
UNIVERSIDADE
FEDERAL
DE PERNAMBUCO

CENTRO DE ARTES E COMUNICAÇÃO

DEPARTAMENTO DE CIÊNCIA DA INFORMAÇÃO

CURSO DE GESTÃO DA INFORMAÇÃO

JORGE LUIS DE ARAUJO ROSSITER

# Proposta de um processo de construção de uma base de dados a partir de documentos físicos: estudo de caso dos estágios do curso Gestão da Informação da UFPE

TRABALHO DE CONCLUSÃO DE CURSO

Recife
2019

JORGE LUIS DE ARAUJO ROSSITER

# Proposta de um processo de construção de uma base de dados a partir de documentos físicos: estudo de caso dos estágios do curso Gestão da Informação da UFPE

Trabalho de Conclusão de Curso apresentado ao curso de Gestão da Informação, como parte dos requisitos necessários à obtenção do título de Bacharel em Gestão da Informação.

Orientador: Bruno Tenório Ávila, Dr.

Recife
2019

Catalogação na fonte
Biblioteca Joaquim Cardozo – Centro de Artes e Comunicação

R835p Rossiter, Jorge Luís de Araújo
Proposta de um processo de construção de uma base de dados a partir de documentos físicos: estudo de caso dos estágios do curso Gestão da Informação da UFPE / Jorge Luís de Araújo Rossiter. – Recife, 2019.
108f.: il.

Orientador: Bruno Tenório Ávila.
Trabalho de Conclusão de Curso (Graduação) – Universidade Federal de Pernambuco. Centro de Artes e Comunicação. Departamento de Ciência da Informação. Curso de Gestão da Informação, 2019.

Inclui referências, apêndices e anexos.

1. Gestão da Informação. 2. Base de Dados. 3. Auditoria de Dados. 4. Qualidade de Dados. 5. Extração da Informação. 6. Modelagem de Dados. 7. Completude de Dados. I. Ávila, Bruno Tenório (Orientador). II. Título.

020 CDD (22. ed.) UFPE (CAC 2019-165)

Serviço Público Federal
Universidade Federal de Pernambuco
Centro de Artes e Comunicação
Departamento de Ciência da Informação

**FOLHA DE APROVAÇÃO**

**Proposta de um processo de construção de uma base de dados a partir de documentos físicos: estudo de caso dos estágios do curso Gestão da Informação da UFPE**

(Título do TCC)

**Jorge Luís de Araújo Rossiter**
(Autor)

Trabalho de Conclusão de Curso submetido à Banca Examinadora, apresentado no Curso de Gestão da Informação, do Departamento de Ciência da Informação, da Universidade Federal de Pernambuco, como requisito parcial para obtenção do título de Bacharel em Gestão da Informação.

TCC aprovado em 11 de julho de 2019.

Banca Examinadora:

Bruno Tenório Ávila

Prof. - Orientador
Universidade Federal de Pernambuco

Sandra de Albuquerque Siebra

Prof. - Examinador 1
Universidade Federal de Pernambuco

Eduardo Garcia Wanderley

Prof. - Examinador 2
Instituto Federal de Pernambuco

## Agradecimentos

À Deus, pela grande conquista.

À minha família, pelo irrestrito apoio.

Aos funcionários do Departamento de Ciência da Informação da UFPE: Tereza, Paulo, Lucas, Cláudio e “Seu” Carlos, pela amizade, paciência e boas risadas.

Aos meus colegas de turma, pelos excelentes momentos que (re)vivi na Universidade.

Aos professores do Departamento de Ciência da Informação da UFPE, pela amizade e absoluto respeito.

A IVIA Tecnologia, pelo contínuo incentivo na busca por conhecimento, crescimento pessoal e profissional.

À Polícia Federal, especialmente ao NTI - Núcleo de Tecnologia da Informação, por ser meu bastião ético-moral, fonte de orgulho e respeito.

A Red Bull, pela cafeína e taurina, e pelas noites em claro.

Ao Google, por motivos óbvios.

## Resumo

Em um mundo cada vez mais conectado, é importante que as informações estejam disponíveis em suportes que sejam acessíveis por várias pessoas, como bases de dados, por exemplo. As bases de dados visam, principalmente, armazenar e organizar os dados para que sejam recuperados facilmente. Diante disso, muitas organizações têm realizado a digitalização dos dados, que consiste no processo de migração dos dados oriundos de documentos físicos para sistemas digitais, a fim de sustentar a manutenção do conhecimento. Este trabalho apresenta um estudo que teve como finalidade construir uma base de dados a partir de documentos físicos dos estágios de Gestão da Informação da UFPE entre 2011.1 e 2019.1. Para desenvolvimento do estudo foi realizada uma fundamentação teórica que aborda os conceitos para digitalização da informação a partir de termos de compromisso de estágio para uma base de dados. Posteriormente, a fundamentação teórica tem seu foco voltado para auditoria, melhoria e monitoramento da qualidade dos registros. Por fim, é realizado um estudo de caso, onde os termos de compromisso de estágio de Gestão da Informação da UFPE são digitalizados para a base de dados e a qualidade do mesmo é verificada.



**Pavras-chave:** Gestão da Informação. Base de Dados. Completude de Colunas Não-Obrigatórias. Completude de Relacionamento Entre Tabelas. Auditoria de Dados. Qualidade de Dados. Extração da Informação. Modelagem de Dados.

**Abstract**

In an increasingly connected world, it is important that information be available on media that are accessible to various people, such as databases. Databases are primarily designed to store and organize data so that it can be easily retrieved. Therefore, many organizations have performed the data digitization, which consists of the process of migrating data from physical documents to digital systems, in order to sustain the maintenance of knowledge. This work presents a study that had the purpose of constructing a database from the physical documents of the Stages of Information Management of UFPE between 2011.1 and 2019.1. For the development of the study was carried out a theoretical foundation that approaches the concepts for information digitization from terms of commitment of stage to a database. Subsequently, the theoretical basis is focused on auditing, improvement and monitoring of the quality of records. Finally, a case study is carried out, where the terms of commitment of the Information Management stage of UFPE are scanned into the database and the quality of the same is verified.

**Keywords:** Information Management. Data base. Completeness of Non-Mandatory Columns. Completeness of Relationship Between Tables. Data Auditing. Data Quality. Extraction of Information. Data Modeling.

## Lista de ilustrações

## Lista de quadros

## Lista de diagramas

## Lista de tabelas

## Lista de abreviaturas e siglas

| | |
|---|---|
| 1FN | Primeira Forma Normal |
| 2FN | Segunda Forma Normal |
| 3FN | Terceira Forma Normal |
| ABNT | Associação Brasileira de Normas Técnicas |
| ANSI | American National Standarts Institute (Instituto Nacional Americano de Padrões) |
| BD | Banco de Dados |
| DBA | Database Administrator (Administrador de Banco de Dados) |
| DDL | Data Definition Language (Linguagem de Definição de Dados) |
| DER | Diagrama Entidade e Relacionamento |
| ER | Entidade-Relacionamento (entity-relationship) |
| FK | Foreign Key (Chave Estrangeira) |
| FN | Formas Normais |
| GI | Gestão da Informação |
| MER | Modelo Entidade-Relacionamento |
| PK | Primary Key (Chave Primária) |
| SGBD | Sistema de Gerenciamento de Banco de Dados |
| SPARC | Standards Planning and Requirement Committee (Comitê de Planejamento e Requisitos de Padrões) |
| SQL | Structured Query Language (Linguagem de Consulta Estruturada) |
| TI | Tecnologias da Informação |
| UFPE | Universidade Federal de Pernambuco |

## Sumário

# 1 Introdução

Na Sociedade da Informação, os dados e as informações, em sua grande maioria, estão presentes na forma digital, principalmente na W*eb*. Sua aquisição, armazenamento, processamento e distribuição utilizam meios eletrônicos, como computadores, servidores de dados ou celulares. Essas novas tecnologias tem o poder de transformar a sociedade, política, cultural, social e economicamente (ANTUNES, 2008). O papel perde portanto sua importância e é substituído por sistemas que armazenam informações de forma mais eficiente, precisa e confiável. O processo de digitalização dos dados armazenados em meios físicos para arquivos digitais tem se tornado uma prática recorrente nas organizações. O excesso de informações armazenadas em documentos físicos dificulta seu armazenamento. As informações não sistematizadas dificultam seu acesso e consequentemente a geração de conhecimento. A conversão do suporte físico de dados (papel, microfilme, entre outros) para o formato digital visa dinamizar o acesso e disseminação das informações, mediante visualização dos dados a multi-usuários e consiste em transformar uma imagem ou sinal analógico em código digital. O processo propõe-se a armazenar, indexar, administrar, entre outros (SEBRAE, 2019).

Esta mudança tecnológica força as organizações a transferirem seus dados e informações para suportes digitais. Estes dados podem ser armazenados em arquivos simples (arquito de texto plano, planilha, entre outros) ou ainda em estruturas mais complexas, que tratam os dados de forma mais eficiente. Uma solução bastante utilizada é o banco de dados (BD) (ALECRIM, 2018).

Segundo Damas (2007, p.15), um BD “consiste em uma coleção de dados estruturados, organizados e armazenados de forma persistente”. BD pode ainda utilizar um *software* que possui recursos para armazenamento e manipulação eficiente de uma grande quantidade de dados. Denotada por um sistema de gerenciamento de banco de dados (SGBD) (MARTELLI; SANTANA FILHO; CABRAL, 2017). Um banco de dados pode gerenciar uma ou mais bases de dados. Bases de dados são um conjunto de dados que possuem relação entre si, organizados de forma a permitir a recuperação da informação (ALBRECHT; OHIRA, 2000).

As bases de dados são alimentadas pelos usuários ou ainda através da digitalização das informações a partir de documentos físicos. Durante o processo de digitalização, podem ocorrer vários problemas. O principal problema abordado neste trabalho foi o da qualidade dos dados resultante tanto do processo manual de extração e entrada de dados, quanto da falta de controle no preenchimento dos dados nos formulários físicos. A solução consistiu em adotar uma técnica de auditoria de dados integrado com indicadores de qualidade de dados. Em particular, utilizou-se indicadores

de completude para colunas não-obrigatórias e de um novo conceito de indicador de completude para relacionamentos entre tabelas.

## 1.1 Objetivos

O objetivo geral deste trabalho é construir uma base de dados a partir de documentos físicos dos estágios de Gestão da Informação da UFPE entre 2011.1 e 2019.1 que abordem os problemas de acurácia e completude relativos à qualidade dos dados.

Os objetivos específicos deste trabalho são:

- Desenvolver uma técnica de auditoria de dados assistida por *software* e indicadores de qualidade de dados;
- Propor um novo conceito de indicadores de completude para relacionamento de tabelas;
- Criar uma base de dados sobre os estágios do curso de Gestão da Informação da UFPE.

## 1.2 Visão geral do documento

O presente trabalho encontra-se dividido em 5 capítulos: o primeiro capítulo compreende a introdução deste trabalho. O segundo capítulo apresenta os fundamentos teóricos necessários para entendimento deste trabalho. O terceiro capítulo detalha a metodologia utilizada neste trabalho. O quarto capítulo expõe o estudo de caso realizado e os indicadores de completude utilizados. O quinto capítulo mostra a conclusão e a indicação para trabalhos futuros.

# 2 Fundamentação teórica

## 2.1 Dados e informações

Primeiramente, é importante entender a sutil diferença entre dado e informação, além da acepção usada neste trabalho.

Segundo Milani (2008, p.311), “dados podem ser definidos como um conjunto de bits (ou de caracteres em uma macrovisão) para o armazenamento de caracteres e textos no formato alfanumérico ou, até mesmo, de arquivos e imagens”. Em outras palavras, dado é um signo que não possui significado ou representatividade de forma isolada, sendo inutilizável para o sujeito que desconheça o contexto no qual o dado está inserido, de forma a não conseguir interpretá-lo (MILANI, 2008).

O dado bruto “3,14” é um número que representa uma quantidade. Pode representar o valor de uma constante ou ainda o valor monetário de algo devido à falta de contexto sobre a representatividade deste dado. Ainda que seja possível ler o que está escrito, não se pode utilizá-lo, porque é preciso conhecer seu significado. Esta é a diferença principal em relação à informação (MILANI, 2010).

De acordo com Milani (2010, p.97), “informação é a agregação de um determinado conhecimento a um dado. Uma informação pode ser interpretada, enquanto um dado apenas pode ser visualizado ou lido”. Ao manipular os dados, atribuindo-os finalidade, significado ou contexto (em outras palavras, processando os dados), tem-se a produção de informações. Segundo Sordi (2015, p.13), “A informação requer, obrigatoriamente, a mediação humana para definir o propósito a ser atendido pelo processamento de dados a ser realizado, de acordo com uma unidade de análise”. Ao definirmos que “3,14” trata-se do valor da constante $\pi$ (pi), ainda que represente algo, para muitos ainda estamos falando de um dado, haja vista que não foi definido onde e como será utilizado. Cenário diferente quando tal dado é processado para realização de um cálculo matemático com o objetivo de definir a área de um círculo. Sendo assim, é importante para o público envolvido compreender os atributos, seus valores ou intervalo de valores, que determinam as diretrizes para sobrevir uma interpretação coerente dos dados (SORDI, 2015).

O Quadro 1 apresenta as principais diferenças entre dados e informação de acordo com cada especificidade relatada.

Quadro 1 – Comparativo entre dado e informação

| CARACTERÍSTICAS | DADO | INFORMAÇÃO |
|---|---|---|
| Estruturação, captura e transferência | Fácil | Difícil |
| Principal requisito para sua geração | Observação | Interpretação concensual |
| Natureza | Explícita | Predominantemente explícita |

Fonte: adaptado de SORDI (2015, p.17)

Os dados podem ser facilmente estruturados, como vetores, listas, pilhas, entre outros (RICARTE, 2008). Sua captura também é facilitada, bastando apenas acesso ao dado e sua observação, seja por texto, imagem estática, vídeo, áudio, anotação, seja por qualquer suporte que se faça entender algum dado. A transferência pode ser realizada através dos suportes citados anteriormente.

Já a informação necessita que todos os envolvidos entendam a conjuntura no qual o dado está inserido, dificultando sua estruturação, captura e transferência, determinando uma natureza predominantemente explícita. Sordi (2015) explica tais características com o exemplo de um livro escrito em japonês. Para que o sujeito entenda o que está escrito ali, ele deve saber e entender o idioma do livro, caso contrário o conteúdo não passa de dados sem sentido algum.

## 2.2 Banco de dados

Banco de dados (BD) é um conjunto de dados reunidos de forma organizada, consistente, protegida e acessível em tempo hábil (ALECRIM, 2018). Seu objetivo é tornar a vida dos usuários mais fácil, prática, precisa, rápida e confiável, facilitando a obtenção dos dados no processo de recuperação da informação a partir de aplicações.

Historicamente, o banco de dados evoluiu de antigas fichas de controle em papel, organizados em pastas e armazenados em arquivos físicos, para arquivos digitais. Estes eram acompanhados de *softwares* simples, que realizavam algumas operações como cadastro, alteração, exclusão e consulta dos registros nos referidos arquivos, que poderiam ser de texto plano, por exemplo. Ainda que tenha facilitado a consulta, era tão somente uma automatização dos arquivos físicos. Entretanto, com a massificação do uso de computadores nos anos 1950, surgiu a necessidade de gerenciar mais volume de dados, além de sua complexidade. Os sistemas de gerenciamento de banco de dados (SGBD) deveriam ser capazes de incluir, alterar, indexar e manter dados íntegros e seguros. Uma grande quantidade de modelos de armazenamento foram apresentadas; entre eles: os pré-relacionais, como o hierárquico e de rede, nos anos

1960; o relacional, nos anos 1970; o orientado a objetos, nos anos 1980; e o NoSQL, nos anos 2000 (ALVES, 2013; AMARAL, 2016).

O modelo de banco de dados relacional proposto por Codd (1970) tem como objetivo criar um modelo sucessor aos hierárquico e de rede, mantendo a integridade das transações e reduzindo a redundância dos dados, no qual os dados são representados de forma bidimensional, utilizando estruturas específicas, como tabelas (AMARAL, 2016; MARTELLI; SANTANA FILHO; CABRAL, 2017). O banco de dados relacional baseia-se na álgebra relacional. Para Martelli, Santana Filho e Cabral (2017, p.36), as tabelas são:

> [...] compostas por linhas e colunas. Cada linha é composta por valores denominados "campos", e cada campo possui um nome. As colunas representam um conjunto de informações extraídas de várias linhas, mas de apenas uma única coluna. [...] A grande vantagem é que esse modelo permite o cruzamento de informações.

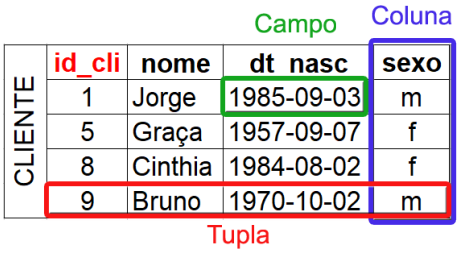
Segundo Damas (2007, p.47):

> A estrutura de dados utilizada no modelo relacional é a **relação** [...]. Uma relação é uma estrutura bidimensional que obedece a um esquema determinado e possui zero ou mais instâncias. O esquema de uma relação é constituído por um ou mais atributos que traduzem o tipo de dados a ser armazenado. Cada instância de uma relação é chamada **tupla.**

Em oposição aos registros vinculados (como os modelos hierárquico e de rede), Codd (1970) sugeriu a criação de tabelas, de estrutura fixa e organizada. A normalização (mais detalhes na Seção 2.4) distribui os dados entre várias tabelas, variando de acordo com o tamanho do arranjo armazenado. Uma tabela deve possuir um identificador único, a chamada **chave primária** (*PK - Primary Key*). A chave primária pode ter um valor semântico ou um valor incremental gerado pelo próprio SGBD. A **relação entre tabelas** se dá entre a chave principal da tabela de origem e a **chave estrangeira** (*FK - Foreign Key*), na tabela relacionada (AMARAL, 2016).

Na Figura 1, cada coluna é identificada com um nome diferente na primeira linha. "id_cli", "nome", "dt_nasc" e "sexo" correspondem aos **atributos** da tabela. O "id_cli" é o identificador único de cada registro, ou seja, a chave primária. Cada linha abaixo dos atributos corresponde a instância da ocorrência de um cliente diferente, também chamada de **tupla**. Por exemplo, o cliente de identificador 9 chama-se Bruno, nasceu em 2 de outubro de 1970 e é do sexo masculino. A interseção de uma linha e coluna formam um **campo**. Neste caso, a data de nascimento do cliente com id_cli 1 é 3 de setembro de 1985 (SCHARDOSIM; CURY, 2015).

**Figura 1 – Exemplo de modelo relacional**




Fonte: elaborado pelo autor, 2019

"A grande vantagem é que esse modelo permite o cruzamento de informações" (MARTELLI; SANTANA FILHO; CABRAL, 2017, p.36). Damas (2007, p.50), ainda complementa:

> Com a utilização deste modelo, de repente aquilo que parecia tão complexo tornou-se algo tão simples, pois se podiam fazer alterações no esquema do banco de dados sem afetar a capacidade do sistema em manipular os dados, pois neste modelo:
>
> Toda informação é armazenada sob a forma de tabelas bidimensionais.
>
> As relações são estabelecidas através de colunas (campos) comuns em tabelas distintas [. . . ].

O modelo relacional mostrou-se um grande sucesso, sendo utilizando em diversas categorias de negócios e plataformas computacionais. Ganhou implementações, tanto comerciais como *open source* (gratuitas). As modernizações do modelo tornaram esses sistemas maduros, estáveis e eficientes, além de acrescentar recursos importantes, como *backup* incremental, replicação, operação em *clusters*, tolerância a falhas, distribuição de carga, entre outros (AMARAL, 2016).

### 2.2.1 Linguagem SQL

A linguagem SQL (*Structured Query Language*) é utilizada para manipular dados armazenados nos principais bancos relacionnais disponíveis no mercado. Surgiu nos anos 1970 e corroborava a abstração do modelo de dados relacional. Várias empresas produziram suas próprias versões da linguagem e, devido ao seu grande crescimento, levou o ANSI[1] a padronizar a linguagem em meados dos anos 80 (VASCONCELOS et al., 2002; MILANI, 2008).

A principal vantagem do SQL reside na padronização, tornando-a segura para utilização em bancos de dados. Outra vantagem é que ela não é procedural, ou seja, realizará a tarefa independente da forma que a fará, tornando os procedimentos mais

[1] American National Standarts Institute (Instituto Nacional Americano de Padrões) - equivalente norte-americano à Associação Brasileira de Normas Técnicas (ABNT).

fáceis de serem assimilados por quem deseja aprender a linguagem. Apesar de ser um padrão, a linguagem pode sofrer algumas alterações e/ou acréscimos dependendo do SGBD utilizado. *Access, SQL Server, Oracle, DB2, Postgres*, são só alguns exemplos de SGBDs que utilizam a linguagem SQL (GOMES, 2018).

Damas (2007, p.131) orienta que a linguagem SQL possui 3 sublinguagens:

- DML - *Data Manipulation Language* (SELECT, INSERT, UPDATE, DELETE, etc.)
- DDL - *Data Definition Language* (CREATE, ALTER, DROP, etc.)
- DCL - *Data Control Language* (GRANT, REVOKE, etc.)

As linguagens do tipo DDL, também chamadas de Linguagens de Definição de Dados, criam e alteram os formatos de atributos (campos) e entidades (tabelas) em um banco de dados. É possível ainda criar relacionamento entre entidades e definir domínios para os atributos. Não realiza a direta manipulação dos dados. Já as Linguagens de Manipulação de Dados (DML), realizam a manipulação dos dados contidos no banco de dados. A partir dela que são realizadas as consultas, inclusões, alterações e exclusões de registros. Finalmente, a DCL, sigla para Linguagem de Controle de Acesso, fiscaliza o controle de usuários e acessos ao banco de dados (MILANI, 2008).

Várias aplicações utilizam o SQL para comunicação com o banco de dados. Assim, é possível executar ações como inserir, alterar, consultar ou excluir dados. Alguns recursos disponibilizados pelo SQL são (MILANI, 2010):

- **Consultas (*queries*):** capacidade de obter dados do banco de dados para uso em aplicativos ou para satisfazer uma busca realizada pelo usuário.
- **Atualizações:** provê a capacidade de atualizar informações em qualquer estrutura no banco de dados a partir de aplicações que possuam acesso ao mesmo. Isto inclui exclusão, manutenção e inserção de dados.
- **Filtros e ordenações:** a linguagem permite que os dados retornados a partir de uma consulta sejam ordenados e formatados de acordo com os critérios da consulta.

A linguagem efetiva os conceitos do modelo relacional, que é largamente admitido e proposto. Portanto, reduz as incompatibilidades e custos já que não se torna necessário optar por arquiteturas proprietárias, diminuindo o custo adicional em contratá-las e o esforço humano dos envolvidos (DAMAS, 2007).

No Quadro 2 está disponível um exemplo simples de tabela. Se faz necessária a compreensão da linguagem SQL para criar e preencher as tabelas.

Quadro 2 – Exemplo de criação de tabela em linguagem SQL

```
CREATE TABLE agencias
(
  nome_fantasia character varying(255) NOT NULL,
  id_agencia integer NOT NULL DEFAULT nextval('agencias_id_agencia_seq'::regclass),
  cnpj character varying(14) NOT NULL,
  auditado timestamp without time zone,
  CONSTRAINT pk_agencias PRIMARY KEY (id_agencia)
)
```

Fonte: elaborado pelo autor, 2019

A instrução *CREATE TABLE*, solicita a criação de uma tabela com o nome “agencias”. Todos os atributos e chaves da tabela estão entre as linhas 2 e 8. O primeiro atributo chama-se “nome_fantasia”, é do tipo cadeia de caracteres, tem capacidade de armazenamento para 255 caracteres, é de preenchimento obrigatório, devido ao *NOT NULL* e armazena o nome fantasia de uma empresa. O segundo atributo, “id_agencia” é do tipo inteiro (número), é de preenchimento obrigatório e seu valor padrão (*DEFAULT*) é um número sequencial automático que incrementa o valor cada vez que uma nova inserção é realizada. Já o atributo “cnpj” cria uma cadeia de caracteres com 14 posições para armazenar o número do cadastro de pessoa jurídica. Para armazenar a data e hora em que os dados daquela tupla foram auditados, foi criado a coluna “auditado” que é do tipo data e hora sem fuso horário. Finalmente, *CONSTRAINT* cria as restrições para colunas no banco de dados, como regras. Neste caso, é criada uma chave primária a partir do comando *PRIMARY KEY* chamada “pk_agencias” que utiliza o valor do atributo “id_agencia” para identificar aquela tupla.

É importante entender algumas instruções da linguagem SQL de modo a realizar uma série de operações no banco de dados. Aqui foram escolhidos os mais importantes para este trabalho, baseados em Milani (2008), Milani (2010).

A instrução *INSERT* serve para inserir dados na tabela. Pode ser utilizado de algumas maneiras, mas a sintaxe básica se resume ao descrito abaixo:

```
INSERT INTO nome_tabela (lista_campos) VALUES (lista_dados)
```

*INSERT INTO* solicita a inserção de dados na tabela “nome._tabela” e especifica o nome das colunas onde os dados serão inseridos através do parâmetro “lista_campos”. Os valores dos dados a serem inseridos, que devem estar na mesma sequência dos campos das tabelas, são repassados pela instrução *VALUE* e correspondem ao parâmetro “lista_dados”.

Por exemplo, na inserção descrita no Quadro 3, a tabela “agencia” tem as colunas “nome_fantasia”, “cnpj” e “auditado” preenchidas pelos valores “cie”, “00394494003313” e “NULL”.

Quadro 3 – Exemplo de inserção de dados em uma tabela

```
INSERT INTO  agencia (nome_fantasia, cnpj, auditado)
VALUES ('cie', '00394494003313', NULL)
```

Fonte: elaborado pelo autor, 2019

A instrução mais importante do SQL é o *SELECT*. Ele é utilizado para realização de consultas no banco de dados, retornando os dados solicitados em formato de tabela. Serve para verificar o resultado da aplicação das demais instruções como alteração, inserção e exclusão (MILANI, 2008). O Quadro 4 apresenta a sintaxe básica do *SELECT*.

Quadro 4 – Sintaxe da instrução SELECT

```
SELECT campo 1, campo 2, ..., campo n, *
FROM tabela 1, ..., tabela n
[WHERE condição]
[GROUP BY ...]
[ORDER BY ...]
```

Fonte: elaborado pelo autor, 2019

Para utilizar a instrução, é necessário informar os campos que desejamos consultar. Após a instrução *SELECT*, os nomes dos campos (campo 1 ... campo n) devem ser informados ou substituídos por "*", caso queira consultar toda a tabela. Toda consulta é originada de algum lugar, então utiliza-se *FROM* para informar a tabela de origem da consulta, tendo o nome da tabela em seguida. É possível consultar várias tabelas ao mesmo tempo, bastanto separar seus nomes por vírgulas. Os comandos entre colchetes são opcionais e serão explicados no decorrer desta seção.

O comando abaixo apresenta uma consulta básica. Primeiramente, nota-se que a consulta será realizada em todos os campos da tabela, pois os nomes dos campos foram substituídos por asterisco ("*"). A tabela a ser consultada chama-se "agencia".

```
SELECT * FROM agencia
```

A consulta retorna todas as tuplas cadastradas nos atributos da tabela "agencia", conforme a Figura 2.

