**FACULDADES INTEGRADAS UPIS**

**DIGITE AQUI O NOME DO DEPARTAMENTO OU COORDENAÇÃO**

**CIÊNCIAS ECONÔMICAS**

**ALEXANDRE PEREIRA GUIMARÃES**

**PREVIDÊNCIA SOCIAL: ESTUDO DE IMPACTOS ECONÔMICOS, UMA ABORDAGEM ANALÍTICA AGREGADA UTILIZANDO POSTGRESQL, PYTHON E WEKA**

**TRABALHO DE CONCLUSÃO DE CURSO**

**BRASÍLIA**

**2020ALEXANDRE PEREIRA GUIMARÃES**

**PREVIDÊNCIA SOCIAL: ESTUDO DE IMPACTOS ECONÔMICOS, UMA ABORDAGEM ANALÍTICA AGREGADA UTILIZANDO POSTGRESQL, PYTHON E WEKA**

Trabalho de Conclusão de Curso apresentada como requisito parcial à obtenção do título de Bacharel em Ciências Econômicas, do (nome do Departamento / Coordenação), da Faculdade integrada UPIS.

Orientador: Prof. Dr. Nome Completo

Co-orientador: (se houver) Prof. Dr. Nome Completo

**BRASÍLIA**

**2020**

Folha destinada à inclusão da **Ficha Catalográfica** (elemento obrigatório somente para teses e dissertações) a ser solicitada ao Departamento de Biblioteca da UTFPR e posteriormente impressa no verso da Folha de Rosto (folha anterior).



|  |  |  |
| --- | --- | --- |
|  | Ministério da Educação  **Universidade Tecnológica Federal do Paraná**  Nome do Campus  Nome da Diretoria / Coordenação / Departamento  Preencher com o Nome do Curso |  |
|  |  |  |

**TERMO DE APROVAÇÃO**

**(A SER FORNECIDA PELA SECRETARIA DO CURSO)**

TÍTULO DO TRABALHO

por

NOME DO(A) ALUNO(A)

Este(a) preencher se Trabalho de Conclusão de Curso (TCC / Monografia / Dissertação / Tese foi apresentado(a) em preencher o dia de preencher o mês de preencher o ano como requisito parcial para a obtenção do título de preencher se Bacharel / Tecnólogo / Especialista / Mestre / Doutor em preencher o nome do curso. O(a) candidato(a) foi arguido(a) pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(escreva aqui o nome do orientador)

Prof.(a) Orientador(a)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(escreva aqui o nome do membro titular)

Membro titular

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(escreva aqui o nome do membro titular)

Membro titular

- O Termo de Aprovação assinado encontra-se na Coordenação do Curso -

.

Dedico este trabalho à, absolutamente, todas as pessoas.

**AGRADECIMENTOS**

Agradeço aos meus colegas de trabalho por incentivarem o desenvolvimento do sistema aqui utilizado.

Agradeço ao Henrique Santos Fonseca Veloso por me mostrar o caminho que viria a trilhar. E à Mirian Izawa Mitusuko por me fazer reconhecer o impacto positivo do meu trabalho.

Agradeço à todas as comunidades Open source, sem os seus esforços não haveria como este trabalho existir.

Espaço destinado à epígrafe (elemento opcional). Nesta folha, o autor usa uma citação, seguida de indicação de autoria e ano, relacionada com a matéria tratada no corpo do trabalho.

Exemplo:Eu denomino meu campo de Gestão do Conhecimento, mas você não pode gerenciar conhecimento. Ninguém pode. O que você pode fazer, o que a empresa pode fazer é gerenciar o ambiente que otimize o conhecimento. (PRUSAK, Laurence, 1997)

**RESUMO**

GUIMARÃES, Alexandre**.** **Previdência social:** Estudo de Impactos Econômicos, uma abordagem analítica agregada, utilizando PostgreSQL, Python e WEKA. 2020. Número total de folhas. Trabalho de Conclusão de Curso (Bacharelado em Ciências econômicas) - Faculdade Integrada UPIS. Brasília, 2020.

Ferramentas de Inteligência de Negócios são frequentemente utilizados para suprir as necessidades de informação para tomadas de decisão das diretorias de empresas de pequeno, médio e grande porte, no entanto, não encontrei registro de nenhum sistema sendo utilizado na área de estudos da economia por parte das universidades. A complexidade dos sistemas de informação, estudos e disponibilização de gráficos varia de acordo com as necessidades e restrições de cada instituição, filtrando, assim, as ferramentas utilizadas e tornando únicas as características de sistemas distintos. Visando estudar os impactos econômicos gerados pelos benefícios da previdência social por meio da melhoria da qualidade de vida dos habitantes de todos os estados brasileiros as ferramentas escolhidas para este estudo são: PostgreSQL, Python, WEKA e Metabase.

**Palavras-chave:** Inteligência. Economia. Previdência. PostgreSQL. Python. WEKA. Metabase.

**ABSTRACT**

GUIMARÃES, Alexandre**.** **Social Security:** Economic Impact, an aggregate analytical approach, using PostgreSQL, Python and WEKA. 2020. Número total de folhas. Trabalho de Conclusão de Curso (Bacharelado em Ciências econômicas) - Faculdade Integrada UPIS. Brasília, 2020.

Business Intelligence tools are often used to meet the information needs for decision-making by the boards of small, medium and large companies, however, I have not found a record of any system being used in ​​economic studies by universities. The complexity of the information systems, studies and graphics availability varies according to the needs and restrictions of each institution, thus filtering the tools used and making each systems characteristics unique. In order to study the economic impacts generated by social security benefits by improving the quality of life of the inhabitants of all Brazilian states, the tools chosen for this study are: PostgreSQL, Python, WEKA and Metabase.

**Keywords:** Business. Intelligence. Economics. PostgreSQL. Python. WEKA. Metabase.

**LISTA DE ILUSTRAÇÕES**

Elemento opcional. São ilustrações: figuras, quadros, gráficos, fotografias, retratos, desenhos, gravuras, imagens, fluxogramas, organogramas, esquemas, mapas, plantas.

Recomenda-se a elaboração de listas **específicas e separadas** para cada tipo de ilustração a partir da existência de cinco elementos da mesma espécie. Para uma lista de ilustrações, organizá-la por ordem alfabética dos elementos.

Para atualizar a lista, clicar com o botão direito do mouse sobre o sumário em Atualizar campo e selecionar uma opção disponível de acordo com a necessidade.

[Figura 1 - As dimensões curriculares de pré-escolar 19](#_heading=h.49x2ik5)

[Figura 2 - Capa do livro: Normas para Elaboração de Trabalhos Acadêmicos 33](#_heading=h.4f1mdlm)

[Fotografia 1 - Entrada da Biblioteca da UTFPR Ponta Grossa 19](#_heading=h.2u6wntf)

[Gráfico 1 - Estatística de Empréstimos em Janeiro de 2009 18](#_heading=h.19c6y18)

[Quadro 1 - Áreas de Desenvolvimento de Competências 18](#_heading=h.3tbugp1)

**LISTA DE TABELAS**

Elemento opcional. É a relação das tabelas contidas no trabalho. Estas devem vir em lista própria, de acordo com a ordem apresentada no texto, com cada item designado por seu nome específico, acompanhado do respectivo número da página.

Para atualizar a lista, clicar com o botão direito do mouse sobre o sumário em Atualizar campo e selecionar uma opção disponível de acordo com a necessidade.

[Tabela 1 - Desempenho dos alunos na prova de conhecimentos específicos 20](#_heading=h.147n2zr)

[Tabela 2 - Situação da Educação Brasileira em 2002 – Ensino Médio 20](#_heading=h.3o7alnk)

**SUMÁRIO**

O sumário deve ser colocado como último elemento pré-textual. O sumário deve ser apresentado conforme o que segue:

* os elementos pré-textuais não aparecem no sumário;
* a ordem dos elementos do sumário deve ser conforme os indicativos das seções que compõem o trabalho;
* o texto de um documento deve ser dividido em seções, sendo a seção primária a principal (1), seguida de outras divisões, como a seção secundária (1.1); seção terciária (1.1.1); seção quaternária (1.1.1.1). Deve-se limitar a numeração progressiva até a seção quinária (1.1.1.1.1), inclusive;
* o espaçamento entre títulos no sumário deve ser simples.

Para atualizar as listas e o sumário, clique com o botão direito do mouse sobre o índice em Atualizar campo e selecione **Atualizar apenas o número de página** ou **Atualizar o índice inteiro**.

[**1**](#_heading=h.28h4qwu) **INTRODUÇÃO 13**

[**2**](#_heading=h.3j2qqm3) **SEÇÃO PRIMÁRIA (EM CAIXA ALTA E EM NEGRITO) 15**

[2.1](#_heading=h.nmf14n) SEÇÃO SECUNDÁRIA (EM CAIXA ALTA E SEM NEGRITO) 15

[2.1.1](#_heading=h.37m2jsg) Seção Terciária (Com as Iniciais Maiúsculas) 15

[2.1.1.1](#_heading=h.1mrcu09) Seção quaternária (Primeira letra do título Maiúscula) 15

[*2.1.1.1.1*](#_heading=h.46r0co2) *Seção quinária (Primeira letra do título maiúscula e em itálico) 15*

[**3**](#_heading=h.2bn6wsx) **DESENVOLVIMENTO 16**

[**4**](#_heading=h.2lwamvv) **CITAÇÕES 22**

[**5**](#_heading=h.111kx3o) **CONCLUSÃO 23**

[**REFERÊNCIAS 24**](#_heading=h.41mghml)

[**APÊNDICE A -**](#_heading=h.3l18frh) **Questionário de Pesquisa 25**

[**APÊNDICE B -**](#_heading=h.206ipza) **Roteiro da Entrevista 27**

[**ANEXO A -**](#_heading=h.3fwokq0) **Direitos autorais - Lei nº 9.610, de 19 de fevereiro de 1998. Disposições preliminares 29**

[**ANEXO B -**](#_heading=h.1v1yuxt) **Capa do livro: Normas para Elaboração de Trabalhos 32**

1. **INTRODUÇÃO**

Atualmente a utilização de banco de dados está focada em grande parte, dentro das empresas, poucas universidades utilizam bancos de dados de forma intensiva e ostensiva, com a grande quantidade de informação existente na internet de órgãos públicos, se faz necessária a utilização de linguagens de programação e banco de dados para coleta e armazenamento de dados de forma eficiente, ao adicionar uma ferramenta de análise e outra para disponibilização de gráficos a partir dos dados obtidos o resultado será um sistema altamente eficaz para estudos de grandes quantidades de dados de qualquer órgão de qualquer local do mundo, desde que seus dados sejam acessíveis ao público.

Este estudo visa identificar quais variáveis sociais e econômicas são afetadas pela previdência e evidenciar os seus impactos sobre a economia, mas além disso tem como objetivo mostrar a utilização da Linguagem de programação Python, do Banco de Dados PostgreSQL, da ferramenta de análise de dados WEKA e a utilização dos gráficos criados pela ferramenta Metabase, por isso, o objetivo primário deste estudo é mostrar o impacto econômico da previdência social em todos os estados brasileiros enquanto o objetivo secundário será ensinar ao leitor como utilizar as técnicas e metodologias para chegar aos mesmos resultados no fim deste estudo.

**2 FUNDAMENTOS TEÓRICOS**

2.1 POSTGRESQL

2.1.1 O que é PostgreSQL

Segundo a própria documentação do PostgreSQL, ele é um sistema de banco de dados de código livre (open source) relacionado à objeto que utiliza e estende a linguagem SQL para combinar muitos recursos para armazenar e dimensionar as quantidades de cargas de dados necessárias à execução de trabalhos de forma segura.

O PostgreSQL ganhou uma forte reputação pela sua arquitetura, confiabilidade, integridade dos dados, conjunto robusto de recursos, possibilidade de utilização de novas extensões e pela comunidade open source que cria, consistentemente, novas soluções performáticas e inovativas no desenvolvimento das atualizações do software. O PostgreSQL funciona em todos os principais sistemas operacionais (FreeBDS, OpenBDS, Linux, MacOS, Solaris e Windows) e possui várias complementações importantes, como a extensão de banco de dados geoespacial PostGIS.

2.1.2 A história de PostgreSQL

2.1.2.1 Software Open source

O termo “Software de código aberto” é um programa que tem seu código fonte lançado sob uma licença onde o dono compartilha o direito de edição, estudo e distribuição do software para todas as pessoas e para qualquer propósito, de forma comumente conhecida, programas de código aberto tem colaboração pública em seu desenvolvimento.

