

# Relatório

## Projeto de Banco de Dados

*Banco de Dados (INF1383)*

Rio, 13/10/2024

Prof. Sérgio Lifschitz

<b>Enzo Milman</b>	2110959
<b>Filipe Quintans</b>	2020857
<b>Gabriel Valente</b>	2310488
<b>Paulo Monção</b>	2310736
<b>Pedro Barizon</b>	2211350



# 1. Texto-Enunciado do Problema Proposto

Com os intuitos de representar um cenário geral da Educação Superior brasileira e de construir um banco de dados que o descreva, foi elaborado o texto abaixo:

*Na base de dados do setor de Educação Superior, os indivíduos são representados como pessoas, sobre as quais se deve armazenar seu número identificador e seu nome completo. Uma pessoa pode ser um discente ou um docente, não sendo excluída a possibilidade de ser os dois em simultâneo.*

*Sobre um docente é necessário conhecer seu mais alto título acadêmico, o ano em que o obteve, um e-mail de contato profissional, sua área de atuação acadêmica, sua bolsa do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) e, finalmente, sua nacionalidade.*

*Quanto ao discente, é imperioso saber sua raça autodeclarada, o gênero com que se identifica, seus tipos de deficiência (se houver), sua faixa de renda familiar e, enfim, se cursou todo o Ensino Médio (EM) em instituições públicas. Esta última informação é necessária, porque só poderão concorrer a vagas por cotas os alunos que tenham cursado o EM integralmente em instituições públicas, conforme se observa no edital deste ano da UFRJ (UNIVERSIDADE FEDERAL DO RIO DE JANEIRO, 2024).*

*Há também as instituições de ensino superior (IES). Sobre uma IES é preciso guardar seu identificador no sistema e-MEC (BRASIL, 2024c), seu nome, sua sigla (se houver), o Cadastro Nacional de Pessoa Jurídica (CNPJ) de sua mantenedora, um e-mail de contato, seu estado de atividade (ativa, inativa ou em extinção), seu tipo de organização acadêmica, sua categoria administrativa, seu credenciamento pelo Ministério da Educação (MEC) e seu Conceito Institucional (CI) mais recente. Vale mencionar que uma IES pode ter um ou mais campi. A respeito desses locais, é devido conhecer a unidade federativa (UF) e o município em que se encontram.*

*Além disso, existem as categorias de cursos superiores (como Direito, Engenharia Mecânica, dentre outros), que, para evitar discrepâncias, deverão ser armazenadas independentemente das IESs, segundo descrição da Classificação Internacional Normalizada da Educação (CINE) (BRASIL, 2019). Assim, para cada categoria de curso, será necessário manter seu Rótulo e sua Área Geral.*

*As IESs ministram um ou mais cursos, cujo código identificador do e-MEC está atrelado à IES e ao único campus onde é ministrado. Cada curso pertence a uma única categoria de curso. Além dessas informações, será útil conservar o nome que o curso recebe na IES, sua situação de atividade (ativo, inativo ou em extinção), a modalidade em que é ministrado (presencial, remoto ou híbrido), o grau conferido aos concluintes (bacharel, licenciado ou tecnólogo), sua quantidade de vagas ofertadas anualmente e sua última nota no Exame Nacional de Desempenho dos Estudantes (ENADE).*

*Voltando aos docentes, estes são funcionários de uma ou mais IESs atualmente. Sobre isso, é necessário guardar o cargo ocupado. Por conta do artigo 37, inciso XVI, da Constituição Federal de 1988 (BRASIL, 1988), um*

*docente pode lecionar em, no máximo, duas IESs públicas simultaneamente. Ademais, os docentes podem lecionar vários cursos.*

*Volvendo aos discentes, estes podem candidatar-se a vários cursos ministrados nas IESs. Sobre uma candidatura é requerido armazenar a modalidade de ingresso (Ampla Concorrência, Cota Racial, por Renda ou por Deficiência), o vestibular pelo qual se concorreu, o período letivo no qual se planeja ingressar, o tipo de bolsa atrelada à vaga (se houver) e, finalmente, o resultado da candidatura, isto é, se o aluno foi matriculado, desistiu da vaga, foi negado ou, enfim, se ainda está à espera do resultado. Vale ressaltar que, por conta da Lei 12.089, de 2009 (BRASIL, 2009), um discente não pode ocupar simultaneamente duas vagas de curso de graduação vindas de IESs públicas.*

*Ademais, evidentemente, os candidatos devem satisfazer as condições previstas nas cotas para adotarem essa modalidade de ingresso. Seja como for, depois de matriculado, o discente poderá trancar sua matrícula, devendo o período de trancamento ser persistido na base de dados. Por fim, o discente poderá sair de um curso por jubramento, por abandono, por troca ou, evidentemente, por conclusão, devendo essa informação ser preservada juntamente com o período letivo em que ocorreu.*

A seguir, será apresentado o diagrama ER correspondente a essa descrição.

## 2. Diagrama Entidade-Relacionamento (ER)

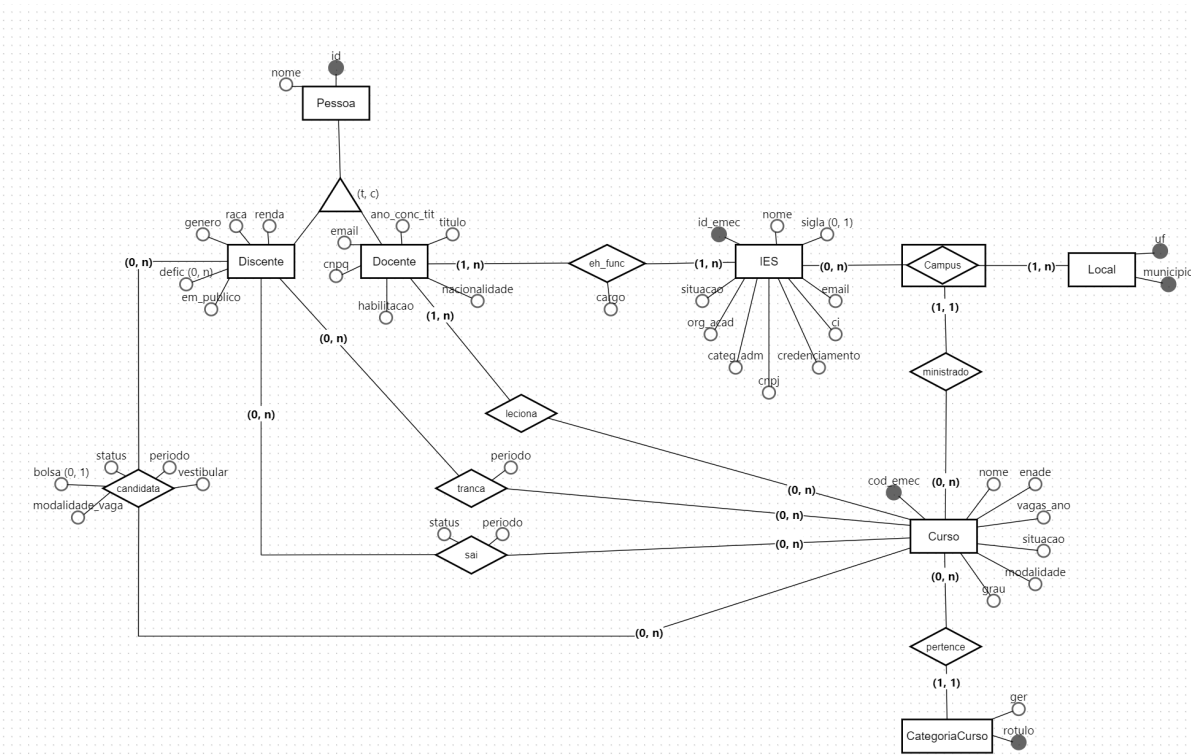


Figura 1 Diagrama ER. Disponível em: <https://app.brmodeloweb.com/#!/publicview/6713aa5ac4c271927df1f448>

O *link* de acesso ao modelo na página do BRmodelo, ferramenta usada na construção do diagrama, encontra-se na descrição da imagem, caso haja dificuldades em sua leitura. De agora em diante, será realizado o detalhamento do diagrama. Destina-se um tópico para cada entidade, dentro do qual serão abertos outros, destinados à especificação de cada atributo separadamente. Ao descrever um atributo, preenche-se em detalhes uma lista enumerada de 1 a 3, cujos índices referem-se, respectivamente, à sua descrição, ao seu domínio de existência e às fontes das quais se extraem tais informações; deve-se ressaltar que o terceiro item é opcional. No caso de a cardinalidade do atributo ser diferente de (1, 1), ela é explicitamente colocada à sua direita. Evidentemente, todo domínio — exceto identificadores, futuras chaves primárias — contém o *placeholder* NULL. Por isso, para evitar redundância, não será explicitado. Por fim, os identificadores são antecidos pelo prefixo marcador << id >>, para que possam ser devidamente apontados, além de as generalizações serem devidamente explicitadas colocando-se, ao lado das entidades especializadas, uma indicação à entidade genérica de que provêm.

## 2.1. Entidades

### 2.1.1. Pessoa

#### << id >> id

1. Identificador da pessoa. Se for docente, usaremos preferencialmente o identificador da plataforma Lattes (IDLattes); se apenas discente, usaremos seu Cadastro de Pessoa Física (CPF).
2. Numérico. Se for IDLattes, são 16 dígitos numéricos; se CPF, 11 dígitos numéricos.
3. IDLattes é extraível da plataforma Lattes. CPF é informação sensível segundo LGPD. Por isso, usaremos dados fictícios neste caso.

#### nome

1. Nome próprio da pessoa e, ao menos, um sobrenome.
2. Alfabético.

### 2.1.2. Docente (*estende* Pessoa)

Os atributos de *Pessoa* herdados por *Docente* não serão novamente descritos. Todos os atributos são coletáveis da plataforma Lattes.

#### título

1. Mais alto título acadêmico obtido pelo docente. Apesar de formalmente não o ser, consideraremos *Pós-Doutor* como um título.

2. Alfabético. Compreende:
- Ensino fundamental incompleto (EFI);
  - Ensino fundamental completo (EFC);
  - Ensino médio completo (EMC);
  - Ensino superior completo (ESC);
  - Mestre (MES);
  - Doutor (DOC); e
  - Pós-Doutor (PHD).

### **ano\_conc\_tit**

1. Ano de conclusão da última titulação. Tentativa de escapar da idade, informação sensível e de mais difícil obtenção.
2. Apenas ano.

### **email**

1. *E-mail* de contato profissional do docente.
2. Alfanumérico com caractere '@' indicador de domínio do *e-mail*.