**Figura 2 – Resultado da instrução SELECT na tabela “agencia”**

| nome_fantasia<br>character varying(255) | id_agencia<br>integer | cnpj<br>character varying(14) | auditado<br>timestamp without time zone |
|---|---|---|---|
| CIEE - CENTRO DE INTEGRAÇÃO EMPRESA-ESCOLA DE PERNAI | 1 | 10998292000157 | 2018-09-20 19:36:42.724 |
| ABRE - AGÊNCIA BRASILEIRA DE EMPREGO E ESTÁGIO | 3 | 10329228000183 | 2018-10-30 20:33:59.683 |
| IEL - INSTITUTO EUVALDO LODI - NÚCLEO REGIONAL DE PI | 2 | 11000361000154 | 2018-11-12 20:15:42.839 |
| ASSESPRO - ASSOCIAÇÃO DAS EMPRESAS BRASILEIRAS DE TI | 19 | 11546934000140 | 2018-12-04 17:39:26.003 |
| CIAPE - CENTRO INTEGRADO DE APOIO PEDAGÓGICO E EDUCA | 4 | 08588210000154 | 2018-12-04 19:36:03.289 |
| NUDEP - NÚCLEO DE DESENVOLVIMENTO PROFISSIONAL LTDA | 21 | 06195488000136 | 2018-09-20 19:08:06.312 |

Fonte: captura de tela elaborada pelo autor no PgAdmin III, 2019

A cláusula *WHERE* é utilizada para refinar a consulta e obter dados de campos específicos. *WHERE*, como sua tradução sugere, define ONDE o dado poderá ser encontrado. A cláusula *WHERE* pode (e deve) ser utilizada em outros comandos. O Quadro 5 apresenta uma forma de utilização.

Quadro 5 – uso da instrução SELECT com cláusula WHERE

```
SELECT *
FROM agencia
WHERE id_agencia = 2
```

Fonte: elaborado pelo autor, 2019

O comando retorna todas as tuplas cadastradas nos atributos da tabela “agencia” que o campo “id_agencia” tenha o valor 2 (dois), como na figura 3.

**Figura 3 – Resultado da instrução SELECT com cláusula WHERE**

| nome_fantasia<br>character varying(255) | id_agencia<br>integer | cnpj<br>character varying(14) | auditado<br>timestamp without time zone |
|---|---|---|---|
| IEL - INSTITUTO EUVALDO LODI - NÚCLEO REGIONAL DE PI | 2 | 11000361000154 | 2018-11-12 20:15:42.839 |

Fonte: captura de tela elaborada pelo autor no PgAdmin III, 2019

Em alguns casos, pode ser desejável o retorno de uma consulta sem repetição de valores. Neste caso, aconselhamos utilização da cláusula *GROUP BY*, de acordo com o Quadro 6.

Quadro 6 – Uso da instrução SELECT com cláusula GROUP BY

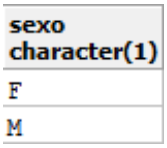
```
SELECT sexo
FROM estudantes
GROUP BY sexo
```

Fonte: elaborado pelo autor, 2019

O comando seleciona o atributo “sexo” a partir da tabela “estudantes” e os agrupa de acordo com o próprio atributo “sexo”. O resultado será retornado conforme

a Figura 4. Note que não há repetição de valores. Este comando é bastante utilizado para encontrar inconsistência nos dados e facilitar a normalização dos mesmos.

**Figura 4 – Resultado da instrução SELECT com cláusula GROUP BY**



| sexo character(1) |
|---|
| F |
| M |

Fonte: captura de tela elaborada pelo autor no PgAdmin III, 2019

É possível ainda utilizar a função agregada *COUNT*. Esta função agrega um conjunto de valores e retorna um resultado, realizando uma contagem (MONTEIRO, 2018). O Quadro 7 ilustra uma consulta em que se faz uma contagem do campo "sexo" na tabela "estudantes", separando-os por sexo (*GROPU BY* sexo). Caso a linha "GROUP BY sexo" não existisse, a quantidade total de estudantes seria exibida.

Quadro 7 – Uso da função agregada COUNT em conjunto com GROUP BY

```
SELECT COUNT (sexo)
FROM estudantes
GROUP BY sexo
```

Fonte: elaborado pelo autor, 2019

Pode ser desejável retornar uma consulta com valores ordenados por uma ou mais determinadas colunas. Para essas situações, devemos utilizar a cláusula *ORDER BY*. O Quadro 8 ilustra um exemplo de uso da cláusula.

Quadro 8 – Uso da instrução SELECT com cláusula ORDER BY

```
SELECT *
FROM agencia
ORDER BY cnpj
```

Fonte: elaborado pelo autor, 2019

O comando retornará todas as tuplas cadastradas nos atributos da tabela "agencia", ordenados pelo campo "cnpj", apresentados na Figura 5.

**Figura 5 – Resultado da instrução SELECT com cláusula ORDER BY**

| nome_fantasia<br>character varying(255) | id_agencia<br>integer | cnpj<br>character varying(14) | auditado<br>timestamp without time zone |
|---|---|---|---|
| NUDEP - NÚCLEO DE DESENVOLVIMENTO PROFISSIONAL LTDA | 21 | 06195488000136 | 2018-09-20 19:08:06.312 |
| CIAPE - CENTRO INTEGRADO DE APOIO PEDAGÓGICO E EDUC | 4 | 08588210000154 | 2018-12-04 19:36:03.289 |
| ABRE - AGÊNCIA BRASILEIRA DE EMPREGO E ESTÁGIO | 3 | 10329228000183 | 2018-10-30 20:33:59.683 |
| CIEE - CENTRO DE INTEGRAÇÃO EMPRESA-ESCOLA DE PERNAI | 1 | 10998292000157 | 2018-09-20 19:36:42.724 |
| IEL - INSTITUTO EUVALDO LODI - NÚCLEO REGIONAL DE PI | 2 | 11000361000154 | 2018-11-12 20:15:42.839 |
| ASSESPRO - ASSOCIAÇÃO DAS EMPRESAS BRASILEIRAS DE TI | 19 | 11546934000140 | 2018-12-04 17:39:26.003 |

Fonte: captura de tela elaborada pelo autor no PgAdmin III, 2019

Parecido com a instrução *INSERT*, o *UPDATE*, ao invés de inserir um novo dado na tabela, realiza a alteração de algum campo já cadastrado. O Quadro 9 indica a sinxate da instrução.

Quadro 9 – Sintaxe da instrução UPDATE

```
UPDATE nome_da_tabela
SET nome_do_campo = novo_valor
WHERE nome_do_campo = valor
```

Fonte: elaborado pelo autor, 2019

A instrução *UPDATE* deve ser sucedida pelo nome da tabela que sofrerá as alterações, neste caso, “nome_da_tabela”. Em seguida, *SET* define qual o nome do campo (nome_do_campo) que sofrerá alteração e o novo valor (novo_valor) que será atribuído ao campo. *WHERE* define que condição deverá ser satisfeita para que a alteração aconteça.

Definidos os campos, caso a cláusula *WHERE* seja satisfeita, um novo valor é configurado para o campo especificado.

A instrução do Quadro 10 primeiramente tenta localizar a tupla em que o valor do campo “id_agencia” esteja valorado em 1. Caso a tupla exista, terá seu nome anterior alterado para “Teste”.

Quadro 10 – Uso da instrução UPDATE

```
UPDATE estagio
SET nome_fantasia = 'Teste'
WHERE id_agencia = 1
```

Fonte: elaborado pelo autor, 2019

Assim como sua tradução sugere, a instrução *DELETE* é utilizada para apagar um dado específico. Deve-se tomar muito cuidado ao utilizar esta instrução, pois sem a cláusula WHERE pode apagar todos os dados da tabela.

Quadro 11 – Sintaxe da instrução DELETE

```
DELETE FROM tabela
WHERE nome_do_campo = valor_para_exclusão
```

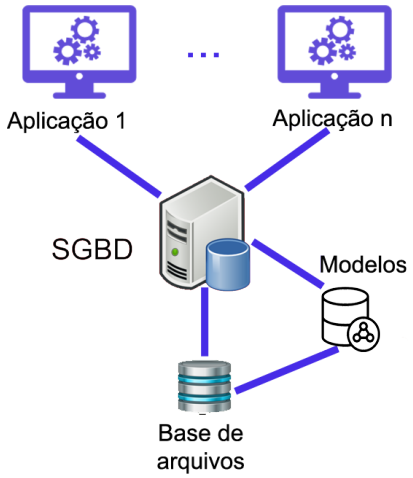
Fonte: elaborado pelo autor, 2019

*DELETE FROM* define o nome da tabela (nome_da_tabela) que terá algum registro removido. *WHERE* define que condição deverá ser satisfeita para que a remoção ocorra. Caso esta condição seja satisfeita, registro (valor_para_exclusão) do campo (nome_do_campo) é removido da tabela.

### 2.2.2 Sistema de gerenciamento de banco de dados

SGBD é um conjunto de programas de computador que faz a interligação entre o usuário e os dados que estão fisicamente armazenados em um banco de dados. A organização, armazenamento e pesquisa dos dados, além do fornecimento de todos os serviços de acesso, ficam a cargo do SGBD e o usuário pode ser uma pessoa ou uma aplicação (DAMAS, 2007). A partir da criação das estruturas e inserção dos dados, os dados são armazenados em um ou mais arquivos que podem ser lidos e/ou manipulados pelo *software*. Este armazenamento de arquivos resultante do SGBD pode ser chamado de base de dados (RIBEIRO, 2018). A Figura 6 apresenta o funcionamento de um SGBD.

**Figura 6 – Funcionamento de um SGBD**




Fonte: elaborado pelo autor, 2019

Um SGBD é composto por módulos que executam tarefas específicas (DAMAS, 2007):

- **Gerenciador de arquivos:** executa a gestão do espaço em disco e das estruturas utilizadas para representar o dado gravado.

- **Gerenciador de banco de dados:** promove a troca de dados entre o nível dos dados e o nível das aplicações e consultas (*queries*).
- **Processador de consultas:** realiza a tradução dos comandos da consulta - *queries* - (que estão em linguagem específica) para instruções de baixo nível que sejam inteligíveis pelo gerenciador de banco de dados.
- **Pré-compilador:** converte os comandos da linguagem utilizada (*SQL*, por exemplo) em chamadas para procedimentos/funções da linguagem de suporte.
- **Compilador DDL (*Data Definition Language*):** converte as instruções da linguagem DDL no conjunto de tabelas que contêm os metadados armazenados em um dicionário de dados. Mais detalhes sobre DDL na Seção 2.2.1.

Visando estabelecer um padrão para todos os fornecedores de bancos de dados, a ANSI/SPARC definiu uma arquitetura (Figura 7) para que SGBDs pudessem ser utilizados por diferentes categorias de usuários, respeitando suas características. São 3 níveis autônomos, cada um com sua visão do banco de dados em seu próprio nível de abstração (DAMAS, 2007). Para Pillou (2017):

> [...] o nível interno (ou físico), que define a maneira pela qual são armazenados os dados e os métodos para acessá-los; o nível conceitual, também chamado de MCD (Modelo Conceitual dos Dados) ou MLD (Modelo Lógico dos Dados), que define a disposição das informações no banco de dados; e o nível externo, que define as visualizações dos usuários.

Destarte, este modelo de arquitetura permite **independência física**, em que as mudanças realizadas no nível interno não afetam o nível conceitural; a **independência lógica**, onde as mudanças realizadas no esquema conceitual, não alteram, necessariamente o esquema externo; e a i**ndependência de programas/dados**, nas quais alterações na estrutura de dados ou sua implementação física não forcem alterações no nível da aplicação (DAMAS, 2007).

**Figura 7 – Arquitetura ANSI/SPARC**


Fonte: adaptado de DAMAS (2007)

Após alguns anos desde a criação do modelo relacional, Codd (1970) estabeleceu 13 regras que deveriam ser satisfeitas a fim que um SGBD seja considerado do tipo relacional (DAMAS, 2007, p.51):

> 1 - Capacidade relacional total: Um SGBD relacional tem de ser capaz de gerenciar os seus bancos de dados recorrendo unicamente às suas capacidades relacionais.
>
> 2 - Representação da informação em tabelas: Em um SGBD relacional, todos os dados, incluindo o próprio dicionário de dados, são armazenados de uma só forma, em tabelas bidimensionais.
>
> 3 - Garantia de acesso: Cada dado armazenado no banco de dados deve poder ser acessado logicamente pela combinação de: - Nome da tabela onde está armazenado; - valor da chave primária; - coluna (atributo).
>
> 4 - Suporte sistemático de valores *NULL*: Os valores nulos (*NULL*) são suportados para representar informação não-disponível ou não-aplicável, independentemente do domínio dos respectivos atributos. O valor *NULL* é diferente de zero e diferente da cadeia de caracteres (*string*) vazia. Pode ser atribuído a qualquer coluna, independentemente do tipo de dados que esta suporta.
>
> 5 - Catálogo ativo e disponível: Os metadados são definidos, representados e acessados da mesma forma que os próprios dados. Os metadados[2] (descrição do banco de dados), tal como os dados, são armazenados em tabelas no banco de dados. Assim, também os metadados podem ser acessados usando a mesma linguagem utilizada para acesso aos dados.

[2] "Os metadados oferecem uma maneira de classificar, organizar e caracterizar dados ou conteúdo" (PISCITELLO, 2016, p.1).

> 6 - Listagem de acesso aos dados: Deve existir pelo menos uma linguagem ou sublinguagem suportada pelo SGBD relacional, com uma sintaxe bem definida e compreensível, na qual são suportadas as operações sobre dados. Apesar de um SGBD relacional poder suportar várias linguagens, deverá existir pelo menos uma linguagem com as seguintes características: - Definição de dados; - definição de *views*; - manipulação de dados, com possibilidade de utilização interativa ou em programas de aplicação; - definição de regras de integridade; - definição de autorização de acesso; - controle de transações (*COMMIT*, *ROLLBACK*).
>
> 7 - Atualização de *views*: As atualizações de dados feitas em uma *view* (atualizável) devem ser repercutidas nas tabelas originais do banco de dados a que pertencem.
>
> 8 - Operações de alto nível: Um SGBD relacional deve ser capaz de tratar uma tabela (base ou *view*) como se fosse um simples operando, tanto em operações de consultas como de inserção e atualização. As operações de inserção, alteração e remoção devem ser orientadas para o processamento de conjuntos por comandos de alto nível e não registro a registro.
>
> 9 - Independência física dos dados: Alterações na organização física dos arquivos do banco de dados ou nos métodos de acesso a esses arquivos (nível interno) não devem afetar o nível conceitual.
>
> 10 - Independência lógica dos dados: Alterações no esquema do banco de dados (nível conceitual), que não envolvam remoção de elementos, não devem afetar o nível externo.
>
> 11 - Independência da integridade: As regras e restrições de integridade devem poder ser especificadas em uma linguagem relacional, independentemente dos programas de aplicação, e armazenadas no diretório de dados.
>
> 12 - Independência (transparência) de localização: O fato de um banco de dados estar centralizado em uma máquina, ou distribuído por várias máquinas, não deve repercutir ao nível da manipulação dos dados.
>
> 13 - Não-subversão: Se existir no sistema uma linguagem de mais baixo nível (*record oriented*) ou recursos de uma linguagem que permitam o processamento de arquivos em baixo nível, o SGBD relacional não deverá permitir a violação das restrições de integridade e segurança. O SGBD relacional deve impedir que os dados sejam manipulados por acessos feitos por partes externas a ele através de linguagens de mais baixo nível.

Para este trabalho foi escolhido o PostgreSQL (www.postgresql.org). É *open-source*, ou seja, é de código aberto, no qual é permitido estudá-lo, modificá-lo e distribuí-lo para qualquer finalidade e a qualquer pessoa. Outra vantagem é que, além de fazer uso do padrão SQL, oferece alguns recursos extras, como gatilhos (*triggers)*. Destarte, permite ao usuário utilizá-lo para outros propósitos, permitindo adição de novos tipos de dados, funções, operadores, funções agregadas, métodos de indexação e linguagens procedurais. também conta com outros recursos, como subconsultas, identificadores entre aspas, *type casting*, entre outros (ATANAZIO, 2018). Todos estes recursos podem ser programados em algumas linguagens de programação, como C, SQL, plSQL. As linguagens PlpgSQL e SQL foram usadas neste trabalho. Alguns recursos extras deste trabalho serão discutidos nesta seção.

Historicamente, o PostgreSQL surgiu do projeto POSTGRES, da Universidade de Berkeley (Califórnia - EUA). A primeira versão estável foi lançada em 1989 e

passou por diversas atualizações até seu código ser adquirido pela *Illustra Information Technologies,* que se fundiu com outra empresa de tecnologia, pertencente à IBM. Sua primeira grande mudança ocorreu em 1994. Seu projeto anterior foi alterado e seu novo projeto trazia grande vantagem, que era a incorporação da linguagem SQL. Destarte, o programa foi compatibilizado com o padrão ANSI C, tornando-o portável para esta plataforma. Em 1996 seu nome mudou para o atual PostgreSQL (MILANI, 2008). A versão escolhida para este trabalho é a 11.2.

Foram necessárias duas ferramentas que servirão como *interface* de administração entre o SGBD e o administrador do banco de dados neste trabalho: phpPgAdmin (http://phppgadmin.sourceforge.net/doku.php) e pgAdmin (www.pgadmin.org). O phpPgAdmin é uma ferramenta de administração *web* para PostgreSQL. É voltado para *Database Administrators* (administradores de banco de dados) - DBA, novatos e serviços de hospedagem. Já o pgAdmin é uma das plataformas de desenvolvimento e administração gratuita para PostgreSQL mais utilizadas no mundo (PGADMIN DEVELOPMENT TEAM, 2019; PHP, 2019).

As funções (*functions*) são códigos armazenados na estrutura do banco de dados que podem ser executados a qualquer momento e seu objetivo é processar algum dado. As funções realizam pequenas operações auxiliares que sejam invocadas em uma transação. Destarte, executam atividades repetitivas e rotineiras, podendo ser utilizadas em diversas transações diferentes, ficando responsáveis pelo tratamento de textos e variáveis (MILANI, 2008).

O Quadro 12 revela a sintaxe de criação de uma função chamada “nome” a partir da instrução *CREATE FUNCTION*. Caso alguma função de mesmo nome já exista, pode-se utilizar a intrução *REPLACE* para definir sua substituição. Os argumentos são parâmetros repassados para utilização interna da função. *RETURNS* determina que a função retornará um valor. O uso de *AS $$* indica uma delimitação de *string*, ou seja, possibilita o uso de caracteres especiais no corpo da função. *DECLARE* indica o bloco onde as variáveis locais da função serão utilizadas. Estas variáveis só existirão durante a execução da função. O trecho entre *BEGIN* e *END* determina o corpo da função, enquanto *RETURN* retorna um valor de acordo com o tipo declarado no cabeçalho da função. Já *END* indica o fim do corpo da função. Finalmente, *LANGUAGE* determina a linguagem da função (MILANI, 2008).

Quadro 12 – Sintaxe para criação de uma função

```
CREATE [OR REPLACE] FUNCTION nome(argumentos) RETURNS tipo_de_retorno AS $$
DECLARE
    -- bloco de declaração de variáveis
BEGIN
    -- corpo da função
    [RETURN variável;]
END;
$$ LANGUAGE linguagem;
```

Fonte: elaborado pelo autor, 2019

O Quadro 13 mostra a criação da função “tr_upper_agencias”. Esta função retorna um gatilho que coloca o “nome_fantasia” em caixa alta.

Quadro 13 – Criação de uma função

```
CREATE [OR REPLACE] tr_upper_agencias () RETURNS TRIGGER AS $$
BEGIN
    NEW.nome_fantasia := UPPER (NEW.nome_fantasia);
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

Fonte: elaborado pelo autor, 2019

*Stored Procedure* (procedimento armazenado) é o recurso que visa armazenar um conjunto de instruções no servidor para processamento de determinados valores e ações, de modo a atingir um resultado. No caso do PostgreSQL, o único recurso que diferencia as funções é o conceito utilizado, já que os procimentos dividem o mesmo espaço na aplicação. Especificamente, as *stored procedures* executam transações completas, como um cadastro completo de usuário, e não somente formatação de algum dado, como as funções. Uma *stored procedure* pode (e deve) utilizar funções para validação de dados cadastrados, por exemplo. Destarte, traz uma série de benefícios ao banco de dados e aplicações, apresentados a seguir (MILANI, 2008):

- **Centralização:** quando o banco de dados é acessado por diversas aplicações, escritas em várias linguagens, estando cada sistema sob a responsabilidade de uma equipe diferente, os códigos de interação dos programas com o BD podem ser utilizados de forma centralizada através das *stored procedures*. Então, caso haja necessidade de manutenção nos códigos, basta ajustar apenas o procedimento, já que as aplicações o invocam para obtenção dos resultados.
- **Segurança**: ainda no contexto anterior, em que o BD é acessado por diversas aplicações e equipes, o fato da *stored procedure* estar armazenado de

forma interna ao banco de dados assegura que quaisquer informações do mesmo encontrar-se-ão disponíveis somente através dos procedimentos de armazenamento. Isto garante que nenhum código malicioso ou mal produzido seja executado.

- **Velocidade**: blocos de códigos, armazenados de forma nativa no servidor, e não uma única expressão SQL completa, podem tornar o processamento mais rápido. Ao invés de serem enviadas pelas mais diversas aplicações (consequentemente ocupando mais espaço na rede), apenas as chamadas são enviadas.
- **Suporte a transações**: com a utilização destes blocos de códigos, é possível assegurar que as transações serão completamente executadas, ou as alterações realizadas até ali sejam desfeitas. Isso acontece porque a transação segue o princípio da atomicidade, abordado na Seção 2.2, ou seja, caso algum problema ocorra (uma indisponibilidade do banco de dados, dentre outros fatores que possam coibir sua execução), as operações não serão executadas de forma icompleta.

Transação, "[. . . ] é uma sequência de operações executadas como uma única unidade lógica de trabalho" (BARROS, 2016). As operações de transação de um SGBD atendem um conjunto de propriedades, denominadas pela abreviação ACID - *Atomicity, Consistency, Isolation, Durability* (Atomicidade, Consistência, Isolamento e Durabilidade (BARROS, 2016; AMARAL, 2016):

- **Atomicidade:** em uma transação que contenha duas ou mais operações de banco de dados, caso uma delas não ocorra, a transação não será executada, garantindo sua indivisibilidade e irredutibilidade. Um procedimento de atualização do banco de dados, por exemplo, não pode ser executado parcialmente, ou dados desatualizados poderão ser utilizados por outros usuários.
- **Consistência:** garante que a execução da transação saia de um estado consistente para outro igualmente consistente. Caso alguma falha venha a ocorrer, o dado é retornado ao valor anterior à execução da transição. O estado consistente é satisfeito quando as regras de integridade dos dados são respeitadas. Por exemplo, evita uma duplicidade de campos que devem ser únicos para cada entidade; ou ainda, não permite inserção de valores inválidos, como o número do CPF "000.000.000-00".
- **Isolamento:** uma transação em execução deverá permanecer separada das demais transações até que a mesma tenha finalizado. Esta situação é muito comum quando um usuário está alterando um registro e que outro também

deseja fazer o mesmo. Até que a transação do primeiro usuário tenha acabado, o registro permanece inacessível para o segundo usuário.

- **Durabilidade:** garante que os dados validados pelo sistema sejam registrados, ainda que imediatamente ocorra alguma falha no banco de dados, travamento, falta de energia elétrica, entre outros eventos. É importante que esses dados sejam registrados em um suporte com memória não volátil.

O Quadro 14 apresenta um exemplo de *stored procedure* de um cadastro de agência de estágios chamado “sp_agencia” recebendo os valores de texto por parâmetro. O valor retornado será um inteiro. O procedimento utiliza as variáveis internas “v_cnpj”, “v_nome_fantasia” e “v_id_agencia”, a partir da instrução *DECLARE* (linha 3). A transação de cadastro inicia com o comando *BEGIN* (linha 7) e termina com *END* (linha 16). Dentro da transação, é executado um comando *SELECT* que armazena em “v_id_agencia” o valor de uma “id_agencia” da tabela “agencias”, cujo cnpj (agencias.cnpj), seja igual ao repassado por parâmetro para o procedimento. Em seguida, é verificado se “v_id_agencia” recebeu algum valor, ou seja, se algum cnpj foi encontrado. Caso positivo, retorna-se o identificador do registro da agencia, através de “v_id_agencia”; caso contrário, uma nova inserção será realizada (linha 12) e o valor do índice do registro recém efetuado é retornado. Por fim, se algum problema ocorra dentro desses blocos, a operação é desfeita e nenhum dado é registrado no banco de dados.

Quadro 14 – Exemplo de criação de um procedimento de armazenamento

```
CREATE OR REPLACE FUNCTION sp_agencia(text, text) RETURNS integer AS
$BODY$
    DECLARE
    v_cnpj ALIAS FOR $1;
    v_nome_fantasia ALIAS FOR $2;
    v_id_agencia integer;
BEGIN
    SELECT id_agencia INTO v_id_agencia FROM agencias WHERE agencias.cnpj = v_cnpj;
    IF v_id_agencia IS NOT NULL THEN
      RETURN v_id_agencia;
    ELSE
      INSERT INTO agencias(cnpj, nome_fantasia)
      VALUES(v_cnpj, v_nome_fantasia);
      RETURN currval ('agencias_id_agencia_seq');
    END IF;
END;
$BODY$
  LANGUAGE plpgsql;
```

Fonte: elaborado pelo autor, 2019

Conforme Milani (2008), os *triggers* (gatilhos) são um recurso importante ao banco de dados, porque dá ao mesmo independência para realizar certas operações sem necessidade de acionamento explícito pelo usuário.

Os gatilhos são mecanismos programáveis que “disparam” a execução de códigos armazenados no servidor ou outros códigos/comandos SQL sem necessidade de acionamento manual, como acontece com as funções. Os acionamentos podem ocorrer quando alguma condição do servidor é satisfeita, antes e depois de algum evento SQL, como inserção, alteração ou exclusão, ou ainda baseado em tempo, ou repetição (MILANI, 2008).

A principal vantagem deste recurso é a automatização do banco de dados, porque determinadas situações necessitam de verificação constante. Por exemplo, a verificação de inserção de um determinado valor, gerenciamento de estoques, valores financeiros, controles de entrada e saída, entre outros. Aplicações poderiam realizar essas tarefas, mas teriam de ser programadas para serem acionadas de tempo em tempo, sem garantia efetiva de controle. Os gatilhos podem ser acionados sempre que ocorrer uma determinada situação, quantas vezes fossem necessárias, sem necessidade de acionamentos desnecessários (MILANI, 2008).

Os gatilhos podem ser classificados de duas formas. Primeiramente, se seu acionamento ocorre antes (*BEFORE*) ou depois (*AFTER*) de um comando SQL, que pode ser uma inserção, exclusão ou alteração. Não é possível configurar gatilho para comando *SELECT*. Também pode ser classificado como de execução por comando, ou por alteração de linha na tabela. Na primeira situação, o gatilho será acionado uma única vez sempre que uma condição for satisfeita; já na segunda, o gatilho acionará um único comando SQL diversas vezes, sempre que uma linha da tabela sofre alteração (MILANI, 2008).

Ainda segundo Milani (2008), é possível combinar dois modos (tabela 1) desde que as operações sejam em comum. Por exemplo, “*AFTER INSERT OR UPDATE“,* ou seja, o acionamento acontecerá após uma ação de inserção ou atualização.

**Tabela 1 – Modos de operação de um *trigger***

| Modo | Descrição |
|---|---|
| *BEFORE INSERT* | O gatilho é disparado antes de uma ação de inserção |
| *BEFORE UPDATE* | O gatilho é disparado antes de uma ação de atualização |
| *BEFORE DELETE* | O gatilho é disparado antes de uma ação de exclusão |
| *AFTER INSERT* | O gatilho é disparado após uma ação de inserção |
| *AFTER UPDATE* | O gatilho é disparado após uma ação de atualização |
| *AFTER DELETE* | O gatilho é disparado após uma ação de exclusão |

Fonte: adaptado de MILANI (2008), p.211

O Quadro 15 exibe a sintaxe básica para a criação de um gatilho “nome”. O “tipo_trigger” indica se a execução do trigger que será criado deverá ser realizada antes ou depois do evento configurado: *BEFORE*, para executar o trigger antes do evento configurado, ou *AFTER*, para executá-lo depois. “eventos” indica qual ou quais eventos deverão executar o *trigger*: INSERT, para executá-lo em uma operação de inserção; *UPDATE*, em uma operação de atualização; e *DELETE*, em uma operação de exclusão. É possível mesclar os tipos de eventos utilizando o operador *OR*. Por exemplo: *INSERT OR UPDATE OR DELETE*. *ON* tabela indica onde o *trigger* será configurado e disparado. “tipo_execução” designa se o *trigger* deve ser executado uma vez por comando ou se quando alguma linha sofrer alteração: *EACH STATEMENT* para dispará-lo uma única vez, independete de quantas linhas forem alteradas e *EACH ROW* para dispará-lo cada vez que uma linha da tabela for alterada. *EXECUTE PROCEDURE* indica que função será executada, além dos parâmetros que serão repassados para ela.

Quadro 15 – Sintaxe de criação de um gatilho