2.1.2.2 Origem do Postgres

De acordo com Momjian (2002, p. 84), “*O antecessor do PostgreSQL era chamado de Ingres, desenvolvido na Universidade de Califórnia em Berkeley (1977-1985). O código do Ingres foi posteriormente aprimorado pela Relational Technologies / Ingres Corporation, que produziu um dos primeiros servidores de banco de dados relacional que foram bem sucedidos comercialmente. Acompanhado a isso, Michael Stonebraker liderou uma equipe para desenvolver um banco de dados relacionado à objeto chamado Postgres (1984-1994). Illustra pegou o código e o desenvolveu em um produto comercial. Dois graduados de Berkeley, Jolly Chen e Andrew Yu, posteriormente adicionaram os recursos de SQL ao Postgres. O resultado foi o projeto chamado Postgres95 (1994-1995).*’’

2.1.2.2 Desenvolvimento fora de Berkeley

Seguindo as palavras de Momjian (2002, p. 85), *“No verão de 1996 ficou claro que havia uma grande demanda por um servidor de banco de dados SQL open source e formou-se uma equipe para continuar o desenvolvimento. Então, Marc Q. Fournier de Toronto, Canadá, se ofereceu para fazer o hosteamento da lista de emails e providenciar uma servidor para servir de host para o código fonte. Posteriormente mil inscrições da lista de emails foram movidas para a nova lista, um servidor foi configurado, dando acesso à algumas pessoas para aplicar patches ao código fonte usando CVS.*

*No início o projeto necessitava de poucas pessoas com muito tempo livre, não muitas pessoas com pouco tempo livre, então, com por volta de 250.000 linhas de código escritas em C, quatro pessoas estavam fortemente envolvidas no projeto: Marc Fournier no Canadá, Thomas Lockhart em Pasadena, Califórnia, Vadim Mikheev em Krasnoyarsk, Rússia e Bruce Momjian em Filadélfia, Pensilvânia.*

*O primeiro objetivo deles foi vasculhar discussões antigas nos emails para avaliar patches que já foram postados para solucionar vários problemas, o sistema, na época, era frágil e não era de fácil compreensão, nos primeiros seis meses de desenvolvimento os desenvolvedores temiam que um único patch fosse quebrar o sistema e não conseguissem resolver o problema, a cada relatório de erro gerado os desenvolvedores deveriam não só encontrar o que estava de errado mas também como o servidor executava a maioria de suas funções.*

*Eles tinham herdado uma base instalada enorme, um dos erros típicos era: “Quando eu faço “isso”, o banco de dados trava”, com uma lista enorme de erros ficou claro que uma forma de organização era necessária, a maioria dos erros precisava significativamente de pesquisas para consertar e muitos erros estavam duplicados, então, a lista do “Para fazer” incluía queries de SQL com muitos erros. Essa abordagem ajudava a identificar os erros e fazia com que os usuários estivessem cientes dos erros também, assim, reduzindo os relatórios de erros duplicados.*

*Embora tivessem desenvolvedores ansiosos para ajudar, a curva de aprendizado de como o banco de dados funcionava atrapalhava significantemente desenvolvedores novos a corrigir os erros, por causa disso, muitos desenvolvedores corrigiram pequenos pedaços do código, como interfaces da linguagem e ferramentas do banco de dados, que eram coisas mais fáceis de entender, outros desenvolvedores focaram especificamente em problemas de queries, tentando encontrar a fonte do erro no código.*

*Era fascinante ver que muitos erros foram consertados utilizando apenas uma linha de código em C, em razão de o Postgres ter evoluído em um ambiente acadêmico, ele não foi exposto ao espectro total das “queries do mundo real”. Durante esse período foi discutida a possibilidade de adicionar novos recursos mas a instabilidade do sistema fazia com que a correção dos erros fosse o foco prioritário dos desenvolvedores.”*

2.1.2.3 Desenvolvimento global do PostgreSQL

Nas palavras de Momjian (2002, p. 87 - 88), *“No final de 1996 o nome do projeto foi mudado de Postgres95 para PostgreSQL, o código fonte foi distribuído remotamente via CVS, que permitia que as pessoas manterem suas cópias atualizadas da árvore de desenvolvimento sem ter que fazer download de um conjunto de arquivos todos os dias.*

*As atualizações ocorriam a cada de 3 a 5 meses, cada período consistia de 2 a 3 meses de desenvolvimento, um mês de testes beta, o lançamento da versão principal e algumas semanas de micro atualizações para corrigir erros graves, os desenvolvedores nunca tentaram lançar versões muito próximas uma da outra nem tentaram corrigir mais erros do que o possível dentro dessa quantidade limitada de tempo, um banco de dados não é um simples processador de palavras nem um jogo, onde é possível reiniciar quando um problema ocorre, bancos de dados são multiusuários e bloqueiam os dados dos usuários dentro do banco, por isso eles devem ser tão confiáveis quanto o possível.*

*O desenvolvimento de um código fonte dessa escala e complexidade não é para iniciantes, inicialmente, houveram problemas para capturar interesse de desenvolvedores nesse projeto com uma curva de aprendizado tão íngreme, no entanto, o tempo, uma atmosfera civilizada e confiabilidade e performance aumentadas ajudaram a atrair talentos experientes que eram necessários.*

*Dar aos desenvolvedores os conhecimentos que eles precisavam para ajudar a desenvolver o PostgreSQL era claramente uma prioridade. Eles tinham uma lista de “Coisas para fazer” que descrevia o que deveria ser feito, mas com 250.000 linhas de código abordar qualquer item é um projeto de mérito próprio. Perceberam que a educação de desenvolvedores traria grandes resultados em ajudar as pessoas que estavam começando. Foi escrito um fluxograma detalhado dos módulos do banco de dados, assim como, o FAQ dos desenvolvedores, que respondia às perguntas frequentes dos desenvolvedores do PostgreSQL, com esse tipo de informação os desenvolvedores se tornaram mais produtivos na correção de erros e adicionamento de ferramentas.*

*Embora o código fonte tenha sido herdado de Berkeley fosse muito modular, a maioria dos coders de Berkeley usavam PostgreSQL como banco de ensaios para projetos de pesquisa, por causa disso, aprimorar códigos existentes não era uma prioridade.*

*Uma ferramenta foi criada para reformatar o código fonte de forma consistente, escreveram um script para achar funções que poderiam estar marcadas como static ou funções que não estavam sendo utilizadas que poderiam ser removidas, esses scripts são executados todas antes de uma nova atualização.*

*Enquanto ganhavam conhecimento a respeito do código os desenvolvedores se tornavam capazes de executar correções mais complicadas e adicionar novos recursos, mas ainda com uma estrutura fraca para as novas adições. Então eles entraram em um novo modo de lançamentos de atualizações onde a cada atualização da versão principal novos recursos seriam adicionados, ao invés de ser apenas correções de erros, com o tempo, melhoraram a conformidade com o SQL, adicionaram sub-seleções, melhoraram o bloqueio e adicionaram funcionalidades SQL ausentes. Uma empresa foi formada para oferecer suporte por telefone.*

*Mais tarde, grupos de discussão da Usenet começaram a divulgar o PostgreSQL, em algum momento, os desenvolvedores pesquisaram “PostgreSQL” e encontravam pessoas recomendando bancos de dados distintos, embora estivessem resolvendo os problemas dos usuários o mais rápido possível, um ano após isso o PostgreSQL passou a ser recomendado por várias pessoas que necessitavam de suporte a transações, queries complexas, suporte de qualidade comercial para SQL, tipos de dados complexos e confiabilidade, claramente os pontos fortes do PostgreSQL, outros tipos de bancos de dados eram recomendados quando a velocidade era a prioridade. A integração do PostgreSQL à distribuição Red Hat do Linux expandiu rapidamente a base de usuários*

*Hoje, cada versão do PostgreSQL é uma grande melhoria em relação à última. Em 2002 a equipe de desenvolvedores ainda dominava o código fonte herdado de Berkeley, além disso, todo o módulo era compreendido por pelo menos um membro do grupo, e a adição de novos recursos se torna cada vez mais fácil graças ao aumento do tamanho e da experiência da equipe de desenvolvimento global do PostgreSQL.”*

2.1.3 Utilização do PostgreSQL

Como o PostgreSQL é apenas um banco de dados, que pode, ou em certas ocasiões deve, ser configurado durante ou após a instalação, é necessária a instalação de uma ferramenta gráfica para que seja possível observar e monitorar o comportamento do banco de dados, assim como, tornar mais fácil escrever códigos e ver seus resultados à medida que o usuário for criando os cruzamentos de dados necessários para qualquer estudo necessário em qualquer organização que utilizar o PostgreSQL, caso a ferramenta gráfica não seja instalada o PostgreSQL somente poderá ser utilizado utilizando a via de comando no Windows ou os terminais do Linux e Mac.

A ferramenta gráfica que foi utilizada neste trabalho é também a recomendada pela própria documentação do PostgreSQL no tutorial de instalação do banco de dados, é chamada de “PgAdmin 4”, a instalação dessa ferramenta é recomendada para iniciantes, quase chegando a ser obrigatória, que ainda estão aprendendo os comandos do banco de dados, mas a utilização de ferramentas gráficas é opcional como veremos mais adiante no desenvolvimento deste trabalho.

2.1.4 Funcionalidades básicas

Aqui estarão disponíveis para consulta os comandos essenciais e mais utilizados do PostgreSQL na elaboração deste trabalho.

Todos os comandos listados estão disponíveis na documentação original do PostgreSQL do ano 2020 na url: <https://www.postgresql.org/docs/> .

2.1.4.1 A linguagem SQL

A linguagem SQL é utilizada em banco de dados relacionais, sua sigla significa “Structured Query Language” sua tradução equivale a Linguagem de Consulta Estruturada, como toda linguagem de programação tudo que será escrito deverá estar conforme os códigos descritos na documentação em forma de comando ao banco de dados, caso contrário os comandos não serão executados por não serem compreendidos pelo banco de dados.

A forma como o banco de dados interpreta os comando é invariável e sequencial por natureza como uma função matemática.

2.1.4.2 Comandos básicos

Os comandos básicos do PostgreSQL são relativos às necessidades de criar, inserir dados e atualizar tabelas, para isso são utilizados inúmeras formas de estruturação dos comandos básicos para atingir os mesmo resultados.

Entre os comandos básicos do PostgreSQL os mais simples, e fáceis de entender visualmente foram utilizados neste estudo para facilitar o seu entendimento, entre eles são: “Create Table”, “Drop Table”, “Insert Into”, “Into”, “Select”, “From”, “Update”, “Set” e ”Where”.

2.1.4.3 Create Table

A utilização do “Create Table” neste trabalho se dá, como exemplo a seguinte forma:

***“Create Table schema.tabela (“coluna 1“ tipo\_de\_dado\_da\_coluna\_1, “coluna 2” tipo\_de\_dado\_da\_coluna\_2, “coluna 3” tipo\_de\_dado\_da\_coluna\_3);”***

Assim, o comando Create Table cria uma tabela em um “schema” já criado posteriormente e adiciona as colunas: “coluna 1”, “coluna 2” e “coluna 3” utilizando um tipo de dados para cada coluna individual, podendo variar de acordo com qualquer necessidade.

Os “schemas” dentro do banco de dados funcionam como as pastas dentro de qualquer computador e utilizando os códigos pode-se definir qual deles o usuário quer retirar informação de alguma tabela.

2.1.4.4 Insert Into

A forma como o comando “Insert Into” foi utilizado está como descrito abaixo:

***“Insert Into schema.tabela (“coluna 1”, “coluna 2”, “coluna 3”, “coluna 4”)”***

O comando acima está incompleto, pois é a forma básica do comando, onde ele determina o schema e a tabela que as colunas 1, 2, 3 e 4 deverão ter algum valor adicionado a elas, mas não determina se o valor é determinado pelo próprio usuário ou se é algum valor dentro de outra tabela que deve ser levado à tabela, ao longo deste trabalho todos os códigos escritos vão ser analisados e assim cada utilização nova será comentada, aqui está somente a versão básica do comando caso seja necessária o entendimento básico da função, é altamente aconselhado o acompanhamento deste trabalho com a documentação do PostgreSQL online ou em mãos.

2.1.4.5 Select \* From

O comando “Select” é obrigatoriamente seguido do comando “From” e é utilizado da seguinte forma:

***“Select “coluna 1”, “coluna 2”, “coluna 3”, “coluna 4”, “coluna 5” From schema.tabela”***

Select é o comando para selecionar as colunas de uma determinada tabela, o nome das colunas escritas no código deve ser igual ao nome das colunas dentro de uma tabela já criada, caso não haja uma tabela na tabela selecionada ocorrerá um erro mostrando onde se encontra o erro.

From é o comando que especifica o schema e a tabela de onde os dados serão retirados, seu uso é necessário para que a seleção das colunas seja realizado.

Alternativamente, pode-se escrever “Select \* From schema.tabela” onde o comando selecionará todas as colunas existentes dentro de uma tabela e irá disponibilizar tudo de uma vez.

2.1.4.6 Update, Set, Where

O comando “Update” é utilizado para atualizar dados de uma tabela, e é necessário que seja acompanhado pelos comandos “Set”, “From” e opcionalmente ”Where”, o código pode ser escrito da seguinte forma:

***“update schema.tabela set “coluna a ser atualizada” = “variável ou outra coluna de outra tabela” from “tabela da coluna que servirá para atualizar a coluna a ser atualizada” where (condição específica)”***

“Update” é o comando que especifica qual tabela em qual schema deve ser atualizado enquanto “Set” determina qual coluna ou dado deve ser atualizado dentro da tabela selecionada pelo “Update” e where especifìca uma condição para se o dados selecionado é atualizado ou não, ou especifica a forma como é alterado.

O comando “From” já teve a sua utilidade explicada anteriormente.

2.1.5 Demais informações

De forma a não estender este trabalho, todo o resto dos comandos e informações a respeito do PostgreSQL serão explicadas à medida que a apresentação deste trabalho for desenvolvida.