### **habilitacao**

1. Área de atuação do docente. Exemplos: "Física", "Linguística" ou "Computação".
2. Alfabético.

### **cnpq**

1. Tipo de bolsa dada pelo CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico) ao docente. Indiretamente, indica a relevância acadêmica do docente.

2. Alfanumérica. Compreende:

- a. 1A;
- b. 1B;
- c. 1C;
- d. 1D;
- e. 2;
- f. NA;
- g. SR.

## **nacionalidade**

- 1. País de origem do docente.
- 2. Alfabético.

### 2.1.3. Discente (*estende Pessoa*)

Os atributos de *Pessoa* herdados por *Discente* não serão novamente descritos. Ademais, uma vez que os atributos envolvem informações sensíveis, eles serão futuramente preenchidos com dados fictícios.

## **genero**

- 1. Gênero com que se identifica o discente.
- 2. Alfabético. Compreende:
  - a. masculino (M);
  - b. feminino (F);
  - c. não binário (N);
  - d. outro (O).

## **raca**

1. Raça/cor do docente.
2. Alfabético. Compreende o padrão do IBGE (2024b):
  - a. branca;
  - b. preta;
  - c. parda;
  - d. amarela;
  - e. indígena;
  - f. outra.

## **defic (0, N)**

1. Tipos de deficiência que o discente possui.
2. Alfabético. Compreende (por simplicidade):
  - a. auditiva;
  - b. intelectual;
  - c. motora ou física;
  - d. visual.

## **em\_publico**

1. Indica se o discente cursou o ensino médio integralmente em instituições públicas.
2. Booleano. Se verdadeiro, indica que o discente cursou integralmente.

## **renda**

1. Faixa de renda a que pertence a família do discente.
2. Alfabético. Compreende:



- a. A (mais de 4 salários mínimos *per capita*);
- b. B (até 4 salários mínimos *per capita*);
- c. C (até 3 salários mínimos *per capita*);
- d. D (até 2 salários mínimos *per capita*);
- e. E (até 1 salário mínimo *per capita*).

#### 2.1.4. IES

IES é sigla para Instituição de Ensino Superior. Todos os dados são extraíveis do sistema e-MEC (BRASIL, 2024c).

##### << id >> id\_emec

- 1. Identificador adotado pelo sistema e-MEC.
- 2. Numérico.

##### nome

- 1. Nome por extenso da IES.
- 2. Alfanumérico.

##### sigla (0, 1)

- 1. Se houver, sigla adotada pela IES, como “UFRJ” para se referir à Universidade Federal do Rio de Janeiro.
- 2. Alfanumérica.

## **situacao**

1. Situação em que se encontra a IES.
2. Alfabética. Compreende:
  - a. ativa;
  - b. em extinção;
  - c. extinta.

## **org\_acad**

1. Tipo de organização acadêmica. Por exemplo, a PUC-Rio é uma *Universidade*.
2. Alfabética.

## **categ\_adm**

1. Categoria administrativa da IES. Indica, por exemplo, se é pública ou privada, com ou sem fins lucrativos.
2. Alfabética.

## **credenciamento**

1. Tipo de credenciamento que possui a IES perante o MEC. Indica se pode ministrar cursos presencialmente ou (inclusive) em regime remoto.
2. Alfabético.

## **ci**

1. *Conceito Institucional*. Atribuído pelo MEC, visa avaliar a qualidade de uma IES (DESAFIOS DA EDUCAÇÃO, 2021).
2. Numérico inteiro limitado entre 1 e 5.

## **email**

1. *E-mail* de contato da IES.
2. Alfanumérico com caractere '@' indicador de domínio do *e-mail*.

## **cnpj**

1. Cadastro Nacional de Pessoa Jurídica (CNPJ) da mantenedora da IES.
2. Numérico da forma EE.EEE.EEE/UUU-VV, em que todas as letras representam dígitos. 'E' indica a empresa; U, a unidade da empresa em questão; e 'V', um dígito verificador.

### **2.1.5. Local**

Será utilizado o cadastro de cidades e estados do IBGE (2024a).

## **<< id >> uf**

1. Unidade federativa em que se encontra o local.
2. Alfabético. Siglas de duas letras dos 26 estados e do distrito federal ("DF"), com a inclusão de "Remoto", para poder relacionar-se com os cursos totalmente remotos.

## << id >> municipio

1. Município do Brasil em que se encontra o local.
2. Alfabético, com a inclusão de “Remoto”, para poder relacionar-se com os cursos totalmente remotos.

### 2.1.6. Categoria Curso

Será utilizado o padrão de codificação da *Classificação Internacional Normalizada da Educação* (CINE), padrão desenvolvido pelo INEP a partir do *International Standard Classification of Education* (Isced), da UNESCO (BRASIL, 2019). A ideia é que cursos oferecidos em IESs diferentes podem, apesar de ligeira diferença de nome, ser equivalentes, como quem diz “Faço Engenharia **da** Computação na UFRJ”, e um outro responde “É mesmo? Faço Engenharia **de** Computação na PUC”.

## << id >> rotulo

1. Nome do curso extraído do manual da CINE. Não considera o prefixo serial NNNNLNN usado no manual, em que N é numérico, e L é uma letra (inicial do nome do curso).
2. Alfabético.

## ger

1. Área geral a que pertence o curso segundo a CINE.
2. Alfabético. Compreende:
  - a. Programas básicos;
  - b. Educação;
  - c. Artes e humanidades;
  - d. Ciências Sociais, comunicação e informação;
  - e. Negócios, administração e direito;

- f. Ciências naturais, matemática e estatística;
- g. Computação e Tecnologias da Informação e Comunicação (TIC);
- h. Engenharia, produção e construção;
- i. Agricultura, silvicultura, pesca e veterinária;
- j. Saúde e bem-estar;
- k. Serviços.

#### 2.1.7. Curso

##### **<< id >> cod\_emec**

1. Código de cadastro do curso ministrado em uma IES na plataforma governamental e-MEC.
2. Numérico.

##### **nome**

1. Nome do curso em sua respectiva IES (não necessariamente igual ao rótulo do CINE Brasil).
2. Alfabético.

##### **situacao**

1. Situação em que se encontra o curso.
2. Alfabético. Compreende:
  - a. ativo;
  - b. em extinção;
  - c. extinto.

## **modalidade**

1. Modo como é ministrado o curso em termos do comparecimento às aulas.
2. Alfabético. Compreende:
  - a. presencial;
  - b. híbrido;
  - c. remoto.

## **grau**

1. Tipo de diploma que será emitido ao término do curso.
2. Alfabético. Compreende:
  - a. bacharelado;
  - b. licenciatura;
  - c. tecnólogo.

## **vagas\_ano**

1. Número de novas vagas anuais ofertadas.
2. Numérico inteiro maior ou igual a 0.

## **enade**

1. Nota seguida do ano em que foi computada obtida no Exame Nacional de Desempenho dos Estudantes (ENADE).
2. Numérico inteiro limitado entre 1 e 5 com apenas ano, isto é, N(AAAA), em que  $1 \leq N \leq 5$ , e AAAA é um ano.

## 2.2. Entidades Associativas

### 2.2.1. (0, N) IES [possui] *Campus* em Local (1, N)

Diz respeito à associação entre a IES e o local onde reside algum de seus *campi*. As cardinalidades adotadas foram pensadas sob a ótica de que, mesmo que uma IES seja totalmente remota, deverá se associar a pelo menos o local “Remoto”.

## 2.3. Relacionamentos

De modo parecido ao das entidades, será feito um detalhamento sobre os relacionamentos existentes no diagrama, envolvendo o mesmo padrão adotado para descrição de atributos, além de serem exibidas as entidades envolvidas.

### 2.3.1. (1, N) Docente *eh\_func* [na] IES (1, N)

Esse relacionamento está representando o ato de um docente ser funcionário — para simplificar o modelo, optou-se por representar apenas cargos de professor — de uma universidade. Nota-se que todo docente deve possuir, ao menos, algum cargo na universidade, caso contrário não seria possível exercer tal função, além de que não há universidade sem algum docente. Tais considerações vieram à tona para justificar as cardinalidades de (1, N) para (1, N) adotadas. Por fim, esses dados são obtíveis da plataforma Lattes.

#### **cargo**

1. Cargo ocupado na respectiva universidade.
2. Alfabético. Compreende:
  - a. Professor Substituto (SUBS);
  - b. Professor Adjunto A (ADJ\_A);
  - c. Professor Associado (ASSOC);
  - d. Professor Assistente (ASSIST);
  - e. Professor Adjunto (ADJ);
  - f. Professor Titular (TIT);
  - g. Outro (OUTRO)

### 2.3.2. (0, N) Discente *candidata*[-se] [ao] Curso (0, N)

Todos os dados serão fictícios neste caso. Registra a tentativa do candidato de ingressar em um curso superior, o que engloba o estado atual (resultado) desse processo, o meio de ingresso por vestibular, eventuais bolsas de estudo ou cotas. A generalidade da situação levou às cardinalidades escolhidas serem do tipo (0, N), deixando claro que pode haver discente que ainda não se candidatou a curso algum, como também cursos sem candidatos.

#### **status**

1. Estado resultante da candidatura do discente.
2. Alfabético. Compreende:
  - a. matrícula (discente matriculou-se com sucesso);
  - b. reprovação (discente não foi aprovado);
  - c. espera: (o resultado ainda não foi definido);
  - d. desistência: (discente desistiu de pleitear a vaga).

#### **vestibular**

1. Tipo de vestibular usado na candidatura.
2. Alfabético. Compreende:
  - a. ENEM.
  - b. próprio (isto é, vestibular da própria instituição à qual se realiza a candidatura).

#### **bolsa (0, 1)**

1. Bolsa a que se encontra atrelada a candidatura.
2. Alfabético. Compreende:
  - a. ProUni.



- b. FIES;
- c. Institucional (isto é, da própria instituição à qual se realiza a candidatura).

### **modalidade\_vaga**

1. Modalidade da vaga a que se candidatou o docente no que diz respeito ao uso de cotas.
2. Alfabético. Compreende:
  - a. Ampla concorrência (AC);
  - b. Pessoas com deficiência (PD);
  - c. Racial (RA);
  - d. Renda (RE).

### **periodo**

1. Ano e semestre de início das aulas do curso ministrado a que o discente se candidatou.
2. Numérico da forma AAAA.S, em que AAAA é o ano, e S ora é 1 (primeiro semestre), ora é 2 (segundo semestre).

#### **2.3.3. (0, N) Discente *tranca* Curso (0, N)**

Relacionamento simplesmente representando o ato de trancamento de um curso por parte do estudante. Como não há qualquer exigência para a concretização desse relacionamento entre duas entidades, foi escolhida a cardinalidade (0, N).