```
CREATE TRIGGER nome tipo_trigger eventos ON tabela FOR tipo_execução
    EXECUTE PROCEDURE função (parâmetros);
```

Fonte: elaborado pelo autor, 2019

O Quadro 16 apresenta um exemplo de gatilho chamado “trigger_UPPERCASE”, que é executado antes de uma inserção ou atualização na tabela “agencias”. Para cada linha (*FOR EACH ROW*) ele executa um procedimento (*EXECUTE PROCEDURE*) chamado “tr_upper_agencias” e não passa nenhum parâmetro.

Quadro 16 – Uso de um gatilho

```
CREATE TRIGGER "trigger_UPPERCASE"
    BEFORE INSERT OR UPDATE
    ON agencias
    FOR EACH ROW
    EXECUTE PROCEDURE tr_upper_agencias();
```

Fonte: elaborado pelo autor, 2019

Um domínio é, basicamente, uma lista limitada de valores. É amplamente utilizado em situações que necessitam de um dado que se repete inúmeras vezes ou ainda para restringir o que é armazenado no banco de dados. O Quadro  exibe a sintaxe de criação de um domínio chamado “name”. *AS* indica um comando de atribuição de tipo, enquanto “tipo_de_dado” será o tipo de dado que será utilizado. *DEFAULT* utiliza o valor “expressão” como padrão, caso nenhum valor seja atribuído. Por fim, *constraint* determina uma condição a ser satisfeita. Ela pode ser: *NULL/NOT NULL,* para valores nulos/não nulos; ou *CHECK,* que verifica uma condição especificada.

Quadro 17 – Sintaxe de criação de um domínio

```
CREATE DOMAIN nome [AS] tipo_de_dado
[DEFAULT expressão]
[CONSTRAINT [...]]
```

Fonte: elaborado pelo autor, 2019

No Quadro 18, um domínio “sexo” é criado. Ele é do tipo caractere (*char)* e suporta apenas 1 letra. A opção padrão (*DEFAULT*) é “M” (Masculino), não pode ser NULO (*NOT NULL*), ou seja, seu preenchimento é obrigatório. Por fim, é checado se o valor digitado é “M” ou “F” (Feminino). Caso o valor seja diferente da checagem, o banco de dados retornará um erro informando que apenas os valores da checagem serão aceitos.

Quadro 18 – Criação de um domínio

```
CREATE DOMAIN sexo AS CHAR(1)
DEFAULT 'M'
NOT NULL
CHECK (VALUE IN ('M','F'));
```

Fonte: elaborado pelo autor, 2019

Este tipo de controle garante que nenhum dado será inserido erroneamente. É possível ainda utilizar expressões regulares para definir outros controles.

Apesar do banco de dados possuir diversos tipos de dados a sua complexidade tem aumentado nos últimos anos. Para atender tais necessidades, é possível estender o PostgreSQL para suportar a criação de tipos (*types*) definidos pelo usuário. Por exemplo, em uma aplicação de biblioteca, deseja-se cadastrar algumas informações para cada livro: título, lista de autores, editora, conjunto de palavras-chaves. Note que os autores podem ser cadastrados em um *array*, mas para encontrar todos os livros de um determinado autor, é preciso acessar uma subparte do domínio autores. O mesmo funciona com as palavras-chaves. Para obter todos os livros que incluam uma ou mais delas, logo é preciso acesso à uma subparte deste domínio (SILBERSCHATZ; KORTH; SUDARSHAN, 2006).

O Quadro 19 apresenta a criação de um tipo chamado “name” que utilizará atributos (*attribute_name*) do tipo especificado (*data_type*). É possível ainda utilizar *COLLATE* para determinar a ordenação de *strings*.

Quadro 19 – Sintaxe de criação de um tipo

```
CREATE TYPE name AS
    ([attribute name data type [COLLATE collation]] [, ...])
```

Fonte: elaborado pelo autor, 2019

O Quadro 20 mostra a criação de um tipo chamado “dados”. Ele é do tipo composto e possui cinco atributos de tipos diferentes: “total_registros”, “total_nulos”, “completude”, “coluna” e “tabela”, sendo os três primeiros do tipo numérico e os demais como cadeia de caracteres. É possível ainda (mas não é o presente caso) adicionar outro tipo definido pelo usuário como atributo.

Quadro 20 – Criação de um tipo

```
CREATE TYPE dados AS
   (total_registros numeric,
    total_nulos numeric,
    completude numeric,
    coluna character varying(100),
    tabela character varying(100));
```

Fonte: elaborado pelo autor, 2019

As sequências são utilizadas para geração de campos auto-incrementais que variam conforme inserções são realizadas. O comando abaixo mostra a sintaxe de criação de uma sequência:

