L e Zhou Aoying: *Peerdb: A p2p-based system for distributed data sharing.* Em *Data Engineering, 2003. Proceedings. 19th International Conference on*, páginas 633–644. IEEE, 2003. 19

[40] Gruber, Thomas R e outros: *A translation approach to portable ontology specifications.* Knowledge acquisition, 5(2):199–220, 1993. 19

[41] Freitas, Frederico Luiz Gonçalves de: *Ontologias e a web semântica.* Jornada de Mini-Cursos em Inteligencia Artificial, SBC, 8, 2003. 19

[42] Kavitha, C e Sadasivam, G Sudha e Shenoy Sangeetha N: *Ontology based semantic integration of heterogeneous databases*. European Journal of Scientific Research, 64(1):115–122, 2011. 20, 23

[43] Mizoguchi, Riichiro: *Part 1: introduction to ontological engineering*. New Generation Computing, 21(4):365–384, 2003. 20

[44] Dou, Dejing e LePendu, Paea: *Ontology-based integration for relational databases*. Em *Proceedings of the 2006 ACM symposium on Applied computing*, páginas 461–466. ACM, 2006. 21

[45] Phatak, Shirish H e Badrinath, BR: *Bounded locking for optimistic concurrency control*. Relatório Técnico, Rutgers University TR DCS-TR, 1999. 21

[46] Ito, Giani C e Ferreira, Maurício e Sant'Ana Nilson: *Computação móvel: Aspectos de gerenciamento de dados*. Instituto Nacional de Pesquisas espaciais - INPE, 2003. 21

[47] Palazzo, Sergio e Puliafito, Antonio e Scarpa Marco: *Design e evaluation of a replicated database for mobile systems*. Wireless Networks, 6(2):131–144, 2000. 21

[48] Karasneh, Yaser, Hamidah Ibrahim, Mohamed Othman e Razali Yaakob: *Matching schemas of heterogeneous relational databases*. Em *Applications of Digital Information and Web Technologies, 2009. ICADIWT'09. Second International Conference on the*, páginas 1–7. IEEE, 2009. 22, 23

[49] Jian-min, Han e Jun, Tong e Xi yu Li: *An adaptive heterogeneous database integration framework based on web service composition techniques*. Em *Granular Computing, 2008. GrC 2008. IEEE International Conference on*, páginas 265–268. IEEE, 2008. 23

[50] Madhavan, Jayant, Philip A Bernstein e Erhard Rahm: *Generic schema matching with cupid*. Em *vldb*, volume 1, páginas 49–58, 2001. 23

[51] Chien, Been Cfflan e Shiang Yi He: *A hybrid approach for automatic schema matching*. Em *Machine Learning and Cybernetics (ICMLC), 2010 International Conference on*, volume 6, páginas 2881–2886. IEEE, 2010. 23, 24

[52] Chen, Wei, Huiling Guo, Fang Zhang, Xiaowei Pu e Xuefei Liu: *Mining schema matching between heterogeneous databases*. Em *Consumer Electronics, Communications and Networks (CECNet), 2012 2nd International Conference on*, páginas 1128–1131. IEEE, 2012. 23, 24

[53] Ying, Qian, Yue Liwen e Liu Zhenglin: *Discovering complex matches between database schemas*. Em *Control Conference, 2008. CCC 2008. 27th Chinese*, páginas 663–667. IEEE, 2008. 23, 24

[54] Tseng, Frank: *Xml-based heterogeneous database integration for data warehouse creation*. PACIS 2005 Proceedings, página 48, 2005. 23, 24

[55] Dhamankar, Robin, Yoonkyong Lee, AnHai Doan, Alon Halevy e Pedro Domingos: *imap: discovering complex semantic matches between database schemas*. Em *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, páginas 383–394. ACM, 2004. 23, 24