### **periodo**

1. Ano e semestre em que o discente trancou sua matrícula.
2. Numérico da forma AAAA.S, em que AAAA é o ano, e S ora é 1 (primeiro semestre), ora é 2 (segundo semestre).

#### 2.3.4. (0, N) Discente *sai* [do curso] Ministrado (0, N)

Encerramento de qualquer vínculo entre o estudante e um curso, seja por conclusão, abandono, jubramento ou migração para outra formação acadêmica. Novamente, as cardinalidades são genéricas, pois não há qualquer obrigatoriedade imposta nesse relacionamento.

##### **periodo**

1. Ano e semestre em que o discente saiu do curso ministrado.
2. Numérico da forma AAAA.S, em que AAAA é o ano, e S ora é 1 (primeiro semestre), ora é 2 (segundo semestre).

##### **status**

1. Estado em que o discente saiu do curso.
2. Alfabético. Compreende:
  - 2.1. conclusão;
  - 2.2. abandono;
  - 2.3. jubramento;
  - 2.4. troca.

#### 2.3.5. (1, N) Docente *leciona* Curso (0, N)

Como não há atributos, justifica-se apenas as cardinalidades porque docentes cadastrados podem ou não lecionar vários cursos, os quais devem ser ministrados por, no mínimo, um docente.

#### 2.3.6. (0, N) Curso *ministrado* [em] Campus (1, 1)

Nesse caso, admite-se a interpretação de que um *campus* pode não ministrar curso algum — caso tenha acabado de ser inaugurado, por exemplo. Por outro lado, todo curso é ministrado em um único *campus*, nem que este seja remoto.

### 2.3.7. (0, N) Curso *pertence* CategoriaCurso (1, 1)

Estabelece uma relação de uma categoria de curso para muitos cursos. Por outro lado, um curso *pertence* a uma única categoria.

## 2.4. Observações

Nesta subseção, justificaremos as alternativas de modelagem escolhidas. Em primeiro lugar, escolheu-se a generalização de Pessoa em Docente e em Discente, porque, além de Docente e Discente serem ocupações não mutuamente excludentes, se separados, ambos teriam os atributos **id** e **nome** em comum. Logo, definindo-se a entidade Pessoa como aquela que possui **id** e **nome**, pôde-se generalizar perfeitamente ambas as entidades, evitando-se a redundância no caso de indivíduos que ocupem as duas posições simultaneamente.

Em segundo, optamos por definir *Campus* como entidade associativa entre Lugar e IES, para que pudéssemos traçar diretamente a relação entre *Campus* e Curso. Do contrário, se tivéssemos optado pela ternária Local-IES-Curso, não seria estruturalmente necessário que Local fosse um *Campus* da IES, o que tornaria esta alternativa menos precisa que a escolhida.

Em terceiro, escolhemos incluir a entidade CategoriaCurso, haja vista que gostaríamos de poder agrupar cursos equivalentes mas ministrados em IESs diferentes sob um mesmo nome. Por exemplo, gostaríamos de, ao dizermos “Direito”, englobar tanto o “Direito da PUC-Rio” quanto o “Direito da UERJ”, quanto tantos outros. Isso seria difícil se não existisse uma entidade que se propusesse a representar a ideia abstrata “curso de Direito”. Por isso, separando em parte abstrata (CategoriaCurso) e parte concreta (Curso), podemos melhor navegar pelos dados.

### 3. Regras de Negócio

Para simplificarmos o modelo, como vimos, haverá quatro modalidades de vagas:

1. AC (ampla concorrência);
2. PD (pessoas com deficiência);
3. RA (racial, que engloba pretos, pardos e indígenas);
4. RE (renda).

Diante disso, de pesquisas e do texto, foram identificadas estas regras de negócio:

- *Um discente não pode ocupar simultaneamente duas vagas de curso de graduação vindas de instituições de ensino superior (IES) públicas (BRASIL, 2009).*
- *Um docente pode lecionar simultaneamente em no máximo duas instituições de ensino superior (IES) públicas (BRASIL, 1988).*
- *Se a instituição de ensino superior for privada, toda vaga ofertada deverá ter modalidade AC.*
- *Para se inscrever em qualquer das modalidades diferentes de AC, um discente deve ter feito seu ensino médio integralmente em instituições de ensino públicas (aferível por meio do atributo “**em\_publico**”).*
- *Para se inscrever via PD, um discente deve possuir ao menos um tipo de deficiência (aferível por meio do atributo multivalorado “**defic**”).*
- *Para se inscrever via RA um discente deve se autodeclarar preto, pardo ou indígena (aferível por meio do atributo “**raca**”).*
- *Para se inscrever via RE, um discente deve ter renda per capita familiar inferior a 1 (um) salário mínimo (aferível por meio do atributo “renda”).*

Todas as considerações basearam-se no modelo de cotas da UFRJ (UNIVERSIDADE FEDERAL DO RIO DE JANEIRO, 2024).

## 4. Esquema Lógico-Relacional

Em princípio, deve-se compreender a definição do diagrama abordado nas seções anteriores e modelado sob o viés do Modelo de Entidades e Relacionamentos (MER), com o auxílio da ferramenta BRmodelo, como a consolidação do esquema conceitual do projeto. Subsequente a essa etapa do processo de concepção de um banco de dados, é extraído o esquema lógico-relacional a partir do ER previamente modelado, bastando apenas aplicar regras de transformação de entidades e relacionamentos para a construção de tabelas. Sendo assim, encontram-se ilustrados abaixo o modelo lógico relativo à modelagem conceitual previamente proposta e, posteriormente, esclarecimentos a respeito das transformações realizadas. Adotou-se como notação que chaves primárias estão sublinhadas e que **FK R(A<sub>1</sub>, A<sub>2</sub>, ..., A<sub>n</sub>) referencia S(B<sub>1</sub>, B<sub>2</sub>, ..., B<sub>n</sub>)** simboliza que os atributos **A<sub>1</sub>, A<sub>2</sub>, ..., A<sub>n</sub>** de **R** são uma chave estrangeira que referencia, respectivamente, os atributos **B<sub>1</sub>, B<sub>2</sub>, ..., B<sub>n</sub>** da relação **S**.

### 4.1. Tabelas

4.1.1. Pessoa(id, nome)

4.1.2. Discente(id\_pessoa, genero, raca, renda, em\_publico)

FK Discente(id\_pessoa) referencia Pessoa(id)

4.1.3. Deficiencia(id\_discente, defic)

FK Deficiencia(id\_discente) referencia Discente(id\_pessoa)

4.1.4. Docente(id\_pessoa, titulo, ano\_conclusao, email, cnpq, habilitacao, nacionalidade)

FK Docente(id\_pessoa) referencia Pessoa(id)

4.1.5. eh\_func(id\_docente, id\_ies, cargo)

FK eh\_func(id\_docente) referencia Docente(id\_pessoa)

FK eh\_func(id\_ies) referencia IES(id\_emec)

- 4.1.6. IES(id\_emec, nome, sigla, email, ci, credenciamento, cnpj, categ\_adm, org\_acad, situacao)
- 4.1.7. Local(uf, municipio)
- 4.1.8. Campus(id\_ies, uf\_local, municipio\_local)  
FK Campus(id\_ies) referencia IES(id\_emec)  
FK Campus(uf\_local, municipio\_local) referencia Local(uf, municipio)
- 4.1.9. Categoria\_Curso(ger, rotulo)
- 4.1.10. Curso(vagas\_ano, enade, nome, cod\_emec, modalidade, grau, situacao, rotulo\_categoria\_curso, uf\_campus, municipio\_campus, id\_ies\_campus)  
FK Curso(id\_ies\_campus, uf\_campus, municipio\_campus) referencia Campus(id\_ies, uf\_local, municipio\_local)  
FK Curso(rotulo\_categoria\_curso) referencia Categoria\_Curso(rotulo)
- 4.1.11. candidata(id\_discente, cod\_curso, vest\_status, periodo, bolsa, vestibular, modalidade\_vaga)  
FK candidata(id\_discente) referencia Discente(id\_pessoa)  
FK candidata(cod\_curso) referencia Curso(cod\_emec)
- 4.1.12. sai(id\_discente, cod\_curso, periodo, disc\_status)  
FK sai(id\_discente) referencia Discente(id\_pessoa)  
FK sai(cod\_curso) referencia Curso(cod\_emec)
- 4.1.13. tranca(id\_discente, cod\_curso, periodo)  
FK tranca(id\_discente) referencia Discente(id\_pessoa)

FK tranca(cod\_curso) referencia Curso(cod\_emec)

#### 4.1.14. leciona(id\_docente, cod\_curso)

FK leciona(id\_docente) referencia Docente(id\_pessoa)

FK leciona(cod\_curso) referencia Curso(cod\_emec)

## 4.2. Esclarecimentos

Em primeiro lugar, todas as colunas das tabelas criadas são provenientes de algum atributo previamente definido no diagrama entidade-relacionamento da modelagem conceitual. Logo, por meio de simples senso de intuição, pode-se mapear as colunas aos atributos respectivos do DER. Em segundo lugar, por mais que seja inerente às suas definições, há de se lembrar que o esquema deve obedecer às restrições de integridade, por exemplo a referencial, na qual toda FK ou recebe *NULL* ou referencia a PK de outra tabela, e da restrição de integridade de entidade, na qual PKs não admitem *NULL* em nenhum de seus valores. Outro ponto relevante a ser levado em consideração pauta-se no embasamento utilizado para realizar as transformações utilizadas. Seguem abaixo listados os critérios postos em prática:

- 1) Em geral, entidades viram tabelas próprias, sendo as colunas os próprios atributos e os identificadores transformados em PKs.
- 2) Para a generalização, criou-se uma tabela para cada entidade que compõe a hierarquia, incluindo-se a chave primária da tabela correspondente à entidade genérica em cada tabela correspondente a uma entidade especializada.
- 3) Tratamento de atributo multivalorado foi feito mediante criação de uma nova tabela para o atributo, inserindo-se uma FK que o ligue à tabela correspondente à entidade a que pertencia no ER.
- 4) Relacionamentos foram abordados por meio da elaboração de novas tabelas próprias, da fusão de tabelas ou da adição de colunas. A dúvida de qual alternativa escolher foi sanada segundo as recomendações de boas práticas incluídas no livro do professor Carlos Heuser (1999), intitulado *Projeto de Banco de Dados*.
- 5) Entidades associativas transformaram-se em tabela própria.

Em especial, vale mencionar que a relação **Curso** poderia ter sido subdividida em três esquemas menores, conforme será visto no tópico 5. Optou-se, porém, pela versão escolhida, porque, apesar de desnormalizada, reduz o número de *joins* necessários para a obtenção de dados referentes aos cursos. Enfim, trata-se de uma questão de desempenho. Seja como for, esse mapeamento também está previsto na obra do professor Heuser.