De toda forma a documentação original do PostgreSQL contém informações mais detalhadas a respeito das funcionalidades do PostgreSQL, tutoriais de instalação, tudo a respeito do banco de dados se encontra em sua própria documentação, por essa razão é de extrema utilidade a utilização dela para acompanhamento deste trabalho.

2.2 PYTHON

2.2.1 O que é Python?

Python é uma linguagem de programação orientada a objetos, com semântica dinâmica e simples, que conta com atuação de desenvolvedores para expandir suas capacidades de programação na forma de livrarias e módulos que podem ser importados para o código para realizar alguma função específica que seja necessária.

A sua forma orientada a objetos é uma forma de programação que visa controle e a estabilidade quando utilizadas grandes quantidades de dados enquanto a sua semântica dinâmica torna programar em python mais eficiente pois variáveis com o mesmo nome podem ser usadas, sem ocasionar erro quando são utilizadas de forma adequada e sua simplicidade torna a linguagem mais fácil para ser compreendida e aprendida por todos.

2.2.2 A história do Python

2.2.2.1 Os primeiros momentos

Nas palavras de Guido Van Rossum (2009), em seu blog pessoal, no artigo “Personal History - part 1, CWI”, a respeito da história do desenvolvimento da linguagem Python, *“O desenvolvimento inicial de Python começou em instituto de pesquisa em Amsterdam chamado CWI, uma sigla holandesa para uma frase que se traduz em português como Centro para Matemática e Ciência da Computação. CWI era um local interessante; financiado pelo Departamento de Educação do governo holandês e outros fundos de pesquisa, conduzia pesquisa acadêmica em ciência da computação e matemática. Em qualquer época havia vários estudantes de doutorado andando pelo local e veteranos na profissão ainda devem lembrar do nome original, o Centro de Matemática. Sob esse nome, provavelmente o centro ganhou fama pela invenção de Algol 68.*

*Comecei a trabalhar no CWI no final de 1982, recém saído da universidade, como um programador do grupo ABC liderado por Lambert Meertens e Steven Pemberton. Depois de 4 ou 5 anos, o projeto ABC foi encerrado devido à óbvia falta de sucesso e eu fui para o grupo Amoeba liderado por Sape Mullender. Amoeba era um sistema distribuído baseado em microkernel desenvolvido conjuntamente pelo CWI e Vrije Universiteit de Amsterdam, sob liderança de Andrew Tanenbaum. Em 1991, Sape deixou o CWI para lecionar na Universidade de Twente e eu terminei no recentemente formado grupo de multimídia do CWI liderado por Dick Bulterman.*

*Python é um produto direto da minha experiência no CWI. Como explicarei depois, ABC me deu a inspiração crucial para Python, Amoeba a motivação imediata e o grupo de multimídia fomentou seu crescimento. Entretanto, pelo que eu sei, nenhum fundo no CWI foi oficialmente reservado para seu desenvolvimento. Em vez disso, Python meramente se desenvolveu como uma ferramenta importante tanto para o grupo Amoeba quanto para o grupo multimídia.*

*Minha motivação para a criação de Python foi a percepção da necessidade de uma linguagem de alto nível no projeto Amoeba. Percebi que o desenvolvimento de utilitários para administração de sistema em C estava tomando muito tempo. E fazê-los no shell Bourne não funcionaria por diversas razões. O mais importante foi que, sendo um sistema distribuído de microkernel, as operações primitivas do Amoeba eram bem diferentes (e refinadas) que as operações primitivas disponíveis no shell Bourne. Portanto, havia necessidade de uma linguagem que "preencheria o vazio entre C e o shell". Por um tempo longo, esse foi o principal lema de Python.*

*Nesse ponto, você poderia perguntar "por que não adaptar uma linguagem existente?" No meu ponto de vista, não havia linguagens adequadas à época. Eu era familiar com Perl 3, mas essa estava ainda mais ligada ao Unix que o shell Bourne. Eu também não gostava da sintaxe de Perl -- meu gosto por sintaxes de linguagens de programação foram fortemente influenciadas por linguagens como Algol 60, Pascal, Algol 68 (todas eu havia aprendido anteriormente), e finalmente, ABC, na qual eu passei quatro anos da minha vida. Portanto, decidi projetar uma linguagem por mim mesmo a qual poderia pegar tudo que gostei de ABC enquanto, ao mesmo tempo, consertar todos seus problemas (como os percebia).*

*O primeiro problema que decidi consertar foi o nome! Da forma que ocorreu, o time ABC teve alguns problemas ao escolher um nome para a linguagem. O nome original, B, teve que ser abandonado por causa de uma confusão com outra linguagem chamada B, que era mais velha e melhor conhecida. De qualquer modo, a ideia era usar B como título de projeto apenas (a piada era que B era o nome da variável contendo o nome da linguagem). A equipe lançou uma competição pública para escolher o novo nome, mas nenhuma das submissões levou e, no fim, a segunda opção interna prevaleceu. O nome era para transmitir a ideia de uma linguagem que faria programação "tão simples quanto ABC", mas nunca me convenceu a tanto.*

*Portanto, em vez de analisar demasiadamente o problema de nomenclatura, decidi subanalisá-lo. Escolhi a primeira coisa que me veio à cabeça, que foi Monty Python’s Flying Circus, um dos meus grupos de comédia favoritos. A referência era convenientemente irreverente para o que era essencialmente um "projeto skunkworks", um projeto desenvolvido por um grupo fracamente estruturado e com alto grau de autonomia. A palavra "Python" também era cativante, um pouco irritante e, ao mesmo tempo, encaixava na tradição de nomear linguagens em homenagens a pessoas famosas, como Pascal, Ada e Eiffel. O grupo Monty Python pode não ser famoso pelos avanços em ciência ou tecnologia, mas certamente é um favorito dos geeks. Também se encaixou na tradição do grupo Amoeba no CWI de batizar programas com nomes de programas de TV.*

*Por muitos anos eu resisti à tentação de associar a linguagem com cobras. Finalmente desisti quando O'Reilly queria colocar uma cobra na capa de seu primeiro livro de Python "Programming Python". Era tradição da O'Reilly usar imagens de animais e, se deveria ser um animal, que fosse uma cobra.*

*Com o problema do nome resolvido, eu comecei a trabalhar em Python no fim de dezembro de 1989 e tinha uma versão operacional nos primeiros meses de 1990. Eu não anotava, mas lembro vividamente que o primeiro trecho de código que escrevi para implementação de Python era um simples gerador de parser LL(1) que chamei de "pgen". Esse gerador de parser ainda é parte do código-fonte de Python e provavelmente o menos modificado em todo o código. Essa versão inicial foi usada por uma pequena quantidade de pessoas no CWI, mas não exclusivamente pelo grupo Amoeba em 1990. Desenvolvedores chave além de mim eram meus colegas de escritório, os programadores Sjoerd Mullender (irmão mais novo de Sape) e Jack Jansen (que permaneceu um dos desenvolvedores líderes da versão para Macintosh vários anos depois que eu saí do CWI).*

*Em 20 de fevereiro de 1991, eu liberei Python para o mundo pela primeira no grupo de notícias alt.sources (em 21 partes codificadas com uuencode que deveriam ser reunidas e decodificadas com uudecode para compor um arquivo tar comprimido). Essa versão foi numerada como 0.9.0 e liberada sob uma licença que era quase uma cópia exata da licença MIT usada pelo projeto X11 na época, substituindo "Stichting Mathematisch Centrum", organização pai do CWI, como entidade legal responsável. Então, como quase tudo que eu escrevi, Python era de código aberto antes que o termo fosse inventado por Eric Raymond e Bruce Perens ao final de 1997.*

*Imediatamente houve muito retorno e com esse encorajamento eu mantive um fluxo constante de lançamentos pelos próximos anos. Comecei a usar CVS para rastrear mudanças e facilitar compartilhamento de responsabilidades sobre a codificação com Sjoerd e Jack (coincidentemente, CVS foi originalmente desenvolvido como um conjunto de scripts shell por Dick Grune, que era um membro anterior do grupo ABC). Eu escrevi um FAQ, que era regularmente publicado em algum grupo de notícias, como era costume para FAQs naqueles dias antes da web, iniciei uma lista de discussão por e-mail e em março de 1993 o grupo comp.lang.python foi criado com meu encorajamento, mas sem meu envolvimento direto. O grupo e a lista de e-mails foram unificados através de um gateway bidirecional que ainda existe, embora atualmente implementado como uma funcionalidade do mailman - o gerenciador de listas dominante de código aberto, escrito em Python.*

*No verão de 1994, o grupo estava agitado com uma discussão intitulada "Se Guido fosse atingido por um ônibus?" sobre a dependência da crescente comunidade de Python em relação às minhas contribuições pessoais. Isso culminou em um convite por Michael McLay para que eu passasse dois meses como pesquisador convidado no NIST, o Instituto Nacional de Padrões e Tecnologias dos EUA, anteriormente o Escritório Nacional de Padrões, em Gaithersburg, Maryland. Michael possuía uma certa quantidade de "clientes" no NIST que estavam interessado em usar Python para vários projetos relacionados a padrões e os fundos para minha estadia foram motivados pela necessidade de ajudá-los a melhorar as habilidades com Python, bem como melhorar Python para suas necessidades.*

*O primeiro workshop de Python ocorreu enquanto eu estava lá em novembro de 1994, com o programador do NIST Ken Manheimer provendo assistência e encorajamento importantes. De aproximadamente 20 participantes, por volta da metade ainda são participantes ativos na comunidade Python e uns poucos se tornaram grandes líderes de projetos de código aberto (Jim Fulton do Zope e Barry Warsaw do GNU mailman). Com o suporte do NIST eu também dei uma palestra para aproximadamente 400 pessoas na conferência Usenix Little Languages em Santa Fe, organizada por Tom Christiansen, um advogado de Perl de mente aberta que me apresentou ao criador de Perl, Larry Wall, e ao autor de Tcl/Tk, John Ousterhout.”*

2.2.2.2 CNRI e além

Segundo Guido Van Rossum (2009), em seu blog pessoal, no artigo “Personal History - part 2, CNRI and beyond”, a respeito da história do desenvolvimento da linguagem Python, *“O workshop do Python resultou em uma oferta de trabalho para trabalhar com agentes móveis no CNRI (Corporation for National Research Initiatives). O CNRI é um laboratório de pesquisa sem fins lucrativos em Reston, Virgínia. Entrei em abril de 1995. O diretor da CNRI, Bob Kahn, foi o primeiro a me indicar o quanto o Python tinha em comum com o Lisp, apesar de ser completamente diferente em um nível superficial (sintático). O trabalho em Python no CNRI foi financiado indiretamente por uma doação da DARPA para pesquisa de agentes móveis. Embora houvesse suporte da DARPA para projetos que usavam Python, não havia muito suporte direto para o desenvolvimento da linguagem.*

*Na CNRI, liderei e ajudei a contratar uma pequena equipe de desenvolvedores para criar um sistema de agente móvel em Python puro. Os membros iniciais da equipe foram Roger Masse e Barry Warsaw, que estiveram no workshop do Python no NIST. Além disso, contratamos os membros da comunidade Python Ken Manheimer e Fred Drake. Jeremy Hylton, um graduado do MIT originalmente contratado para trabalhar na recuperação de texto, também se juntou à equipe. A equipe foi inicialmente gerenciada por Ted Strollo e mais tarde por Al Vezza.*

*Essa equipe me ajudou a criar e manter uma infraestrutura adicional da comunidade Python, como o site python.org, o servidor CVS e as listas de discussão de vários grupos de interesse especial do Python. As versões 1.3 a 1.6 do Python saíram do CNRI. Por muitos anos, o Python 1.5.2 foi a versão mais popular e mais estável.*

*O GNU mailman também nasceu aqui: originalmente usamos uma ferramenta Perl chamada Majordomo, mas Ken Manheimer a considerou impossível de manter e procurou uma solução em Python. Ele descobriu algo escrito em Python por John Viega e assumiu a manutenção. Quando Ken deixou a CNRI para a Digital Creations, Barry Warsaw assumiu o cargo e convenceu a Free Software Foundation a adotá-la como sua ferramenta oficial de lista de discussão. Portanto, Barry licenciou-a sob a GPL (GNU Public License).*

*Os workshops do Python continuaram, primeiro duas vezes por ano, mas devido ao crescimento e ao aumento dos esforços logísticos, eles logo evoluíram para eventos anuais. Eles foram executados pela primeira vez por quem quisesse hospedá-los, como o NIST (o primeiro), o USGS (o segundo e o terceiro) e o LLNL (o quarto e o início da série anual). Eventualmente, o CNRI assumiu a organização e, mais tarde (juntamente com as conferências da WWW e da IETF), isso foi desmembrado como um esforço comercial, a Fortec. O atendimento rapidamente subiu para várias centenas. Quando o Fortec desapareceu um pouco depois que eu deixei o CNRI, a International Python Conference foi incluída na OSCON (Open Source Conference) da O'Reilly, mas, ao mesmo tempo, a Python Software Foundation iniciou uma nova série de conferências de base denominadas PyCon.*