```
CREATE SEQUENCE nome_da_sequencia;
```

## 2.3 Modelos de dados

A modelagem de dados serve para representar uma parte do mundo real e organizá-los em um banco de dados. Esse recurso permite que seja realizada uma análise prévia da construção de um sistema de dados, como um esboço de um projeto final (MARTELLI; SANTANA FILHO; CABRAL, 2017).

Funciona como um plano com todas as informações dos componentes que farão parte do banco de dados e serão acessíveis através do sistema de gerenciamento. Conceitua as estruturas de dados que serão necessárias, além de suas associações e regras de operações dos objetos (Ibid.).

Existem atualmente três categorias de modelos: o modelo conceitual é o mais abstrato. Em seguida, temos o modelo lógico, mais técnico e detalhado. Finalmente, temos o modelo físico, que é a codificação do projeto para o banco de dados (Ibid.).

Neste trabalho, todas as modelagens serão realizadas pelo software brModelo[3]. Outras ferramentas podem ser utilizadas, variando de acordo com as necessidades do projeto e capacidade de superá-las. A grande vantagem desta ferramente é sua facilidade de utilização, inclusive para quem está iniciando no processo de aprendizagem sobre modelagem de dados. Outra vantagem é que o *software* pode gerar os modelos discutidos neste trabalho.

### 2.3.1 Modelo conceitual

Para Heuser (1998), o modelo conceitual descreve o banco de dados a ser implementado sem estar dependente da implantação do SGBD. Este modelo define quais dados aparecerão no banco de dados, mas não se preocupa em como estes dados estão registrados no SGBD. "[. . . ] É uma representação que permite visualizar facilmente todo o conteúdo do banco de dados" (MARTELLI; SANTANA FILHO; CABRAL, 2017, p.41). A técnica mais utilizada neste modelo é abordagem Entidade-Relacionamento (ER).

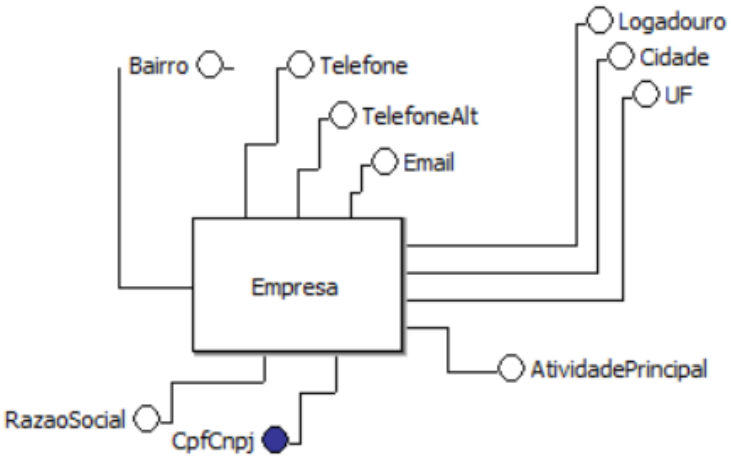
"O [Modelo Endidade-Relacionamento] (MER) trata-se de um modelo conceitual, representado em forma de gráfico, que inicialmente foi introduzido por Peter Chen em 1976, porém desde então já sofreu algumas atualizações em sua estrutura" (CARVALHO, 2011, p.10). O MER é um diagrama, onde objetos são interligados, que possibilita identificar uma forma apropriada de organizar os dados; uma representação gráfica da realidade que deve apresentar de forma clara as entidades envolvidas e seus relacionamentos (DAMAS, 2007). O modelo utilizado neste trabalho pode ser encontrado no Apêndeice A.1.

> Este modelo permite mapear os eventos concretos por meio de uma abstração para o ambiente de banco de dados. Essa abstração descreve os objetos (**entidades**) envolvidos, suas características (**atributos**) e como se relacionam (**relacionamentos**). Para representar melhor o MER, você precisa criar um DER (Diagrama Entidade e Relacionamento), que é a chave para que o cliente compreenda o modelo conceitual. (MARTELLI; SANTANA FILHO; CABRAL, 2017, p.41).

O primeiro passo é definir as entidades que farão parte do modelo conceitual. Entidade é uma representação de um objeto (ente) do mundo real. Se um grupo de entidades corresponde a mesma gênese do mundo real, então tem-se um conjunto de entidades. Cada entidade possui atributos que as diferenciam entre si e podem ser simples, compostos ou multivalorados. Esses atributos multivalorados são direcionados a um grupo de valores. O valor do atributo não se repete em mais de um elemento da entidade (CARVALHO, 2011). Normalmente as entidades são representadas por retângulos onde se coloca o nome da entidade.

[3] www.sis4.com.br/brModelo

**Figura 8 – Exemplo de entidade e seus atributos**




Fonte: captura de tela elaborada pelo autor no brModelo, 2019

De acordo com Martelli, Santana Filho e Cabral (2017), é possível utilizar algumas regras que ajudam a identificar as entidades:

- Sempre serão exibidos no formato de retângulo.
- Seu nome normalmente é um substantivo no singular.
- Para os nomes compostos, utiliza-se o "_" (underline).
- Não deve usar preposições. Ao invés de "Empresa_da_Cidade", utilize "Empresa_Cidade".

As entidades podem ser fortes, quando sua existência independe de outras entidades, em outras palavras, por si só já possuem sentido de existir. Por exemplo, em um sistema de biblioteca, a entidade livro independe de qualquer outra para existir; fracas, quando dependem da existência de outras entidades. Por exemplo, a entidade empréstimo depende da entidade livro, pois a entidade empréstimo isolada não tem sentido; e as associativas, quando associa-se uma entidade a um relacionamento existente.

Somente a existência das entidades não garante que dados de uma entidade se relacione com os dados de outras entidades. Para que isso ocorra é preciso que haja um relacionamento entre elas (Ibid.).

Os relacionamentos são representações conceituais que os objetos (a partir do conjunto de entidades) possuem entre si, ou seja, expostos dois conjuntos de elementos, os componentes de um podem se relacionar com os do outro (CARVALHO, 2011). Mais precisamente, Martelli, Santana Filho e Cabral (2017, p.47) esclarece que:

> Relacionamentos são ligações entre as entidades de dados, que expressam formas de associações, e são representados por losangos. São descritos como ações ou processos que ocorrem entre as entidades associadas.

Através dos relacionamentos podemos estabelecer a ligação entre duas entidades como se fossem apenas uma única estrutura.

Neste modelo, ao efetuar a ligação entre duas entidades, estamos estabelecendo regras que garantam a integridade da informação.

Existem três tipos de relacionamentos:

- Relacionamento um-para-um: neste relacionamento, uma entidade só pode se relacionar com apenas 1 outra entidade, conforme a Figura 9.
- Relacionamento um-para-muitos: neste relacionamento, uma entidade pode estar associada a qualquer número de entidades; porém as demais unidades só se relacionam apenas com a primeira entidade, segundo a Figura 10. Neste tipo de relacionamento o modelo não força a obrigatoriedade entre as entidades.
- Relacionamento muitos-para-muitos: várias entidades podem estar associadas a qualquer número de entidades, de acordo com a Figura 11.

**Figura 9 – Exemplo de relacionamento um-para-um**


Fonte: captura de tela elaborada pelo autor no brModelo, 2019

**Figura 10 – Exemplo de relacionamento um-para-muitos**


Fonte: captura de tela elaborada pelo autor no brModelo, 2019

**Figura 11 – Exemplo de relacionamento muitos-para-muitos**


Fonte: captura de tela elaborada pelo autor no brModelo, 2019

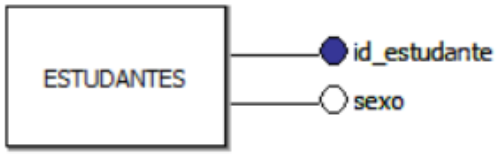
Para ajudar a definir os relacionamentos, é utilizada uma restrição que aparece notadamente entre parênteses ao lado de cada entidade. Essa restrição é também chamada de **cardinalidade**. A cardinalidade é definida em ambas as direções das entidades através do relacionamento. As Figuras 9, 10 e 11 usam este recurso (MARTELLI; SANTANA FILHO; CABRAL, 2017). Na Figura 9, por exemplo, temos a entidade "EMPRESA". Efetuando a leitura, percorrendo o relacionamento e chegando à outra entidade (faz-se a leitura da cardinalidade que fica no destino, ou seja, (1,1)) verifica-se que a "EMPRESA possui uma, e somente uma SEDE". O processo também é realizado no sentido contrário, ou seja, "a SEDE só possui uma, e somente uma EMPRESA".

Os atributos são informações que caracterizam a entidade. São representados por elipses. Sua etiqueta deve ser um substantivo, começar com uma letra e evitar espaços e/ou acentuação (Ibid.).

- Atributo simples: a maioria dos atributos são considerados simples. Não possuem nenhuma característica especial e nem é divido em partes.
- Atributo composto: são atributos que podem ser formados por mais de um atributo. Por exemplo, nome pode conter nome e sobrenome; telefone reúne a entidade DDD e entidade número. No modelo lógico é possível que sejam convertidos para atributos simples.
- Atributo multivalorado: são atributos que podem assumir mais de um valor. Por exemplo, ao informar os números de telefone fixo e celular.
- Atributo determinante: são atributos (normalmente sequenciais), únicos e que servem para identificar a entidade. Normalmente é composto por número de matrícula, CPF, HASH, entre outros. Tal atributo ainda pode ser:

**Atributo-chave**: identifica de forma única um registro armazenado na entidade. Normalmente é representado por um círculo preenchido.

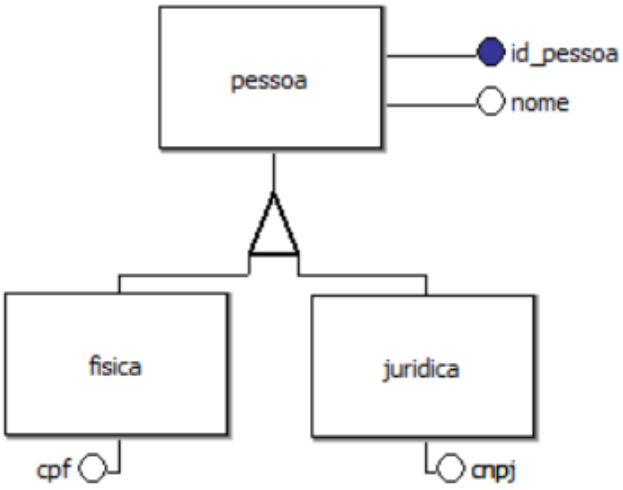
**Figura 12 – Exemplo de entidade com atributo-chave**




Fonte: captura de tela elaborada pelo autor no brModelo, 2019

Alguns atributos possuem especificidades que necessitam da utilização de outros atributos voltados para características específicas. Seu comportamento é muito parecido com uma *interface* de programação orientada à objetos: um determinado objeto possui características em comum com outro objeto, o que os difere são características específicas em cada um. O mesmo funciona aqui, porém, com atributos. Um exemplo bem simples poderia ser o tipo de pessoa de um cadastro: física ou jurídica. Sabemos que ambos compartilham algo em comum, como o nome. Os demais dados são utilizados especificamente em seus contextos. A isso dá-se o nome de **especialização**. Normalmente são representados através de triângulos, de acordo com a Figura 13 (MARTELLI; SANTANA FILHO; CABRAL, 2017).

**Figura 13 – Exemplo de especialização**




Fonte: captura de tela elaborada pelo autor no brModelo, 2019

### 2.3.2 Modelo lógico

O modelo lógico é consequência da conversão do modelo conceitual para um tipo de banco de dados, como relacional, hierárquico, de rede, entre outros. Este modelo é uma abstração do banco de dados na ótica do profissional responsável por criar e gerenciar o banco de dados. Esta etapa é importante, porque é a partir dela que o modelo físico é implementado, então o tipo de banco de dados já é conhecido nesta

etapa (MARTELLI; SANTANA FILHO; CABRAL, 2017). Um exemplo deste modelo pode ser encontrado no Apêndice A.2.

Ainda segundo Barros (2019), algumas regras de conversão devem ser satisfeitas a fim de gerar o modelo lógico. A primeira delas consiste no mapeamento de entidades:

- Para cada entidade comun do modelo conceitual, criar uma tabela no modelo lógico.

Em seguida, o mapeamento de atributos:

- Para cada atributo-chave no modelo conceitual, criar um atributo **chave primária (*Primary Key - PK*)** na entidade do modelo lógico;
- Para cada atributo simples no modelo conceitual, criar um atributo simples na entidade do modelo lógico;
- Para cada atributo composto, criar na entidade do modelo lógico somente os componentes do atributo composto.

Como restrição, é possível determinar se um campo pode receber valores nulos (*NULL*) ou não (*NOT NULL*). Toda chave primária (PK) deverá ser *NOT NULL*, em outras palavras, a chave primária não aceitará valor nulo.

O mapeamento de entidades fracas se dá ao criar uma entidade no modelo conceitual incluindo nesta entidade fraca o atributo-chave da entidade pai. Sua chave primária compreenderá o atributo-chave da entidade pai mais o atributo-chave da entidade fraca.

Por fim, o mapeamento de relacionamentos:

- Para relacionamentos “um para muitos” (1:N), cada atributo-chave da entidade do lado “um” deve ser registrado como **chave secundária (*Foreign Key - FK*)** para cada atributo do lado “muitos”. Neste tipo de relacionamento, o modelo conceitual não força a obrigatoriedade de relacionamento entre as entidades, ou seja, em alguns casos, alguns atributos podem não ser utilizados.
- Para relacionamentos “muitos para muitos” (N:M), deve-se criar uma nova entidade no modelo lógico e utilizar os atributos das entidades participantes do relacionamento como chaves primárias desta nova entidade.
- Para relacionamentos “um para um” (1:1), escolhe-se uma das entidades e a transfere seu atributo-chave para outra unidade como chave secundária.

Ainda que o esquema lógico tenha sido criado, ainda é possível realizar algumas alterações nas entidades, relacionamentos, cardinalidades, tipos, entre outros. Depois, basta gerar o modelo físico.

### 2.3.3 Modelo físico

É uma descrição de um banco de dados no nível de abstração visto pelo usuário do SGBD. Assim, esse modelo depende do SGBD que está sendo usado. Aqui são detalhados os componentes da estrutura, além da estrutura física do banco, como tabelas, campos, tipos de valores, índices, etc. Nesse estágio estamos prontos para criar o banco de dados propriamente dito, usando o SGBD escolhido (LOPES, 2012). O Apêndice A.3 apresenta o modelo físico utilizado neste trabalho.

## 2.4 Normalização

A ordem do armazenamento de dados, assim como sua estruturação variam bastante. É preciso ponderação e planejamento para evitar problemas que venham a acontecer numa má estruturação do banco de dados, o que pode deixá-lo lento, menos preciso, incoerente, redundante ou não confiável. Essas anomalias podem ocorrer na inserção, quando o sistema fica impedido de funcionar por falta de dados; na alteração, quando uma alteração não ocorre por falha de planejamento no relacionamento entre duas tabelas; e na exclusão, quando um dado não pode ser apagado sob o risco de mais dados serem perdidos (MILANI, 2008).

Para minimizar tais anomalias, alguns métodos de normalização podem ser utilizados. É comum ocorrer um aumento de dados nas tabelas do banco de dados, já que assuntos distintos são separados visando melhor gerenciamento. Isso é importante, pois a qualidade está diretamente associada à redução de desperdícios, retrabalhos e erros. A normalização ocorre em etapas chamadas de Formas Normais (FN), que foram descritas por Codd (1970), sendo unidas nas décadas seguintes. Atualmente existem 5 formas normais, mas neste trabalho utilizaremos apenas 3 FN, porque suas aplicações já garantem a integridade e qualidade desejada (FIGUEIREDO, 2018; MILANI, 2008).

A primeira forma normal (1FN) consiste em tornar todos os atributos da tabela em atômicos, ou seja, não devem conter repetições ou mais de um valor. A conversão consiste em identificar a chave primária da tabela, identificar as colunas que possuem mais de um valor e removê-las, criar uma nova tabela para os elementos repetidos juntamente com a chave primária e finalmente criar um relacionamento entre a tabela nova e a que teve as colunas removidas (MELO, 2011). A Tabela 2 apresenta uma entidade que o atributo endereço armazena uma série de valores como Logradouro, Número, CEP, Bairro, Cidade, Estado. É possível também que várias pessoas morem

na mesma localidade, repetindo a maioria dos valores.

**Tabela 2 – Tabela não normalizada**

| PESSOA | | | |
|---|---|---|---|
| ID | NOME | TELEFONE | ENDEREÇO |
| 1 | Fulano | 11111111 | Rua Tal, Nº 45, Bairro das Acácias, Cidade Ali, Estado A |
| 2 | Ciclano | 22222222 | Rua Aquela, Nº 72, Bairro Algum, Cidade Acolá, Estado B |
| 3 | Beltrano | 33333333 | Rua da Flor, Nº 15, Bairro Algum, Cidade Acolá, Estado B |
| 4 | User | 44444444 | Rua Antares, Nº 20, Bairro Qualquer, Cidade Além, Estado C |

Fonte: elaborado pelo autor, 2019

O atributo endereço é desmembrado da tabela pessoa e será dividido nas tabelas 3, 4, 5, 6 e 7, que serão relacionadas a partir das chaves primárias (ID).

**Tabela 3 – Tabela pessoa após primeira forma normal**

| PESSOA | | |
|---|---|---|
| ID | NOME | TELEFONE |
| 1 | Fulano | 11111111 |
| 2 | Ciclano | 22222222 |
| 3 | Beltrano | 33333333 |
| 4 | User | 44444444 |

Fonte: elaborado pelo autor, 2019

**Tabela 4 – Tabela bairros criada durante primeira forma normal**

| BAIRROS | |
|---|---|
| ID | BAIRRO |
| 1 | Acácias |
| 2 | Algum |
| 3 | Qualquer |

Fonte: elaborado pelo autor, 2019

**Tabela 5 – Tabela cidades criada durante a primeira forma normal**

| CIDADES | |
|---|---|
| ID | CIDADE |
| 1 | Acolá |
| 2 | Ali |
| 3 | Além |

Fonte: elaborado pelo autor, 2019

**Tabela 6 – Tabela estados criada durante a primeira forma normal**

| ESTADOS | |
|---|---|
| ID | ESTADO |
| 1 | A |
| 2 | B |
| 3 | C |

Fonte: elaborado pelo autor, 2019

**Tabela 7 – Tabela endereços criada durante a primeira forma normal**

| ENDEREÇOS | | | | |
|---|---|---|---|---|
| PESSOA_ID | LOGRADOURO | BAIRRO_ID | CIDADE_ID | ESTADO_ID |
| 1 | Rua Tal, Nº 45 | 1 | 2 | 1 |
| 2 | Rua Aquela, Nº 72 | 2 | 1 | 2 |
| 3 | Rua da Flor, Nº 15 | 2 | 1 | 2 |
| 4 | Rua Antares, Nº 20 | 3 | 3 | 3 |

Fonte: elaborado pelo autor, 2019

Para executar a segunda forma normal (2FN) é preciso que já se esteja na 1FN, conforme Tabela 8. Além disso, todos os atributos não-chaves da tabela deverão depender unicamente da chave primária (dependente de toda a chave, não apenas de parte dela)(MELO, 2011; MILANI, 2008).

**Tabela 8 – Tabela antes da segunda forma normal**

| IdSetor | NomeSetor | IdFuncionario | NomeFuncionario | TempoServico |
|---|---|---|---|---|
| 1 | RH | 14 | Carlos | 10 |
| 2 | Comercial | 25 | Andre | 2 |
| 3 | Marketing | 10 | Joao | 4 |
| 4 | TI | 30 | Alexandre | 7 |
| 5 | Logistica | 8 | Henrique | 5 |
| 6 | Diretoria | 2 | Fabio | 6 |

Fonte: elaborado pelo autor, 2019

Apesar de estar na 1FN, o atributo “TempoServico” só diz respeito ao funcionário, sem necessidade de estar atrelada também ao setor. Caso algum funcionário seja transferido para outro setor e o registro seja apagado, o tempo de serviço também o será. Para solucionar este problema, é necessário criar uma tabela para o setor, outra para o funcionário e em seguida criar uma tabela para vincular o setor ao funcionário. As tabelas 9, 10 e 11 mostram o resultado da normalização na segunda forma normal.

**Tabela 9 – Tabela funcionarios criada durante a segunda forma normal**

| IdFuncionario | NomeFuncionario | TempoServico |
|---|---|---|
| 14 | Carlos | 10 |
| 25 | Andre | 2 |
| 10 | Joao | 4 |
| 30 | Alexandre | 7 |
| 8 | Henrique | 5 |
| 2 | Fabio | 6 |

Fonte: elaborado pelo autor, 2019

**Tabela 10 – Tabela setores criada durante a segunda forma normal**

| IdSetor | NomeSetor |
|---|---|
| 1 | RH |
| 2 | Comercial |
| 3 | Marketing |
| 4 | TI |
| 5 | Logistica |
| 6 | Diretoria |

Fonte: elaborado pelo autor, 2019

**Tabela 11 – Tabela que relaciona funcionários ao setores criada durante segunda forma normal**

| IdSetor | IdFuncionario |
|---|---|
| 1 | 14 |
| 2 | 25 |
| 3 | 10 |
| 4 | 30 |
| 5 | 8 |
| 6 | 2 |

Fonte: elaborado pelo autor, 2019

Finalmente, para a terceira forma normal (3FN), além de já estar na segunda forma normal (2FN), os atributos não chaves da tabela deverão ser independetes, sendo dependentes unicamente da chave primária, aumentando ainda mais a integridade do banco de dados.É preciso identificar quais colunas são dependentes de outras colunas não chave e movê-las para outra tabela. É bastante parecida com a 2FN, sendo realizada automaticamente pelos administradores de bancos de dados que separam os atributos de cada tabela de acordo com o contexto (MELO, 2011; MILANI, 2008).

**Tabela 12 – Tabela funcionarios antes da terceira forma normal**

| FUNCIONARIOS | | | |
|---|---|---|---|
| ID | Nome | ID_Cargo | Descricao_Cargo |
| 1 | Carlos | 4 | Designer |
| 5 | Danilo | 2 | DBA |
| 8 | Alcides | 5 | Analista |
| 10 | Laura | 1 | Gerente |

Fonte: adaptado de MELO (2011)

O atributo “Descricao_Cargo” depende exclusivamente do atributo não chave “ID_Cargo”. Assim sendo, o mesmo deve ser utilizado em uma nova tabela, conforme as Tabelas 13 e 14.

**Tabela 13 – Tabela funcionarios após a segunda forma normal**

| FUNCIONARIOS | | |
|---|---|---|
| ID | Nome | ID_Cargo |
| 1 | Carlos | 4 |
| 5 | Danilo | 2 |
| 8 | Alcides | 5 |
| 10 | Laura | 1 |

Fonte: adaptado de MELO (2011)

**Tabela 14 – Tabela cargos criada durante a terceira forma normal**

| CARGOS | |
|---|---|
| ID_Cargo | Descricao_Cargo |
| 1 | Gerente |
| 2 | DBA |
| 4 | Designer |
| 5 | Analista |

Adaptado de MELO (2011)

## 2.5 Extração da informação

A etapa de extração da informação é defina por Jiang (2012, p. 11. Tradução nossa.):

> Extração da informação é a tarefa de encontrar informação estruturada a partir de texto não estruturado ou semi-estruturado. É importante na mineração de texto e tem sido extensivamente estudada em vários grupos de pesquisa, incluindo porcessamento de linguagem natural, recuperação da informação e mineração *web*. Possui uma gama de aplicações em domínios como literatura biomédica, mineração e *Business Intelligence*.

O objetivo geral é descobrir informação estruturada a partir de diversos suportes informacionais. Tal informação pode ser apresentada diretamente ao usuário, ou ainda pode ser utilizada por outros sistemas computadorizados, como mecanismos de busca e sistemas de gerenciamento de banco de dados, visando melhorar os serviços. O tipo específico e estrutura da informação a ser extraída depende da necessidade daquela aplicação em particular (Ibid., p. 12). Para Taylor (2018), novas ferramentas estão disponíveis para analisar dados. Muitas destas ferramentas são baseadas em *machine learning* (aprendizado de máquina, em tradução livre). Dados estruturados podem usá-lo, mas o volume massivo e os diferentes tipos de dados (não estruturados ou semi-estruturados) o requerem.

Dados em geral podem ser adquiridos nos mais diversos suportes informacionais: arquivos físicos ou computadorizados, bancos de dados, *datawarehouse*, entre outros (BARANAUSKAS, 2013). Estes dados podem ser **estruturados, semi-estruturados** e **não estruturados**.

- **Dados estruturados**: dados estruturados são aqueles rigidamente organizados e representados, cuja estrutura foi planejada previamente com o intuito de armazená-los. Formulários, por exemplo, representam bem sua utilização. Os campos recebem tipos de valores específicos (somente texto, número, valor binário) criando um padrão bem definido, com estrutura rígida e formato projetado para tal. Os dados de um mesmo cadastro tratam uma única pessoa, logo possuem uma relação. Os dados de outros cadastros, apesar de se referirem à outras pessoas, utilizam a mesma representação para armazenamento dos dados. Logo, possuem os mesmos atributos (campos do formulário) e formatos (com diferentes valores por cadastro) (MONTEIRO, 2019).

- **Dados semi-estruturados**: para Mello et al. (2000), dados semi-estruturados possuem uma representação estrutural heterogênea, que não é completamente “não estruturado”, muito menos estritamente tipado. Dados *web* são exemplos típicos deste tipo de dado, já que em alguns casos os dados se apresentam de forma estruturada (como um catálogo de produtos, ou ainda, um formulário *online*), em outros, ainda se pode identificar um padrão estrutural, flexível (um conjunto de documentos em um artigo), ou ainda, não identificar um padrão estrututal (uma imagem, vídeo, aúdio).

- **Dados não estruturados**: de forma antagônica aos dados estruturados, os dados não estruturados possuem uma estrutura interna flexível, dinâmica, mas não é elaborada via modelos ou esquemas de dados pré-definidos. São exemplos, documentos de texto, *emails*, mídias sociais, *websites*, mídias digitais (vídeos, músicas, fotos), dados de sensores (TAYLOR, 2018; MONTEIRO, 2019).

O processo de extração da informação consiste em algumas etapas que vão desde a definição dos objetivos até a avaliação dos dados e padrões descobertos. Primeiramente, são definidos os objetivos do processo, ou seja, que problema será solucionado e como. Nesta etapa, os envolvidos devem possuir conhecimento prévio do domínio. Em seguida, ocorre o pré-processamento dos dados, em que os mesmos são preparados e, baseados no conhecimento do indivíduo e nos objetivos estabelecidos, são transformados em um conjunto inicial de objetos. Nesta fase são definidos de forma manual quais atributos que descrevem os objetos (entidades de interesse) são importantes para atingir as metas estabelecidas. A definição evita a ocorrência de atributos dependentes ou redundantes. A fase seguinte realiza a integração dos dados provenientes de diferentes fontes para o formato padrão escolhido na primeira etapa. Caso sejam dados relacionais, isto pode requerer uma junção ou projeção de várias tabelas com relações de diferentes cardinalidades. O próximo estágio define a preparação dos dados, já que é possível que uma grande quantidade de dados brutos resulte em um conjunto de dados de tamanho moderado. Neste caso, é necessária a redução de dados. A redução serve para suavizar, discretizar ou agrupar valores de um atributo. O próximo passo consiste na definição do *template* de extração, ou seja, modelos estruturados, como uma base de dados, que devem ser construídos com os componentes obtidos nas etapas anteriores e preenchidos com os dados. A limpeza realiza a verificação de consistência da base de dados e também remove as repetições indevidas. A última etapa consistena na interpretação dos resultados, onde os padrões descobertos são avaliados e visualizados. Os padrões irrelevantes ou redundantes são removidos. Além disso, os dados são analisados e podem sofrer correções, seja para aumentar o desempenho do sistema, seja para eliminar redundâncias (BARBOSA, 2019; ZAMBENEDETTI, 2002; BARANAUSKAS, 2013).

Apesar de eficiente, a extração da informação não está livre de problemas. O primeiro deles consiste na quantidade de informações a serem extraídas. Segundo Baranauskas (2013), a combinação da Lei de Moore (a capacidade de processamento dobra a cada 18 meses) e que a capacidade de armazenamento dobra a cada 10 meses, gera uma lacuna entre nossa habilidade de gerar dados e de fazer uso deles. Outro problema consiste em dados aleatórios, onde nenhum relacionamento entre eles é encontrado. Além disso, os dados podem estar representados de forma errada, dificultando a identificação dos dados relevantes. Mais problemas podem ocorrer, como dados faltantes, incorretos, redundantes ou ainda em formato inadequado.

## 2.6 Qualidade de dados

Segundo a norma ISO 9000 (ABNT, 2000, p.7), qualidade é o “grau no qual um conjunto de características inerentes satisfaz a requisitos”. Neste sentido, qualidade

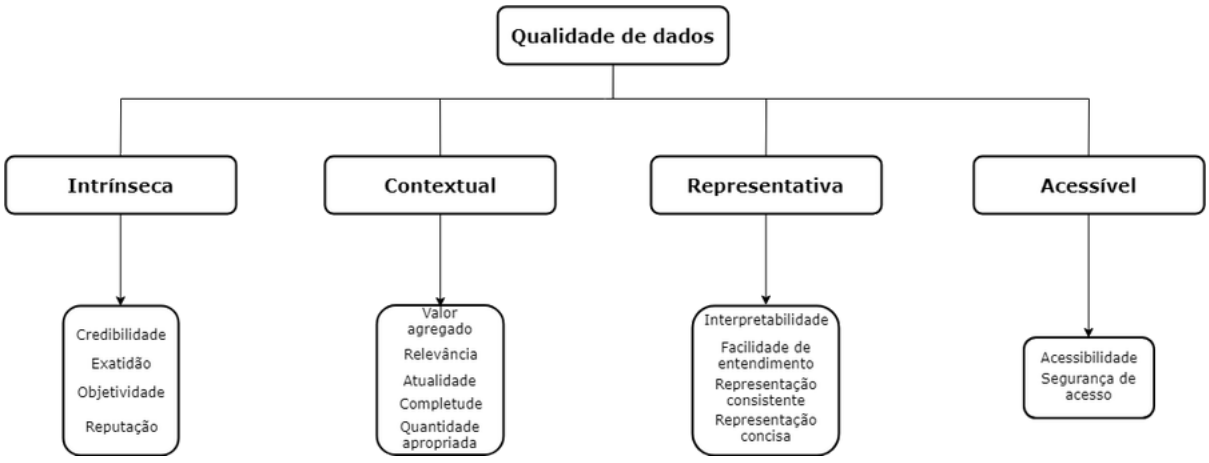
da informação pode ser compreendida como a totalidade de características dos dados que satisfazem as necessidades do usuário ou aplicação.

Segundo Machado (2013), o conceito de qualidade possui um amplo escopo, envolvendo não só ativos tangíveis quanto intangíveis. A qualidade da informação trata os aspectos relevantes da informação, possibilitando ainda identificar atributos e características que são parte dos processos de avaliação e qualificação da informação, com intenção de garantir sua eficácia nas organizações.

É comum que se confunda qualidade da infomação com qualidade de dados e que ambos sejam utilizados como sinônimos. Não há um concenso, mas o termo qualidade de dados normalmente é utilizado pela Ciência da Computação, enquanto qualidade da informação é utilizado pela Ciência da Informação (AROUCK, 2011; MACHADO, 2013). Ainda neste contexto, Madnick et al. (2009, cap. 1) orientam que há uma tendência em utilizar qualidade de dados para problemas técnicos, de abordagem prática, como indicadores numéricos de qualidade ou construção de ontologias/taxonomias; e qualidade da informação para problemas não-técnicos, de abordagem teórica, como avaiação de qualidade nas fases de construção de dados ou um modelo para avaliar a percepção de qualidade a partir da análise de impacto de valor dos dados e informações, por exemplo. Este trabalho utilizará ambos os termos como sinônimos, já que ambas abordagens serão utilizadas neste trabalho.

Este trabalho utilizará o *framework* conceitual de qualidade de dados proposta por Wang e Strong (1996) - Figura 14.

**Figura 14 – Framework conceitual de qualidade de dados**




Fonte: adaptado de WANG e STRONG (1996)

A acessibilidade garante que o dado estará acessível sempre que solicitado. Ao recuperar um dado, por exemplo; a interpretabilidade garante que o dado será

interpretado de forma correta, caso contrário o dado pode estar em outro idioma; a precisão garante que os dados estão corretos e devem vir de fontes confiáveis; a relevância define que o dado precisa ser relevante ao consumidor na tomada de decisão e a completude verifica se algum dado está faltando (MACHADO, 2013).

### 2.6.1 Completude

Compondo uma das propriedades essenciais da qualidade da informação, a completude é a inclusão de todos os dados necessários para responder questões de um ou mais problemas. Também representa o grau com que registros de um sistema possuem valores não-nulos. (PAIM; NEHMY; GUIMARÃES, 1996; CORREIA; PADILHA; VASCONCELOS, 2014).

De acordo com Ortega (2016, tradução nossa), um dado pode ser considerado incompleto se, por exemplo, o identificador de um ativo está incluso, mas o número de série e fabricante não estão. Em outras palavras, de nada importa a presença atributo-chave de uma entidade se os demais atributos estão vazios (*NULL*). Uma das expectativas da completude é que todos os atributos-chave sempre terão valores atribuídos no conjunto de dados.

Correia, Padilha e Vasconcelos (2014) ressaltam a importância da completude como indicador de qualidade ao afirmarem que a completude é a segunda dimensão mais avaliada nas análises referentes aos sistemas de saúde do governo federal. Ressaltam ainda a importância do monitoramento e avaliação contínua desta dimensão como ferramenta para verificar se o preenchimento da base de dados é realizado de forma adequada. Este indicador contribui ainda para identificar fragilidades e potencialidades dos dados produzidos pelos sistemas e com isso ampliar seu uso ou recomendar estratégias, visando a melhoria da qualidade. Além disso, concluem que completude ainda é uma dimensão de qualidade pouco explorada nos sistemas de saúde do governo brasileiro.

Quanto às técnicas de verificação da dimensão de completude, a maioria dos estudos na área de sistemas de informação de saúde, por exemplo, realiza a análise descritiva de indicadores de banco de dados (CORREIA; PADILHA; VASCONCELOS, 2014). Para Margaritopoulos et al. (2011, tradução nossa), a completude é medida baseada na presença ou ausência de valores nos campos. Relata ainda que em muitos esforços de pesquisa, os campos considerados na medição de completude são selecionados baseados em suas importâncias. Apresenta alguns estudos específicos, em que a completude é expressa apenas efetuando a contagem do número de elementos em um registro ou através de um cálculo, que consiste no número de campos que contém valores não-nulos dividido pelo número total de campos do registro.

### 2.6.2 Auditoria

A auditoria consiste em verificar os dados de arquivos, armazenados em meio eletrônico, a fim de garantir sua integridade, confiabilidade e se estão em conformidade com as leis. Pode ser executada isoladamente ou em conjunto com outra auditoria, de forma a complementá-la. Possibilita ainda a verificação de toda a base de dados e permite que bases de dados externas sejam consultadas para verificar os registros auditados (MARTINS, 2009).

Martins (2009) faz algumas advertências. Entre elas, que as informações podem não estar estruturadas de acordo com as melhores práticas de modelagem de dados; ou ainda, que o modelo entidade-relacionamento não é complexo, já que envolve um número limitado de símbolos e diagramas, mas sim o negócio. Esclarece que é essencial entender o negócio, a fim de identificar os riscos e selecionar os dados que responderão as questões de auditoria.

A auditoria pode ser realizada em algumas situações (Ibid.):

- Quando se tem um grande volume de dados a ser analisado;
- Quando 100% das informações podem ser verificadas;
- Utilizar dados processados por computador para respaldar o que foi auditado;
- Determinar se os dados são considerados complexos ou exatos;
- Se informações obtidas de diversas fontes podem ser comparadas.

> "Os princípios de auditoria exigem que qualquer evidência (independentemente de sua fonte ou formato) seja confiável, relevante e suficiente. O teste dos dados tem o objetivo de estabelecer a consistência deles em relação ao uso planejado (BRASIL. Tribunal de Contas da União - TCU, 1998, p. 24)".

É improvável que sistemas contenham dados livres de erros, a confiabilidade não exige dados perfeitos. É pressuposto que alguns procedimentos sejam realizados para avaliar a integridade e autenticidade dos dados, assim como a exatidão de seu processamento pelo computador que engloba testes de integridade de dados que verificam se o universo contém todos os elementos de dados e registros relevantes para a auditoria; testes de autenticidade dos dados, verificam se os dados correspondem com exatidão aos da fonte; e testes de exatidão do processamento, verificam se todos os registros foram processados de forma completa e se atenderam aos objetivos estabelecidos (BRASIL. Tribunal de Contas da União - TCU, 1998).

Martins (2009) descreve uma técnica para auditoria de dados baseada em escopo e planejamento, execução e relatório. Na fase escopo e planejamento, ocorre a

definição de objetivos, visando levantar as deficiências dos dados e a elaboração da matriz de planejamento. Ainda nesta fase, em conjunto com a equipe de Tecnologia da Informação (TI), ocorrem a modelagem dos dados e as definições dos campos a serem auditados. Na fase execução e relatório, busca-se a garantia da conformidade, provando a completude do arquivo e que os dados não estão corrompidos. Recomenda ainda a utilização de técnicas de auditoria auxiliadas por computador. Em seguida, efetua checagens periódicas por indícios de irregularidades.

Este trabalho realiza uma auditoria sem o auxílio do computador, ou seja, o procedimento compara os dados armazenados pelo sistema com dados provinientes de outras fontes (arquivos, registros, formulários, entre outros). Relatórios produzidos pelo governo e outros documentos produzidos por terceiros (Universidades, órgãos privados, entre outros) compõem uma base de dados útil para comparação (Ibid.).

# 3 Metodologia

## 3.1 Caracterização da pesquisa

Esta pesquisa é de natureza aplicada, porque busca solucionar um problema específico, gerando conhecimento, através de uma solução prática. A abordagem é quali-quantitativa, pois as metodologias quantitativas e qualitativas deste trabalho se complementam. Essa abordagem possibilita um cruzamento maior de dados, conferindo-lhes qualidade, aprofundando o peso da pesquisa na medida que valida todas as informações. Caracteriza-se como descritiva, porque exige uma série de informações sobre o objeto pesquisado. Além disso, descreve fatos e fenômenos de determinada realidade. Quanto aos procedimentos, inicia com uma pesquisa bibliográfica, utilizando referências teóricas já publicadas em diversos meios, e termina com um estudo de caso, onde dados de um contexto específico são levantados e analisados individualmente (GERHARDT; SILVEIRA, 2009).

Os dados secundários foram obtidos através de revisão bibliográfica que vão desde os conceitos básicos de dado e informação, passando por ferramentas, linguagens e códigos para criação de bancos de dados, até conceitos de qualidade da informação.

Em seguida, para coleta de dados primários, foi realizado um estudo de caso nos estágios do curso de Gestão da Informação da Universidade Federal de Pernambuco, utilizando os dados finais para geração de uma base de dados que possui um recurso para análise de qualidade dos dados.

## 3.2 Procedimento metodológico

Esta pesquisa buscou realizar um levantamento detalhado dos 391 estágios (obrigatórios, não-obrigatórios, aditivos, entre outros) realizados durante o período de 2011.1 à 2019.1 por 227 estudantes do curso de Gestão da Informação da UFPE. Foi ainda realizado levantamento em 1165 atividades de estágio, 115 empresas que celebraram contratos de estágio e 298 funcionários que respondiam pela representação das empresas ou supervisão dos estágios. Para tal, os formulários de estágio desse período foram analisados e os atributos principais foram selecionados e utilizados na criação de uma base de dados. Em seguida, após o preenchimendo dos dados a partir dos formulários, realizou-se a auditoria dos dados inseridos. Finalmente, também realizou-se o levantamento estatístico, baseado na completude, de todos os atributos que aceitassem valores nulos das tabelas adicionadas ao banco de dados e dos relacionamentos entre tabelas. O procedimento adotado nesta pesquisa consiste de cinco passos, os quais serão descritos a seguir.

O primeiro passo consistiu em determinar quais documentos serão utilizados

para estabelecer as entidades e atributos que farão parte da base de dados, além da sua criação. Os termos de compromisso de estágio (conforme modelos do Anexo A) apresentam elementos suficientes para obtenção das entidades e atributos, além disso são dados semi-estruturados, facilitando o processo de escolha das entidades, seus atributos e tipos de valores, além da inserção de dados na base de dados. Os formuláros podem variar de acordo com o tipo de estágio ou se é realizado através de uma agência intermediadora de estágios.

O segundo passo consistiu na construção dos modelos conceitual, lógico e físico do banco de dados. Para este passo, foi utilizado o programa brModelo, na versão 3.0, que foi desenvolvido pelo setor de sistemas de informação da Pontifícia Universidade Católica de Minas Gerais. Os modelos utilizados neste trabalho encontram-se no Apêndice A.

O terceiro passo consistiu em criar funções, gatilhos, domínios e tipos específicos com o objetivo de impedir a inserção de dados repetidos, visando manter o funcionamento da base de dados e assegurar que as informações foram inseridas de forma correta.

O quarto passo consistiu em popular os dados. A população definida para este estudo são estudantes do curso de Gestão da Informação da Universidade Federal de Pernambuco que exerceram algum tipo de estágio supervisionado entre o período de 2011.1 à 2019.1. A área de estudo restringiu-se ao Departamento de Ciência da Informação (DCI) da referida Universidade. Para a execução do estudo realizou-se uma coleta de dados relacionados aos estágios de Gestão da Informação da UFPE durante o período de 2011.1 à 2019.1. Os estágios foram firmados através dos termos de compromisso de estágios, onde constam diversas informações sobre os participantes do termo. Como várias pessoas estavam envolvidas no processo de coleta, uma planilha foi utilizada como suporte temporário para registro dos dados. Estes dados foram separados na planilha de acordo com os atributos da base de dados e era preciso transcrever os dados dos termos de compromisso de estágio, já que os mesmos encontravam-se impressos em papel. Por isso, alguns dados deixaram de ser registrados ou foram adicionados de forma incorreta. Estes dados foram corrigidos na etapa de auditoria.

O quinto passo consistiu na auditoria, melhoria e monitoramento da qualidade dos registros. Em que uma outra pessoa analisava os dados já inseridos, conferindo-os com o documento físico e corrigindo-os quando necessário.

# 4 Estudo de caso

## 4.1 Contexto

O curso de Gestão da Informação (GI) foi criado em 2008 pelo Departamento de Ciência da Informação (DCI) da Universidade Federal de Pernambuco (UFPE) com o objetivo de suprir a demanda das organizações, onde a informação é produzida, armazenada, recuperada e utilizada. O curso está localizado no Centro de Artes e Comunicação (CAC) e realiza anualmente seleção através do SISU para 55 vagas. Funciona no turno da noite e compreende até 14 semestres de duração. A disciplina de estágio supervisionado é obrigatória e possui carga horária de 220 horas. É possível realizar estágio não-obrigatório supervisionado enquanto está matriculado no curso. A disciplina tem o objetivo de preparar o estagiário para o mercado de trabalho, de forma que o mesmo possa aprender na prática os conceitos apresentados durante o curso (UNIVERSIDADE FEDERAL DE PERNAMBUCO, 2019).

O estágio supervisionado é registrado em termos de compromisso de estágio (Anexo A). Esses formulários possuem vários modelos diferentes e registram todos os participantes do processo, juntamente com as obrigações de cada um e as atividades que serão executadas pelo estagiário.

Os termos de compromisso de estágio estão armazenados na secretaria do curso, dentro de um armário de arquivos, em pastas plásticas, organizadas por ano, sem nenhuma ordem interna, podendo conter formulários de outros anos, já que termos aditivos de estágio estavam na mesma pasta que os formulários originais.

## 4.2 Modelos de dados

O Apêndice A.1 apresenta o modelo conceitual com todas as entidades, atributos e relacionamentos da base de dados de estágios. O Apêndice A.2 apresenta o modelo lógico gerado a partir do MER. O Apêndice A.3 contém as tabelas do banco de dados. Além disso, os atributos das tabelas estão comentados, explicando a função de cada um.

Alguns domínios foram estabelecidos para a base de dados especializada deste trabalho. O Quadro 21 apresenta um domínio que lista todas as siglas dos estados brasileiros.

Quadro 21 – Listagem dos estados brasileiros

```
CREATE DOMAIN public.dom_uf_check
  AS character varying(2)
  COLLATE pg_catalog."default"
  CONSTRAINT dom_uf_check_check CHECK (VALUE::text = ANY
 (ARRAY['AC'::character varying::text,
        'AL'::character varying::text,
        'AP'::character varying::text,
        'AM'::character varying::text,
       'BA'::character varying::text,
       'CE'::character varying::text,
       'DF'::character varying::text,
       'ES'::character varying::text,
       'GO'::character varying::text,
       'MA'::character varying::text,
       'MT'::character varying::text,
       'MS'::character varying::text,
       'MG'::character varying::text,
       'PA'::character varying::text,
       'PB'::character varying::text,
       'PR'::character varying::text,
       'PE'::character varying::text,
       'PI'::character varying::text,
       'RJ'::character varying::text,
       'RN'::character varying::text,
       'RS'::character varying::text,
       'RO'::character varying::text,
       'RR'::character varying::text,
       'SC'::character varying::text,
       'SP'::character varying::text,
       'SE'::character varying::text,
       'TO'::character varying::text]));
```

Fonte: elaborado pelo autor, 2019

Ao realizar a inserção de dados, somentes os valores delimitados no domínio que poderão ser utilizados. O Quadro 22 é um domínio que verifica se o telefone inserido possui 11 dígitos (DDD+telefone celular com o dígito 9 precedente) e, baseado na expressão regular proposta, se somente dígitos de 0 à 9 foram utilizados.

Quadro 22 – Domínio para verificação de telefone