## 5. Formas Normais

As formas normais são regras utilizadas no *design* de bancos de dados relacionais para garantir a organização e a minimização de redundâncias. As principais FNs estão enumeradas a seguir.

### 1) Primeira Forma Normal (1FN)

Um atributo deve ter valores atômicos (sem conjuntos ou listas). Todas as entradas em uma coluna devem ser do mesmo tipo.

### 2) Segunda Forma Normal (2FN)

Atende a 1FN. Todos os atributos não-chave devem ser totalmente dependentes da chave primária (sem dependências parciais).

### 3) Terceira Forma Normal (3FN)

Atende a 2FN. Não deve haver dependências transitivas (ou seja, um atributo não-chave não pode depender de outro atributo não-chave).

### 4) Forma Normal de Boyce-Codd (BCNF)

Atende a 3FN. Para cada dependência funcional não trivial, a parte esquerda deve ser uma superchave, e a direita não poderá ser atributo primo.

Nesse contexto, será feita uma análise sobre a maior tabela criada no modelo lógico, isto é, aquela com maior quantidade de atributos — tabela **Curso** —, verificando sua qualidade em função das formas normais.

## 5.1. Dependências Funcionais da Maior Tabela

A dependência funcional (DF) é uma relação entre dois conjuntos de atributos em um banco de dados relacional. Diz-se que um atributo (ou conjunto de atributos) A determina outro atributo (ou conjunto de atributos) B se, para cada valor de A, existe exatamente um valor correspondente de B. Em termos formais, se temos duas coleções de atributos X e Y em uma tabela, dizemos que X determina funcionalmente Y (notado como  $X \rightarrow Y$ ) se, para cada par de tuplas (linhas) da tabela, se duas tuplas têm o mesmo valor para X, então elas também devem ter o mesmo valor para Y.

Por exemplo, considere uma tabela de estudantes com atributos **id**, **nome** e **curso**. Aqui, pode-se dizer que **id**  $\rightarrow$  **nome** (o **id** do estudante determina o seu nome) e que **id**  $\rightarrow$  **curso** (seu **id** determina o curso em que está matriculado). Se dois estudantes têm o mesmo **id**, eles devem ter o mesmo **nome** e o mesmo **curso**, caso contrário, a relação não é válida. A dependência funcional é fundamental para entender e aplicar as regras de normalização em bancos de dados.

A seguir, a tabela **Curso** será posta sob essa perspectiva, colocando-se:



**Curso**(vagas\_ano, enade, nome, cod\_emec, modalidade, grau, situacao, rotulo\_categoria\_curso, id\_ies\_campus, uf\_campus, municipio\_campus)

A tabela Curso prevê as seguintes dependências funcionais, além das triviais:

- Para todo **A** em **Curso**,  $\text{cod\_emec} \rightarrow \mathbf{A}$ ;
- $(\text{rotulo\_categoria\_curso}, \text{id\_ies\_campus}, \text{uf\_campus}, \text{municipio\_campus}) \rightarrow \text{cod\_emec}$ .
- $(\text{nome}, \text{id\_ies\_campus}) \rightarrow \text{rotulo}$ .

Logo, as chaves candidatas (CKs) são:

- $\text{cod\_emec}$ ;
- $(\text{rotulo\_categoria\_curso}, \text{id\_ies\_campus}, \text{uf\_campus}, \text{municipio\_campus})$ ;
- $(\text{nome}, \text{id\_ies\_campus}, \text{uf\_campus}, \text{municipio\_campus})$ ;

E o fecho de DFs se resume a (serão omitidas as DFs dedutíveis):

- Para todo **A** em **Curso**,  $\text{cod\_emec} \rightarrow \mathbf{A}$ ;
- Para todo **A** em **Curso**,  $(\text{rotulo\_categoria\_curso}, \text{id\_ies\_campus}, \text{uf\_campus}, \text{municipio\_campus}) \rightarrow \mathbf{A}$ ;
- Para todo **A** em **Curso**,  $(\text{nome}, \text{id\_ies\_campus}, \text{uf\_campus}, \text{municipio\_campus}) \rightarrow \mathbf{A}$ ;
- $(\text{nome}, \text{id\_ies\_campus}) \rightarrow \text{rotulo}$ .

- 1) Atende a 1FN, uma vez que a tabela apenas possui atributos monovalorados.
- 2) Primeiramente, ela atende a 1FN. Como a chave primária da tabela não é composta, não há nenhum subconjunto da chave (diferente dela mesma) que determine funcionalmente todos os atributos da tabela. Essas condições garantem que a tabela esteja na 2FN.
- 3) Primeiramente, ela atende a 2FN. Além disso, conforme vemos no conjunto de dependências funcionais, toda dependência não trivial é da forma  $\mathbf{A} \rightarrow \mathbf{B}$ , em que **A** é superchave ou **B** é atributo primo. Logo, por definição, está em 3FN.
- 4) Primeiramente, ela atende a 3FN. Por outro lado, existe a DF não trivial  $(\text{nome}, \text{id\_ies\_campus}) \rightarrow \text{rotulo}$ , em que *rotulo* é primo. Logo, não pode estar em FNBC, pois, se estivesse, toda DF não trivial teria a forma  $\mathbf{A} \rightarrow \mathbf{B}$ , em que **A** é superchave e **B** não é primo.

Para normalizar o esquema com *lossless join*, sugerimos a seguinte divisão:

**Curso**(vagas\_ano, enade, nome, cod\_emec, modalidade, grau, situacao)

**ministrado**(cod\_emec, id\_ies\_campus, uf\_campus, municipio\_campus)

**categoria**(nome, id\_ies\_campus, rotulo\_categoria\_curso)

Fecho das DFs da nova **Curso**:

- Para todo **A** em **Curso**,  $\text{cod\_emec} \rightarrow \mathbf{A}$ ;

Como *cod\_emec* é o único atributo primo, segue que a única DF com atributo primo à direita é trivial. Logo, **Curso** está em FNBC.

Fecho das DFs da nova **ministrado**:

- Para todo **A** em **ministrado**,  $\text{cod\_emec} \rightarrow \mathbf{A}$ ;

Como *cod\_emec* é o único atributo primo, segue que a única DF com atributo primo à direita é trivial. Logo, **ministrado** está em FNBC.

Fecho das DFs da nova **categoria**:

- Para todo **A** em **categoria**,  $(\text{nome}, \text{id\_ies\_campus}) \rightarrow \mathbf{A}$ ;

Como *(nome, id\_ies\_campus)* são os únicos atributos primos, segue que as únicas DFs com atributo primo à direita são triviais. Logo, **categoria** está em FNBC.

Quanto à escolha da PK de Curso na versão desnormalizada, optou-se por **cod\_emec** por determinar funcionalmente todos os outros atributos e de forma, evidentemente, minimal. Por mais que houvesse outras CKs, preferiu-se **cod\_emec** como PK, porque as demais candidatas, além de compostas, eram CKs por questões de regras de negócio, ao passo que **cod\_emec** o é por definição, o que o torna menos suscetível a mudanças, de maneira a favorecer a integridade do banco.

## 6. Criação de Tabelas

As tabelas foram criadas no servidor da disciplina de Banco de Dados, com sistema gerenciador de banco de dados (SGBD) Postgres. Foram usados os comandos:

- **CREATE TABLE new\_table:** criação de tabelas; e
- **ALTER TABLE existing\_table ADD CONSTRAINT constraint:** alteração de tabelas já existentes por meio da adição de restrições.

Vale mencionar que, a fim de aumentar a legibilidade, algumas indentações foram removidas do *script* de criação, que se encontra abaixo, sendo este também acessível na pasta *DDL*, no endereço:

<https://drive.google.com/drive/folders/1aHGMKj5A-15MmD0cHLyKikamW-zMDq09?usp=sharing>

```
CREATE TABLE IES (
    id_emec INT,
    nome VARCHAR(255) UNIQUE,
    sigla VARCHAR(20),
    email VARCHAR(255),
    ci INT CHECK (1 <= ci AND ci <= 5),
    credenciamento VARCHAR(100),
    cnpj CHAR(18),
    categ_adm VARCHAR(100),
    org_acad VARCHAR(100),
    situacao VARCHAR(100),

    CONSTRAINT PK_ies PRIMARY KEY (id_emec)
);

CREATE TABLE Local (
    uf CHAR(2) CHECK (uf IN ('AC', 'AL', 'AP', 'AM', 'BA', 'CE', 'DF', 'ES', 'GO',
    'MA', 'MT', 'MS', 'MG', 'PA', 'PB', 'PR', 'PE', 'PI', 'RJ', 'RN', 'RS', 'RO', 'RR',
    'SC', 'SP', 'SE', 'TO', 'Remoto')),
    municipio VARCHAR(255),

    CONSTRAINT PK_local PRIMARY KEY (uf, municipio)
);

CREATE TABLE Campus (
    id_ies INT,
    uf_local CHAR(2),
    municipio_local VARCHAR(255),

    CONSTRAINT PK_campus PRIMARY KEY (id_ies, uf_local, municipio_local),
    CONSTRAINT FK_campus_ies FOREIGN KEY (id_ies) REFERENCES IES(id_emec),
    CONSTRAINT FK_campus_local FOREIGN KEY (uf_local, municipio_local) REFERENCES
Local(uf, municipio)
);
```

```

CREATE TABLE Categoria_Curso (
    ger VARCHAR(255),
    rotulo VARCHAR(255)
);
/* Adicionando PK via ALTER TABLE */
ALTER TABLE Categoria_Curso ADD CONSTRAINT PK_categoria_curso PRIMARY KEY (rotulo);

CREATE TABLE Curso (
    vagas_ano INT CHECK (vagas_ano >= 0),
    enade INT CHECK (1 <= enade AND enade <= 5),
    nome VARCHAR(255),
    cod_emec CHAR(20),
    modalidade CHAR(10) CHECK (modalidade IN ('presencial', 'híbrido', 'remoto')),
    grau CHAR(12) CHECK (grau IN ('bacharelado', 'licenciatura', 'tecnólogo')),
    situacao CHAR(15) CHECK (situacao IN ('ativo', 'em extinção', 'extinto')),
    rotulo_categoria_curso VARCHAR(255),
    uf_campus CHAR(2),
    municipio_campus VARCHAR(255),
    id_ies_campus INT,

    CONSTRAINT PK_curso PRIMARY KEY (cod_emec),
    CONSTRAINT FK_curso_campus FOREIGN KEY (uf_campus, municipio_campus,
id_ies_campus) REFERENCES Campus(uf_local, municipio_local, id_ies),
    CONSTRAINT FK_curso_categoria_curso FOREIGN KEY (rotulo_categoria_curso)
REFERENCES Categoria_Curso(rotulo)
);

CREATE TABLE Pessoa (
    id CHAR(16), -- CHAR(16) por causa do ID Lattes
    nome VARCHAR(255) NOT NULL,

    CONSTRAINT PK_pessoa PRIMARY KEY (id)
);

CREATE TABLE Docente ( -- estende Pessoa
    id_pessoa CHAR(16),
    titulo CHAR(3) CHECK (titulo IN ('EFI', 'EFC', 'EMC', 'ESC', 'MES', 'DOC', 'PHD')),
    ano_conclusao CHAR(6),
    email VARCHAR(255),
    cnpq CHAR(2) CHECK (cnpq IN ('1A', '1B', '1C', '1D', '2', 'NA', 'SR')),
    habilitacao VARCHAR(100),
    nacionalidade VARCHAR(100),

    CONSTRAINT PK_docente PRIMARY KEY (id_pessoa),
    CONSTRAINT FK_docente_pessoa FOREIGN KEY (id_pessoa) REFERENCES Pessoa(id)
);