*Também criamos a primeira organização (livre) em torno do Python no CNRI. Em resposta aos esforços de Mike McLay e Paul Everitt para criar uma "Python Foundation", que acabou na areia movediça da elaboração de leis, Bob Kahn se ofereceu para criar a "Python Software Activity", que não seria uma entidade legal independente, mas simplesmente um grupo de pessoas que trabalham sob o guarda-chuva legal (sem fins lucrativos) da CNRI. O PSA conseguiu reunir a energia de um grande grupo de usuários comprometidos de Python, mas sua falta de independência limitou sua eficácia.*

*O CNRI também usou o dinheiro da DARPA para financiar o desenvolvimento do JPython (mais tarde abreviado para Jython), uma implementação Python para o Java. Jim Hugunin criou o JPython inicialmente enquanto fazia um trabalho de pós-graduação no MIT. Ele então convenceu a CNRI a contratá-lo para concluir o trabalho. Quando Jim deixou a CNRI menos de dois anos depois para ingressar no projeto AspectJ na Xerox PARC, Barry Warsaw continuou o desenvolvimento do JPython. (Muito mais tarde, Jim também criaria o IronPython, a porta Python para o .NET da Microsoft. Jim também escreveu a primeira versão do Numeric Python.)*

*Outros projetos no CNRI também começaram a usar o Python. Vários novos desenvolvedores principais do Python surgiram disso, em particular Andrew Kuchling, Neil Schemenauer e Greg Ward, que trabalharam no projeto MEMS Exchange. (Andrew havia contribuído com o Python mesmo antes de ingressar na CNRI; seu primeiro grande projeto foi o Python Cryptography Toolkit, uma biblioteca de terceiros que disponibilizou muitos algoritmos criptográficos fundamentais para os usuários do Python.)*

*Com o sucesso do Python, o CNRI tentou criar um modelo para financiar o desenvolvimento do Python mais diretamente do que por meio de bolsas de pesquisa da DARPA. Criamos o Python Consortium, modelado após o X Consortium, com uma taxa de entrada mínima de US $ 20.000. No entanto, além de um grupo da Hewlett-Packard, não tivemos muita atração e, eventualmente, o consórcio morreu por falta de recursos. Outra tentativa de encontrar financiamento foi a Programação de Computador para Todos (CP4E), que recebeu algum financiamento da DARPA. No entanto, o financiamento não foi suficiente para toda a equipe e verificou-se que havia toda uma rede de pessoas que obtinham a maior parte do dinheiro distribuído ao longo de vários anos. Não foi algo que eu gostei e comecei a procurar outras opções.*

*Eventualmente, no início de 2000, o boom das pontocom, que ainda não havia entrado em colapso, convenceu a mim e três outros membros da equipe do CNRI Python (Barry Warsaw, Jeremy Hylton e Fred Drake) a ingressar no BeOpen.com, um estado da Califórnia. startup que estava recrutando desenvolvedores de código aberto. Tim Peters, um membro importante da comunidade Python, também se juntou a nós.*

*Antecipando a transição para o BeOpen.com, uma pergunta difícil era a propriedade futura do Python. O CNRI insistiu em alterar a licença e solicitou o lançamento do Python 1.6 com essa nova licença. A licença antiga usada enquanto eu ainda estava no CWI tinha sido uma versão da licença do MIT. Os lançamentos feitos anteriormente no CNRI usavam uma versão ligeiramente modificada dessa licença, com basicamente uma frase adicionada, onde o CNRI se isentava da maioria das responsabilidades. A licença 1.6, no entanto, era uma longa e extensa lista de advogados elaborada pelos advogados da CNRI.*

*Tivemos várias discussões longas com Richard Stallman e Eben Moglen, da Free Software Foundation, sobre algumas partes desta nova licença. Eles temiam que isso fosse incompatível com a GPL e, portanto, ameaçavam a viabilidade do gerenciador de emails do GNU, que já havia se tornado uma ferramenta essencial para a FSF. Com a ajuda de Eric Raymond, foram feitas alterações na licença CNRI Python que satisfizeram tanto a FSF quanto a CNRI, mas a linguagem resultante não é fácil de entender. A única coisa boa que posso dizer sobre isso é que (novamente graças à ajuda de Eric Raymond) ele tem o selo de aprovação da Open Source Initiative como uma genuína licença de código aberto. Apenas pequenas modificações foram feitas no texto da licença para refletir as duas sucessivas mudanças de propriedade, primeiro o BeOpen.com e depois a Python Software Foundation, mas, em essência, o trabalho dos advogados da CNRI ainda permanece.*

*Como tantas startups na época, o plano de negócios do BeOpen.com falhou de maneira espetacular. Deixou para trás uma grande dívida, algumas sérias dúvidas sobre o papel desempenhado por alguns dos executivos da empresa e resultou em alguns desenvolvedores muito desiludidos, alguns da minha própria equipe.*

*Felizmente, 2001 foi um ano muito bom, minha equipe, agora conhecida como PythonLabs, estava muito entusiasmada, e fomos contratados como uma unidade pela Digital Creations, uma das primeiras empresas a usar o Python. (Ken Manheimer nos precedeu lá alguns anos antes.) A Digital Creations logo renomeou-se Zope Corporation após seu principal produto de código aberto, o sistema de gerenciamento de conteúdo da web Zope. Os fundadores do Zope, Paul Everitt e Rob Page, participaram do primeiro workshop sobre Python no NIST em 1994, assim como seu CTO, Jim Fulton.*

*A história poderia facilmente ter sido muito diferente: além da Digital Creations, também estávamos considerando ofertas do VA Linux e ActiveState. O VA Linux era então uma estrela em ascensão no mercado de ações, mas eventualmente o preço das suas ações (que fizeram de Eric Raymond um multimilionário no papel) entrou em colapso de maneira dramática. Olhando para trás, acho que o ActiveState não teria sido uma má escolha, apesar da personalidade controversa de seu fundador, Dick Hardt, se não estivesse localizado no Canadá.*

*Em 2001, criamos a Python Software Foundation, uma organização sem fins lucrativos, cujos membros iniciais eram os principais desenvolvedores de Python contribuintes na época. Eric Raymond foi um dos membros fundadores. Vou ter que escrever mais sobre esse período outra vez.”*

2.2.3 Funcionalidades do Python

De acordo com Kuhlman, Dave (2012, p 17), o Python é Uma linguagem multi-propósito de alto nível porque:

1. O código é compilado automaticamente para código byte e executado, o Python é adequado para uso como linguagem de script, linguagem de implementação de aplicativo Web, etc.
2. O Python pode ser estendido em C e C ++, e assim, pode fornecer a velocidade necessária para tarefas intensivas de computação.
3. Por causa de suas fortes estruturas de estrutura (blocos de códigos, funções, classes, módulos e pacotes aninhados) e seu uso consistente de objetos e programação orientada a objetos, o Python nos permite escrever aplicativos lógicos e claros para tarefas pequenas e grandes.
4. O python tem Várias características importantes como:

* Tipos de dados integrados de alto nível: strings, listas, dicionários etc.
* As estruturas de controle comum: if, if else, if elif else, while, além de um poderoso iterador de coleção (for).
* Vários níveis de estrutura organizacional: funções, classes, módulos e pacotes. Eles ajudam na organização do código. Um exemplo excelente e grande é a biblioteca padrão do Python.
* Compilar em tempo real para código de bytes O código-fonte é compilado para código de bytes sem uma etapa de compilação separada. Os módulos de código-fonte também podem ser "pré-compilados" para arquivos de código de bytes.
* O Python orientado a objetos fornece uma maneira consistente de usar objetos: tudo é um objeto. E, no Python, é fácil implementar novos tipos de objetos (chamados de classes na programação orientada a objetos).
* Extensões nos módulos de extensão C e C ++ e tipos de extensão podem ser escritos manualmente. Existem também ferramentas que ajudam com isso, por exemplo, SWIG, sip, Pyrex.

1. O Jython é uma versão do Python que é compatível com Java.

Ao analisar como o python foi utilizado neste trabalho, à medida que os códigos forem mencionados haverá uma explicação detalhando a utilização de cada comando que foi escrito.

2.3 WEKA

2.3.1 O que é WEKA?

WEKA significa Waikato Environment for Knowledge Analysis (Ambiente para Análise de Conhecimento de Waikato), foi desenvolvida pela Universidade de Waikato, na Nova Zelândia e é software livre sob a licensa de GNU General Public License.

Segundo Witten, I.H., Frank, E., Trigg, L., Hall, M., Holmes, G. & Cunningham, S.J. (1999, p 2), do departamento de ciência da computação da Universidade de Waikato, *“O Waikato Environment for Knowledge Analysis (Weka) é um conjunto abrangente de bibliotecas de classes Java que implementam muitos algoritmos avançados de aprendizado de máquina e mineração de dados. Weka está disponível gratuitamente na Internet e acompanha um novo texto sobre mineração de dados que documenta e explica completamente todos os algoritmos que ele contém. Os aplicativos gravados usando as bibliotecas de classes Weka podem ser executados em qualquer computador com capacidade de navegação na Web; isso permite que os usuários apliquem técnicas de aprendizado de máquina a seus próprios dados, independentemente da plataforma do computador.“*

De acordo com as palavras de Eibe Frank, Mark A. Hall, and Ian H. Witten (2016, p 7), *“A ferramenta WEKA fornece implementações de algoritmos de aprendizado que você pode aplicar facilmente ao seu conjunto de dados. Também inclui uma variedade de ferramentas para transformar conjuntos de dados, como os algoritmos para discretização e amostragem. Você pode pré-processar um conjunto de dados, alimentá-lo em um esquema de aprendizado, e analisar o classificador resultante e seu desempenho, tudo sem ter escrever absolutamente nenhum código de programa.*

*O ambiente de trabalho inclui métodos para os principais problemas de mineração de dados: regressão, classificação, clustering, mineração de regras de associação e seleção de atributos. Conhecer os dados é uma parte integrante do trabalho, e muitas facilidades de visualização de dados e ferramentas de pré-processamento de dados são fornecidas. Todos os algoritmos recebem sua entrada na forma de uma única tabela relacional que pode ser lida de um arquivo ou gerado por uma consulta ao banco de dados.*

*Uma maneira de usar o WEKA é aplicar um método de aprendizado a um conjunto de dados e analisar sua saída para saber mais sobre os dados. Outra é usar modelos aprendidos para gerar previsões sobre novas instâncias. Um terceiro é aplicar vários algoritmos diferentes e comparar seu desempenho para escolher um para previsão. Na interface interativa do WEKA, você seleciona o método de aprendizado você quer de um menu. Muitos métodos têm parâmetros ajustáveis, que você acessa através de uma página de propriedades ou editor de objetos. Um módulo de avaliação comum é usado para medir o desempenho de todos os classificadores. Implementações de esquemas de aprendizado reais são o recurso mais valioso que o WEKA fornece. Mas as ferramentas para pré-processar os dados, chamados filtros, ficam em segundo lugar. Como classificadores, você seleciona filtros em um menu e os adapta aos seus requisitos.”*

*2.3.2 Como utilizar o WEKA?*

Seguindo as palavras de Eibe Frank, Mark A. Hall, and Ian H. Witten (2016, p 8), *“A maneira mais fácil de usar o WEKA é através de uma interface gráfica do usuário chamada Explorer. Que dá acesso a todas as suas instalações usando a seleção de menus e o preenchimento de formulários. Por exemplo, você pode ler rapidamente um conjunto de dados de um arquivo e criar uma árvore de decisão a partir dele. O Explorer guia você apresentando opções como formulários a serem preenchidos. Dicas úteis sobre ferramentas aparecem quando o mouse passa sobre os itens na tela para explicar o que eles fazem. Valores padrão sensíveis garantem que você possa obter resultados com o mínimo de esforço - mas você terá que pensar no que está fazendo para entender o que os resultados significam.*

*Existem outras três interfaces gráficas de usuário para o WEKA. A interface do fluxo de conhecimento permite criar configurações para processamento de dados em fluxo. Uma desvantagem fundamental do Explorer é que ele mantém tudo na memória principal, quando você abre um conjunto de dados ele carrega tudo imediatamente. Isso significa que ele só pode ser aplicado a pequenas e médias empresas problemas, no entanto, o WEKA contém alguns algoritmos incrementais que podem ser usados ​​para processar conjuntos de dados muito grandes. A interface Fluxo de Conhecimento permite arrastar caixas representando algoritmos de aprendizado e fontes de dados na tela e pode-se junta-los à configuração você quer. Isso permite especificar um fluxo de dados conectando componentes que representam dados fontes, ferramentas de pré-processamento, algoritmos de aprendizado, métodos de avaliação e módulos de visualização. Se os filtros e algoritmos de aprendizado forem capazes de aprendizado incremental, os dados serão carregados e processado incrementalmente.*

*A terceira interface da WEKA, o Experimenter, foi projetada para ajudar você a responder a uma pergunta prática básica ao aplicar técnicas de classificação e regressão: Quais métodos e parâmetros valores funcionam melhor para o problema em questão? Normalmente, não há como responder a essa pergunta a priori, e uma razão pela qual desenvolvemos o ambiente de trabalho foi fornecer um ambiente que permita usuários comparar uma variedade de técnicas de aprendizado. Isso pode ser feito interativamente usando o Explorador. No entanto, o Experimentador permite automatizar o processo, facilitando a execução de classificadores e filtros com diferentes configurações de parâmetros em um conjunto de dados, coletar estatísticas de desempenho e realizar testes de significância. Usuários avançados podem empregar o Experimenter para distribuir a carga de computação em várias máquinas usando a chamada de método remoto Java. Dessa forma, você pode configurar experimentos estatísticos em larga escala e deixá-los em execução.*