```
CREATE DOMAIN public.dom_telefone_check
  AS character varying(11)
  COLLATE pg_catalog."default"
  CONSTRAINT dom_telefone_check_check CHECK (VALUE::text ~ '^([0-9]*)$'::text);
```

Fonte: elaborado pelo autor, 2019

O Apêndice A.3 contém o estado final das tabelas após as normalizações, definições das chaves, relacionamentos, obrigatoriedade de preenchimento, indexadores

e comentários. O gerenciamento das tabelas foi realizado por meio do pgAdmin III e a Tabela 15 apresenta uma breve descrição de cada tabela utilizada neste trabalho.

**Tabela 15 – Tabelas da base de dados de estágios de GI da UFPE**

| Nome da tabela | Descrição |
|---|---|
| agencias | Agências intermediadoras de estágio |
| atividades | Lista com atividades exercidas pelos estagiários nos contratos |
| atividades_categorias | Lista com atividades em nível “macro” e ID’s sequenciais automáticos para organizar as atividades “micro” |
| atividades_padronizadas | Classifica as atividades em nível “micro” |
| cnae | Código Nacional de Atividade Econômica das empresas |
| empresas | Dados de empresas (conforme site da Receita Federal) |
| empresas_cnae_secundarias | Classifica as atividades econômicas secundárias das empresas |
| empresas_locais | Lista com ID’s de locis vinculados às empresas cadastradas |
| empresas_trabalhadores | Lista com ID’s de empresas e trabalhadores que possuem vínculo empregatício entre si |
| estagios | Dados de contratos de estágio celebrados pelo DCI/UFPE |
| estagios_atividades | Relação de atividades exercidas em cada estágio |
| estudantes | Lista com matrícula, sexo e CPF dos estudantes |
| trabalhadores | Lista com dados de trabalhadoresde determinada empresa |
| trabalhadores_representantes | Lista com ID’s dos representantes das empresas |
| trabalhadores_supervisores | Lista com ID’s dos supervidores das empresas para cada estágio |

Fonte: elaborado pelo autor, 2019

## 4.3 Funções e gatilhos

Baseado no código proposto por Araujo (2010), foi criada uma função (Apêndice B.1) que verifica se a numeração do CPF (Cadastro de Pessoa Física) ou CNPJ (Cadastro Nacional de Pessoa Jurídica) inserida está correta. A função recebe um número por parâmetro, elimina a máscara (pontos e traços) e separa os dígitos em vetores, de acordo com a quantidade de dígitos. Em seguida, calcula o dígito verificador utilizando todos os dígitos da cadeia de caracteres (*string*) e compara o resultado com o dígito verificador passado por parâmetro. Se os dígitos contiverem o mesmo valor, "*TRUE*" (verdadeiro) será retornado pela função, caso contráro, o valor retornado será *"FALSE"* (falso).

Outra função (Apêndice B.2) verifica se o CNAE (Código Nacional de Atividade Econômica) digitado está presente na tabela cnae. A função recebe um número por parâmetro, elimina a máscara e armazena somente os números em uma variável interna da função. Em seguida, verifica se o número repassado está presente na tabela 'cnae'. Finalmente, caso esteja presente, retorna o próprio código; caso contrário, emite uma mensagem de erro e desfaz a operação.

O gatilho do Apêndice B.3 é acionado após a inserção de um CNPJ em algum cadastro. A função "fn_cnpj_cpf_validar" é acionada, repassando o número por argumento. Caso a verificação indique que o CNPJ está incorreto, uma mensagem de erro é exibida e a operação é desfeita; caso contrário, o CNPJ é retornado para inserção. Um outro gatilho foi cadastrado para validação de numeração CPF.

Já o gatilho do Apêndice B.4 busca padronizar a formatação dos textos inseridos na base de dados, atualizando os dados para caixa alta. Todas as tabelas com atributo texto possuem esse gatilho.

## 4.4 Entrada de dados

A inserção de dados na base pode ocorrer de duas formas: utilizando procedimentos de armazenamento ou uma interface gráfica de usuário. As chamadas dos procedimentos de armazenamento, conforme o Quadro 23, possuem a desvantagem da dificuldade de assimilação, haja vista que é preciso conhecer cada tipo de variável que será utilizada, além do seu formato de inserção. Qualquer erro impede que o cadastro prossiga. Todas as tabelas possuem uma função para armazenamento de dados, conforme exemplo disponível no Apêndice B.5.

Quadro 23 – Procedimento de armazenamento para inserção de valores por argumento

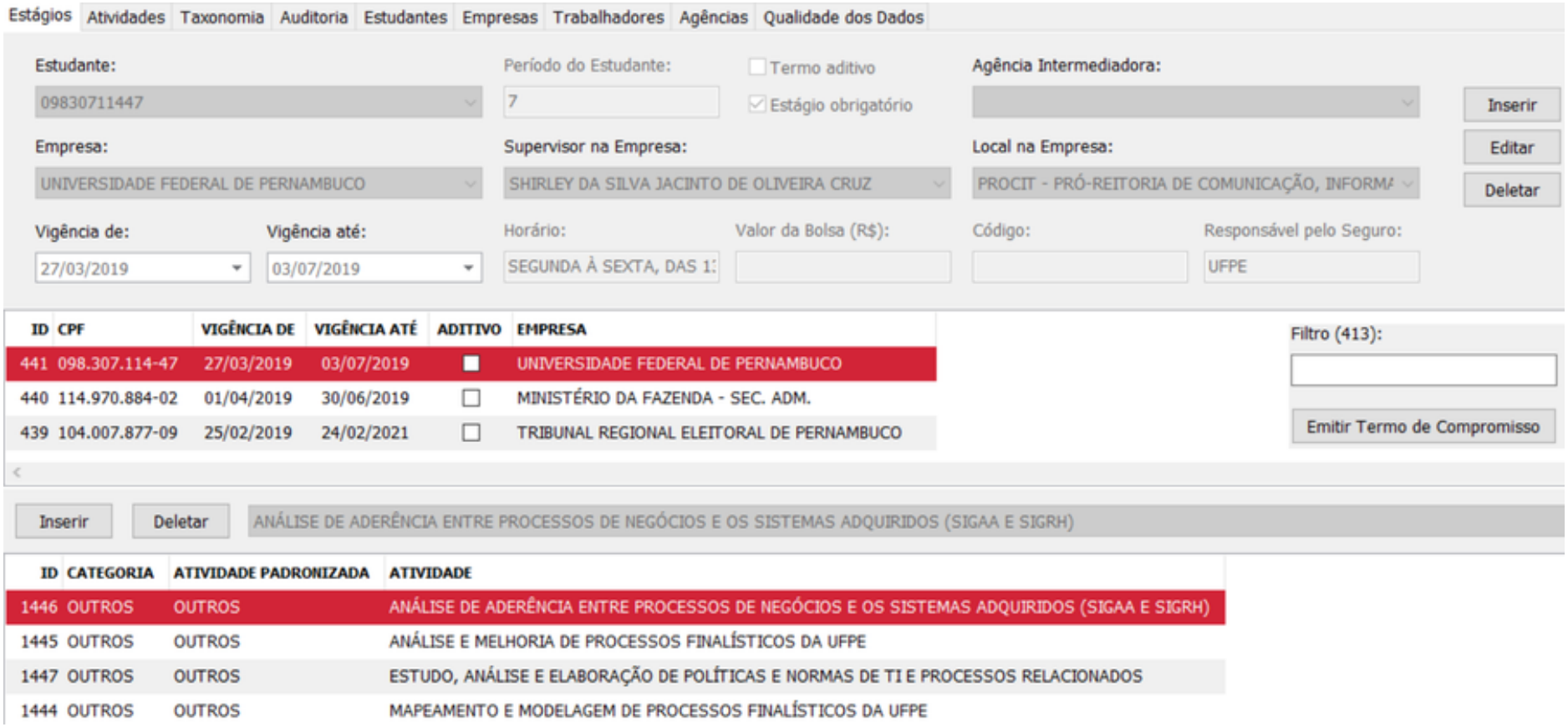
```
SELECT sp_cadastro ('00394494003313' -- número do CNPJ da agência intermediadora de estágios,
'Agencia X'         -- nome fantasia da agência intermediadora de estágios,
'LTDA'              -- tipo da empresa concedente de estágio,
'Recife'            -- cidade da empresa concedente de estágio,
'Cajueiro'          -- bairro da empresa concedente de estágio,
TRUE                -- informa se a empresa possui algum convênio com a universidade,
'8121373952'        -- telefone principal da empresa concedente de estágio,
'8134436358'        -- telefone alternativo da empresa concedente de estágio,
'teste@teste.com'  -- e-mail da empresa concedente de estágio,
'PE'               -- unidade da federação (estado) da empresa concedente de estágio,
'Rua Teste'        -- logradouro da empresa concedente de estágio,
'Empresa Teste'    -- razão social da empresa concedente de estágio,
'Empresa Teste'    -- nome fantasia da empresa concedente de estágio,
'00394494003313'   -- número do CNPJ da empresa concedente de estágio,
'50030-230'        -- CEP da empresa concedente de estágio,
'11.12.7-00'       -- CNAE da empresa concedente de estágio,
'Matriz'           -- local na empresa concedente de estágio onde o mesmo será realizado,
'Trabalhador teste'-- nome do funcionário que será o representante e/ou supervisor do estágio,
'Cargo teste'      -- cargo do representante e/ou supervisor do estágio,
TRUE               -- informa se o funcionário a ser cadastrado é representante da empresa,
'M'                -- sexo do estudante que participará do estágio (M ou F),
'07557808410'      -- CPF do estudante que participará do estágio,
'Teste'            -- horário do estágio (ex: seg. à sex; das 10h às 13h),
'2018-07-20'       -- data de início da vigência do estágio,
'2018-07-20'       -- data do término da vigência do estágio,
FALSE              -- informa se o estágio é aditivo de algum outro estágio,
TRUE               -- informa se o estágio é do tipo obrigatório,
114.15             -- valor da bolsa de estágio,
'123456'           -- código do estágio,
'Universidade'     -- informa quem será responsável pelo seguro do estagiário,
8                  -- período do estudante que participará do estágio,
'Teste'            -- descrição da classificação geral da atividade realizada no estágio,
'Teste'            -- descrição da classificação da atividade realizada no estágio,
'Teste'            -- nome da atividade realizada no estágio)
```

Fonte: elaborado pelo autor, 2019

Os procedimentos de armazenamento deste trabalho recebem os valores por parâmetro, ou seja, através da chamada da função. É possível ainda que um procedimento acione outro, de forma a garantir a inserção correta dos dados. O *stored procedure* do Código 4.3 realiza um cadastro de estágio.

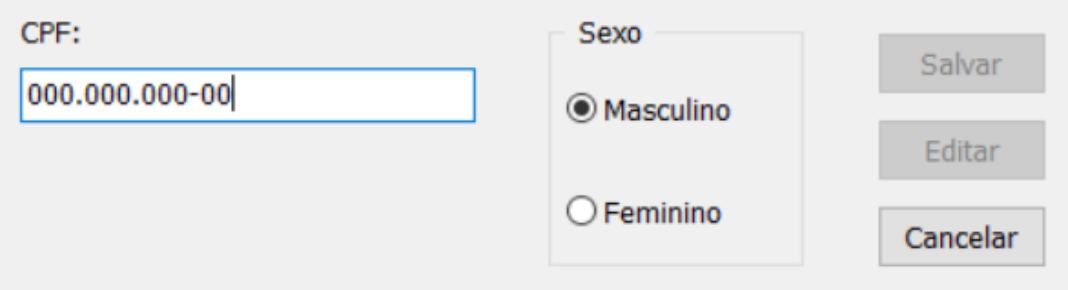
Já a Interface Gráfica de Usuário (GUI), facilita a inserção dos dados, verificação de erros, alterações, remoções e visualizações de registros, conforme a Figura 15. A distribuição uniforme e intuitiva, dividindo as entidades do estágio em abas e seus atributos dentro delas tornou os processos de recuperação, inserção, alteração e exclusão de registros mais fáceis. Por exemplo, na Figura 16 foi realizada uma tentativa de inserção de número de CPF incorreto. Enquanto o dado permanecer incorreto, o botão “Salvar” permanece desabilitado. Isso ocorre porque a função verificadora de número CPF/CNPJ detectou se tratar de uma incorretude, acionando o procedimento do *software*.

**Figura 15 – Software Agadê Estágios**




Fonte: modificado pelo autor a partir de captura de tela, 2019

**Figura 16 – Tentativa de inserção de CPF incorreto**




Fonte: modificado pelo autor a partir de captura de tela, 2019

## 4.5 Auditoria

A auditoria de dados deste trabalho consistiu em analisar manualmente cada um dos termos de estágio cadastrados na base de dados. Primeiramente, adicionou-se uma coluna intitulada 'auditado' com tipo de dados *timestamp without timezone* em todas as tabelas da base, não apenas as tabelas derivadas das entidades, como também dos relacionamentos entre eles. Em seguida, o documento físico contendo o termo de estágio era analisado e cada registro do documento sofria comparação com o registro correspondente na base de dados, a fim de verificar sua igualdade. Quando o registro é auditado manualmente e está correto, o auditor usa o *software* para registrar a auditoria. Nesse momento, essocia-se automaticamente ao campo 'auditado' o dia e a hora da auditoria. Registros ainda não auditados possuem valores nulos. Algumas entidades em especial foram analisadas, uma vez que durante o processo verificou-se que muitos dados foram inseridos de forma incompleta, incorreta ou desatualizada. No caso das empresas, todas foram consultadas na base de dados da Receita Federal do Brasil (www.receita.fazenda.gov.br). Já os estudantes foram consultados através do sistema sig@ da UFPE. As demais inconsistências eram consultadas através da *Web*, principalmente em portais de busca de conteúdo, como o Google (www.google.com). A auditoria também serviu como parâmetro de qualidade da base de dados, uma vez que os dados são verificados, correspondendo com os termos físicos de estágio.

A auditoria de dados deste trabalho também contou com a assistência de um *software* próprio, intitulado Agadê Estágios, que utilizou de recursos visuais para a marcação dos registros auditados exibindo-os em cor verde, laranja ou vermelho, segundo a Figura 17. As cores dos indicadores são modificados de acordo com o grau de completude apresentado na Tabela 16.

**Figura 17 – Definição de cores na verificação do índice de completude**

| TABELA | COLUNA | ÍNDICE DE COMPLETUDE (%) | TOTAL | COMPLETOS |
|---|---|---|---|---|
| atividades_categorias | auditado | 100,0 | 7 | 7 |
| atividades_padronizadas | auditado | 98,3 | 171 | 168 |
| estagios | codigo | 48,9 | 413 | 202 |

Fonte: captura de tela do software Agadê Estágios, 2019

**Tabela 16 – Níveis de completude**

| Cor | Grau de completude |
|---|---|
| Verde | 100% |
| Laranja | 50 à 99,9% |
| Vermelho | 0 à 49,9% |

Fonte: elaborado pelo autor, 2019

### 4.5.1 Indicador de completude de colunas não-obrigatórias

Além de monitorar a qualidade de dados através da contagem de dados auditados, o indicador de completude verifica a quantidade de registros que foram inseridos nas colunas não-obrigatórias da base de dados.

Para execução da função que verifica a completude de dados de colunas não-obrigatórias, foi necessária a criação de um tipo específico para atender as exigências. O Quadro 24 mostra a instrução para criação do novo tipo.

Quadro 24 – Criação de tipo de dado personalizado para medir o índice de completude de colunas não-obrigatórias

```
CREATE TYPE public.dados AS
   (total_registros numeric,
    total_nulos numeric,
    completude numeric,
    coluna character varying(100),
    tabela character varying(100));
```

Fonte: elaborado pelo autor, 2019

O Quadro 25 corresponde à função que efetua a verificação de completude de colunas não-obrigatórias de toda a base de dados. Primeiramente, todas as tabelas do banco de dados são selecionadas e em seguida verifica-se a quantidade total de registros. Depois, efetua-se uma contagem total de linhas de cada coluna que aceite valores nulos. Finalmente, é feito um cálculo simples em que a quantidade de linhas com valor nulo (*NULL*) é divida pela quantidade total de registros, obtendo-se o índice de completude daquela tabela, caso a tabela possua um ou mais registros.

Quadro 25 – Função para verificar completude das colunas

```
1   CREATE OR REPLACE FUNCTION public.fn_verifica_completude()
2     RETURNS SETOF dados AS
3   $BODY$DECLARE
4     nome_tabela TEXT;
5     coluna TEXT;
6     dados_temporarios dados;
7   BEGIN
8   FOR nome tabela in (
9     SELECT table_name
10    FROM information_schema.tables
11    WHERE table_schema = 'public' AND table_type = 'BASE TABLE')
12  LOOP
13      dados_temporarios.tabela := nome_tabela;
14      EXECUTE 'SELECT count(*)::numeric FROM ' || nome_tabela INTO dados_temporarios.total_registros;
15    FOR coluna IN (
16      SELECT column_name
17      FROM information_schema.columns
18      WHERE table_schema = 'public'
19      AND table_name   = nome_tabela
20      AND is_nullable  = 'YES')
21    LOOP
22        dados_temporarios.coluna := coluna;
23        EXECUTE 'SELECT count(' || coluna || ')::numeric FROM ' || nome_tabela INTO dados_temporarios.total_nulos;
24        IF dados_temporarios.total_registros > 0 THEN
25          dados_temporarios.completude := round((dados_temporarios.total_nulos * 100.0 / dados_temporarios.total_registros),2);
26        ELSE
27          dados_temporarios.completude := NULL;
28        END IF;
29        RETURN NEXT dados_temporarios;
30    END LOOP;
31  END LOOP;
32  RETURN;
33  END;$BODY$
34    LANGUAGE plpgsql
```

Fonte: elaborado pelo autor, 2019

A linha 1 informa que uma função de nome “fn_verifica_completude” será criada e retornará um conjunto de dados (linha 2). Em seguida, três variáveis são declaradas: “nome_tabela” e “coluna” do tipo texto, e “dados_temporários” do novo tipo “dados”. O trecho principal da função compreende as linhas 7 à 34. Entre as linhas 8 e 11, a variável “nome_tabela” (que conterá o nome das tabelas da base de dados de estágios) busca todos os nomes de tabelas (linha 9) que fazem parte do esquema principal (linha 10), onde o nome do esquema é “public” e o tipo de tabela é “base table” (linha 11). O próximo passo entra em um loop contínuo que se extende da linha 12 à 31. Nela, a instância “tabela” da variável “dados_temporarios” recebe o nome da tabela pesquisada (linha 13) e executa uma contagem de tuplas da tabela. Essa contagem é armazenada na instância “total_registros” da variável “dados_temporários” (linha 14). Entre as linhas 15 à 20, a variável local “coluna” (que conterá o nome das colunas da tabela da variável “nome_tabela”) recebe o nome de todas as colunas (linha 16) do esquema “public” (18) e da tabela “nome_tabela” que permitam valores nulos (linha 20). A partir daí, entre as linhas 21 à 31, a instância “coluna” da variável “dados_temporarios” recebe o nome das colunas (linha 22) e executa uma contagem das que estão na tabela “nome_tabela” e os armazenam na instância “total_nulos” da variável “dados_temporarios”. Entre as linhas 24 e 28 é feita uma verificação na quantidade total de registros contabilizados. Caso o número total de registros seja superior a zero (linha 24), a instância “completude” da variável “dados_temporários” recebe o valor calculado pela divisão de todos os itens nulos (“total_nulos” da variável “dados_temporarios”) e o número total de registros daquela tabela. Caso contrário, a função retornará *NULL* (nulo) (linha 7). Esta função percorrerá todas as tabelas da base de dados executando a função e retornando os resultados (linha 29).

Essa função poderá ser utilizada em qualquer base de dados que pretende obter o índice de completude, desde que a base em questão esteja em uma linguagem que suporte a função.

A Figura 18 mostra a função de verificaçao de completude através do *software* intitulado Agadê Estágios. A coluna “TABELA” lista todas as tabelas da base de dados em que a “COLUNA” aceite valores nulos; o “ÍNDICE DE COMPLETUDE (%)” é o resultado percentual da divisão entre a coluna “COMPLETOS”, que lista a quantidade de registros preenchidos daquela coluna, e “TOTAL”, que é a quantidade total de registros daquela coluna. Uma vez que qualquer coluna (exceto ’auditado’) atinja o índice máximo de completude, sua propriedade pode ser alterada para preenchimento obrigatório.

**Figura 18 – Índice de completude de colunas das tabelas**

| TABELA | COLUNA | ÍNDICE DE COMPLETUDE (%) | TOTAL | COMPLETOS |
|---|---|---|---|---|
| atividades_categorias | auditado | 100,0 | 7 | 7 |
| atividades_padronizadas | auditado | 98,3 | 171 | 168 |
| empresas_cnae_secundarias | auditado | 94,5 | 456 | 431 |
| empresas | auditado | 94,2 | 121 | 114 |
| empresas_locais | auditado | 94,1 | 170 | 160 |
| empresas_trabalhadores | auditado | 93,2 | 325 | 303 |
| trabalhadores | auditado | 93,1 | 317 | 295 |
| estagios | auditado | 92,7 | 413 | 383 |
| estagios_atividades | auditado | 92,5 | 1624 | 1503 |
| estudantes | auditado | 92,5 | 240 | 222 |
| estagios | horario | 88,9 | 413 | 367 |
| agencias | auditado | 87,5 | 8 | 7 |
| trabalhadores | cargo | 81,1 | 317 | 257 |
| empresas | nome_fantasia | 71,1 | 121 | 86 |
| estagios | codigo | 48,9 | 413 | 202 |
| empresas | email | 47,9 | 121 | 58 |
| estagios | bolsa | 40,9 | 413 | 169 |
| estagios | id_agencia | 34,4 | 413 | 142 |
| empresas | telefone_alternativo | 27,3 | 121 | 33 |
| atividades | auditado | 0,0 | 1225 | 0 |

Fonte: captura de tela do software Agadê Estágios, 2019

A maioria das colunas encontram-se quase completamente auditadas, com mais de 90% da auditoria realizada. As colunas “código” da tabela “estágios” está com 48,9% porque somente os termos de compromisso de estágio celebrados por agências intermediadoras possuem este recurso. A coluna “email” da tabela “empresas” também está com esse nível de completude porque os termos de compromisso de estágio não possuem tal campo, sendo necessária consulta externa no site da Receita Federal e, mesmo assim, nem todas as empresas cadastraram algum e-mail. Por relacionar estágios obrigatórios e não-obrigatórios, a coluna “bolsa” da tabela “estagios” também está com o índice baixo. A coluna “id_agencia” da tabela “estagios” também não sofrerá mudança significativa, pelo fato de alguns estágios serem celebrados diretamente entre

as empresas concedentes e a instituição de ensino, sem necessidade de uma agência intermediadora. O “telefone_alternativo” também configura uma baixa porcentagem porque tal campo não está cadastrado em todas as empresas.

### 4.5.2 Indicador de completude para relacionamentos entre tabelas

Com o indicador de completude de relacionamentos, é possível acompanhar qual grau que os registros se relacionam a outros registros. Relacionamentos de cardinalidades obrigatórias (ou seja, no mínimo 1) não são garantidas pelo modelo relacional como, por exemplo, relacionamento obrigatório de cardinalidade (1,n) para (1,n). O indicador de completude de relacionamento pode mostrar se essa regra foi respeitada e em qual grau.

Para execução da função que verifica a completude de dados de relacionamentos entre tabelas, foi necessária a criação de um tipo específico para atender as exigências. O Quadro 26 mostra a instrução para criação do novo tipo.

Quadro 26 – Criação de tipo de dado personalizado para medir o índice de completude de relacionamentos entre tabelas

```
CREATE TYPE public.resultado_completude AS
   (total_registros numeric,
    total_nulos numeric,
    completude numeric,
    coluna character varying(200),
    tabela character varying(200),
    fk_coluna character varying(200),
    fk_tabela character varying(200),
    chave_estrangeira character varying(200));
```

Fonte: elaborado pelo autor, 2019

Os Quadros 27 e 28 correspondem à função que efetua a verificação de completude de relacionamentos entre tabelas de toda a base de dados. Primeiramente, lista as Chaves Estrangeiras (FK) utilizadas em duas colunas de tabelas diferentes que possuem um relacionamento entre si. Em seguida, realiza o cálculo de registros vazios divididos pelo número total de registros, mostrando o índice de completude das chaves.

Quadro 27 – Função para verificação de completude dos relacionamentos de tabelas (parte 1)

```
1   CREATE OR REPLACE FUNCTION public.fn_verifica_completude_chave_estrangeira()
2     RETURNS SETOF resultado_completude AS
3   $BODY$DECLARE
4     constraint_name varchar;
5     table_id oid;
6     fk_table_id oid;
7     column_ids int2[];
8     fk_column_ids int2[];
9     column_id int2;
10    column_name varchar;
11    table name varchar;
12    fk_column_name varchar;
13    fk_table_name varchar;
14    sql_conditions varchar;
15    sql_fields varchar;
16    sql_agg_fields varchar;
17    sql_columns varchar;
18    sql_fk_columns varchar;
19    i int;
20    resultado resultado_completude;
21  BEGIN
22
23  FOR constraint_name, table_id, column_ids, fk_table_id, fk_column_ids IN (
24      SELECT conname, conrelid, conkey, confrelid, confkey
25      FROM pg_constraint WHERE contype = 'f')
26  LOOP
27    table_name := table_id::regclass;
28    fk_table_name := fk_table_id::regclass;
29
30    sql_conditions := '(1 = 1)';
31    sql_fields := '';
32    sql_agg_fields := '';
33    sql_columns := '';
34    sql_fk_columns := '';
35
36    FOR i IN 1..array_length(column_ids, 1) LOOP
37      SELECT attname INTO column_name FROM pg_attribute WHERE attrelid = table_id AND attnum = column_ids[i];
```

Fonte: elaborado pelo autor, 2019

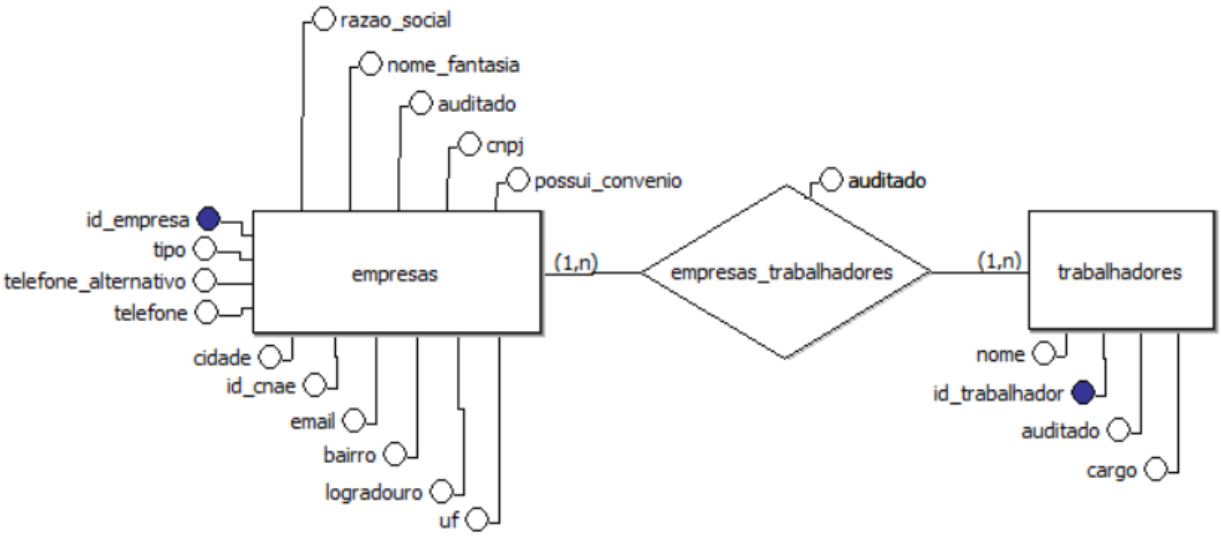
Quadro 28 – Função para verificação de completude dos relacionamentos de tabelas (parte 2)