```

```

CREATE TABLE Discente ( -- estende Pessoa
    id_pessoa CHAR(16),
    genero CHAR(1) CHECK (genero IN ('M', 'F', 'N', 'O')),
    raca CHAR(10) CHECK (raca IN ('branca', 'preta', 'parda', 'amarela',
'indigena', 'outra')),
    renda CHAR(1) CHECK (renda IN ('A', 'B', 'C', 'D', 'E')),
    em_publico BOOLEAN,

    CONSTRAINT PK_discente PRIMARY KEY (id_pessoa),
    CONSTRAINT FK_discente_pessoa FOREIGN KEY (id_pessoa) REFERENCES Pessoa(id)
);

CREATE TABLE Deficiencia (
    id_discente CHAR(20),
    defic CHAR(20) CHECK (defic IN ('auditiva', 'intelectual', 'motora ou física',
'visual')),

    CONSTRAINT PK_deficiencia PRIMARY KEY (defic, id_discente),
    CONSTRAINT FK_deficiencia_discente FOREIGN KEY (id_discente) REFERENCES
Discente(id_pessoa)
);

/* TABELAS ORIGINADAS DE RELACIONAMENTOS */
CREATE TABLE candidata (
    id_discente CHAR(16),
    cod_curso CHAR(20),
    vest_status CHAR(15) CHECK (vest_status IN ('matrícula', 'reprovação',
'espera', 'desistência')),
    periodo CHAR(6),
    bolsa CHAR(20) CHECK (bolsa IN ('ProUni', 'FIES', 'Institucional', 'Nenhuma')),
    modalidade_vaga CHAR(2) CHECK (modalidade_vaga IN ('AC', 'PD', 'RA', 'RE')),
    vestibular CHAR(7) CHECK (vestibular IN ('ENEM', 'Proprio')),

    CONSTRAINT PK_candidata PRIMARY KEY (cod_curso, id_discente, periodo,
vestibular),
    CONSTRAINT FK_candidata_discente FOREIGN KEY (id_discente) REFERENCES
Discente(id_pessoa),
    CONSTRAINT FK_candidata_curso FOREIGN KEY (cod_curso) REFERENCES
Curso(cod_emec)
);

CREATE TABLE eh_func(
    id_docente CHAR(16),
    id_ies INT,
    cargo char(6) CHECK (cargo IN ('SUBS', 'ADJ_A', 'ASSOC', 'ASSIST', 'ADJ',
'TIT', 'OUTRO')),

    CONSTRAINT PK_eh_func PRIMARY KEY (id_docente, id_ies),
    CONSTRAINT FK_eh_func_docente FOREIGN KEY (id_docente) REFERENCES
Docente(id_pessoa),
    CONSTRAINT FK_eh_func_ies FOREIGN KEY (id_docente) REFERENCES IES(id_emec)

```

```

);

CREATE TABLE leciona (
    id_docente CHAR(16),
    cod_curso CHAR(20),

    CONSTRAINT PK_leciona PRIMARY KEY (id_docente, cod_curso)
);

/* Adicionando FKs via ALTER TABLE */
ALTER TABLE leciona ADD CONSTRAINT FK_leciona_docente FOREIGN KEY (id_docente)
REFERENCES Docente(id_pessoa);
ALTER TABLE leciona ADD CONSTRAINT FK_leciona_curso FOREIGN KEY (cod_curso)
REFERENCES Curso(cod_emec);

CREATE TABLE sai (
    id_discente CHAR(16),
    cod_curso CHAR(20),
    periodo CHAR(6),
    disc_status CHAR(12) CHECK (disc_status IN ('abandono', 'conclusão',
'jubilamento', 'troca')),

    -- Supomos que so se sai uma vez, por isso periodo nao compoe a PK
    CONSTRAINT PK_sai PRIMARY KEY (id_discente, cod_curso),
    CONSTRAINT FK_sai_discente FOREIGN KEY (id_discente) REFERENCES
Discente(id_pessoa),
    CONSTRAINT FK_sai_curso FOREIGN KEY (cod_curso) REFERENCES Curso(cod_emec)
);

CREATE TABLE tranca (
    id_discente CHAR(16),
    cod_curso CHAR(20),
    periodo CHAR(6),

    -- Supomos que se pode trancar mais de uma vez um mesmo curso, por isso periodo
compoe a PK
    CONSTRAINT PK_tranca PRIMARY KEY (cod_curso, id_discente, periodo),
    CONSTRAINT FK_tranca_discente FOREIGN KEY (id_discente) REFERENCES
Discente(id_pessoa),
    CONSTRAINT FK_tranca_curso FOREIGN KEY (cod_curso) REFERENCES Curso(cod_emec)
);

```

## 7. Inserção de Tuplas

Os dados foram inseridos por meio do comando **INSERT INTO VALUES values**. Para cada tabela do esquema relacional, um pequeno exemplo de inserção encontra-se abaixo. Para mais detalhes, os *scripts* de todas as inserções encontram-se na pasta *DML*, no endereço:

<https://drive.google.com/drive/folders/1aHGMKj5A-15MmD0cHLyKikamW-zMDq09?usp=sharing>

### 7.1. Tabelas

#### 7.1.1. Pessoa

```
INSERT INTO Pessoa (id, nome)
VALUES
  ('49630236754', 'João Souza'),
  ('21161597580', 'Maria Pereira'),
  ('82520806906', 'Patrícia Silva'),
  ('31287884747', 'Bruna Santos');
```

#### 7.1.2. Discente

```
INSERT INTO Discente (id_pessoa, genero, raca, renda, em_publico)
VALUES
  ('49630236754', 'M', 'branca', 'A', false),
  ('21161597580', 'F', 'amarela', 'A', true),
  ('82520806906', 'M', 'indigena', 'E', false);
```

#### 7.1.3. Deficiencia

```
INSERT INTO Deficiencia (id_discente, defic)
VALUES
  ('49630236754', 'motora ou física'),
  ('49630236754', 'intelectual'),
  ('49630236754', 'visual');
```

#### 7.1.4. Docente

```
INSERT INTO Docente (id_pessoa, titulo, ano_conclusao, email, cnpq,
habilitacao, nacionalidade)
VALUES
  ('17084965300', 'EMC', '2016.2', 'email@exemplo.com', '2 ',
'Literatura', 'Argentino'),
  ('76039415207', 'DOC', '2022.2', 'email@exemplo.com', 'SR',
'Química', 'Liberal');
```

```

('09732514680', 'EFC', '2016.2', 'email@exemplo.com', '1D',
'Matemática', 'Português');

```

#### 7.1.5. eh\_func

```

INSERT INTO eh_func (id_docente, id_ies, cargo)
VALUES
('17084965300', 2079, 'TIT'),
('06945781348', 2288, 'TIT'),
('98275401305', 1813, 'ASSOC');

```

#### 7.1.6. IES

```

INSERT INTO IES (id_emec, nome, sigla, email, ci, credenciamento, cnpj,
categ_adm, org_acad, situacao)
VALUES
(2565, 'ABEU - CENTRO UNIVERSITÁRIO (UNIABEU)', 'UNIABEU',
'pesquisador@abeu.edu.br', 5, 'EAD - Superior / Presencial - Superior',
'30.831.606/0001-30', 'Privada sem fins lucrativos', 'Centro
Universitário', 'Ativa'),
(26777, 'ACADEMIA DA FORÇA AÉREA (AFA)', 'AFA',
'dpl.direns@gmail.com', NULL, 'Presencial - Superior',
'00.394.429/0111-45', 'Pública Federal', 'Faculdade', 'Ativa'),
(26238, 'ACADEMIA DE BOMBEIROS MILITAR (ABM)', 'ABM',
'abm.ste@bombeiros.mg.gov.br', NULL, 'Presencial - Superior',
'03.389.126/0001-98', 'Pública Estadual', 'Faculdade', 'Ativa');

```

#### 7.1.7. Local

```

INSERT INTO Local (uf, municipio)
VALUES
('PA', 'Abaetetuba'),
('SP', 'Adamantina'),
('PE', 'Afogados da Ingazeira');

```

#### 7.1.8. Campus

```

INSERT INTO Campus (id_ies, uf_local, municipio_local)
VALUES
(24253, 'SP', 'São José do Rio Preto'),
(22101, 'MG', 'Patos de Minas'),
(2220, 'MG', 'Juiz de Fora');

```



#### 7.1.9. Categoria\_Curso

```
INSERT INTO Categoria_Curso (ger, rotulo)
VALUES
  ('Programas básicos', 'ABI Educação'),
  ('Programas básicos', 'ABI Artes e humanidades'),
  ('Programas básicos', 'ABI Negócios, administração e direito');
```

#### 7.1.10. Curso

```
INSERT INTO Curso (vagas_ano, enade, nome, cod_emec, modalidade, grau,
situacao, rotulo_categoria_curso, municipio_campus, uf_campus,
id_ies_campus)
VALUES
  (97, 1, 'ABI Educação', '0011A01A', 'remoto', 'bacharelado',
'ativo', 'Filosofia formação de professor', 'SP', 'São José do Rio
Preto', 24253),
  (67, 1, 'ABI Artes e humanidades', '0011A02A', 'remoto',
'bacharelado', 'ativo', 'Filosofia formação de professor', 'MG', 'Patos
de Minas', 22101),
  (95, 3, 'ABI Ciências sociais, comunicação e informação',
'0011A03A', 'remoto', 'licenciatura', 'extinto', 'Filosofia formação de
professor', 'MG', 'Juiz de Fora', 2220);
```