*A quarta interface, chamada Workbench, é uma interface gráfica de usuário unificada que combina os outros três (e todos os plugins que o usuário instalou) em um aplicativo. O Workbench é altamente configurável, permitindo ao usuário especificar quais aplicativos e plugins serão exibidos, juntamente com as configurações relacionadas a eles.*

*Por trás dessas interfaces interativas está a funcionalidade básica do WEKA. Isso pode ser acessado na forma bruta, digitando comandos textuais, que dão acesso a todos os recursos do sistema. Ao iniciar o WEKA, você deve escolher entre cinco interfaces de usuário diferentes via o seletor de GUI do WEKA: o Explorer, o Fluxo de Conhecimento, o Experimentador, o Workbench e a linha de comando de interfaces. A maioria das pessoas escolhe o Explorer, pelo menos inicialmente.”*

*2.3.3 O que é possível fazer?*

Seguindo as palavras de Eibe Frank, Mark A. Hall, and Ian H. Witten (2016, p 9), *“Um recurso importante ao trabalhar com o WEKA é a documentação on-line, que foi gerada automaticamente a partir do código-fonte e reflete de forma concisa sua estrutura. Vamos explicar como usar esta documentação. Também identificaremos os principais componentes da WEKA, destacando quais partes contêm métodos de aprendizado supervisionados, quais ferramentas para pré-processamento de dados e quais métodos para outros esquemas de aprendizado. A documentação on-line fornece a única lista completa de algoritmos disponíveis, porque o WEKA está em constante crescimento e a documentação on-line está sempre atualizada. Além disso, torna-se essencial se você deseja avançar para o próximo nível e acessar a biblioteca a partir de seus próprios programas Java ou escrever e testar seus próprios esquemas de aprendizado.*

*Na maioria dos aplicativos de mineração de dados, o componente de aprendizado de máquina é apenas uma pequena parte de um sistema de software muito maior. Se você pretende escrever um aplicativo de mineração de dados, desejará acessar os programas no WEKA de dentro do seu próprio código. Ao fazer isso, você pode resolver o subproblema de aprendizado de máquina do seu aplicativo com um mínimo de programação adicional. Mostramos como fazer isso, apresentando um exemplo de um aplicativo simples de mineração de dados em Java. Isso permitirá que você se familiarize com as estruturas básicas de dados no WEKA, representando instâncias, classificadores e filtros.*

*Se você pretende se tornar um especialista em algoritmos de aprendizado de máquina (ou se você já é um), provavelmente desejará implementar seus próprios algoritmos sem precisar abordar detalhes mundanos como ler os dados de um arquivo, implementar algoritmos de filtragem ou fornecendo código para avaliar os resultados. Se assim for, temos boas notícias para você: WEKA já inclui tudo isso. Para fazer pleno uso, você deve se familiarizar com as estruturas básicas de dados. Para ajudá-lo a chegar a esse ponto, descreveremos essas estruturas em mais detalhes e explicaremos uma implementação ilustrativa de um classificador.”*

Ilustrações, funcionamento e eventualmente, automatização das regressões serão mostradas em seu devido momento neste trabalho.

2.4 METABASE

2.4.1 O que é Metabase?

Conforme a documentação do Metabase na seção do Guia do Usuário *“Metabase é uma ferramenta de inteligência comercial de código aberto. Ele permite que você faça perguntas sobre seus dados e exibe respostas em formatos que façam sentido, seja um gráfico de barras ou uma tabela detalhada.Suas perguntas podem ser salvas para mais tarde, facilitando o retorno a elas, ou você pode agrupá-las em painéis de ótima aparência. A metabase também facilita o compartilhamento de perguntas e painéis com o restante da sua equipe.”*

De modo simples o Metabase é uma ferramenta para entendimento de dados gerados por outra ferramenta ou processo de refinamento de dados dentro de um banco de dados, e para gerar o entendimento dos dados disponibilizados o Metabase gera gráficos, de várias maneiras, e permite a visualização de tabelas originais ou montadas para uma determinada visualização.

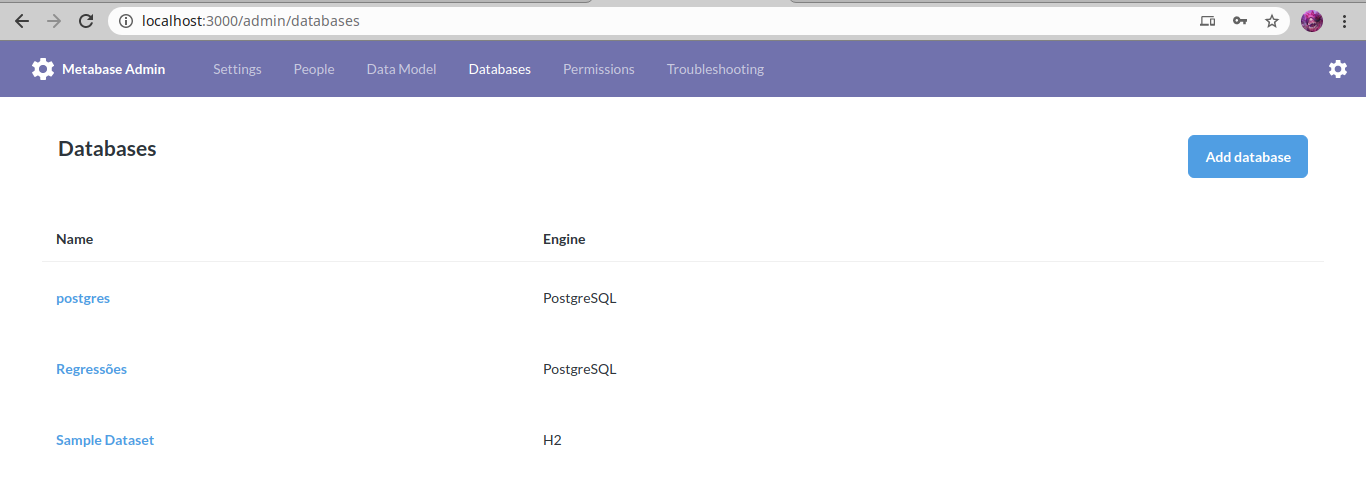
2.4.2 Funcionamento

O Metabase se conecta ao banco de dados e sincroniza com o mesmo, reconhecendo os schemas criados, as tabelas contidas em cada schema, verificando os dados contidos em cada tabela e realizando as importações para o ambiente virtual do Metabase.

2.4.3 Inicializando e configurando o Metabase

O Metabase é uma aplicação escrita em JAVA, portanto a forma mais comum de abrir essa aplicação é por meio da linha de comando de qualquer sistema operacional.

Para inicializar a aplicação, uma das formas é a seguinte: *“java -jar metabase/metabase.jar”*, pois, como a aplicação está escrita em JAVA, deve-se especificar dentro da linha de comando que o Java, “java”, irá executar o arquivo, com extensão jar ,”-jar”, que está numa pasta determinada, “metabase/”, e em seguida definir qual arquivo será executado, “metabase.jar”.

Por padrão o Metabase irá funcionar por meio de um browser, bastando digitar *“*[*http://localhost:3000/*](http://localhost:3000/)*”* para poder acessar o Metabase e configurar-lo.

**Ilustração 1 - Bancos de dados conectados ao Metabase**

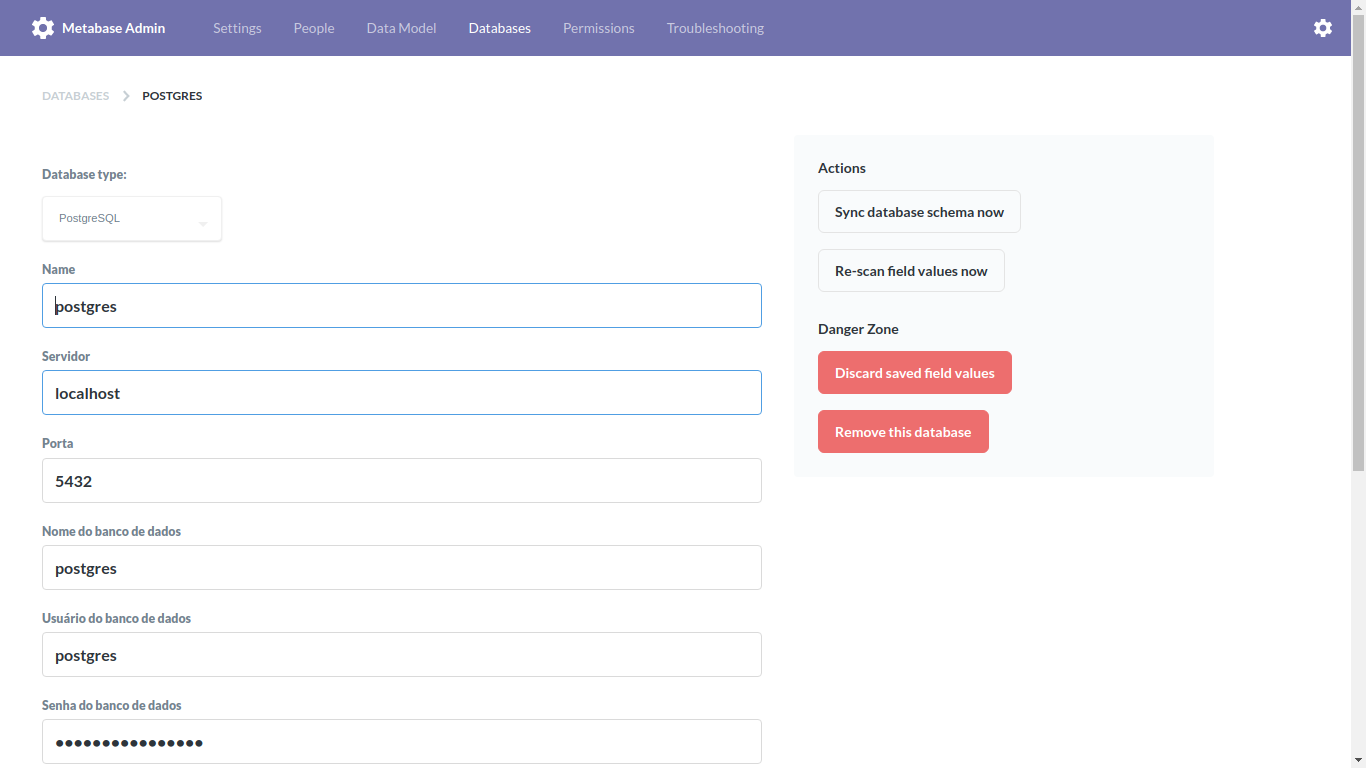
**Fonte: Elaboração própria (2020)**

Como demonstrado pela Ilustração 1, o Metabase pode estar conectado com mais de um banco de dados simultaneamente. Os bancos de dados com nomes “postgres” e “Regressões” são os bancos de dados utilizados para o estudo da previdência social, enquanto o banco de dados “Sample Dataset” é um banco que é disponibilizado pelo metabase para praticar levantamento de visões para entendimento de dados.

O banco de dados “postgres” compreende todo o processo de ETL (Extract Transform and Load) anterior às realizações das regressões para estudo dos dados, onde se encontra todo o processo de tratamento de dados de modo a executar as regressões de maneira otimizada.

Por consequência, o banco de dado “Regressões” abrange todos os procedimentos relacionados à geração das visões posteriores às regressões, é o processo de ETL que está encarregado de transformar os dados gerados em informações por meio de comparações entre os dados originais de cada estado do Brasil com os dados resultantes das regressões realizadas.

Assim a utilização do Metabase é de suma importância devido à sua capacidade de conexão com o banco de dados para elaboração de relatórios de forma rápida e sem gastar recursos, como memória, do computador usuário.

****

**Ilustração 2 - Configurações do Metabase para os bancos de dados**

**Fonte: Elaboração própria (2020)**

Como se pode ver cada banco de dados conectados tem sua própria configuração, como o nome dentro do metabase, o servidor conectado, a porta utilizada para conexão, o nome do banco de dados dentro do servidor, o usuário como administrador do banco de dados e a senha do banco de dados. Adicionalmente há ações que podem ser tomadas em relação ao banco de dados como, sincronizar os schemas do bancos de dados com o metabase (O metabase já faz isso automaticamente quando inicializado, sincronizar o banco de dados manualmente serve para quando houver alguma atualização em relação aos schemas) e escanear os valores das tabelas dentro do banco de dados (novamente, isso serve apenas se os dados forem atualizados)

**3 METODOLOGIA**

3.1 AS CARACTERÍSTICAS DOS DADOS

Como se trata de um estudo econômico, mais especificamente, sobre a previdência social, os dados necessários para realizar as regressões devem ser previdenciários e dados gerais de caráter econômico, então serão utilizados dados como, por exemplo, desemprego, informalidade, condição de habitação, quantidade de aposentados e beneficiários, quantidade de aposentados, renda, entre outros. Para conseguir esse dados foram utilizados os dados abertos da previdência social, disponibilizados no site da previdência social, a PNAD (Pesquisa Nacional por Amostra de Domicílios), obtida a partir do site do IBGE na seção a respeito das estatísticas nacionais de condições de vida, desigualdade e pobreza, as demais tabelas utilizadas foram obtidas no site do Ipeadata.