[56] Melnik, Sergey, Hector Garcia-Molina e Erhard Rahm: *Similarity flooding: A versatile graph matching algorithm and its application to schema matching*. Em *Data Engineering, 2002. Proceedings. 18th International Conference on*, páginas 117–128. IEEE, 2002. 23, 24

[57] Do, Hong Hai e Erhard Rahm: *Coma: a system for flexible combination of schema matching approaches*. Em *Proceedings of the 28th international conference on Very Large Data Bases*, páginas 610–621. VLDB Endowment, 2002. 23, 24

[58] Bíblia, A: *Genesis 5:29. Tradução de João Ferreira de Almeida*. Rio de Janeiro: Editora King Cross, 2008. 33

[59] *Php: Documentação*. `https://secure.php.net/docs.php`. Acessado em 03-11-2017. 34

[60] *Php: Pdo - manual*. `https://secure.php.net/manual/pt_BR/book.pdo.php`. Acessado em 03-11-2017. 34

[61] *Rap - rdf api for php v0.9.6*. `http://wifo5-03.informatik.uni-mannheim.de/bizer/rdfapi/`. Acessado em 03-11-2017. 34

[62] *Protégé*. `https://protege.stanford.edu/`. Acessado em 03-11-2017. 34, 41

[63] *Splice machine*. `http://www.splicemachine.com/`. Acessado em 19-11-2016. 40