#### 7.1.11. candidata

```
INSERT INTO candidata (id_discente, cod_curso, vest_status, periodo,
bolsa, modalidade_vaga, vestibular)
VALUES
  ('49630236754', '0011A01A', 'matricula', 2021.2, 'ProUni', 'AC',
'proprio'),
  ('57840869979', '0312C01A', 'espera', 2017.2, 'institucional',
'RA', 'proprio'),
  ('68422867511', '1088P01A', 'matricula', 2015.1, 'FIES', 'PD',
'proprio');
```

#### 7.1.12. sai

```
INSERT INTO sai (id_discente, cod_curso, periodo, disc_status)
VALUES
  ('49630236754', '0011A01A', '2020.2', 'jubilamento'),
  ('21161597580', '0011A02A', '2024.2', 'troca'),
  ('82520806906', '0011A03A', '2000.2', 'conclusão');
```

#### 7.1.13. tranca

```
INSERT INTO tranca (id_discente, cod_curso, periodo)
VALUES
    ('75955946274', '0716E04A', '2010.1'),
    ('29540554925', '0716E05A', '2014.1'),
    ('84025734817', '0716M01A', '2023.1');
```

#### 7.1.14. leciona

```
INSERT INTO leciona (id_docente, cod_curso)
VALUES
    ('17084965300', '0011A01A'),
    ('17084965300', '0011A02A'),
    ('76039415207', '0011A03A');
```

## 8. Correções Feitas ao Primeiro Trabalho

Uma vez que a nota do primeiro trabalho foi máxima (10.0), pouco houve a ser feito. As três primeiras questões permaneceram inalteradas, ao passo que as demais sofreram pequenas mudanças, descritas a seguir.

Na quarta questão, mudamos os nomes de alguns atributos do esquema relacional, a fim de que houvesse maior uniformidade na nomenclatura de chaves estrangeiras. Para tal, adotamos o padrão *<nome-atributo>\_<tabela-referenciada>*, como visível no atributo de **Curso** *rotulo\_categoria\_curso*, uma FK para *rotulo* de **Categoria\_Curso**. Apesar de verbosa, tal convenção explicita a estrutura do banco e define um padrão facilmente dedutível para a construção de novos atributos, caso a base sofra expansão.

Na quinta, realizaram-se ligeiras adequações, cascadeadas a partir das renomeações feitas na questão anterior, com o objetivo de que o relatório se mantivesse um todo coerente.

Na sexta, decidimos alterar os tipos de alguns atributos com **ALTER COLUMN column TYPE new\_type**, haja vista que, dada a civilizada correria com que entregamos o trabalho, deixamos passar escolhas não muito adequadas. Assim, por exemplo, para atributos com domínios bem definidos, como *modalidade* de **Curso**, trocou-se o tipo *character varying* para *character* somente. Uma vez que sabíamos de antemão aproximadamente quantos caracteres teriam as entradas, consideramos válido um ligeiro desperdício de espaço em troca de ganho de desempenho. Ademais, adicionaram-se algumas restrições de domínio com **CHECK IN (...)**, que se haviam esquecido. Por fim, alteraram-se os nomes de algumas restrições com **RENAME CONSTRAINT**, a fim de uniformizarmos a nomenclatura. Todas essas mudanças foram repercutidas no *script* de DDL, sendo este agora exibido em tema claro, com o intuito de facilitar a leitura (assim esperamos).

Na sétima e última, tivemos o trabalho mais árduo: assegurarmos que as tuplas respeitassem as sete regras de negócio definidas na terceira questão. O porquê é evidente: melhorarmos a qualidade da base de dados. Diante disso, para cada restrição, definimos uma visão auxiliar que apontasse as tuplas inconsistentes, à qual demos um nome da forma *valida\_<regra-de-negocio>*, como *valida\_pcd*. Em seguida, executamos o **DELETE FROM** da visão, removendo-se as inconsistências. Para mais detalhes das visões, acesse o arquivo *validacao\_regras\_negocio.sql*, localizado na pasta *views*, no endereço:

<https://drive.google.com/drive/folders/1aHGMKj5A-15MmD0cHLyKikamW-zMDq09?usp=sharing>

Cabe mencionar que, para mantermos a coerência, os *scripts* de DML da questão 7 foram readequados tanto às mudanças de nome vindas da questão 4, quanto às remoções de tuplas. Além disso, foram adicionadas algumas novas tuplas, a fim de que as consultas do item 9 não resultassem vazias.

## 9. Consultas à Base de Dados

Abaixo, seguem seis consultas em *Structured Query Language* (SQL) realizadas com o intuito de exemplificar o funcionamento do banco de dados criado. A fim de torná-las mais facilmente reproduzíveis, foram criadas sob a forma de visões (*views*). Para mais detalhes, os *scripts* de todas as visões encontram-se na pasta *views*, no endereço:

<https://drive.google.com/drive/folders/1aHGMKj5A-15MmD0cHLyKikamW-zMDq09?usp=sharing>

### 9.1. Área geral com mais vagas anuais

#### 9.1.1. Enunciado em Português

*Qual a área geral das categorias de curso com maior quantidade de novas vagas anuais?*

#### 9.1.2. Consulta em SQL

Uso de **WITH**, **GROUP BY**, **SUM** e **MAX**:

```
CREATE OR REPLACE VIEW area_geral_mais_vagas AS
(
    WITH vagas_por_area_geral AS
    (
        SELECT      ger, SUM(vagas_ano) AS qtd_vagas
        FROM        categoria_curso AS Cat
                   INNER JOIN Curso AS Cur
        ON          Cat.rotulo = Cur.rotulo_categoria_curso
        GROUP BY    ger
    ),

    max_vagas AS
    (
        SELECT      MAX(qtd_vagas) AS qtd_vagas
        FROM        vagas_por_area_geral
    )

    SELECT  ger AS area_geral, qtd_vagas
    FROM    vagas_por_area_geral
           NATURAL INNER JOIN
           max_vagas
)
```

### 9.1.3. Saída

PostgreSQL 15.3 (Ubuntu 15.3-1.pgdg18.04+1) rodando em localhost:5433 -- Você está logado como usuário "bd124201" [SQL](#) | [Histórico](#) | [Encontrar](#) | [Sair](#)

phpPgAdmin: PostgreSQL?: bd124201?: trabalho\_bd?: area\_geral\_mais\_vagas?:

**Navegar**

area_geral	qtd_vagas
Educação	41888

1 linha(s)

[Voltar](#) | [Expandir](#) | [Atualizar](#)

Figura 2 Saída da consulta sobre a área geral com mais vagas anuais.

## 9.2. Professores de Computação remotos

### 9.2.1. Enunciado em Português

*Quais os professores com habilitação em Computação que lecionam ao menos um curso com modalidade remota?*

### 9.2.2. Consulta em SQL

Uso de **INNER JOIN** e de **ORDER BY**:

```
CREATE OR REPLACE VIEW professores_comp_remotos AS
(
    SELECT P.nome, P.id, D.habilitacao, C.cod_emec
    FROM pessoa AS P
        INNER JOIN docente AS D
            ON P.id = D.id_pessoa
        INNER JOIN leciona AS L
            ON L.id_docente = D.id_pessoa
        INNER JOIN curso AS C
            ON C.cod_emec = L.cod_curso
    WHERE C.modalidade = 'remoto'
    AND D.habilitacao LIKE '%Computação%'
    OR D.habilitacao LIKE '%Computacao%'
    ORDER BY P.nome
)
```

### 9.2.3. Saída

PostgreSQL 15.3 (Ubuntu 15.3-1.pgdg18.04+1) rodando em localhost:5433 -- Você está logado como usu

phpPgAdmin: PostgreSQL?: bd124201?: trabalho\_bd?: professores\_comp\_remos?:

## Navegar

nome	id	habilitacao	cod_emec
Camila Melo	14369278546	Computação	0732E02A
Gabriel Martins	91672405858	Computação	0114C02A
Giovanna Aragão	46328579128	Computação	0413E01A
Giovanna da Mota	84675023126	Computação	0722P02A
Luana Almeida	57321698491	Computação	0413A01A
Mariana da Rosa	40169837203	Computação	1015T01A
Rodrigo Santos	39602784547	Computação	0211C02A
Rodrigo Santos	39602784547	Computação	0211C01A
Thomas Fernandes	35829760410	Computação	0114C05A

9 linha(s)

[Voltar](#) | [Expandir](#) | [Atualizar](#)

Figura 3 Saída da consulta sobre professores de Computação remotos.

## 9.3. Alunos de baixa renda em vestibulares de 2020.1

### 9.3.1. Enunciado em Português

*Qual o total de alunos de baixa renda (classes E e D) que se inscreveram em vestibulares em 2020.1?*

### 9.3.2. Consulta em SQL

Uso de **COUNT**, de **INNER JOIN** e de **IN**:

```
CREATE OR REPLACE VIEW baixa_renda_2020_1_vest AS
(
    SELECT COUNT(*) AS qtd_inscritos_baixa_renda
    FROM    candidata AS C
           INNER JOIN
           discente AS D
    ON      C.id_discente = D.id_pessoa
    WHERE   C.periodo = '2020.1'
    AND     D.renda IN ('E', 'D')
)
```

### 9.3.3. Saída

PostgreSQL 15.3 (Ubuntu 15.3-1.pgdg18.04+1) rodando em localhost:5433 -- Você está logado como usuário "bd124201"

phpPgAdmin: PostgreSQL?: bd124201?: trabalho\_bd?: baixa\_renda\_2020\_1\_vest?:

**Navegar**

qtd_inscritos_baixa_renda
10

1 linha(s)

[Voltar](#) | [Expandir](#) | [Atualizar](#)

Figura 4 Saída da consulta sobre alunos de baixa renda em vestibulares de 2020.1.

## 9.4. Pessoas simultaneamente discentes e docentes

### 9.4.1. Enunciado em Português

*Quais pessoas são simultaneamente discentes e docentes?*

### 9.4.2. Consulta em SQL

Uso de **EXISTS** e de **ORDER BY**:

```

CREATE OR REPLACE VIEW discente_e_docente AS
(
    SELECT  P.*
    FROM    pessoa AS P
    WHERE EXISTS
    (
        SELECT  id_pessoa
        FROM    discente
        WHERE   id_pessoa = P.id
    )
    AND EXISTS
    (
        SELECT  id_pessoa
        FROM    docente
        WHERE   id_pessoa = P.id
    )
    ORDER BY P.nome ASC
)

```

#### 9.4.3. Saída

PostgreSQL 15.3 (Ubuntu 15.3-1.pgdg18.04+1) rodando em localhost:5433 -- Você está logado co

phpPgAdmin: PostgreSQL?: bd124201?: trabalho\_bd?: discente\_e\_docente?:

**Navegar**

id	nome
18511592474	Ana Gomes
53946986640	Bruna Pereira
20608506979	José Gomes
12887508963	José Santos
73162251572	Lucas Pereira
73504978749	Patrícia Almeida
54437990244	Patrícia Silva
91392766258	Paulo Costa
89229388046	Paulo Pereira

9 linha(s)

[Voltar](#) | [Expandir](#) | [Atualizar](#)

Figura 5 Saída da consulta sobre pessoas simultaneamente discentes e docentes.