3.1.1 O problema da quantidade de dados

As informações obtidas para a realização deste estudo foram dados relacionados à arrecadação, quantidade de benefícios emitidos e o valor total dos benefícios entre os anos de 2000 e 2018, no entanto, como veremos na parte de filtragem de dados, a parte da arrecadação não foi utilizada para as regressões, pois não era necessário para estudar o impacto da previdência sobre a economia e para não gastar tempo sem necessidade na produção deste trabalho.

Os dados nas planilhas estão detalhadas em 14 colunas e um total de 5570 linhas, demonstrando o nome de todos os municípios do Brasil, tornando muito difícil a execução de regressões em computadores que não tem processadores razoavelmente bons para executar as tarefas demandadas, por isso é necessário usar um banco de dados, como a série histórica é desde 2000 até 2018, fazer qualquer junção ou implementar qualquer função, quando há limitações de hardware, é uma tarefa quase impossível.

Assim, são as demais planilhas do PNAD, apesar de terem ainda mais colunas, linhas, e mais pormenorizadas. As planilhas estão separadas em três classificações: Características gerais dos Domicílios, do ano 2016 até 2018, com 8 colunas e 18312 linhas na aba de “Condições de Habitação”, 10 colunas e 16854 linhas na aba de “Serviços Básicos” e 8 colunas e 3894 linhas na aba de “Posse de Bens e Serviços”; Características gerais dos Moradores de 2012 até 2018 com 14 colunas e 42126 linhas; e Características adicionais do Mercado de Trabalho de 2012 até 2017 com 13 colunas e 31888 linhas.

Com a situação de quantidade demasiada de dados as únicas saídas seriam reduzir a quantidade de dados que seriam utilizados ou eliminar a necessidade de utilizar ferramentas gráficas para manipular os dados, como já deve ter percebido, foi escolhida a segunda opção e para isso, foi instalado o PostgreSQL.

3.1.2 O primeiro código

Com o banco de dados instalado e funcionando restava apenas uma coisa, colocar os dados dentro dele, para isso o Python foi utilizado, como demonstrado abaixo:

**Ilustração 3 - Script em Python para migração de dados para o PostgreSQL**

**Fonte: Elaboração própria (2020)**

O funcionamento deste código é como descrito a seguir. Inicialmente há a importação das funcionalidade necessárias para o programa, onde “import os” importa a funcionalidade de manipulação de arquivos e diretórios do sistema utilizado, “import pandas as pd” puxa as funcionalidades que são relacionadas à manipulação de planilhas de excel e especifica como essa funcionalidade será chamada “pd”, “from sqlalchemy import create\_engine” pega uma dependencia chamada “create\_engine” de uma funcionalidade chamada “sqlalchemy” que é utilizada para criar a conexão com o banco de dados PostgreSQL.

A linha “engine = create\_engine('postgresql://postgres:senha@localhost:5432/bancodedados')” cria a conexão com o banco de dados e utiliza a função “create\_engine” para isso, onde nessa linha são somente substituídos as palavras “senha” por uma senha que o usuário já tenha definido e “bancodedados” pelo nome do banco de dados que será utilizado para armazenar os dados.

A linha “pasta = '/home/alexandre/Documentos/TCC/Estudo de Estado'” define a pasta onde estão localizados os arquivos a serem enviados para o banco de dados.

A linha “excel\_files\_path = [os.path.join(pasta, nome) for nome in os.listdir(pasta)]” realiza as seguintes operações: cria uma lista de cada arquivo que está dentro da pasta definida no código anterior, cria um loop para ler o nome de cada arquivo e armazenar na variável nome e em seguida juntar (concatenar) o diretório da pasta ao nome do arquivo, para fazer com que os próximos comandos possam abrir os arquivos em excel e realizar transformações neles.

A linha “for excel\_files in excel\_files\_path:” cria um loop que irá verificar quantos arquivos existem em “excel\_files\_path” e irá sequencialmente armazenar cada arquivo utilizando um nome (variável) temporário chamado “excel\_files”.

As linhas “print('')” e “print('Excel files: ' + os.path.basename(excel\_files))” não fazem nada em relação à manipulação dos arquivos em excel, a utilidade dessas duas linhas, bem como qualquer comando que utilizar “print()” neste trabalho servirá apenas como forma de identificação de em qual parte do script alguma coisa deu errado. Nesse caso o comando “print('')” irá criar uma linha em branco no console e “print('Excel files: ' + os.path.basename(excel\_files))” irá escrever “Excel files: o nome do arquivo excel que deu problema”.

A linha “xls = pd.ExcelFile(excel\_files)” armazena os dados do arquivo excel dentro da variável “xls”, esse comando tem a mesma função que o computador utiliza para abrir arquivos excel utilizando a ferramenta excel no ambiente gráfico de qualquer sistema operacional.

A linha “for excel\_sheets in xls.sheet\_names:” cria um loop que irá verificar e abrir cada aba dentro de cada arquivo excel que foi aberto anteriormente e armazenará os dados que estão dentro da aba em uma variável chamada “excel\_sheets”.

A linha “print('Excel sheets: ' + os.path.basename(excel\_sheets))” irá escrever no console “Excel sheets: o nome da aba do arquivo excel”.

A linha “df = pd.read\_excel(xls, excel\_sheets)” dá a ordem à funcionalidade “pandas” mas escrita como “pd” para ler o arquivo em excel (read\_excel) utilizando a extensão “xls” cujo nome é “excel\_sheets” e armazenar os resultados na variável “df” que é uma sigla para “dataframe”.

Os comandos que constituem as variáveis “retirar” e “tabela” foram utilizados para retirar os caracteres que não são reconhecidos pelo PostgreSQL, o comando “os.path.basename(excel\_files)” serve para obter o nome dos arquivos excel, o comando “.lower()” foi utilizado para deixar todos os caracteres em letra minúscula e “replace(‘’,’’)” foi utilizado para substituir acentos, números, sinais de extensão, e espaços em branco.

O comando “print('Tabela: ' + tabela)” foi utilizado para retornar no console o nome da tabela que estava sendo enviada para o banco de dados, ele retorna no console “Tabela: nome da tabela a ser exportada para banco de dados”. Enquanto o comando “print(df)” retorna no console todos os dados armazenados pela variável “df”, este comando torna visível quais são os dados que estão sendo enviados ao PostgreSQL.

O comando “df.to\_sql(tabela, con=engine, if\_exists='replace', index=False)” ordena ao computador que os dados contidos dentro da variável “df” seja exportados para o banco de dados (.to\_sql) utilizando a variável “tabela” como nome para a tabela enviada, utilizando como conector a variável “engine”, como configurações adicionais o comando ainda realiza uma substituição caso haja uma tabela com o mesmo nome dentro do banco de dados “if\_exists='replace'” e não adiciona uma indexação às linhas dentro do banco de dados “index=False”.

Por fim, o comando “print('Conversão terminada')” indica que todo o processo de exportação de dados das planilhas em excel para o PostgreSQL terminou.

O script não levou mais de dois dias para ser desenvolvido e utilizando este primeiro script em python pode-se começar o estudo econômico da previdência social em todos os estados do Brasil, todo o processo de exportação de dados não levou mais de 15 minutos.

3.2 DENTRO DO POSTGRESQL

Assim que todas as tabelas dos arquivos excel foram exportadas para o banco de dados, cada uma delas foi armazenada no schema “public”, sem nenhuma ordenação se não alfabética, ao todo 49 tabelas foram armazenadas com nomes distintos mas conteúdos semelhantes, o primeiro passo seria juntar todas as informações semelhantes de forma concisa, mas para que isso acontecesse foi necessário realizar algumas ações antes.

3.2.1 Correções e visões.

Para que todas as tabelas pudessem ser utilizadas de forma a não comprometer o estudo algumas correções deveriam ser realizadas, na tabela “pnad\_domicilios\_bens\_e\_servicos\_2016\_a\_2018” haviam espaços em branco depois dos nomes de algumas das unidades da federação, para garantir que não houvessem erros derivados dessas pequenas incongruências o seguinte script SQL foi escrito:

****

**Ilustração 4 - Script em SQL para correção de tabelas no PostgreSQL**

**Fonte: Elaboração própria (2020)**

A melhor forma de entender o comando acima é se pensarmos como uma sequência lógica de ações que o banco de dados deverá realizar, onde o banco de dados irá atualizar a tabela “*pnad\_domicilios\_bens\_e\_servicos\_2016\_a\_2018*” que está no schema “public” e em seguida determina como “Acre” (sem espaço) todas as ocorrências da palavra “Acre “ (com espaço) na coluna “Abertura Territorial”. O comando se repete mudando apenas os nomes da Unidade da Federação e para cada tabela derivada do PNAD.

Adicionalmente à incongruência dos nomes na planilha, haviam letras onde deveria haver o número “0”, para corrigir isso, os seguinte comandos foram utilizados, onde define como “0” cada ocorrência das letras “ND” nas colunas dos anos de cada tabela do PNAD.

**

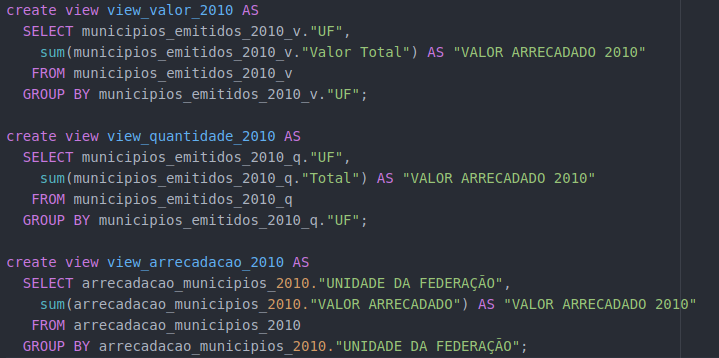
**Ilustração 5 - Script em SQL para correção de tabelas no PostgreSQL**

**Fonte: Elaboração própria (2020)**

Com as correções feitas não resta nada que impeça o prosseguimento do estudo, o próximo passo é criar visões com informações da previdência social dentro do banco de dados para facilitar o uso de informações em processos de tratamento de dados futuros.

As visões foram criadas para conter os valores absolutos de arrecadação, quantidade de benefícios e a soma do valores dos benefícios emitidos nos anos entre 2010 e 2017.

Abaixo segue um exemplo para a criação das visões dos valores da soma dos valores dos benefícios emitidos, arrecadação da previdência e da quantidade de benefícios agrupados por suas respectivas Unidades da Federação:

**

**Ilustração 6 - Script em SQL para criação das visões no PostgreSQL**

**Fonte: Elaboração própria (2020)**

Os comandos acima são compreendidos da seguinte forma; seleciona-se a partir da tabela desejada as Unidades da Federação e o valor total dos valores para aquele ano naquela Unidade da federação (quando não há esse valor na tabela utiliza-se a função “sum” para criar o valor total) e por fim agrupa-se todos os valores segundo a Unidade da Federação para não gerar um resultado desorganizado.

A utilização das visões não era necessário, mas foi utilizado apenas como teste visando saber se a sua utilização é eficiente para ciências de dados, como conclusão, a sua eficácia é limitada, a função de visões é melhor utilizada, por exemplo, para gerar relatórios quando as informações de tabelas são atualizadas de forma frequente, como no caso o estudo é feito a partir de uma série histórica e portanto os dados são fixos a sua utilização não tão eficiente neste estudo.

A função das visões no PostgreSQL é para que um script gerado anteriormente seja automaticamente executado, gerando assim resultados novos quando a tabela original for atualizada, por isso, a sua utilização é mais adequada para gerar gráficos em aplicações desenvolvidas em plataformas online, sendo âmbito de web development.

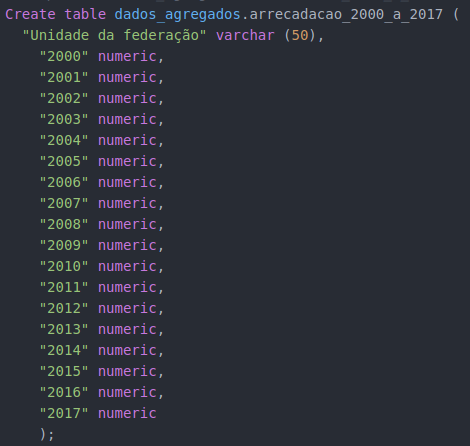
3.2.2 Extrações, Transformações e Cargas (ETL).

Com as correções e as visões criadas, resta agregar as tabelas por tópicos, começando com as tabelas mais importantes, as que detêm as informações a respeito da previdência social. Há ao todo 26 tabelas e 24 visões com as informações necessárias da previdência gerando uma série histórica desde o ano de 2000 até 2017 para o total de arrecadações, benefícios emitidos e valor total dos benefícios.