```
38      SELECT attname INTO fk_column_name FROM pg_attribute WHERE attrelid = fk_table_id AND attnum = fk_column_ids[i];
39      sql_conditions := sql_conditions || format(' AND (et.%s = e.%s)', column_name, fk_column_name);
40      sql_fields := sql_fields || format('e.%s AS x%s, et.%s AS y%s, ', fk_column_name, i, column_name, i);
41      sql_agg_fields := sql_agg_fields || format('a.x%s, ', i);
42      sql_columns := sql_columns || column_name || ', ';
43      sql_fk_columns := sql_fk_columns || fk_column_name || ', ';
44    END LOOP;
45
46    EXECUTE 'SELECT count(*)::numeric FROM ' || fk_table_name INTO resultado.total_registros;
47    EXECUTE '
48    SELECT count(*)::numeric FROM (
49      SELECT ' || sql_agg_fields || ' count(a.y1)
50      FROM (
51        SELECT ' || sql_fields || ' 1
52        FROM ' || fk_table_name || ' e LEFT OUTER JOIN ' || table_name || ' et ON ' || sql_conditions || '
53      ) AS a
54      GROUP BY ' || sql_agg_fields || ' 1
55      HAVING count(a.y1) > 0
56    ) AS b' INTO resultado.total_nulos;
57
58    IF resultado.total_registros > 0 THEN
59      resultado.completude := round((resultado.total_nulos * 100.0 / resultado.total_registros), 2);
60    ELSE
61      resultado.completude := NULL;
62    END IF;
63
64    resultado.tabela := table_name;
65    resultado.coluna := sql_columns;
66    resultado.fk_tabela := fk_table_name;
67    resultado.fk_coluna := sql_fk_columns;
68    resultado.chave_estrangeira := constraint_name;
69
70    RETURN NEXT resultado;
71  END LOOP;
72
73  END;$BODY$
74    LANGUAGE plpgsql
```

Fonte: elaborado pelo autor, 2019

Para facilitar a compreensão do código apresentado, utilizaremos como exemplo o relacionamento da Figura 19 retirada da base de dados e o *software* intitulado Agadê Estágios como exemplo.

**Figura 19 – Relacionamento entre empresas e trabalhadores**




Fonte: elaborado pelo autor, 2019

Os dados do indicador de qualidade do relacionamento estão de acordo com a Tabela 17.

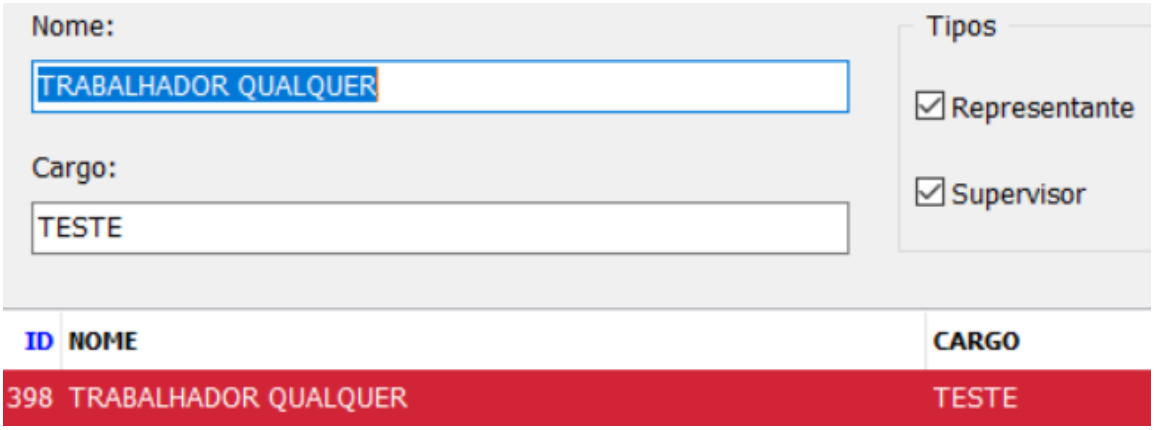
**Tabela 17 – Situação dos parâmetros antes de inserção controlada**

| | |
|---|---|
| Chave Estrangeira | fk_empresas_trabalhadores_id_trabalhador |
| Tabela | empresas_trabalhadores |
| Coluna | id_trabalhador |
| FK Tabela | trabalhadores |
| FK Coluna | id_trabalhador |
| Índice de Completude (%) | 100,0 |
| Total | 317,0 |
| Completos | 317,0 |

Fonte: elaborado pelo autor, 2019

Um cadastro de trabalhador é realizado com o objetivo de entender o comportamento do indicador de qualidade, conforme Figura 20.

**Figura 20 – Cadastro de usuário como trabalhador**




Fonte: captura de tela do software Agadê Estágios, 2019

De acordo com a Tabela 18, uma vez que o trabalhador foi cadastrado, os indicadores de qualidade de relacionamento sofrem uma leve alteração.

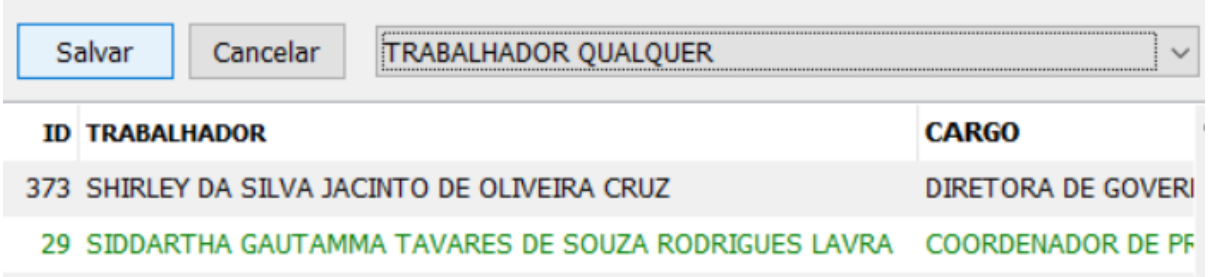
**Tabela 18 – Situação dos parâmetros após inserção controlada**

| | |
|---|---|
| Chave Estrangeira | fk_empresas_trabalhadores_id_trabalhador |
| Tabela | empresas_trabalhadores |
| Coluna | id_trabalhador |
| FK Tabela | trabalhadores |
| FK Coluna | id_trabalhador |
| Índice de Completude (%) | 99,7 |
| Total | 318,0 |
| Completos | 317,0 |

Fonte: elaborado pelo autor, 2019

Nota-se que o índice de completude o total de registros foram afetados. Ao realizar o cadastramento de um trabalhador, a quantidade total de registros “id_trabalhador” da tabela “trabalhadores” passou a ser 318, enquanto a quantidade total de registros “id_trabalhador” da tabela “empresas_trabalhadores” permaneceu em 317. Isso ocorre porque o trabalhador ainda não foi vinculado à empresa, correspondete à chave “fk_empresas_trabalhadores_id_trabalhador”. O procedimento para aumentar o índice de completude de registros é vincular o trabalhador recém cadastrado com uma empresa na guia “empresa” do *software* Agadê Estágios, conforme Figura 21.

**Figura 21 – Trabalhador sendo vinculado à empresa**




Fonte: captura de tela do software Agadê Estágios, 2019

Na Figura 22 é possível aferir, por exemplo, que todos os trabalhadores da tabela “trabalhadores_supervisores” estão listados como supervisor na tabela estágios. É possível verificar também que nem todos os estágios possuem atividades (tabela “estagios_atividades” e fk_coluna “id_estagio”).

**Figura 22 – Índice de completude de chaves estrangeiras das tabelas**

| CHAVE ESTRANGEIRA | TABELA | COLUNA | FK TABELA | FK COLUNA | ÍNDICE DE COMPLETUDE (%) | TOTAL | COMPLETOS |
|---|---|---|---|---|---|---|---|
| fk_atividades_id_atividade_padronizada | atividades | id_atividade_padronizada, | atividades_padronizadas | id_atividade_padronizada, | 100,0 | 171,0 | 171,0 |
| fk_atividades_padronizadas_id_atividade_categoria | atividades_padronizadas | id_atividade_categoria, | atividades_categorias | id_atividade_categoria, | 100,0 | 7,0 | 7,0 |
| fk_empresas_locais_id_empresa | empresas_locais | id_empresa, | empresas | id_empresa, | 100,0 | 121,0 | 121,0 |
| fk_empresas_trabalhadores_id_empresa | empresas_trabalhadores | id_empresa, | empresas | id_empresa, | 100,0 | 121,0 | 121,0 |
| fk_empresas_trabalhadores_id_trabalhador | empresas_trabalhadores | id_trabalhador, | trabalhadores | id_trabalhador, | 100,0 | 317,0 | 317,0 |
| fk_estagios_id_agencia | estagios | id_agencia, | agencias | id_agencia, | 100,0 | 8,0 | 8,0 |
| fk_estagios_id_estudante | estagios | id_estudante, | estudantes | id_estudante, | 100,0 | 240,0 | 240,0 |
| fk_estagios_id_local | estagios | id_local, | empresas_locais | id_local, | 100,0 | 170,0 | 170,0 |
| fk_estagios_id_supervisor | estagios | id_supervisor, | trabalhadores_supervisores | id_trabalhador, | 100,0 | 214,0 | 214,0 |
| fk_estagios_atividades_id_atividade | estagios_atividades | id_atividade, | atividades | id_atividade, | 98,5 | 1225,0 | 1207,0 |
| fk_estagios_atividades_id_estagio | estagios_atividades | id_estagio, | estagios | id_estagio, | 79,7 | 413,0 | 329,0 |
| fk_trabalhadores_supervisores_id_trabalhador | trabalhadores_supervisores | id_trabalhador, | trabalhadores | id_trabalhador, | 67,5 | 317,0 | 214,0 |
| fk_empresas_cnae_secundarias_id_empresa | empresas_cnae_secundarias | id_empresa, | empresas | id_empresa, | 66,1 | 121,0 | 80,0 |
| fk_trabalhadores_representantes_id_trabalhador | trabalhadores_representantes | id_trabalhador, | trabalhadores | id_trabalhador, | 52,1 | 317,0 | 165,0 |
| fk_empresas_cnae_secundarias_id_cnae | empresas_cnae_secundarias | id_cnae, | cnae | id_cnae, | 15,8 | 1329,0 | 210,0 |
| fk_empresas_cnae_id_cnae | empresas | id_cnae, | cnae | id_cnae, | 5,3 | 1329,0 | 71,0 |

Fonte: captura de tela do software Agadê Estágios, 2019

## 4.6 Problemas e limitações

Durante todo o processo de concepção, ajuste, criação e inserção de dados na base de dados especializada, algumas limitações dificultaram o andamento dos processos.

Primeiramente, o simples fato de estar nos documentos físicos, já limitava o acesso aos dados, porque era necessário acesso aos documentos primeiro. Isso demandou presença de uma pessoa na secretaria do curso para verificação dos termos de compromisso. A extração foi completamente manual, sem utilização de *software* de apoio, exigindo atenção redobrada para evitar quaisquer errros por parte do responsável pela extração. Isso demandou também um longo prazo até que todos os termos fossem verificados. Os documentos físicos estavam desorganizados, dificultando qualquer iniciativa em recuperar quaisquer dados. Destarte, continham campos semi-estruturados e vários tipos de *layout,* que variavam de acordo com o tipo de termo de compromisso. Os dados estavam desestruturados e/ou incompletos.

Ao utilizar Sistemas de Gerenciamento de Banco de Dados (SGBD), verificou-se como se comportam com mensagens simples. Em alguns casos, como o phpPgMyAdmin, essas mensagens eram consideradas uma exceção e toda a operação de inserção, ainda que correta, não era completada. Foi necessário então efetuar a migração para uma plataforma mais amigável. O pgAdmin III não apresentou problemas com as mensagens, principalmente das *stored procedures*.

No processo de auditoria, uma série de erros foram constatados. Entre eles, erros de digitação, extrações erradas da informação, dados não encontrados nos termos de compromisso, extrações ou inserções incompletas e fontes externas incompletas ou desatualizadas.

## 4.7 Resultados

Utilizando os conceitos apresentados neste trabalho foi criada uma base de dados que registra os dados dos termos de compromisso de estágio de Gestão da Informação da UFPE no período entre 2011.1 e 2019.1. Esta base possui registrado 413 estágios, 240 estudantes de Gestão da Informação, 1.225 atividades de estágio, 121 empresas concedentes de estágio e 317 profissionais entre representantes das empresas concedentes e supervisores dos estágios.

Após o preenchimento da base, a auditoria realizou um processo de auditoria meticulosa, onde todos os termos de compromisso de estágio foram analisados novamente e seus dados foram comparados aos cadastrados na base de dados. Os dados que não estavam presentes na base de dados e na documentação física eram

consultados em bases externas, como a Receita Federal do Brasil[1] e buscadores *Web*, como o Google[2]. Com isso, vários dados complementares foram adicionados aos registros, como telefone, e-mail e endereço.

Utilizando o indicador de completude de colunas não-obrigatórias constatou-se que mais de 90% de todas os registros foram auditados; alguns dados tornaram-se obrigatórios após o índice de completude atingir 100%, como o telefone das empresas concedentes de estágio e algumas constatações, como o baixo índice de completude entre algumas colunas, entre elas os códigos de estágios, porque algumas modalidades de estágio não exigem cadastro de código. Ou ainda, o fato de menos de 42% dos estágios possuírem o valor de bolsa auxílio cadastrados. A justificativa para tal índice é o fato da maioria dos estágios serem voluntários.

Utilizando os indicacador de completude de relacionamentos entre tabelas constatou-se que após a inserção de dados na base, muitas tabelas não estavam bem relacionadas entre si, o que demandou uma segunda análise dos termos de compromisso atrás de verificar que tipo de informação se mostrou incompleta. Alguns relacionamentos entre tabelas permaneceu com índice de completude baixo, como o relacionamento entre as tabelas estágios e atividades. A justificativa é que vários estágios não possuem atividade cadastrada, principalmente pela falta do mesmo, seja porque perdeu-se o relatório de atividades ou ausência deste, seja pelo fato do dado não estar presente no termo de compromisso de estágio. O mesmo constatou-se entre as tabelas empresas e cnae, já que são muitas atividades cnae cadastradas (que contemplam todas as atividades possíveis das empresas) e poucas empresas cadastradas.

[1] www.receita.fazenda.gov.br
[2] www.google.com.br

# 5 Conclusão

A sociedade ainda precisa conviver com dados e informações em suportes físicos e digitais enquanto ocorre progressivamente o processo de digitalização. Assim, este trabalho abordou alguns dos desafios que ocorrem nos processos e propôs novos conceitos que busquem resolver os problemas.

Uma base de dados de termos de estágios do curso de Gestão da Informação da UFPE foi desenvolvida com intuito de armazenar os registros dos termos de compromisso de estágios e resolver alguns problemas, como recuperação ágil da informação, alteração e exclusão de dados, corretude na inserção de dados, completude e qualidade dos dados.

O principal problema abordado foi o da digitalização de dados a partir de documentos físicos, onde a qualidade dos dados foi verificada e metrificada. A solução consistiu em 3 etapas, que foram desde projetar e construir uma base de dados em conjunto com um *software* de apoio, passando por extração e inserção dos dados, até adotar uma técnica de auditoria de dados assistida por *software*, integrada com indicadores de qualidade de dados: indicadores de completude para colunas não-obrigatórias e um novo conceito de indicador de completude para relacionamentos entre tabelas. Com a auditoria, os campos incompletos foram preenchidos reexaminando o documento original, os relatórios de estágios e outras fontes. O progresso da auditoria foi acompanhado pelos indicadores.

Como proposta de trabalhos futuros, recomenda-se utilizar o novo conceito apresentado em outras bases de dados. Além disso, recomenda-se a criação de uma estrutura que organize as atividades de estágios de acordo com técnicas já conhecidas, como taxonomia e tesauro.

## Referências

ABNT. **ABNT NBR ISO 9000 - Sistemas de gestão da qualidade – Fundamentos e vocabulário**. 2000. Disponível em: http://www.jvasconcellos.com.br/unijorge/wp-content/uploads/2012/09/Abnt-Nbr-Iso-9000.pdf. Acesso em: 05 junho 2019.

ALBRECHT, R. F.; OHIRA, M. L. B. Bases de dados: metodologia para seleção e coleta de documentos. **ACB: Biblioteconomia em Santa Catarina**, Santa Catarina, v. 5, n. 5, p. 131 – 144, janeiro 2000.

ALECRIM, E. **Bancos de dados são mais importantes nas nossas vidas do que a gente imagina**. 2018. Disponível em: https://tecnoblog.net/245120/banco-de-dados-importancia/. Acesso em: 12 dez 2018.

ALVES, G. F. de O. **A história dos bancos de dados**. 2013. Disponível em: https://dicasdeprogramacao.com.br/a-historia-dos-bancos-de-dados/. Acesso em: 12 dez 2018.

AMARAL, F. Introdução à ciência de dados: mineração de dados e big data. In: AMARAL, F. (ed.). **Introdução à ciência de dados**: mineração de dados e big data. 1. ed. Rio de Janeiro: Alta Books, 2016. cap. 3, p. 23 – 37. ISBN 978-85-7608-934-6.

ANTUNES, A. **Sociedade da Informação**. 2008. Disponível em: http://www4.fe.uc.pt/fontes/trabalhos/2008007.pdf. Acesso em: 14 junho 2019.

ARAUJO, E. **Função para validar Cpf/Cnpj – PostgreSQL**. 2010. Disponível em: https://eacshm.wordpress.com/2010/08/25/funcao-para-validar-cpfcnpj-postgresql/. Acesso em: 14 junho 2019.

AROUCK, O. **Atributos de qualidade da informação**. 2011. 117 p. Dissertação (pós-graduação em ciência da Informação) — Universidade de Brasília. Disponível em: http://repositorio.unb.br/bitstream/10482/9501/1/2011_OsmarCarmoArouckFerreira.pdf. Acesso em: 12 junho 2019.

ATANAZIO, J. **PostgreSQL - SQL Básico**. 2018. Apostila on-line. Disponível em: https://github.com/juliano777/pgsql_fs2w/blob/master/postgresql_sql_basico.pdf. Acesso em: 11 dez 2018.

BARANAUSKAS, J. A. **Extração de Conhecimento & Mineração de Dados**. 2013. Disponível em: http://dcm.ffclrp.usp.br/~augusto/teaching/ami/AM-I-KDD-DM.pdf. Acesso em: 05 junho 2019.

BARBOSA, L. **Recuperação de Informação: Extração de Dados na Web**. 2019. 63 slides. Disponível em: https://www.cin.ufpe.br/~luciano/courses/ir/extracao_texto.pdf. Acesso em: 02 julho 2019.

BARROS, F. M. M. de. **Tecnologia de banco de dados**. 2019. Apostila. Disponível em: http://adm.online.unip.br/img_ead_dp/6812.doc. Acesso em: 30 junho de 2019.

BARROS, P. **O que é ACID?** 2016. Disponível em: https://medium.com/opensanca/o-que-%C3%A9-acid-59b11a81e2c6. Acesso em: 13 dez 2018.

BRASIL. Tribunal de Contas da União - TCU. Portaria n. 455-GP, de 30 de setembro de 1998. **Manual de Auditoria de Sistemas**, Brasília, p. 12 – 105, 1998. Disponível em: http://lncc.br/~borges/ist/SAS/AuditoriaSistemasTCU.pdf. Acesso em: 13 junho 2019.

CARVALHO, G. C. de. **Sistema de Banco de Dados Orientado a Objeto**. 2011. 51 p. Monografia (Tecnólogo em Processamento de Dados) — Faculdade de Tecnologia de São Paulo. Disponível em: http://www.fatecsp.br/dti/tcc/tcc0002.pdf. Acesso em: 17 dez 2018.

CODD, E. F. A relational model of data for large shared data banks. **Association of Computer Machinery (ACM) Journal - Communications of the ACM**, IBM Research Laboratory, San Jose, v. 13, n. 6, p. 377 – 387, Junho 1970. Disponível em: https://www.seas.upenn.edu/~zives/03f/cis550/codd.pdf. Acesso em: 13 dez 2018.

CORREIA, L. O. dos S.; PADILHA, B. M.; VASCONCELOS, S. M. L. Métodos para avaliar a completitude dos dados dos sistemas de informação em saúde do Brasil: uma revisão sistemática. **Ciência & Saúde Coletiva**, Scielo, São Paulo, v. 19, n. 11, p. 4467 – 4478, Novembro 2014. Disponível em: https://doi.org/10.1590/1413-812320141911.02822013. Acesso em: 14 junho 2019.

DAMAS, L. **SQL, Structured Query Language**. 6. ed. Rio de Janeiro: LTC, 2007. ISBN 978-85-216-1558-3.

FIGUEIREDO, D. L. M. **Qualidade e normalização**. Belo Horizonte: [s.n.], 2018. Disponível em: http://www.demc.ufmg.br/tec3/Qualidade%20e%20Normaliza%E7%E3o.doc. Acesso em: 13 junho 2019.

GERHARDT, T. E.; SILVEIRA, D. T. (org.). **Métodos de pesquisa**. 1. ed. Porto Alegre: Universidade Federal do Rio Grande do Sul, 2009. 120 p. ISBN 978-85-386-0071-8. Disponível em: http://www.ufrgs.br/cursopgdr/downloadsSerie/derad005.pdf. Acesso em: 17 julho 2019.

GOMES, E. H. **Linguagem SQL: Linguagem de Manipulação, Consulta e Controle de Dados**. 2018. Disponível em: http://ehgomes.com.br/disciplinas/bdd/sql.php. Acesso em: 11 dez 2018.

HEUSER, C. A. **Projeto de Banco de Dados**. 4. ed. Porto Alegre: Sagra Luzatto, 1998. v. 4. (Livros Didáticos, v. 4).

JIANG, J. Information Extraction from Text. In: JIANG, J. (ed.). **Mining Text Data**. Nova Iorque: Springer-Verlag, 2012. cap. 2, p. 11 – 41. ISBN 978-1-4614-3222-7. Disponível em: https://ink.library.smu.edu.sg/sis_research/1711.

LOPES, A. **Modelo Conceitual, Lógico e Físico, Entidade-Relacionamento**. 2012. 41 slides. Disponível em: http://docente.ifrn.edu.br/abrahaolopes/semestre-2012.1/4.401.1m-banco-de-dados/slides-modelo-conceitual-fisico-logico-er. Acesso em: 18 dez 2018.

MACHADO, O. A. **Qualidade da informação**: uma abordagem orientada para o contexto. 2013. 175 p. Tese (Ciências) — Universidade de São Paulo. Disponível em: http://www.teses.usp.br/teses/disponiveis/3/3141/tde-23052014-001437/. Acesso em: 12 junho 2019.

MADNICK, S. E. et al. Overview and Framework for Data and Information Quality Research. **ACM Journal of Data and Information Quality**, Nova Iorque, v. 1, n. 1, p. 1 – 22, junho 2009. ISSN 1936-1955. Disponível em: https://irandoc.ac.ir/sites/fa/files/attach/event/overviewandframeworkfordatamadnick.pdf. Acesso em: 12 junho 2019.

MARGARITOPOULOS, M. et al. Quantifying and Measuring Metadata Completeness. **Journal of The American Society for Information Science and Technology**, ASIS&T, v. 63, n. 4, p. 724 – 737, Dezembro 2011. Disponível em: https://onlinelibrary.wiley.com/doi/abs/10.1002/asi.21706. Acesso em: 03 julho 2019.

MARTELLI, R.; SANTANA FILHO, O. V.; CABRAL, A. de L. **Modelagem e banco de dados**. São Paulo: Senac, 2017. ISBN 978-85-396-1194-2.

MARTINS, R. R. de Q. **Auditoria de Dados**. 2009. 56 slides coloridos. Disponível em: https://portal.tcu.gov.br/lumis/portal/file/fileDownload.jsp?fileId=8A8182A14E01F8FC014E02CA13FA2AA0. Acesso em: 13 junho 2019.

MELLO, R. dos S. et al. Dados Semi-Estruturados. In: ANAIS. .., 2000, João Pessoa. **XV Simpósio Brasileiro de Banco de Dados / XIV Simpósio Brasileiro de Engenharia de Software - Mini-Cursos / Tutoriais**. João Pessoa, 2000. p. 475 – 513. Disponível em: https://www.ime.usp.br/~jef/semi-estruturado.pdf. Acesso em: 5 junho 2019.

MELO, I. V. de A. **Normalização de Bancos de Dados Relacionais**. 2011. Disponível em: http://www.dsc.ufcg.edu.br/~pet/jornal/maio2011/materias/recapitulando.html. Acesso em: 13 junho 2019.

MILANI, A. **PostgreSQL** : guia do programador. 1. ed. São Paulo: Novatec, 2008. ISBN 978-85-7522-157-0.

MILANI, A. **Construindo aplicações web com PHP e MySQL**. 1. ed. São Paulo: Novatec, 2010. ISBN 978-85-7522-219-5.

MONTEIRO, D. **Você sabe o que é uma FUNCTION?** 2018. Disponível em: http://db4beginners.com/blog/voce-sabe-o-que-e-uma-function/. Acesso em: 18 dez 2018.

MONTEIRO, L. P. **Dados Estruturados e Não Estruturados**. 2019. Disponível em: https://universidadedatecnologia.com.br/dados-estruturados-e-nao-estruturados/. Acesso em: 05 junho 2019.

ORTEGA, D. **The dimensions of data quality: completeness and duplication**. 2016. Disponível em: https://www.blazent.com/dimensions-data-quality-completeness-duplication/. Acesso em: 01 julho de 2019.

PAIM, I.; NEHMY, R. M. Q.; GUIMARÃES, C. G. Problematização do conceito "Qualidade" da Informação. **Perspec. Ci. Inf.**, v. 1, n. 1, p. 111 – 119, jan./jun. 1996. Disponível em: http://portaldeperiodicos.eci.ufmg.br/index.php/pci/article/download/8/27. Acesso em: 14 junho 2019.

PGADMIN DEVELOPMENT TEAM. **pgAdmin - PostgreSQL Tools**. 2019. Disponível em: https://www.pgadmin.org/. Acesso em: 17 julho 2019.

PHP. **phpPgAdmin - start**. 2019. Disponível em: http://phppgadmin.sourceforge.net/doku.php. Acesso em: 17 julho 2019.

PILLOU, J. **Os níveis de dados**. 2017. Glossário on-line de bando de dados. Disponível em: https://br.ccm.net/contents/63-os-niveis-de-dados. Acesso em: 17 dez 2018.

PISCITELLO, D. **Parte I: o que são metadados?** 2016. Blog do ICANN. Disponível em: https://www.icann.org/news/blog/parte-i-o-que-sao-metadados. Acesso em: 16 dez 2018.

RIBEIRO, L. C. **Banco de dados e SQL – Diferença entre Base de Dados e Banco de Dados**. 2018. Disponível em: https://www.luis.blog.br/banco-de-dados-e-sql-diferenca-entre-base-de-dados-e-banco-de-dados/. Acesso em: 08 out 2018.

RICARTE, I. L. M. **Estrutura de dados**. 2008. Disponível em: http://calhau.dca.fee.unicamp.br/wiki/images/0/01/EstruturasDados.pdf. Acesso em: 13 dez 2018.

SCHARDOSIM, L.; CURY, T. **Banco de dados relacional**. 2015. Disponível em: http://www.thiagocury.eti.br/disciplinas-conteudo/banco-de-dados-relacional/banco-de-dados-relacional.html#. Acesso em: 15 dez 2018.

SEBRAE. **Digitalização de documentos**. 2019. Disponível em: http://www.sebrae.com.br/sites/PortalSebrae/ideias/como-montar-um-servico-de-digitalizacao-de-documentos,f1887a51b9105410VgnVCM1000003b74010aRCRD. Acesso em: 27 junho 2019.

SILBERSCHATZ, A.; KORTH, H. F.; SUDARSHAN, S. **Sistema de Banco de Dados**. 5. ed. Rio de Janeiro: Elsevier, 2006. ISBN 978-85-352-1107-8.

SORDI, J. O. D. **Administração da informação**: fundamentos e práticas para uma nova gestão do conhecimento. 2. ed. São Paulo: Saraiva, 2015. ISBN 978-85-02-63480-0.

TAYLOR, C. **Structured vs. Unstructured Data**. 2018. Disponível em: https://www.datamation.com/big-data/structured-vs-unstructured-data.html. Acesso em: 5 junho 2019.

UNIVERSIDADE FEDERAL DE PERNAMBUCO. **Departamento de Ciência da Informação**. 2019. Disponível em: https://www.ufpe.br/dci. Acesso em: 17 julho 2019.

VASCONCELOS, A. B. et al. **SQL - Um pouco de história**. 2002. 3ª ed. Disponível em: http://pcleon.if.ufrgs.br/~leon/Livro_3_ed/node116.html. Acesso em: 11 dez 2018.

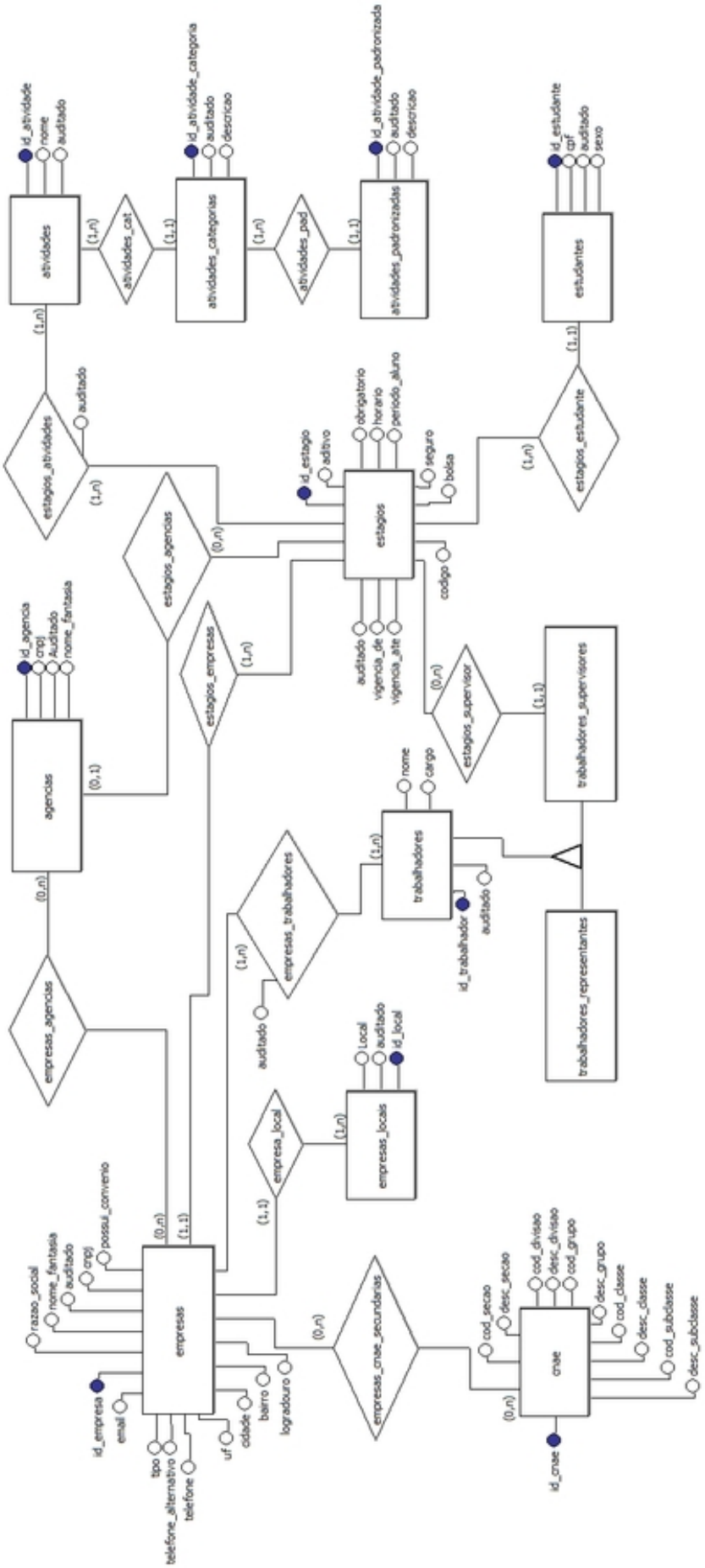
WANG, R. Y.; STRONG, D. M. Beyond Accuracy: What Data Quality Means to Data Consumers. **Journal of Management Information Systems**, M. E. Sharpe, Nova Iorque, v. 12, n. 4, p. 5 – 33, junho 1996. ISSN 07421222. Disponível em: http://mitiq.mit.edu/Documents/Publications/TDQMpub/14_Beyond_Accuracy.pdf. Acesso em: 12 junho 2019.

ZAMBENEDETTI, C. **Extração de informação sobre bases de dados textuais**. 2002. 144 p. Dissertação (Ciência da Computação) — Universidade Federal do Rio Grande do Sul. Disponível em: https://lume.ufrgs.br/bitstream/handle/10183/1628/000353940.pdf. Acesso em: 02 julho 2019.

# APÊNDICE A – Modelos de dados

## A.1 Modelo conceitual

Diagrama 1 – MER da base de dados de estágios de GI da UFPE




Fonte: elaborado pelo autor

## A.2 Modelo lógico

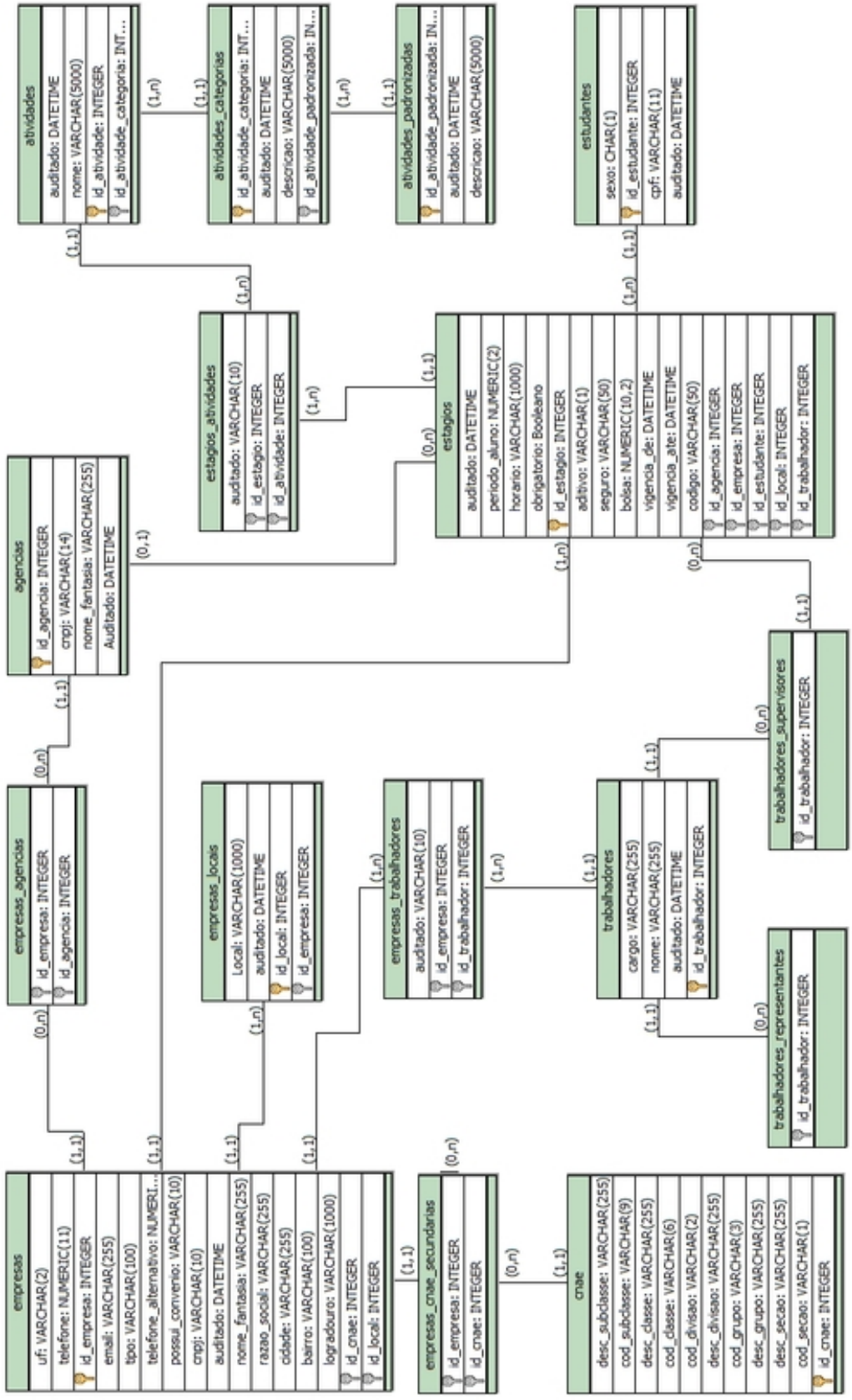



Diagrama 2 – Esquema lógico da base de dados de estágios de GI da UFPE

Fonte: elaborado pelo autor

## A.3 Modelo físico

**Código A.1 – Tabelas da base de dados especializada em estágios**