## 9.5. Alunos com deficiência auditiva e visual

### 9.5.1. Enunciado em Português

*Quais discentes possuem simultaneamente deficiência auditiva e visual?*

### 9.5.2. Consulta em SQL

Uso de **VALUES**, **ORDER BY** e de **NOT EXISTS** com **EXCEPT** para construir operador de **DIVISÃO RELACIONAL**:

```
CREATE OR REPLACE VIEW pcd_auditiva_visual AS
(
    SELECT *
    FROM discente AS Disc
    WHERE NOT EXISTS
    (
        (
            SELECT defic
            FROM (VALUES ('auditiva'), ('visual')) AS Aux(defic)
        )
        EXCEPT
        (
            SELECT defic
            FROM deficiencia
            WHERE id_discente = Disc.id_pessoa
        )
    )
    ORDER BY Disc.id_pessoa ASC
)
```

### 9.5.3. Saída

id_pessoa	genero	raca	renda	em_publico
00040373070	F	parda	B	FALSE
00330531725	M	branca	B	TRUE
02560351019	F	parda	E	FALSE
04628382484	F	parda	D	TRUE
04991659373	M	branca	A	FALSE
06641982143	M	amarela	E	FALSE
12041063430	O	amarela	E	FALSE
13863682368	O	parda	A	FALSE
14562299143	M	amarela	C	TRUE
16339906811	O	amarela	B	TRUE
18104474165	F	parda	B	TRUE
21857862007	O	branca	E	FALSE
24869468106	O	preta	E	FALSE
25063232565	M	indigena	A	TRUE
27136783444	M	indigena	E	TRUE
27187670549	F	preta	A	FALSE
30970451841	F	parda	E	FALSE
31241115592	O	indigena	B	TRUE
34766580952	F	indigena	E	FALSE
36572275505	M	branca	E	FALSE
41346567284	M	preta	B	FALSE
41730707786	M	preta	D	FALSE
42408756514	O	branca	A	FALSE
45201217916	F	amarela	D	TRUE
48828604769	M	amarela	A	FALSE
49963224152	F	preta	A	TRUE
50552914807	F	amarela	D	FALSE
53635410892	O	amarela	A	FALSE
54269491677	M	indigena	C	FALSE
54437990244	O	amarela	E	TRUE

30 linha(s)

**Figura 6** Primeira parte da saída da consulta sobre discentes com deficiência auditiva e visual.

PostgreSQL 15.3 (Ubuntu 15.3-1.pgdg18.04+1) rodando em localhost:5433 -- Você está logado como

phpPgAdmin: PostgreSQL: bd124201: trabalho\_bd: pcd\_auditiva\_visual:

**Navegar**

id_pessoa	genero	raca	renda	em_publico
56254583435	M	parda	A	TRUE
58658939474	M	amarela	E	TRUE
61952484936	F	parda	C	TRUE
62284744742	M	parda	B	TRUE
64541660465	M	parda	B	FALSE
72420604867	O	branca	C	FALSE
72693100850	O	branca	C	TRUE
73686991794	F	parda	A	TRUE
75080128446	O	preta	B	TRUE
77760686582	M	indigena	C	FALSE
78433813757	F	indigena	A	TRUE
83598246339	O	parda	C	FALSE
84246633215	O	parda	D	FALSE
85556066913	O	preta	C	FALSE
85697634033	M	indigena	E	FALSE
87859606696	O	parda	D	TRUE
89229388046	F	branca	D	TRUE
90901548163	O	indigena	B	TRUE
95884394781	O	preta	E	FALSE
98241098168	M	amarela	B	FALSE

20 linha(s)

[Voltar](#) | [Expandir](#) | [Atualizar](#)

Figura 7 Segunda parte da saída da consulta sobre discentes com deficiência auditiva e visual.

## 9.6. IESs com discentes não homens em Engenharia ou em Computação

### 9.6.1. Enunciado em Português

*Quais as Instituições de Ensino Superior (IESs) e os respectivos números de candidatos nas quais se candidataram dois ou mais discentes não homens para cursos de Engenharia ou de Computação?*

### 9.6.2. Consulta em SQL

Uso de **WITH**, de **GROUP BY**, de **HAVING** e de **JOIN** feito à moda antiga, com **PRODUTO CARTESIANO** seguido de **SELEÇÃO**:

```
CREATE OR REPLACE VIEW _2_mais_candidatos_nao_homens_eng_ou_comp AS
(
    WITH cursos_eng_ou_comp AS
    (
        SELECT  C.cod_emec
        FROM    curso AS C
        WHERE   nome LIKE '%Engenharia%'
        OR      nome LIKE '%engenharia%'
        OR      nome LIKE '%Computação%'
        OR      nome LIKE '%computação%'
    ),

    homens AS
    (
        SELECT  id_pessoa
        FROM    discente
        WHERE   genero = 'M'
    ),

    candidatos_eng_ou_comp_nao_homens AS
    (
        SELECT  cod_curso, COUNT(*) AS total_nao_homens
        FROM    candidata
        WHERE   cod_curso IN
        (
            SELECT  cod_emec
            FROM    cursos_eng_ou_comp
        )
        AND id_discente NOT IN
        (
            SELECT  id_pessoa
            FROM    homens
        )
    )
)
```

```

    )
    GROUP BY cod_curso
)

-- Para diferenciar, fazemos o JOIN final à moda antiga, com produto
cartesiano e seleção:
SELECT I.sigla AS ies, SUM(total_nao_homens) AS
total_candidatos_nao_homens
FROM candidatos_eng_ou_comp_nao_homens AS CECNH, curso AS C, ies AS I
WHERE CECNH.cod_curso = C.cod_emec
AND C.id_ies_campus = I.id_emec
GROUP BY I.sigla
HAVING SUM(total_nao_homens) >= 2
)

```

### 9.6.3. Saída

PostgreSQL 15.3 (Ubuntu 15.3-1.pgdg18.04+1) rodando em localhost:5433 -- Você está logado como usuário "bd124201" [SQL](#) | [Histórico](#) | [Encontrar](#) | [Sair](#)

phpPgAdmin: PostgreSQL?: bd124201?: trabalho\_bd?: \_2\_mais\_candidatos\_nao\_homens\_eng\_ou\_comp?:

**Navegar**

ies	total_candidatos_nao_homens
FAEL CURITIBA	2
NULL	5

2 linha(s)

[Voltar](#) | [Expandir](#) | [Atualizar](#)

Figura 8 Saída da consulta sobre IESs com dois ou mais discentes não homens em Engenharia ou em Computação.

## 10. Visões propriamente ditas

### 10.1. Visões auxiliares

#### 10.1.1. candidaturas\_bolsa

Exibe todas as candidaturas atreladas a bolsas de estudo. Compreende o nome e o identificador do candidato juntamente com o tipo de bolsa e com o código do curso ao qual se destina a candidatura. É definida por:

```
CREATE OR REPLACE VIEW candidaturas_bolsa AS
(
    SELECT  P.nome, P.id, C.bolsa, C.cod_curso
    FROM    pessoa AS P
           INNER JOIN discente AS D ON
               P.id = D.id_pessoa
           INNER JOIN candidata AS C ON
               C.id_discente = D.id_pessoa
    WHERE   C.bolsa <> 'nenhum'
    AND     C.bolsa IS NOT NULL
)
```

#### 10.1.2. docentes\_remotos

Exibe os docentes que lecionam ao menos um curso em modalidade remota. Compreende o nome e o identificador do docente juntamente com o código do curso lecionado. É definida por:

```
CREATE OR REPLACE VIEW docentes_remotos AS
(
    SELECT  P.nome, P.id, Cur.cod_emec
    FROM    pessoa AS P
    JOIN    docente AS DOC ON P.id = Doc.id_pessoa
    JOIN    leciona AS L ON L.id_docente = Doc.id_pessoa
    JOIN    curso AS Cur ON Cur.cod_emec = L.cod_curso
    WHERE   Cur.modalidade = 'remoto'
    ORDER BY P.nome
)
```

## 10.2. Exemplos de uso

### 10.2.1. candidaturas\_bolsa

Enunciado

*Qual é o curso, o nome, e o número de identificação dos estudantes que se candidataram para Computação com uma bolsa de estudos?*

Consulta em SQL

```
CREATE OR REPLACE VIEW candidaturas_bolsa_comp AS
(
    SELECT C.nome AS nome_curso, C.cod_emec, E.nome AS nome_estudante, E.id
    FROM candidaturas_bolsa AS E
    INNER JOIN
    curso AS C
    ON cod_curso = cod_emec
    WHERE C.nome LIKE '%Computação%'
)
```

Saída

PostgreSQL 15.3 (Ubuntu 15.3-1.pgdg18.04+1) rodando em localhost:5433 -- Você está logado como usuário "bd124201"			
phpPgAdmin: PostgreSQL?: bd124201?: trabalho_bd?: candidaturas_bolsa_comp?:			
Navegar			
nome_curso	cod_emec	nome_estudante	id
Computação formação de professor	0114C05A	Fernanda Almeida	61861705831
ABI Computação e Tecnologias da Informação e ...	0011A06A	Fernanda Oliveira	07382452624
Computação formação de professor	0114C05A	Lucas Silva	26691312700

3 linha(s)

[Voltar](#) | [Expandir](#) | [Atualizar](#)

Figura 9 Saída da consulta sobre candidaturas com bolsa para Computação.