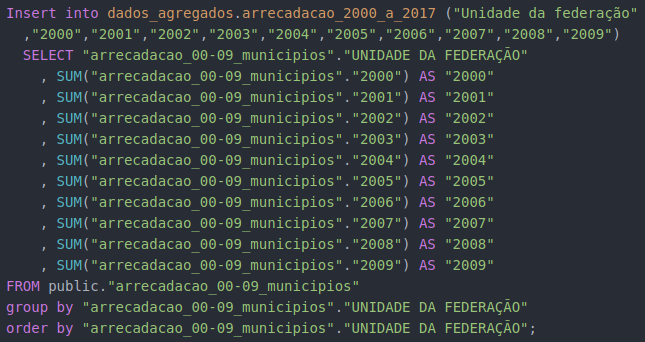
Para construir uma tabela única com as séries históricas de cada Unidade da Federação para as categorias “arrecadação”, “quantidade de benefícios” e “valor dos benefícios” foram utilizados os seguintes scripts:

**Ilustração 7 - Script em SQL para criação de colunas dentro do PostgreSQL**

**Fonte: Elaboração própria (2020)**

O script da ilustração 4 foi utilizado para criar as colunas dentro do schema “dados\_agregados” e da tabela “arrecadação\_2000\_2017” que irão receber os dados das tabelas da previdência social referentes à arrecadação de todas as Unidades da Federação, nos anos de 2000 até 2017.

Para isso foi preciso definir a coluna “Unidade da Federação” como “varchar” pois irá receber os nomes das Unidades da Federação enquanto as colunas dos anos são definidos como “numeric” por receberem dados numéricos.

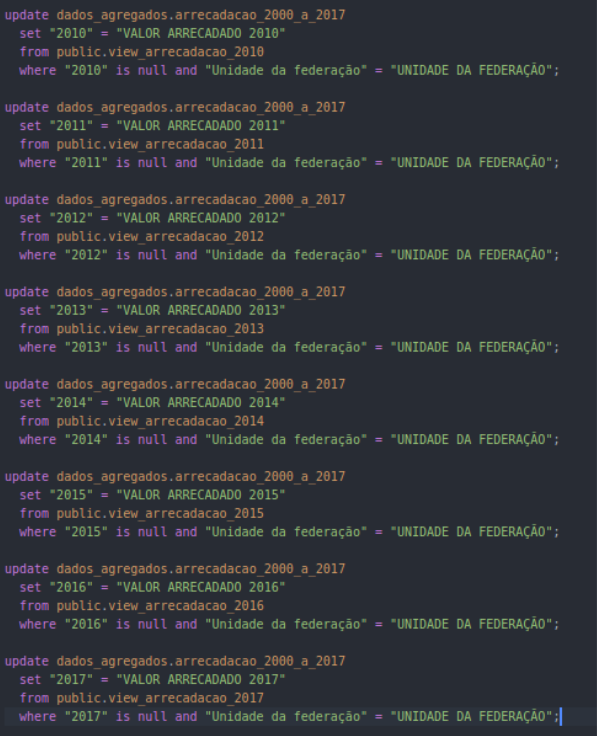
**Ilustração 8 - Script em SQL para inserção de dados em colunas no PostgreSQL**

**Fonte: Elaboração própria (2020)**

O script da ilustração 5 foi utilizado para inserir os dados de arrecadação da tabela “arrecadacao\_00-09\_municipios” localizada no schema “public”. Os dados serão inseridos nas colunas “Unidade da federação”, “2000”, "2001", "2002", "2003", "2004", "2005", "2006", "2007", "2008" e "2009".

A função de inserção de dados precisa essencialmente da função de seleção de dados para executar o comando, e para se obter o valor total da arrecadação foi utilizado a função “sum” que soma todas as ocorrências numéricas dentro de uma coluna, por isso há no final do script as funções “group by” e “order by”, onde o “group by” agrupa os dados que são somados por meio das unidades da federação, assim não há como o valor de uma unidade da federação interferir no valor de outra, e o “order by” ordena alfabeticamente os dados por meio das unidades da federação, o comando “order by” não é necessário para o funcionamento do script.

Com metade da tabela construído ainda falta os anos entre 2010 e 2017, para isso será utilizado o comando “update”.

**Ilustração 9 - Script em SQL para atualização de dados de colunas no PostgreSQL**

**Fonte: Elaboração própria (2020)**

O script atualiza a tabela com os dados do valor de arrecadação a partir da visão criada anteriormente, onde “VALOR ARRECADADO 2010” é o valor total de arrecadação do ano 2010 que irão preencher as células vazias da coluna “2010”.

No script exposto pela ilustração 6 vale notar que para que cada ano seja atualizado deve ser alterado apenas o ano no código, o script em si é longo demais para ser colado neste trabalho, pois há sequências de comando com pequenas alterações. Além disso a estrutura do código se repete para as demais categorias, pois inicialmente foi criada seguindo um padrão, então, em razão disso as tabelas que englobam os dados dos valores do benefícios e da quantidade de benefícios das unidades da federação seguem a mesma estrutura sendo os nomes das tabelas criadas e importadas as únicas alterações.

**DESENVOLVIMENTO**

Parte principal do trabalho, que contém a exposição ordenada e pormenorizada do assunto. É composta de revisão de literatura, dividida em seções e subseções, material e método(s) e/ou metodologia e resultados, agora descritos detalhadamente. Cada seção ou subseção deverá ter um título apropriado ao conteúdo.

Deve-se utilizar sempre a terceira pessoa do singular na elaboração do texto, mantendo-se a forma impessoal com o uso do pronome SE.[[1]](#footnote-0)

**REGRAS GERAIS DE APRESENTAÇÃO**

Constituem-se como padrão para apresentação de trabalhos acadêmicos:

* tipo de papel: papel branco, formato A4 (21 cm x 29,7 cm), para a versão final a ser depositada na Biblioteca da UTFPR, segundo critérios de avaliação. O uso de papel reciclado pode ser aceito nas versões intermediárias;
* impressão: em trabalhos de até 100 páginas, a impressão deverá ser feitas apenas no anverso da folha. Neste caso, recomenda-se utilizar a gramatura 90g para papel branco. Na impressão de trabalhos acima de 101 páginas, deverão ser usados os anversos e versos. Neste caso, recomenda-se utilizar a gramatura 120g para papel branco. Para os elementos pré-textuais deve-se utilizar apenas o anverso da folha, excetuado-se a ficha catalográfica. A partir dos elementos textuais devem ser usados o anverso e o verso, como indicado anteriormente;
* tamanho da fonte: Arial (sem serifa) ou Times New Roman (com serifa), tamanho 12, quando da utilização de fontes proprietárias (sistema Windows). Deve-se utilizar apenas um dos tipos escolhidos em todo o trabalho. Recomenda-se o uso de fonte tamanho 12 para o texto e de tamanho 10 para citações de mais de três linhas, notas de rodapé e legendas das ilustrações e tabelas. Utilizar fontes e tamanhos correspondentes quando do uso de software livre. Em citações com mais de três linhas, deve-se utilizar um recuo de 4 cm da margem esquerda;
* formato do título: o título do trabalho, na capa e na folha de rosto, deve aparecer em CAIXA ALTA, negrito, centralizado e usando fonte Arial ou Times New Roman, tamanho 14. Havendo subtítulo, este deve ser precedido por dois pontos, escrito também em CAIXA ALTA, negrito, centralizado e usando fonte Arial ou Times New Roman, tamanho 14 e sem ponto final;
* parágrafo: deve aparecer com recuo na primeira linha de 1,5 cm, justificado, sem espaçamento anterior ou posterior.

**MARGEM**

* Deve-se usar margens esquerda e superior de 3 cm; e margens direita e inferior de 2 cm;
* na folha de rosto e na folha de aprovação, a natureza do trabalho, o objetivo, o nome da instituição a que é submetida e a área de concentração devem ser alinhados do meio da mancha (espelho/área ocupada pelo texto) para a margem direita, em espaço simples.

**ILUSTRAÇÕES**

São ilustrações: figuras, quadros, gráficos, fotografias, retratos, desenhos, gravuras, imagens, fluxogramas, organogramas, esquemas, mapas, plantas e diferenciam-se das tabelas. As ilustrações devem ser inseridas o mais próximo possível do texto a que se refere.

Qualquer que seja o tipo de ilustração, sua identificação aparece na parte inferior, precedida da palavra designativa, seguida de seu número de ordem de ocorrência no texto, em algarismos arábicos, do respectivo título e/ou legenda. A fonte deve ser tamanho 10 e em negrito. Após a legenda, deve-se citar a fonte de onde foi retirada a ilustração, precedida da palavra “Fonte” (Autor, data), bem como, deve-se referenciá-la, de forma completa, na seção Referências.

\* Para inserir legendas nas ilustrações e tabelas:

1) Na guia **Referências**, no grupo **Legendas**, clique em **Inserir Legenda**.

2) Na lista **Rótulo**, selecione o rótulo que descreva melhor o objeto. Se a lista não contiver o rótulo correto, clique em **Novo Rótulo**, digite o novo rótulo na caixa **Rótulo** e clique em **OK**.

3) Digite o texto, incluindo a pontuação, que você deseja exibir depois do rótulo e clique **Enter** para utilizar a formatação apropriada para a Fonte.

Modelo de formatação de gráfico (Estilo utilizado é “Parágrafo para Ilustrações”):

**Gráfico 1 - Estatística de Empréstimos em Janeiro de 2009**

**Fonte: Biblioteca UTFPR (2010)**

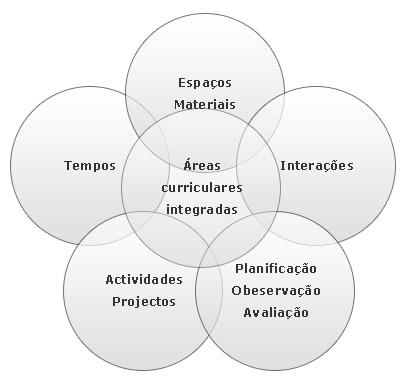
Modelo de formatação de quadro (em geral, trazem apenas informações textuais) e diferenciam-se das tabelas:

|  |  |
| --- | --- |
| **Áreas de Desenvolvimento** | **Descrição** |
| 1. Competências sobre processos | Conhecimento nos processos de trabalho |
| 2. Competências técnicas | Conhecimento técnico nas tarefas a serem desempenhadas e tecnologias empregadas nestas tarefas |
| 3. Competências sobre a organização | Saber organizar os fluxos de trabalho |
| 4. Competências de serviço | Aliar as competências técnicas com o impacto que estas ações terão para o cliente consumidor |
| 5. Competências sociais | Atitudes que sustentam o comportamento do indivíduo: saber comunicar-se e responsabilizar-se pelos seus atos. |

**Quadro 1 - Áreas de Desenvolvimento de Competências**

**Fonte: Zarifian (1999) apud Fleury e Fleury (2004).**

Modelo de formatação de figura (Estilo utilizado é “Parágrafo para Ilustrações”).



**Figura 1 - As dimensões curriculares de pré-escolar**

**Fonte: Centro Social de Azurva (2009)**

Insira o seu texto.

Modelo de formatação de fotografia:



**Fotografia 1 - Entrada da Biblioteca da UTFPR Ponta Grossa**

**Fonte: Autoria própria**

Insira o seu texto.

**TABELAS**

Uma tabela deve apresentar dados numéricos de modo resumido e é utilizada principalmente para a apresentação de comparações. Deve-se seguir tal padrão em todo o trabalho, constando também na lista de tabelas, separada da lista de ilustrações. Os quadros não devem ser chamados de tabelas, uma vez que se diferenciam destas por apresentarem as laterais fechadas e o conteúdo não numérico.[[2]](#footnote-1)

Exemplo de tabela

**Tabela 1 - Desempenho dos alunos na prova de conhecimentos específicos**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Média** | **CEFET** | | **BRASIL** | |
| Curso | concluintes | ingressantes | concluintes | ingressantes |
| Matemática | 27,8 | 22,5 | 27,1 | 22,4 |
| Letras | 32,3 | 31,5 | 30,9 | 26,5 |
| Geografia | 38,4 | 34,2 | 34,6 | 29,5 |
| Ciências Biológicas | 26,4 | 23,6 | 26,6 | 21,9 |

**Fonte: Relatório dos cursos avaliados em 2005 – INEP/SINAES (2006)**

**Exemplo de tabela que ocupa mais de uma folha**

**Tabela 2 - Situação da Educação Brasileira em 2002 – Ensino Médio**

**(continua)**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Taxa de repetência no**  **Ensino Médio (%)** | **Taxa de evasão no Ensino Médio (%)** | **Taxa de analfabetismo da população de 15 a 17** |
| **Sul** | … | … | … |
| Paraná | 19,3 | 8 | 1,4 |
| Rio Grande do Sul | 23,3 | 7,7 | 1,1 |
| Santa Catarina | 20,6 | 9,5 | 1,4 |

**Tabela 2 - Situação da Educação Brasileira em 2002 – Ensino Médio**

**(conclusão)**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Taxa de repetência no**  **Ensino Médio (%)** | **Taxa de evasão no Ensino Médio (%)** | **Taxa de analfabetismo da população de 15 a 17** |
| **Sudeste** | ... | ... | ... |
| Espírito Santo | 17,4 | 5,2 | 2,2 |
| Minas Gerais | 14,2 | 7,2 | 2,1 |
| Rio de Janeiro | 22,4 | 6,5 | 1,3 |
| São Paulo | 11,5 | 7,6 | 0,8 |

**Fontes: IBGE: PNAD 1999 / Contagem populacional 1996; MEC/SEEC/INEP: Censo Educacional 1999.**

1. **CITAÇÕES**

É fundamental nesta etapa a ética e a honestidade intelectual, atribuindo autoria a quem, realmente, contribuiu para o desenvolvimento do estudo em questão. Neste processo usam-se as citações, definidas como “menção de uma informação extraída de outra fonte” (ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS, 2002, p. 1)[[3]](#footnote-2) ou ainda (ABNT, 2002, p. 1).