Apêndice A

Ontologia de Aplicação da SEEDF

```
<?xml version="1.0"?>
<rdf:RDF xmlns="http://ontologia.se.df.gov.br/"
     xml:base="http://ontologia.se.df.gov.br/"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns:xml="http://www.w3.org/XML/1998/namespace"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
    <owl:Ontology rdf:about="http://ontologia.se.df.gov.br/"/>

    <!--
    ////////////////////////////////////////////////////////////////////////////////
    //
    // Object Properties
    //
    ////////////////////////////////////////////////////////////////////////////////
     -->

    <!-- http://ontologia.se.df.gov.br/cid_unidade_escolar -->

    <owl:ObjectProperty rdf:about="http://ontologia.se.df.gov.br/cid_unidade_escolar">
        <owl:equivalentProperty rdf:resource="http://ontologia.se.df.gov.br/cidade"/>
        <rdfs:subPropertyOf rdf:resource="http://www.w3.org/2002/07/owl#topObjectProperty"/>
        <rdfs:domain rdf:resource="http://ontologia.se.df.gov.br/unidade_escolar"/>
    </owl:ObjectProperty>

    <!-- http://ontologia.se.df.gov.br/cidade -->

    <owl:ObjectProperty rdf:about="http://ontologia.se.df.gov.br/cidade">
        <rdfs:subPropertyOf rdf:resource="http://www.w3.org/2002/07/owl#topObjectProperty"/>
        <rdfs:domain rdf:resource="http://ontologia.se.df.gov.br/colegio"/>
        <rdfs:domain rdf:resource="http://ontologia.se.df.gov.br/escola"/>
    </owl:ObjectProperty>

    <!-- http://ontologia.se.df.gov.br/nm_escola -->

    <owl:ObjectProperty rdf:about="http://ontologia.se.df.gov.br/nm_escola">
        <owl:equivalentProperty
        rdf:resource="http://ontologia.se.df.gov.br/nm_unidade_escolar"/>
        <rdfs:subPropertyOf rdf:resource="http://www.w3.org/2002/07/owl#topObjectProperty"/>
        <rdfs:domain rdf:resource="http://ontologia.se.df.gov.br/escola"/>
    </owl:ObjectProperty>

    <!-- http://ontologia.se.df.gov.br/nm_unidade_escolar -->

    <owl:ObjectProperty rdf:about="http://ontologia.se.df.gov.br/nm_unidade_escolar">
        <owl:equivalentProperty rdf:resource="http://ontologia.se.df.gov.br/nome"/>
        <rdfs:subPropertyOf rdf:resource="http://www.w3.org/2002/07/owl#topObjectProperty"/>
        <rdfs:domain rdf:resource="http://ontologia.se.df.gov.br/unidade_escolar"/>
    </owl:ObjectProperty>

    <!-- http://ontologia.se.df.gov.br/nome -->

    <owl:ObjectProperty rdf:about="http://ontologia.se.df.gov.br/nome">
        <rdfs:subPropertyOf rdf:resource="http://www.w3.org/2002/07/owl#topObjectProperty"/>
        <rdfs:domain rdf:resource="http://ontologia.se.df.gov.br/colegio"/>
    </owl:ObjectProperty>

    <!-- http://ontologia.se.df.gov.br/tel_unidade_escolar -->

    <owl:ObjectProperty rdf:about="http://ontologia.se.df.gov.br/tel_unidade_escolar">
        <owl:equivalentProperty rdf:resource="http://ontologia.se.df.gov.br/telefone"/>
        <rdfs:subPropertyOf rdf:resource="http://www.w3.org/2002/07/owl#topObjectProperty"/>
        <rdfs:domain rdf:resource="http://ontologia.se.df.gov.br/unidade_escolar"/>
    </owl:ObjectProperty>

    <!-- http://ontologia.se.df.gov.br/telefone -->
```

```
    <owl:ObjectProperty rdf:about="http://ontologia.se.df.gov.br/telefone">
        <rdfs:subPropertyOf rdf:resource="http://www.w3.org/2002/07/owl#topObjectProperty"/>
        <rdfs:domain rdf:resource="http://ontologia.se.df.gov.br/colegio"/>
        <rdfs:domain rdf:resource="http://ontologia.se.df.gov.br/escola"/>
    </owl:ObjectProperty>

    <!-- http://ontologia.se.df.gov.br/tp_unidade_escolar -->

    <owl:ObjectProperty rdf:about="http://ontologia.se.df.gov.br/tp_unidade_escolar">
        <owl:equivalentProperty rdf:resource="http://ontologia.se.df.gov.br/type"/>
        <rdfs:subPropertyOf rdf:resource="http://www.w3.org/2002/07/owl#topObjectProperty"/>
        <rdfs:domain rdf:resource="http://ontologia.se.df.gov.br/unidade_escolar"/>
    </owl:ObjectProperty>

    <!-- http://ontologia.se.df.gov.br/type -->

    <owl:ObjectProperty rdf:about="http://ontologia.se.df.gov.br/type">
        <rdfs:subPropertyOf rdf:resource="http://www.w3.org/2002/07/owl#topObjectProperty"/>
        <rdfs:domain rdf:resource="http://ontologia.se.df.gov.br/colegio"/>
        <rdfs:domain rdf:resource="http://ontologia.se.df.gov.br/escola"/>
    </owl:ObjectProperty>

    <!--
    ///////////////////////////////////////////////////////////////////////////////////////
    //
    // Classes
    //
    ///////////////////////////////////////////////////////////////////////////////////////
     -->

    <!-- http://ontologia.se.df.gov.br/colegio -->

    <owl:Class rdf:about="http://ontologia.se.df.gov.br/colegio">
        <owl:equivalentClass rdf:resource="http://ontologia.se.df.gov.br/unidade_escolar"/>
    </owl:Class>

    <!-- http://ontologia.se.df.gov.br/escola -->

    <owl:Class rdf:about="http://ontologia.se.df.gov.br/escola">
        <owl:equivalentClass rdf:resource="http://ontologia.se.df.gov.br/unidade_escolar"/>
    </owl:Class>

    <!-- http://ontologia.se.df.gov.br/unidade_escolar -->

    <owl:Class rdf:about="http://ontologia.se.df.gov.br/unidade_escolar"/>
</rdf:RDF>

<!-- Generated by the OWL API (version 4.2.8.20170104-2310) https://github.com/owlcs/owlapi
-->
```