```
1 -- Cria a tabela agencias
2
3 CREATE TABLE public.agencias
4 (
5   nome_fantasia character varying(255) NOT NULL, -- Sigla/
      nome da agência intermediadora. Ex: IEL - Instituto
      Evaldo Lodi
6   id_agencia integer NOT NULL DEFAULT nextval('agencias_id_
      agencia_seq'::regclass), -- ID sequencial automático da
      Agência Intermediadora
7   cnpj character varying(14) NOT NULL, -- CNPJ da agência
      intermediadora. Vale a pena consultar no site da Receita
       Federal se está correto.
8   auditado timestamp without time zone,
9   CONSTRAINT pk_agencias PRIMARY KEY (id_agencia)
10 )
11 WITH (
12   OIDS=FALSE
13 );
14 ALTER TABLE public.agencias
15   OWNER TO postgres;
16
17 CREATE UNIQUE INDEX un_agencia_cnpj
18   ON public.agencias
19   USING btree
20   (cnpj COLLATE pg_catalog."default");
21
22 -- Cria a tabela atividades
23
24 CREATE TABLE public.atividades
25 (
26   nome character varying(5000) NOT NULL, -- Nome da atividade
27   id_atividade integer NOT NULL DEFAULT nextval('atividades_
      id_atividade_seq'::regclass), -- ID sequencial automá
      tico da Atividade
28   id_atividade_padronizada integer NOT NULL, -- ID sequencial
       automático da classificação (presente na tabela "
      atividades_classificacao")
29   auditado timestamp without time zone,
30   CONSTRAINT pk_atividades PRIMARY KEY (id_atividade),
31   CONSTRAINT fk_atividades_id_atividade_padronizada FOREIGN
      KEY (id_atividade_padronizada)
32       REFERENCES public.atividades_padronizadas (id_atividade
           _padronizada) MATCH SIMPLE
33       ON UPDATE RESTRICT ON DELETE RESTRICT
34 )
35 WITH (
36   OIDS=FALSE
```

```
37 );
38 ALTER TABLE public.atividades
39   OWNER TO postgres;
40
41 CREATE UNIQUE INDEX un_atividades_nome
42   ON public.atividades
43   USING btree
44   (nome COLLATE pg_catalog."default");
45
46 -- Cria a tabela atividades_categorias
47
48 CREATE TABLE public.atividades_categorias
49 (
50   id_atividade_categoria integer NOT NULL DEFAULT nextval('
       atividades_categorias_id_atividade_categoria_seq'::
       regclass), -- ID sequencial automático para cada
       classificação GERAL
51   descricao character varying(5000) NOT NULL, -- Descrição da
        classificação GERAL
52   auditado timestamp without time zone,
53   CONSTRAINT pk_atividades_categorias PRIMARY KEY (id_
       atividade_categoria)
54 )
55 WITH (
56   OIDS=FALSE
57 );
58 ALTER TABLE public.atividades_categorias
59   OWNER TO postgres;
60
61 CREATE UNIQUE INDEX un_atividades_categorias
62   ON public.atividades_categorias
63   USING btree
64   (descricao COLLATE pg_catalog."default");
65
66 -- Cria a tabela atividades_padronizadas
67
68 CREATE TABLE public.atividades_padronizadas
69 (
70   id_atividade_padronizada integer NOT NULL DEFAULT nextval('
       atividades_padronizadas_id_atividade_padronizada_seq'::
       regclass), -- ID sequencial automático para cada
       classificação GERAL
71   descricao character varying(5000) NOT NULL, -- Descrição da
        classificação
72   id_atividade_categoria integer NOT NULL, -- ID da
       classificação GERAL obtida na tabela "atividades_
       classificacao_geral"
73   auditado timestamp without time zone,
74   CONSTRAINT pk_atividades_padronizadas PRIMARY KEY (id_
       atividade_padronizada),
75   CONSTRAINT fk_atividades_padronizadas_id_atividade_
       categoria FOREIGN KEY (id_atividade_categoria)
76       REFERENCES public.atividades_categorias (id_atividade_
           categoria) MATCH SIMPLE
```

```
77        ON UPDATE RESTRICT ON DELETE RESTRICT
78 )
79 WITH (
80   OIDS=TRUE
81 );
82 ALTER TABLE public.atividades_padronizadas
83   OWNER TO postgres;
84
85 CREATE UNIQUE INDEX un_atividades_padronizadas_descricao
86   ON public.atividades_padronizadas
87   USING btree
88   (descricao COLLATE pg_catalog."default");
89
90 -- Cria a tabela cnae
91
92 CREATE TABLE public.cnae
93 (
94   id_cnae integer NOT NULL,
95   cod_secao character varying(1) NOT NULL,
96   desc_secao character varying(255) NOT NULL,
97   cod_divisao character varying(2) NOT NULL,
98   desc_divisao character varying(255) NOT NULL,
99   cod_grupo character varying(3) NOT NULL,
100  desc_grupo character varying(255) NOT NULL,
101  cod_classe character varying(6) NOT NULL,
102  desc_classe character varying(255) NOT NULL,
103  cod_subclasse character varying(9) NOT NULL,
104  desc_subclasse character varying(255) NOT NULL,
105  CONSTRAINT pk_cnae PRIMARY KEY (id_cnae)
106 )
107 WITH (
108   OIDS=FALSE
109 );
110 ALTER TABLE public.cnae
111   OWNER TO postgres;
112
113 -- Cria a tabela empresas
114
115 CREATE TABLE public.empresas
116 (
117   tipo character varying(100) NOT NULL, -- Tipo da empresa.
        Ex: S/A; LTDA; ME; MEI
118   cidade character varying(255) NOT NULL, -- Cidade onde a
        empresa está instalada (Consultar no site da Receita
        Federal a partir do CNPJ)
119   bairro character varying(100) NOT NULL, -- Bairro onde a
        empresa está instalada (Consultar no site da Receita
        Federal a partir do CNPJ)
120   possui_convenio boolean NOT NULL, -- Se a empresa possui
        convênio com a UFPE
121   telefone dom_telefone_check NOT NULL, -- Telefone
        cadastrado da empresa (Consultar no site da Receita
        Federal a partir do CNPJ)
```

```
122   telefone_alternativo dom_telefone_check, -- Telefone "
        alternativo" (celular; outro telefone) cadastrado da
        empresa (Consultar no site da Receita Federal a partir
        do CNPJ)
123   email character varying(255), -- E-mail cadastrado da
        empresa (Consultar no site da Receita Federal a partir
        do CNPJ)
124   uf dom_uf_check NOT NULL, -- Unidade da Federação onde a
        empresa está instalada (Consultar no site da Receita
        Federal a partir do CNPJ). Ex: PE; PB; RJ; SP
125   logradouro character varying(1000) NOT NULL, -- Rua/Avenida
        /Travessa/Etc; número e complemento onde a empresa está
        instalada (Consultar no site da Receita Federal a partir
         do CNPJ)
126   id_empresa integer NOT NULL DEFAULT nextval('empresas_id_
        empresa_seq'::regclass), -- ID sequencial automático da
        empresa cadastrada
127   razao_social character varying(255) NOT NULL, -- Nome jurí
        dico da empresa (Obtido no site da Receita Federal,
        através de consulta ao CNPJ)
128   nome_fantasia character varying(255), -- Nome fantasia da
        empresa (Obtido no site da Receita Federal, através de
        consulta ao CNPJ)
129   cnpj character varying(14) NOT NULL, -- CNPJ da empresa
130   cep dom_cep_check NOT NULL, -- Código Postal da empresa (
        obtido no site da Receita Federal, através de consulta
        ao CNPJ)
131   id_cnae integer NOT NULL,
132   auditado timestamp without time zone,
133   CONSTRAINT pk_empresas PRIMARY KEY (id_empresa),
134   CONSTRAINT fk_empresas_cnae_id_cnae FOREIGN KEY (id_cnae)
135       REFERENCES public.cnae (id_cnae) MATCH SIMPLE
136       ON UPDATE RESTRICT ON DELETE RESTRICT
137 )
138 WITH (
139   OIDS=FALSE
140 );
141 ALTER TABLE public.empresas
142   OWNER TO postgres;
143
144 -- Cria a tabela da relacao empresas e cnae secundarias
145
146 CREATE TABLE public.empresas_cnae_secundarias
147 (
148   id_empresa integer NOT NULL,
149   id_cnae integer NOT NULL,
150   auditado timestamp without time zone,
151   CONSTRAINT pk_empresas_cnae_secundarias PRIMARY KEY (id_
        empresa, id_cnae),
152   CONSTRAINT fk_empresas_cnae_secundarias_id_cnae FOREIGN KEY
         (id_cnae)
153       REFERENCES public.cnae (id_cnae) MATCH SIMPLE
154       ON UPDATE RESTRICT ON DELETE RESTRICT,
```

```
155   CONSTRAINT fk_empresas_cnae_secundarias_id_empresa FOREIGN
          KEY (id_empresa)
156       REFERENCES public.empresas (id_empresa) MATCH SIMPLE
157       ON UPDATE RESTRICT ON DELETE RESTRICT
158 )
159 WITH (
160   OIDS=FALSE
161 );
162 ALTER TABLE public.empresas_cnae_secundarias
163   OWNER TO postgres;
164
165 -- Cria a tabela da relacao empresas e locais
166
167 CREATE TABLE public.empresas_locais
168 (
169   local character varying(1000) NOT NULL, -- Sigla/nome/local
          onde o estágio será realizado. (Ex: UFPE / CAC / DCI /
          laboratório de informática)
170   id_local integer NOT NULL DEFAULT nextval('empresas_locais_
          id_local_seq'::regclass), -- ID sequencial automático do
          local de estágio
171   id_empresa integer NOT NULL, -- ID da empresa, obtida a
          partir da tabela "empresa", com o objetivo de vinculá-lo
          ao local
172   auditado timestamp without time zone,
173   CONSTRAINT pk_empresas_locais PRIMARY KEY (id_local),
174   CONSTRAINT fk_empresas_locais_id_empresa FOREIGN KEY (id_
          empresa)
175       REFERENCES public.empresas (id_empresa) MATCH SIMPLE
176       ON UPDATE RESTRICT ON DELETE RESTRICT,
177   CONSTRAINT un_empresas_locais_id_empresa_local UNIQUE (id_
          empresa, local)
178 )
179 WITH (
180   OIDS=FALSE
181 );
182 ALTER TABLE public.empresas_locais
183   OWNER TO postgres;
184
185 -- Cria a tabela da relacao empresas e trabalhadores
186
187 CREATE TABLE public.empresas_trabalhadores
188 (
189   id_empresa integer NOT NULL, -- ID obtida a partir da
          tabela "empresa", com os demais dados da empresa onde o
          trabalhador está empregado (endereço, telefone, e-mail,
          etc.)
190   id_trabalhador integer NOT NULL, -- ID do trabalhado obtido
          a partir da tabela "trabalhadores", para vinculá-lo à
          empresa
191   auditado timestamp without time zone,
192   CONSTRAINT pk_empresas_trabalhadores PRIMARY KEY (id_
          empresa, id_trabalhador),
```

```
193   CONSTRAINT fk_empresas_trabalhadores_id_empresa FOREIGN KEY
          (id_empresa)
194       REFERENCES public.empresas (id_empresa) MATCH SIMPLE
195       ON UPDATE RESTRICT ON DELETE RESTRICT,
196   CONSTRAINT fk_empresas_trabalhadores_id_trabalhador FOREIGN
          KEY (id_trabalhador)
197       REFERENCES public.trabalhadores (id_trabalhador) MATCH
             SIMPLE
198       ON UPDATE RESTRICT ON DELETE RESTRICT
199 )
200 WITH (
201   OIDS=FALSE
202 );
203 ALTER TABLE public.empresas_trabalhadores
204   OWNER TO postgres;
205
206 -- Cria a tabela de estagios
207
208 CREATE TABLE public.estagios
209 (
210   horario character varying(1000), -- Horário do estagiário
        na empresa. Ex: Segunda a sexta, das 13h às 18h;
        intervalo de 15 minutos
211   vigencia_de date NOT NULL, -- Data de início da vigência do
          contrato de estágio. Ex: 2017-01-16
212   vigencia_ate date NOT NULL, -- Data de término da vigência
        do contrato de estágio. Ex: 2017-01-16
213   aditivo boolean NOT NULL, -- Determina se o contrato de est
        ágio é um aditivo de algum contrato de estágio anterior
214   obrigatorio boolean NOT NULL, -- Determina se o contrato de
          estágio é "obrigatório" ou "não-obrigatório"
215   bolsa numeric(10,2), -- Valor (total) de todos os benefí
        cios pecuniários recebidos pelo estagiário. Ex: 520.00
216   id_estagio integer NOT NULL DEFAULT nextval('estagios_id_
        estagio_seq'::regclass), -- ID sequencial automático do
        estágio cadastrado
217   codigo character varying(50), -- Código do contrato de está
        gio
218   seguro character varying(50) NOT NULL, -- Responsável pelo
        pagamento do seguro. Ex: UFPE
219   periodo_aluno numeric(2,0) NOT NULL, -- Período que o
        estudante do estágio está matriculado (utilizar o menor
        período das cadeiras cursadas)
220   id_supervisor integer NOT NULL, -- ID do supervisor, obtido
          a partir da tabela "supervisor", com o objetivo de
        vinculá-lo ao estágio
221   id_estudante integer NOT NULL, -- ID do estudante, obtida a
          partir da tabela "estudantes", com o objetivo de vincul
        á-lo ao estágio
222   id_local integer NOT NULL, -- ID do local, obtido a partir
        da tabela "estagios_locais", com o objetivo de vinculá-
        lo ao estágio
223   id_agencia integer, -- ID da agência intermediadora, obtido
          através da tabela "agencias"
```

```
224   auditado timestamp without time zone,
225   CONSTRAINT pk_estagios PRIMARY KEY (id_estagio),
226   CONSTRAINT fk_estagios_id_agencia FOREIGN KEY (id_agencia)
227       REFERENCES public.agencias (id_agencia) MATCH SIMPLE
228       ON UPDATE RESTRICT ON DELETE RESTRICT,
229   CONSTRAINT fk_estagios_id_estudante FOREIGN KEY (id_
          estudante)
230       REFERENCES public.estudantes (id_estudante) MATCH
              SIMPLE
231       ON UPDATE RESTRICT ON DELETE RESTRICT,
232   CONSTRAINT fk_estagios_id_local FOREIGN KEY (id_local)
233       REFERENCES public.empresas_locais (id_local) MATCH
              SIMPLE
234       ON UPDATE RESTRICT ON DELETE RESTRICT,
235   CONSTRAINT fk_estagios_id_supervisor FOREIGN KEY (id_
          supervisor)
236       REFERENCES public.trabalhadores_supervisores (id_
              trabalhador) MATCH SIMPLE
237       ON UPDATE RESTRICT ON DELETE RESTRICT
238 )
239 WITH (
240   OIDS=FALSE
241 );
242 ALTER TABLE public.estagios
243   OWNER TO postgres;
244
245 -- Cria a tabela do relacionamento estagios e atividades
246
247 CREATE TABLE public.estagios_atividades
248 (
249   id_estagio integer NOT NULL, -- ID obtido através da tabela
          "estagios" para informar a que estágio a(s) atividade(s
          ) será(ão) vinculada(s)
250   id_atividade integer NOT NULL, -- ID obtido através da
          tabela "atividades" para informar que atividade(s) será(
          ão) vinculada(s) ao estágio
251   auditado timestamp without time zone,
252   CONSTRAINT pk_estagios_atividades PRIMARY KEY (id_estagio,
          id_atividade),
253   CONSTRAINT fk_estagios_atividades_id_atividade FOREIGN KEY
          (id_atividade)
254       REFERENCES public.atividades (id_atividade) MATCH
              SIMPLE
255       ON UPDATE RESTRICT ON DELETE RESTRICT,
256   CONSTRAINT fk_estagios_atividades_id_estagio FOREIGN KEY (
          id_estagio)
257       REFERENCES public.estagios (id_estagio) MATCH SIMPLE
258       ON UPDATE RESTRICT ON DELETE RESTRICT
259 )
260 WITH (
261   OIDS=FALSE
262 );
263 ALTER TABLE public.estagios_atividades
264   OWNER TO postgres;
```

```
265
266 -- Cria a tabela estudantes
267
268 CREATE TABLE public.estudantes
269 (
270   sexo character(1) NOT NULL, -- Sexo (M ou F) do estudante
271   id_estudante integer NOT NULL DEFAULT nextval('estudantes_
        id_estudante_seq'::regclass), -- Número sequencial autom
        ático com ID do estudante
272   cpf character varying(11) NOT NULL, -- Número do CPF (sem
        pontos e traço) do estudante
273   auditado timestamp without time zone,
274   CONSTRAINT pk_estudantes PRIMARY KEY (id_estudante),
275   CONSTRAINT un_estudantes_cpf UNIQUE (cpf),
276   CONSTRAINT ck_estudantes_sexo CHECK (sexo = ANY (ARRAY['M'
        ::bpchar, 'F'::bpchar]))
277 )
278 WITH (
279   OIDS=FALSE
280 );
281 ALTER TABLE public.estudantes
282   OWNER TO postgres;
283
284 -- Cria a tabela trabalhadores
285
286 CREATE TABLE public.trabalhadores
287 (
288   nome character varying(255) NOT NULL, -- Nome da pessoa que
        será vinculada às tabelas "empresa", "representantes" e
        /ou "supervisores"
289   cargo character varying(255), -- Cargo da pessoa que será
        vinculada às tabelas "empresa", "representantes" e/ou "
        supervisores". Ex: Supervisor(a); Diretor(a)
290   id_trabalhador integer NOT NULL DEFAULT nextval('
        trabalhadores_id_trabalhador_seq'::regclass), -- ID
        sequencial automático do trabalhador cadastrado
291   auditado timestamp without time zone,
292   CONSTRAINT pk_trabalhadores PRIMARY KEY (id_trabalhador)
293 )
294 WITH (
295   OIDS=FALSE
296 );
297 ALTER TABLE public.trabalhadores
298   OWNER TO postgres;
299
300 -- Cria a tabela da entidade especializada trabalhadores do
      tipo representantes
301
302 CREATE TABLE public.trabalhadores_representantes
303 (
304   id_trabalhador integer NOT NULL, -- ID obtido através da
        tabela "trabalhadores" visando vincular o mesmo ao está
        gio na condição de "representante da empresa"
```

```
305   CONSTRAINT pk_trabalhadores_representantes PRIMARY KEY (id_
        trabalhador),
306   CONSTRAINT fk_trabalhadores_representantes_id_trabalhador
        FOREIGN KEY (id_trabalhador)
307       REFERENCES public.trabalhadores (id_trabalhador) MATCH
            SIMPLE
308       ON UPDATE RESTRICT ON DELETE RESTRICT
309 )
310 WITH (
311   OIDS=FALSE
312 );
313 ALTER TABLE public.trabalhadores_representantes
314   OWNER TO postgres;
315
316 -- Cria a tabela da entidade especializada trabalhadores do
      tipo supervisores
317
318 CREATE TABLE public.trabalhadores_supervisores
319 (
320   id_trabalhador integer NOT NULL, -- ID obtido através da
        tabela "trabalhadores" visando vincular o mesmo ao está
        gio na condição de "supervisor do estágio"
321   CONSTRAINT pk_trabalhadores_supervidores_id_trabalhador
        PRIMARY KEY (id_trabalhador),
322   CONSTRAINT fk_trabalhadores_supervisores_id_trabalhador
        FOREIGN KEY (id_trabalhador)
323       REFERENCES public.trabalhadores (id_trabalhador) MATCH
            SIMPLE
324       ON UPDATE RESTRICT ON DELETE RESTRICT
325 )
326 WITH (
327   OIDS=FALSE
328 );
329 ALTER TABLE public.trabalhadores_supervisores
330   OWNER TO postgres;
```

# APÊNDICE B – Outros comandos SQL

## B.1 Função para verificar CPF/CNPJ

**Código B.1 – Função para verificar CPF/CNPJ**