A transcrição, literal ou não, seguida da referência adequada, torna-se uma citação como classificada por normas. Entretanto, a mesma transcrição ou cópia, sem atender os ditames da norma, torna-se um plágio. O plágio é passível de restrições legais e penais. A Lei nº 9.610, de 19/02/1998, regula os direitos autorais no Brasil, prevendo as sanções legais pertinentes; já o Código Penal, no seu Artigo 184, aponta as sanções penais.

Todos os autores e obras, consultados e efetivamente citados, deverão constar na lista final de referências.

As citações curtas diretas, aquelas com até três linhas, aparecem diretamente no texto, destacadas entre aspas (“...”), precedidas ou sucedidas da indicação de autoria. Exemplo: O autor lembra, contudo, a análise precursora de Leonard-Barton (1998) sobre alguns aspectos limitantes das competências, ou aptidões, essenciais, que as transformam em “limitações estratégicas” (LEONARD-BARTON, 1998, p. 48).

Exemplo de citação direta longa (mais de 3 linhas)

O contexto capacitante não significa necessariamente um espaço físico. Em vez disso, combina aspectos de espaço físico (como o projeto de um escritório ou operações de negócios dispersas), espaço virtual (e-mail, Intranets, teleconferências) e espaço mental (experiências, idéias e emoções compartilhadas). Acima de tudo, trata-se de uma rede de interações, determinada pela solicitude e pela confiança dos participantes. (KROGH; ICHIJO; NONAKA, 2001, p. 66).

1. **CONCLUSÃO**

Parte final do texto, na qual se apresentam as conclusões do trabalho acadêmico, usualmente denominada Considerações Finais. Pode ser usada outra denominação similar que indique a conclusão do trabalho.

**REFERÊNCIAS**

As referências constituem-se de elementos descritivos que permitem a identificação e a localização do documento original[[4]](#footnote-3). O estilo utilizado é: “Formatação das Referências”.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR 6023:** informação e documentação: referências: elaboração. Rio de Janeiro, 2002. (modelo de referência com autoria coletiva)

ANDRADE, M. M. **Competências requeridas pelos gestores de Instituições de ensino superior privadas:** um estudo em Curitiba e região Metropolitana. 2005. 173 f. Dissertação (Mestrado) – Programa de Pós-Graduação em Tecnologia, Universidade Tecnológica Federal do Paraná. Curitiba, 2005. (modelo de referência de dissertação).

BARROS, A. J. S; LEHFELD, N. A. S. **Fundamentos de metodologia:** um guia para a iniciação científica. 2. ed. São Paulo: Makron, 2000. (modelo de referência de livro com subtítulo e edição)

CÓDIGO de Catalogação Anglo-Americano. 2. ed. São Paulo: FEBAB, 1983-1985. (modelo de referência sem autoria - entrada pelo título da obra).

MONTEIRO, S. Breve espaço entre cor e sombra: o romance da maturidade literária de Cristóvão Tezza. **Revista de Letras,** Curitiba (PR), v. 13, n. 11, p. 183-200, dez. 2009. (modelo de referência de artigo de periódico).

RENAUX, D. P. B.; et al. Gestão do conhecimento de um laboratório de pesquisa: uma abordagem prática. In: SIMPÓSIO INTERNACIONAL DE GESTÃO DO CONHECIMENTO. 4., 2001, Curitiba. **Anais...** Curitiba: PUC-PR, 2001. p. 195-208. (modelo de referência de trabalho publicado em evento).

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ. Sistema de Bibliotecas. Normas para elaboração de trabalhos acadêmicos. Curitiba: UTFPR, 2009. Disponível em: <http://www.utfpr.edu.br/documentos/normas\_trabalhos\_utfpr.pdf>. Acesso em: 11 nov. 2011. (modelo de referência de material disponível na versão eletrônica).

1. Questionário de Pesquisa

(esta é a página de apresentação - em seguida, incluir o apêndice)

**CENTRALIZAR O TÍTULO VERTICALMENTE E ALINHAR COM OS DEMAIS ELEMENTOS**

Ministério da Educação

**Universidade Tecnológica Federal do Paraná**

Diretoria de Graduação e Educação Profissional

*Secretaria de Gestão Acadêmica*

Departamento de Biblioteca

**APLICAÇÃO DO QUESTIONÁRIO PARA TRABALHOS ACADÊMICOS**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1. Você tem conhecimento do trabalho que está sendo realizado na UTFPR que criará o padrão da instituição para elaboração de trabalhos acadêmicos? | | | | | | |
|  | **EM** | **G** | **PG** | **P** | **TA** | **TOTAL** |
| Sim |  |  |  |  |  |  |
| Não |  |  |  |  |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 2. Se a resposta da pergunta anterior foi afirmativa, de que maneira tomou conhecimento? | | | | | | |
|  | **EM** | **G** | **PG** | **P** | **TA** | **TOTAL** |
| Pela Internet, na página da instituição |  |  |  |  |  |  |
| Pelo jornal da instituição |  |  |  |  |  |  |
| Por outra maneira |  |  |  |  |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 3. Na realização de trabalhos acadêmicos (relatório, TCC, dissertação, tese, etc.) você costuma consultar normas que norteiam a elaboração dos mesmos? | | | | | | |
|  | **EM** | **G** | **PG** | **P** | **TA** | **TOTAL** |
| Sempre |  |  |  |  |  |  |
| Nunca |  |  |  |  |  |  |
| Às vezes |  |  |  |  |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 4. Se utiliza normas para elaboração de trabalhos acadêmicos, quais costuma consultar? | | | | | | |
|  | **EM** | **G** | **PG** | **P** | **TA** | **TOTAL** |
| ABNT |  |  |  |  |  |  |
| UFPR |  |  |  |  |  |  |
| A que seu orientador passou |  |  |  |  |  |  |
| A elaborada pela biblioteca e professores de nosso Campus |  |  |  |  |  |  |
| De outra instituição |  |  |  |  |  |  |

1. Roteiro da Entrevista

**Roteiro de Entrevista**

1- Identificação Pessoal:

Nome:

D/N:

Nacionalidade:

Sexo:

Idade:

Outras pessoas que moram na casa:

Informante:

2- Encaminhado por:

Motivo da solicitação:

3 - Antecedentes Pessoais:

3.1- Concepção

Quanto tempo após o casamento?

Foi desejada?

Sexo esperado?

Abortos anteriores (espontâneos ou provocados e época)

Observações:

* 1. Direitos autorais - Lei nº 9.610, de 19 de fevereiro de 1998. Disposições preliminares

|  |  |
| --- | --- |
| Brastra.gif (4376 bytes) | **Presidência da República Casa Civil Subchefia para Assuntos Jurídicos** |

**LEI Nº 9.610, DE 19 DE FEVEREIRO DE 1998.**

|  |  |
| --- | --- |
| Mensagem de veto | Altera, atualiza e consolida a legislação sobre direitos autorais e dá outras providências. |

**O PRESIDENTE DA REPÚBLICA** Faço saber que o Congresso Nacional decreta e eu sanciono a seguinte Lei:

Título I

Disposições Preliminares

        Art. 1º Esta Lei regula os direitos autorais, entendendo-se sob esta denominação os direitos de autor e os que lhes são conexos.

        Art. 2º Os estrangeiros domiciliados no exterior gozarão da proteção assegurada nos acordos, convenções e tratados em vigor no Brasil.

        Parágrafo único. Aplica-se o disposto nesta Lei aos nacionais ou pessoas domiciliadas em país que assegure aos brasileiros ou pessoas domiciliadas no Brasil a reciprocidade na proteção aos direitos autorais ou equivalentes.

        Art. 3º Os direitos autorais reputam-se, para os efeitos legais, bens móveis.

        Art. 4º Interpretam-se restritivamente os negócios jurídicos sobre os direitos autorais.

        Art. 5º Para os efeitos desta Lei, considera-se:

        I - publicação - o oferecimento de obra literária, artística ou científica ao conhecimento do público, com o consentimento do autor, ou de qualquer outro titular de direito de autor, por qualquer forma ou processo;

        II - transmissão ou emissão - a difusão de sons ou de sons e imagens, por meio de ondas radioelétricas; sinais de satélite; fio, cabo ou outro condutor; meios óticos ou qualquer outro processo eletromagnético;

        III - retransmissão - a emissão simultânea da transmissão de uma empresa por outra;

        IV - distribuição - a colocação à disposição do público do original ou cópia de obras literárias, artísticas ou científicas, interpretações ou execuções fixadas e fonogramas, mediante a venda, locação ou qualquer outra forma de transferência de propriedade ou posse;

        V - comunicação ao público - ato mediante o qual a obra é colocada ao alcance do público, por qualquer meio ou procedimento e que não consista na distribuição de exemplares;

        VI - reprodução - a cópia de um ou vários exemplares de uma obra literária, artística ou científica ou de um fonograma, de qualquer forma tangível, incluindo qualquer armazenamento permanente ou temporário por meios eletrônicos ou qualquer outro meio de fixação que venha a ser desenvolvido;

        VII - contrafação - a reprodução não autorizada;

        VIII - obra:

        a) em co-autoria - quando é criada em comum, por dois ou mais autores;

        b) anônima - quando não se indica o nome do autor, por sua vontade ou por ser desconhecido;

        c) pseudônima - quando o autor se oculta sob nome suposto;

        d) inédita - a que não haja sido objeto de publicação;

        e) póstuma - a que se publique após a morte do autor;

        f) originária - a criação primígena;

        g) derivada - a que, constituindo criação intelectual nova, resulta da transformação de obra originária;

        h) coletiva - a criada por iniciativa, organização e responsabilidade de uma pessoa física ou jurídica, que a publica sob seu nome ou marca e que é constituída pela participação de diferentes autores, cujas contribuições se fundem numa criação autônoma;

        i) audiovisual - a que resulta da fixação de imagens com ou sem som, que tenha a finalidade de criar, por meio de sua reprodução, a impressão de movimento, independentemente dos processos de sua captação, do suporte usado inicial ou posteriormente para fixá-lo, bem como dos meios utilizados para sua veiculação;

        IX - fonograma - toda fixação de sons de uma execução ou interpretação ou de outros sons, ou de uma representação de sons que não seja uma fixação incluída em uma obra audiovisual;

        X - editor - a pessoa física ou jurídica à qual se atribui o direito exclusivo de reprodução da obra e o dever de divulgá-la, nos limites previstos no contrato de edição;

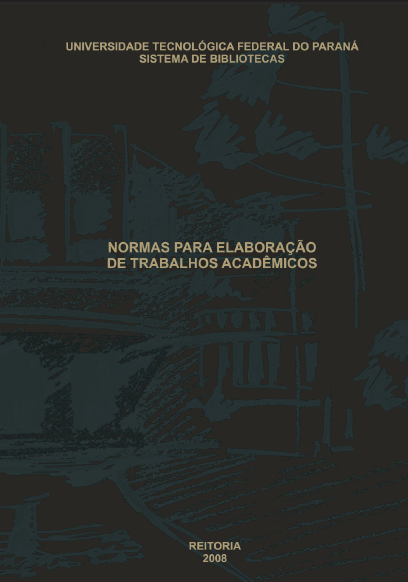
        XI - produtor - a pessoa física ou jurídica que toma a iniciativa e tem a responsabilidade econômica da primeira fixação do fonograma ou da obra audiovisual, qualquer que seja a natureza do suporte utilizado*;*

        XII - radiodifusão - a transmissão sem fio, inclusive por satélites, de sons ou imagens e sons ou das representações desses, para recepção ao público e a transmissão de sinais codificados, quando os meios de decodificação sejam oferecidos ao público pelo organismo de radiodifusão ou com seu consentimento;

        XIII - artistas intérpretes ou executantes - todos os atores, cantores, músicos, bailarinos ou outras pessoas que representem um papel, cantem, recitem, declamem, interpretem ou executem em qualquer forma obras literárias ou artísticas ou expressões do folclore.

        Art. 6º Não serão de domínio da União, dos Estados, do Distrito Federal ou dos Municípios as obras por eles simplesmente subvencionadas.

* 1. Capa do livro: Normas para Elaboração de Trabalhos



**Figura 2 - Capa do livro: Normas para Elaboração de Trabalhos Acadêmicos**

**Fonte: UTFPR (2008)**

1. Para as regras gerais de apresentação das citações, ver página 65 das Normas para Elaboração de Trabalhos Acadêmicos da UTFPR. Dica: para autores entre parênteses, sempre em CAIXA ALTA. [↑](#footnote-ref-0)
2. Para as regras gerais de apresentação das referências, ver página 77 das Normas para Elaboração de Trabalhos Acadêmicos da UTFPR, disponível em: <http://www.utfpr.edu.br/documentos/normas_trabalhos_utfpr.pdf> [↑](#footnote-ref-1)
3. [↑](#footnote-ref-2)
4. [↑](#footnote-ref-3)