```
1  CREATE OR REPLACE FUNCTION public.fn_cnpj_cpf_validar(text)
2    RETURNS boolean AS
3  $BODY$-- Função para validar CPF/CNPJ, com ou sem máscara
4  DECLARE
5   v_string text := $1;
6   v_caldv1 int4;
7   v_caldv2 int4;
8   v_dv1 int4;
9   v_dv2 int4;
10  v_array1 text[] ;
11  v_array2 text[] ;
12  v_tst_string int4;
13 BEGIN
14  v_string := translate(v_string, './-', '');
15  IF (char_length(v_string)::int4) = 14 THEN
16
17   SELECT INTO v_array1 '{5,4,3,2,9,8,7,6,5,4,3,2}';
18   SELECT INTO v_array2 '{6,5,4,3,2,9,8,7,6,5,4,3,2}';
19   v_dv1 := (substring(v_string, 13, 1))::int4;
20   v_dv2 := (substring(v_string, 14, 1))::int4;
21   /* COLETA DIG VER 1 CNPJ */
22   v_caldv1 := 0;
23   FOR va IN 1..12 LOOP
24    v_caldv1 := v_caldv1 + ((SELECT substring(v_string, va, 1)
        )::int4 * (v_array1[va]::int4));
25   END LOOP;
26   v_caldv1 := v_caldv1 % 11;
27    IF (v_caldv1 = 0) OR (v_caldv1 = 1) THEN
28     v_caldv1 := 0;
29    ELSE
30     v_caldv1 := 11 - v_caldv1;
31    END IF;
32   /* COLETA DIG VER 2 CNPJ */
33   v_caldv2 := 0;
34   FOR va IN 1..13 LOOP
35    v_caldv2 := v_caldv2 + ((SELECT substring(v_string || v_
        caldv1::text, va, 1))::int4 * (v_array2[va]::int4));
36   END LOOP;
37   v_caldv2 := v_caldv2 % 11;
38    IF (v_caldv2 = 0) OR (v_caldv2 = 1) THEN
39     v_caldv2 := 0;
40    ELSE
41     v_caldv2 := 11 - v_caldv2;
42    END IF;
43   /* TESTA */
```

```
44   IF (v_caldv1 = v_dv1) AND (v_caldv2 = v_dv2) THEN
45    RETURN TRUE;
46   ELSE
47    RETURN FALSE;
48   END IF;
49
50  ELSIF (char_length(v_string)::int4) = 11 THEN
51
52   v_dv1 := (substring(v_string, 10, 1))::int4;
53   v_dv2 := (substring(v_string, 11, 1))::int4;
54   v_string := substring(v_string, 1, 9);
55   /* COLETA DIG VER 1 CPF */
56   v_caldv1 := 0;
57   FOR va IN 1..9 LOOP
58    v_caldv1 := v_caldv1 + ((SELECT substring(v_string, va, 1)
        )::int4 * (11 - va));
59   END LOOP;
60   v_caldv1 := v_caldv1 % 11;
61   IF (v_caldv1 = 0) OR (v_caldv1 = 1) THEN
62    v_caldv1 := 0;
63   ELSE
64    v_caldv1 := 11 - v_caldv1;
65   END IF;
66   /* COLETA DIG VER 2 CPF */
67   v_caldv2 := 0;
68   FOR va IN 1..10 LOOP
69    v_caldv2 := v_caldv2 + ((SELECT substring((v_string || v_
        caldv1::text), va, 1))::int4 * (12 - va));
70   END LOOP;
71   v_caldv2 := v_caldv2 % 11;
72   IF (v_caldv2 = 0) OR (v_caldv2 = 1) THEN
73    v_caldv2 := 0;
74   ELSE
75    v_caldv2 := 11 - v_caldv2;
76   END IF;
77   /* TESTA */
78   IF (v_caldv1 = v_dv1) AND (v_caldv2 = v_dv2) THEN
79    RETURN TRUE;
80   ELSE
81    RETURN FALSE;
82   END IF;
83
84  END IF;
85
86 RETURN FALSE;
87 END;
```

## B.2 Função para verificar CNAE

**Código B.2 – Função de verificação de CNAE**

```
1 CREATE OR REPLACE FUNCTION public.fn_verifica_cnae(text)
2   RETURNS integer AS
3 $BODY$-- A descrição de subclasse pode ser obtida a partir do
      "CÓDIGO DA ATIVIDADE ECONÔMICA PRINCIPAL" disponível na
     consulta de CNPJ a partir do site da Receita Federal do
     Brasil (http://www.receita.fazenda.gov.br/PessoaJuridica/
     CNPJ/cnpjreva/Cnpjreva_Solicitacao.asp)
4
5 DECLARE
6     v_cod_atividade ALIAS FOR $1;
7 BEGIN
8     v_cod_atividade := TRANSLATE(v_cod_atividade, './-', '');
9     IF EXISTS (SELECT 1 FROM cnae WHERE cnae.id_cnae = CAST(v
         _cod_atividade AS INT)) THEN
10      RETURN v_cod_atividade;
11    ELSE
12      RAISE EXCEPTION 'Código da atividade econômica
           principal da empresa não encontrada! Consulte o CNPJ
            no site da Receita Federal e verifique a descrição
           CORRETA!';
13    END IF;
14 END;
```

## B.3 Gatilho para verificar CPF/CNPJ

**Código B.3 – Gatilho para acionar a função de verificação do CPF/CNPJ**

```
1 CREATE OR REPLACE FUNCTION public.tr_cnpj_validar()
2   RETURNS trigger AS
3 $BODY$    BEGIN
4         IF fn_cnpj_cpf_validar(NEW.cnpj)=FALSE THEN
5             RAISE EXCEPTION 'CNPJ % inválido! Empresa não
                 cadastrada/atualizada!',NEW.cnpj;
6         ELSE
7             RETURN NEW;
8         END IF;
9 END;
```

## B.4 Gatilho para normalizar dados

**Código B.4 – Gatilho para formatar o texto em CAIXA ALTA**

```
1 CREATE OR REPLACE FUNCTION public.tr_upper_empresas() -- Cada
     tabela com texto possui tal gatilho
2   RETURNS trigger AS
3 $BODY$BEGIN -- Abaixo cada atributo da tabela que seja do
    tipo texto é atualizado para caixa alta
4     NEW.tipo := UPPER(NEW.tipo);
```

```
5      NEW.cidade := UPPER(NEW.cidade);
6      NEW.bairro := UPPER(NEW.bairro);
7      NEW.email := UPPER(NEW.email);
8      NEW.uf := UPPER(NEW.uf);
9      NEW.logradouro := UPPER(NEW.logradouro);
10     NEW.razao_social := UPPER(NEW.razao_social);
11     NEW.nome_fantasia := UPPER(NEW.nome_fantasia);
12     RETURN NEW;
13 END;
```

## B.5 Função para inserir um estágio

**Código B.5 – Procedimento para armazenamento de dados de estágio**

```
CREATE OR REPLACE FUNCTION public.sp_cadastro(text,text,text,
   text,text,boolean,dom_telefone_check,dom_telefone_check,
   text,dom_uf_check,text,text,text,text,dom_cep_check,text,
   text,text,text,boolean,text,text,text,date,date,boolean,
   boolean,numeric,text,text,integer,text,text,text) RETURNS
   void AS
$BODY$
DECLARE
    v_agencia_cnpj ALIAS FOR $1;
    v_agencia_nome_fantasia ALIAS FOR $2;
    v_empresa_tipo ALIAS FOR $3;
    v_empresa_cidade ALIAS FOR $4;
    v_empresa_bairro ALIAS FOR $5;
    v_empresa_possui_convenio ALIAS FOR $6;
    v_empresa_telefone ALIAS FOR $7;
    v_empresa_telefone_alternativo ALIAS FOR $8;
    v_empresa_email ALIAS FOR $9;
    v_empresa_uf ALIAS FOR $10;
    v_empresa_logradouro ALIAS FOR $11;
    v_empresa_razao_social ALIAS FOR $12;
    v_empresa_nome_fantasia ALIAS FOR $13;
    v_empresa_cnpj ALIAS FOR $14;
    v_empresa_cep ALIAS FOR $15;
    v_empresa_id_cnae ALIAS FOR $16;
    v_empresa_local_local ALIAS FOR $17;
    v_trabalhador_nome ALIAS FOR $18;
    v_trabalhador_cargo ALIAS FOR $19;
    v_trabalhador_representante_eh_representante ALIAS FOR
       $20;
    v_estudante_sexo ALIAS FOR $21;
    v_estudante_cpf ALIAS FOR $22;
    v_estagio_horario ALIAS FOR $23;
    v_estagio_vigencia_de ALIAS FOR $24;
    v_estagio_vigencia_ate ALIAS FOR $25;
    v_estagio_aditivo ALIAS FOR $26;
    v_estagio_obrigatorio ALIAS FOR $27;
    v_estagio_bolsa ALIAS FOR $28;
```

```
    v_estagio_codigo ALIAS FOR $29;
    v_estagio_seguro ALIAS FOR $30;
    v_estagio_periodo_aluno ALIAS FOR $31;
    v_atividade_classificacao_geral_descricao ALIAS FOR $32;
    v_atividade_classificacao_descricao ALIAS FOR $33;
    v_atividade_nome ALIAS FOR $34;
    v_agencia_id_agencia integer;
    v_empresa_id_empresa integer;
    v_empresa_local_id_local integer;
    v_trabalhador_id_trabalhador integer;
    v_supervisor_id_supervisor integer;
    v_estudante_id_estudante integer;
    v_estagio_id_estagio integer;
    v_atividade_classificacao_geral_id_classificacao_geral
       integer;
    v_atividade_classificacao_id_classificacao integer;
    v_atividade_id_atividade integer;

BEGIN

    SELECT sp_agencia (v_agencia_cnpj, v_agencia_nome_
       fantasia) INTO v_agencia_id_agencia;
    RAISE NOTICE 'ID Agência --> %', v_agencia_id_agencia;

    SELECT sp_empresa (v_empresa_tipo, v_empresa_cidade, v_
       empresa_bairro, v_empresa_possui_convenio, v_empresa_
       telefone, v_empresa_telefone_alternativo, v_empresa_
       email, v_empresa_uf, v_empresa_logradouro, v_empresa_
       razao_social, v_empresa_nome_fantasia, v_empresa_cnpj,
        v_empresa_cep, v_empresa_id_cnae) INTO v_empresa_id_
       empresa;
    RAISE NOTICE 'ID Empresa --> %', v_empresa_id_empresa;

    SELECT sp_empresa_local (v_empresa_local_local, v_empresa
       _id_empresa) INTO v_empresa_local_id_local;
    RAISE NOTICE 'ID Local --> %', v_empresa_local_id_local;

    SELECT sp_trabalhador (v_trabalhador_nome, v_trabalhador_
       cargo) INTO v_trabalhador_id_trabalhador;
    RAISE NOTICE 'ID Trabalhador --> %', v_trabalhador_id_
       trabalhador;

    PERFORM sp_representante (v_trabalhador_id_trabalhador, v
       _trabalhador_representante_eh_representante);

    SELECT sp_supervisor (v_trabalhador_id_trabalhador) INTO
       v_supervisor_id_supervisor;
    RAISE NOTICE 'ID Supervisor --> %', v_supervisor_id_
       supervisor;

    PERFORM sp_empresa_trabalhador (v_empresa_id_empresa, v_
       trabalhador_id_trabalhador);
```

```
    SELECT sp_estudante (v_estudante_sexo, v_estudante_cpf)
       INTO v_estudante_id_estudante;
    RAISE NOTICE 'ID Estudante --> %', v_estudante_id_
       estudante;

    SELECT sp_estagio (v_estagio_horario, v_estagio_vigencia_
       de, v_estagio_vigencia_ate, v_estagio_aditivo, v_
       estagio_obrigatorio, v_estagio_bolsa, v_estagio_codigo
       , v_estagio_seguro, v_estagio_periodo_aluno, v_
       supervisor_id_supervisor, v_estudante_id_estudante, v_
       empresa_local_id_local, v_agencia_id_agencia) INTO v_
       estagio_id_estagio;
    RAISE NOTICE 'ID Estágio --> %', v_estagio_id_estagio;

    SELECT sp_atividade_classificacao_geral (v_atividade_
       classificacao_geral_descricao) INTO v_atividade_
       classificacao_geral_id_classificacao_geral;
    RAISE NOTICE 'ID Classificação Geral --> %', v_atividade_
       classificacao_geral_id_classificacao_geral;

    SELECT sp_atividade_classificacao (v_atividade_
       classificacao_descricao, v_atividade_classificacao_
       geral_id_classificacao_geral) INTO v_atividade_
       classificacao_id_classificacao;
    RAISE NOTICE 'ID Classificação --> %', v_atividade_
       classificacao_id_classificacao;

    SELECT sp_atividade (v_atividade_nome, v_atividade_
       classificacao_id_classificacao) INTO v_atividade_id_
       atividade;
    RAISE NOTICE 'ID Atividade --> %', v_atividade_id_
       atividade;

    PERFORM sp_estagio_atividade (v_estagio_id_estagio, v_
       atividade_id_atividade);

EXCEPTION WHEN others THEN -- trecho abaixo modificado do
   obtido em http://www.anicehumble.com/2011/08/postgresql-
   catch-exception-rocks.html

    RAISE NOTICE 'O erro foi --> %. Código do erro -> %',
       SQLERRM, SQLSTATE;
END;
```

# ANEXO A – Termos de compromisso de estágio

## A.1 Termo de compromisso de estágio curricular obrigatório

**SERVIÇO PÚBLICO FEDERAL**
**UNIVERSIDADE FEDERAL DE PERNAMBUCO**
**PRÓ-REITORIA PARA ASSUNTOS ACADÊMICOS**

**TERMO DE COMPROMISSO DE ESTÁGIO CURRICULAR OBRIGATÓRIO**
**(Estágios ofertados através de Agente Integração conveniados com a UFPE)**

### 1. CONCEDENTE:

---

*(nome da concedente)* _____,
adiante **CONCEDENTE**

CNPJ nº

Natureza jurídica da instituição:

Endereço: *(endereço completo e CEP)*

Representada por _____ CPF nº _____, RG nº _____,

---

### 2. ESTAGIÁRIO:

---

*(nome do estagiário)* _____,
adiante **ESTAGIÁRIO**

CPF nº _____, RG nº _____,

---

*(nome do estagiário)* _____,
adiante **ESTAGIÁRIO**

data de nascimento: __ de_____ de _____; residente a
_____

_____,

Bairro_____,CEP_____-_____
Cidade_____, Estado _____; Telefone (____)
_____ / (____) _____;

Estudante do _____ período do Curso de Graduação em
_____

**3. INSTITUIÇÃO DE ENSINO:**

UNIVERSIDADE FEDERAL DE PERNAMBUCO, adiante **UFPE**

CNPJ n° 24.134.488/0001-08

Natureza jurídica da instituição: autarquia federal vinculada ao Ministério da Educação

Av. Prof. Moraes Rego, n° 1235 - Cidade Universitária, Recife/PE – CEP: 50670-901

Representada por
_____,
professor(a) universitário(a), CPF nº _____, RG nº
_____, residente nesta cidade, na qualidade de Coordenador(a) de Estágio do Curso de Graduação em _____.

**4. FUNDAMENTO LEGAL**: Lei 11.788 de 25 de setembro de 2008.

**5. OBJETO**: Condições para a realização do estágio curricular obrigatório do **ESTAGIÁRIO,** junto à **CONCEDENTE** e à **UFPE,** intermediado por _____*(nome do Agente de Integração)*_____*,* no papel de Agente de Integração, conveniado com a UFPE.

**6. PLANEJAMENTO DO ESTÁGIO:**

6.1**.** Vigência: de ___ / ____ / 20___ até __ / __ / 20___;

6.2. Dias da Semana e Horários: *consultar o plano de estágio em anexo*;

6.3. Jornada Semanal: _____ horas;

6.4. **Supervisor do Estágio ou Preceptor:** _____, Registro Profissional no órgão de classe ou CPF nº _____;

6.5. **Prof. Orientador**: _____ SIAPE: _____;

6.6. As atividades de estágio serão realizadas conforme Plano de Atividades de Estágio, aprovado pela Coordenação de Estágio do Curso e parte integrante desse documento.

**7. COMPROMISSOS DA UFPE**, através da Coordenação de Estágio do Curso de Graduação ao qual o estudante está vinculado:

7.1 Verificar a correlação das atividades previstas no plano de estágio, em anexo, com o conteúdo formativo do curso do estagiário;

7.2 Verificar se o aluno está regularmente matriculado e frequentando o curso;

7.3 Verificar a compatibilidade do horário estabelecido no plano de estágio com as atividades escolares;

7.4 Informar à **CONCEDENTE** a frequência e o desempenho acadêmico do estagiário, sempre que solicitado; bem como os períodos de avaliação escolar e outras atividades acadêmicas obrigatórias do estudante;

7.5 Comunicar à **CONCEDENTE** a integralização curricular, colação de grau, trancamento ou abandono do curso por parte do estagiário;

7.6 Acompanhar a realização do estágio, garantindo o cumprimento do Plano de Atividades.

**8. COMPROMISSOS DO ESTAGIÁRIO:**

8.1. Observar as normas e regulamentos internos da **CONCEDENTE**;

8.2. Cumprir a programação do estágio;

8.3. Zelar pelos materiais, equipamentos e bens em geral da **CONCEDENTE**, sob os seus cuidados;

8.4. Manter em absoluto sigilo, durante e após o estágio, quaisquer informações de caráter confidencial a que tiver acesso;

8.5. Apresentar relatório circunstanciado de estágio, monografia, trabalho de conclusão de curso e/ou submeter-se a outras formas de avaliação definidas pelo

Colegiado do Curso;

8.6. Comparecer aos acompanhamentos periódicos programados pela **UFPE** e previstos no Plano de Estágio.

**9. COMPROMISSOS DA CONCEDENTE:**

9.1. Orientar profissionalmente o **ESTAGIÁRIO**, supervisionando sistematicamente o desenvolvimento das atividades realizadas;

9.2. Comunicar mensalmente à Coordenação do Curso a avaliação da assiduidade e do desempenho do **ESTAGIÁRIO;**

9.3. Comunicar à Coordenação de Estágio do Curso, quaisquer atitudes tomadas, diante de irregularidades e faltas cometidas pelo **ESTAGIÁRIO**;

9.4. Garantir o recesso, obedecido o prazo de realização do estágio;

9.5 Aplicar ao estagiário a legislação relacionada à saúde e segurança no trabalho.

**10. SEGURO DE COBERTURA DE ACIDENTES PESSOAIS:** Responsabilidade da **CONCEDENTE** Seguradora:_____; Nº da Apólice: _____. Início da Vigência: _____ .

**11. BOLSA e AUXILIO TRANSPORTE** de responsabilidade da **CONCEDENTE:**

_____

*(Caso sejam concedidos informar os valores. Em caso contrário, informar que não serão concedidos)*

**12. DISPOSIÇÕES GERAIS:**

12.1. A realização do estágio não acarretará vínculo empregatício de qualquer natureza entre o estagiário e a **CONCEDENTE**;

12.2. A prorrogação do estágio depende de prévia e expressa autorização da **UFPE**;

12.3. O termo de compromisso de estágio poderá ser rescindido em qualquer período de realização por solicitação da **CONCEDENTE**, do **ESTAGIÁRIO** ou da **UFPE**, mediante comunicação por escrito no prazo mínimo de 10 (dez) dias, explicitando o motivo da rescisão.

**13. FORO:** O foro da Justiça Federal em Pernambuco é o competente para dirimir quaisquer questões oriundas da execução deste convênio ou para a interpretação deste instrumento.

Este instrumento é firmado em 04 (quatro) vias de igual teor e forma, cabendo a 1ª à **CONCEDENTE,** a 2ª ao **ESTAGIÁRIO,** a 3ª à **UFPE** e a 4ª ao **AGENTE DE INTEGRAÇÃO**.

Recife, ____ de _____ de _____

_____ _____

**ESTAGIÁRIO** (*Nome e Cargo do representante da Concedente)*

**CONCEDENTE**

_____ _____

**INSTITUIÇÃO DE ENSINO** (*Nome e Cargo do rep. do Ag. de Int.)*

**(Coord.de Estágio do Curso** **AGENTE DE INTEGRAÇÃO**

*(indicar o Curso)* **– UFPE)**

## A.2 Termo de compromisso de estágio curricular não-obrigatório

**SERVIÇO PÚBLICO FEDERAL**
**UNIVERSIDADE FEDERAL DE PERNAMBUCO**
**PRÓ-REITORIA PARA ASSUNTOS ACADÊMICOS**

**TERMO DE COMPROMISSO DE ESTÁGIO CURRICULAR NÃO OBRIGATÓRIO**

## 1. CONCEDENTE:

*(nome da concedente)* ,
adiante **CONCEDENTE**

CNPJ nº

Natureza da instituição:

Endereço: *(endereço completo e CEP)*

Representada por

_____

,

RG nº CPF nº

## 2. ESTAGIÁRIO:

*(nome do estagiário)* ,
adiante **ESTAGIÁRIO**

CPF nº , RG
nº ,

data de nascimento: de de ; residente a

Bairro_____,
CEP_____-_____ Cidade_____, Estado
;Telefone ( )/( );

Estudante do período do Curso de Graduação em

## 3. INSTITUIÇÃO DE ENSINO:

UNIVERSIDADE FEDERAL DE PERNAMBUCO, adiante **UFPE**

UNIVERSIDADE FEDERAL DE PERNAMBUCO, adiante **UFPE**

CNPJ n° 24.134.488/0001-08

Natureza da instituição: autarquia federal vinculada ao Ministério da Educação

Av. Prof. Moraes Rego, n° 1235 - Cidade Universitária, Recife/PE – CEP: 50670-901

Representada por _____ professor(a) universitário(a), CPF nº _____ , RG nº _____ , residente nesta cidade, na qualidade de Coordenador(a) do Curso de Graduação em .

**4. FUNDAMENTO LEGAL**: Lei 11.788 de 25 de setembro de 2008.

**5. OBJETO**: Formalizar as condições para a realização de estágio no campo de formação do **ESTAGIÁRIO,** junto à **CONCEDENTE** e à **UFPE.**

**6. PLANEJAMENTO DO ESTÁGIO:**

6.1**.** Vigência: de ___ / ____ / 20___ até __ / __ / 20___ ;

6.2. Dias da Semana e Horários: *consultar o plano de estágio em anexo*;

6.3. Jornada Semanal: _____ horas;

6.4. **Supervisor do Estágio ou Preceptor:** _____, Registro Profissional no órgão de classe ou CPF nº _____;

6.5. **Prof. Orientador**: _____SIAPE: _____;

6.6. As atividades de estágio serão realizadas conforme Plano de Atividades de Estágio, aprovado pela Coordenação do Curso e parte integrante desse documento.

**7. COMPROMISSOS DA UFPE**, através da Coordenação do Curso de Graduação ao qual o estudante está vinculado:

7.1 Verificar a correlação das atividades previstas no plano de estágio, em anexo, com o conteúdo formativo do curso do estagiário;

7.2 Verificar se o aluno está regularmente matriculado e frequentando o curso;

7.3 Verificar a compatibilidade do horário estabelecido no plano de estágio com as atividades escolares; 7.4 Não permitir que a realização do estágio não obrigatório provoque atrasos na conclusão do curso;

7.5 Informar à **CONCEDENTE** a frequência e o desempenho acadêmico do estagiário, sempre que solicitado; bem como os períodos de avaliação escolar e outras atividades acadêmicas obrigatórias do estudante;

7.6 Comunicar à **CONCEDENTE** a integralização curricular, colação de grau, trancamento ou abandono do curso por parte do estagiário;

7.7 Acompanhar a realização do estágio, garantindo o cumprimento do Plano de Atividades.

**8. COMPROMISSOS DA CONCEDENTE:**

8.1 Orientar profissionalmente o **ESTAGIÁRIO**, zelando pelo desenvolvimento das atividades realizadas; com fiel cumprimento do plano de estágio;

8.2 Comunicar à Coordenação do Curso, quaisquer atitudes tomadas, diante de irregularidades e faltas cometidas pelo **ESTAGIÁRIO;**

8.3 Enviar à Coordenação do Curso ao qual o aluno está vinculado na **UFPE,** a cada 6 meses de duração do estágio, e ao final, avaliação do desempenho do **ESTAGIÁRIO**, acompanhado do relatório das atividades desenvolvidas;

8.4 Emitir termo de realização do estágio quando do desligamento do **ESTAGIÁRIO**;

8.5 Garantir o gozo de recesso, de 30 dias, remunerado, quando a duração do estágio for igual ou superior a 1 ano, e recesso proporcional quando o estágio tiver duração inferior a um ano;

8.6. Contratar em favor do **ESTAGIÁRIO** seguro contra acidente de trabalho;

8.7 Assegurar ao **ESTAGIÁRIO** o pagamento de Bolsa e Auxílio Transporte, mensais;

8.8 Efetuar o pagamento do seguro de cobertura de acidentes pessoais;

8.9. Aplicar ao estagiário a legislação relacionada à saúde e segurança no trabalho.

**9. COMPROMISSOS DO ESTAGIÁRIO:**

9.1 Observar as normas e regulamentos internos da **CONCEDENTE**;

9.2 Cumprir a programação do estágio;

9.3 Zelar pelos bens e equipamentos da **CONCEDENTE**, sob os seus cuidados;

9.4 Manter em absoluto sigilo, durante e após o estágio, quaisquer informações de caráter confidencial a que tiver acesso;

9.5 Apresentar ao supervisor do estágio, ao professor orientador e a Coordenação do Curso ao final do período de seis meses e ao final do estágio relatório de atividades.

**10. SEGURO DE COBERTURA DE ACIDENTES PESSOAIS:** Seguradora:__ ; Nº da Apólice: _____. Início da Vigência: ____ /____ /_____; Término da Vigência: ____ /____ /_____ .

**11. BENEFÍCIOS :** O **ESTAGIÁRIO** fará jus, na vigência do estágio:

11.1 Bolsa de R$ _____ (_____), mensais;

11.2 Auxílio Transporte no valor de R$ _____ (_____), mensais;

11.3 Recesso remunerado de 30 dias após 1 ano de vigência do estágio, ou proporcional a sua duração, quando a mesma for inferior 01 ano;

11.4 Outros benefícios de caráter não compulsório:_____

**12. DISPOSIÇÕES GERAIS:**

12.1. A concessão do estágio não acarretará vínculo empregatício de qualquer natureza entre o estagiário e a **CONCEDENTE**;

12.2. As atividades desenvolvidas através deste Estágio, não substituem o estágio curricular obrigatório do estudante;

12.3. O período de vigência do estágio será de 06 (seis) meses, podendo ser renovado por igual período, através de Termo Aditivo e mediante apresentação do relatório de atividades e da avaliação do desempenho do estagiário, incluindo o desempenho acadêmico;

12.4. O período total do estágio não poderá ser superior a dois anos;

12.5. O termo de compromisso de estágio poderá ser rescindido em qualquer período de realização por solicitação da **CONCEDENTE**, do **ESTAGIÁRIO** ou da **UFPE** sem nenhuma indenização de qualquer das partes, salvo a garantia do recesso remunerado proporcional ao período de estágio realizado, mediante comunicação por escrito realizada com antecedência mínima de 5 dias.

**13. FORO:** O foro da Justiça Federal em Pernambuco é o competente para dirimir quaisquer questões oriundas da execução deste convênio ou para a interpretação deste instrumento.

Este instrumento é firmado em 03 (três) vias de igual teor e forma, cabendo a 1ª à **CONCEDENTE,** a 2ª ao **ESTAGIÁRIO** e a 3ª à **UFPE.**

Recife, ____ de _____ de _____

_____ _____

**ESTAGIÁRIO** *Nome e Cargo do rep. da Concedente)*

**CONCEDENTE**

_____

**INSTITUIÇÃO DE ENSINO**

**(Coordenador(a) do Curso de** *(indicar o Curso)* **– UFPE)**