### 10.2.2. docentes\_remotos

Enunciado

*Encontre os docentes estrangeiros que lecionam cursos remotos.*

Consulta em SQL

```
CREATE OR REPLACE VIEW docentes_estrangeiros_remotos AS
(
    SELECT DISTINCT DR.nome, DR.id, DOC.nacionalidade
    FROM    docentes_remotos AS DR
    JOIN    docente AS DOC ON DR.id = DOC.id_pessoa
    WHERE   nacionalidade NOT LIKE '%Brasileir%'
    OR      nacionalidade NOT LIKE '%brasileir%'
    ORDER BY DOC.nacionalidade
)
```

Saída

Por simplicidade, serão exibidas apenas as páginas inicial e final da saída.

nome	id	nacionalidade
Luiz Fernando Ribeiro	69420817520	Andorrano
Vitor Hugo Correia	51237480871	Antiguano
Davi Lucca Correia	17084965300	Argentino
Gabriel da Costa	60387512977	Argentino
Giovanna Aragão	46328579128	Argentino
Luigi Carvalho	75362408135	Argentino
Leonardo Barbosa	60194523870	Australiano
Manuela da Costa	59873140839	Austriaco
Lucas Duarte	57430691234	Azerbaijano
Luiz Fernando da Mata	07218348653	Azerbaijano
Clara Alves	07148593204	Bielorrusso
Marina Viana	04619837539	Bielorrusso
Bruno Lima	81035697203	Bissau-Guineense
Yuri Nunes	84955173075	Bissau-Guineense
Maitê Duarte	81835607288	Bosnio
Mariane Pinto	24985310689	Bosnio
Davi Lucas Lopes	00732148840	Sávaro
Augusto da Rocha	56123789473	Bélgica
João Rocha	61732095434	Búlgaro
Francisco da Mata	85126430726	Cabo-verdiano
Isabella Peixoto	76035426800	Camarones
Pedro Lucas Aragão	91326745034	Camarones
Thomas Fernandes	35829780410	Camarones
Antônio Sales	67902854310	Cazaque
Benjamin Nascimento	15468367228	Cazaque
Davi Lucca Souza	29153674006	Cazaque
Emanuella Nunes	07934815204	Cazaque
Bruno Rodrigues	17596384200	Chadiano
Paulo Peixoto	12748609549	Chinês
Pedro Miguel Moraes	32149708523	Colombiano

30 linha(s)

Figura 10 Primeira das quatro páginas da saída com docentes estrangeiros que lecionam cursos remotos.



<< Primeiro < Anterior 1 2 3 4

nome	id	nacionalidade
Paulo Moura	70948523123	Tunisiano
Laura Cardoso	45196730893	Turco
Caroline Freitas	48163579200	Zambiano

3 linha(s)

Figura 11 Última das quatro páginas da saída com docentes estrangeiros que lecionam cursos remotos.

## 10.3. Visão verificadora de integridade

### 10.3.1. Definição em SQL

A fim de que se assegure o cumprimento da regra de negócio *Um discente só poderá candidatar-se a uma vaga com modalidade de baixa renda (RE), se tiver cursado integralmente o ensino médio em instituições públicas e pertencer à classe E*, foi criada a seguinte visão, com opção **CHECK OPTION**:

```
CREATE OR REPLACE VIEW insere_candidata_modalidade_renda AS
(
    SELECT  *
    FROM    candidata
    WHERE   id_discente IN
    (
        SELECT  id_pessoa
        FROM    discente
        WHERE   em_publico = TRUE
        AND     renda = 'E'
    )
    AND     modalidade_vaga = 'RE'
)
WITH CHECK OPTION
```

Note, então, que, para passarem no teste de integridade semântico da *view*, as tuplas a serem inseridas devem conter identificadores de discentes que tenham cursado ensino médio integralmente em instituições públicas (**em\_publico = TRUE**) e que pertençam à classe E (**renda = 'E'**), além de possuírem modalidade de vaga respectiva à cota por baixa renda (**modalidade\_vaga = 'RE'**).

### 10.3.2. Inserção bem-sucedida

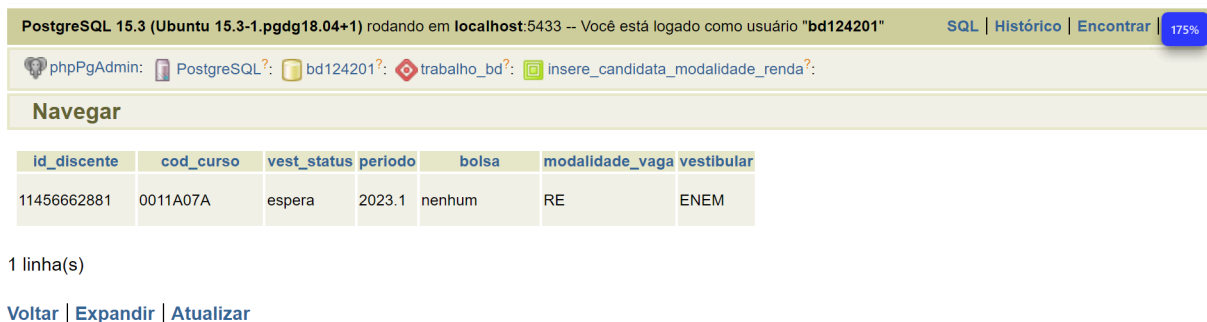
Tentamos inserir, inicialmente, uma tupla válida, pois possuía um discente que cumpria ambas as condições e modalidade de vaga de baixa renda. Evidentemente, esperava-se uma inserção bem-sucedida, o que de fato ocorreu, conforme demonstra a saída.

SQL

```
-- Tupla em discente que satisfaz ambas as condições:
('11456662881', 'F', 'indigena', 'E', TRUE)

INSERT INTO insere_candidata_modalidade_renda (id_discente, cod_curso,
vest_status, periodo, bolsa, modalidade_vaga, vestibular)
VALUES
('11456662881', '0011A07A', 'espera', '2023.1', 'nenhum', 'RE', 'ENEM');
```

Saída



PostgreSQL 15.3 (Ubuntu 15.3-1.pgdg18.04+1) rodando em localhost:5433 -- Você está logado como usuário "bd124201" [SQL](#) | [Histórico](#) | [Encontrar](#) 175%

phpPgAdmin: PostgreSQL? bd124201? trabalho\_bd? insere\_candidata\_modalidade\_renda?

Navegar

id_discente	cod_curso	vest_status	periodo	bolsa	modalidade_vaga	vestibular
11456662881	0011A07A	espera	2023.1	nenhum	RE	ENEM

1 linha(s)

[Voltar](#) | [Expandir](#) | [Atualizar](#)

Figura 12 Inserção bem-sucedida em insere\_candidata\_modalidade\_renda.

### 10.3.3. Inserção falha

Tentamos inserir, em seguida, duas tuplas inválidas: uma, porque o discente não pertencia à classe E; a outra, porque o discente não havia cursado ensino médio integralmente em instituições públicas. Os dois testes correram como esperado, consoante revelam as saídas.

Renda inválida

SQL

```
-- Tupla em discente que satisfaz em_publico, mas não satisfaz renda:
('21161597580', 'F', 'amarela', 'A', 'TRUE')

INSERT INTO insere_candidata_modalidade_renda (id_discente, cod_curso,
vest_status, periodo, bolsa, modalidade_vaga, vestibular)
VALUES
('21161597580', '0533F02A', 'espera', '2023.1', 'nenhum', 'RE', 'ENEM');
```

## Saída

PostgreSQL 15.3 (Ubuntu 15.3-1.pgdg18.04+1) rodando em localhost:5433 – Você está logado como usuário "bd124201" [SQL](#) | [Histórico](#) | [Encerrar](#)

phpPgAdmin: PostgreSQL: bd124201

Resultados da consulta

**Erro de SQL:**

ERROR: new row violates check option for view "insere\_candidata\_modalidade\_renda"  
DETAIL: Failing row contains (21161597580, 0533F02A, espera, 2023.1, nenhum, RE, ENEM ).

**No bloco:**

```
INSERT INTO insere_candidata_modalidade_renda (id_discente, cod_curso, vest_status, periodo, bolsa, modalidade_vaga, vestibular)
VALUES
('21161597580', '0533F02A', 'espera', '2023.1', 'nenhum', 'RE', 'ENEM');
```

Tempo de execução total: 2.874 ms

SQL executado.

[Editar SQL](#)

Figura 13 Inserção falha em insere\_candidata\_modalidade\_renda por conta de modalidade\_vaga.

## Ensino médio público inválido

### SQL

```
-- Tupla em discente que satisfaz renda, mas não satisfaz em_publico:
('82520806906', 'M', 'indigena', 'E', 'FALSE')

INSERT INTO insere_candidata_modalidade_renda (id_discente, cod_curso,
vest_status, periodo, bolsa, modalidade_vaga, vestibular)
VALUES
('82520806906', '0533F02A', 'espera', '2023.1', 'nenhum', 'RE', 'ENEM');
```

## Saída

PostgreSQL 15.3 (Ubuntu 15.3-1.pgdg18.04+1) rodando em localhost:5433 – Você está logado como usuário "bd124201" [SQL](#) | [Histórico](#) | [Encerrar](#)

phpPgAdmin: PostgreSQL: bd124201

Resultados da consulta

**Erro de SQL:**

ERROR: new row violates check option for view "insere\_candidata\_modalidade\_renda"  
DETAIL: Failing row contains (82520806906, 0533F02A, espera, 2023.1, nenhum, RE, ENEM ).

**No bloco:**

```
INSERT INTO insere_candidata_modalidade_renda (id_discente, cod_curso, vest_status, periodo, bolsa, modalidade_vaga, vestibular)
VALUES
('82520806906', '0533F02A', 'espera', '2023.1', 'nenhum', 'RE', 'ENEM');
```

Tempo de execução total: 2.182 ms

SQL executado.

[Editar SQL](#)

Figura 14 Inserção falha em insere\_candidata\_modalidade\_renda por conta de em\_publico.

## 10.3.4. Atualização falha

Por fim, para testarmos a integridade da modalidade de vaga, tentamos atualizar a modalidade das tuplas presentes na view para 'AC', o que contradiz a última das restrições. Conforme esperado, o SGBD bloqueou a operação e apontou um erro.

## SQL

```
UPDATE insere_candidata_modalidade_renda
SET modalidade_vaga = 'AC'
WHERE id_discente = '11456662881'
```

## Saída

PostgreSQL 15.3 (Ubuntu 15.3-1.pgdg18.04+1) rodando em localhost:5433 -- Você está logado como usuário "bd124201"

phpPgAdmin: PostgreSQL: bd124201

Resultados da consulta

**Erro de SQL:**

ERROR: new row violates check option for view "insere\_candidata\_modalidade\_renda"  
DETAIL: Failing row contains (11456662881, 0011A07A, espera, 2023.1, nenhum, AC, ENEM ).

**No bloco:**

```
UPDATE insere_candidata_modalidade_renda
SET modalidade_vaga = 'AC'
WHERE id_discente = '11456662881'
```

Tempo de execução total: 3.707 ms

SQL executado.

[Editar SQL](#)

**Figura 15** Inserção falha em `insere_candidata_modalidade_renda` por conta de renda.

## 11. Funções, Gatilhos e Procedimentos Armazenados

Nesta seção, serão abordados as funções, os gatilhos (*triggers*) e os procedimentos armazenados (*stored procedures* ou, simplesmente, SPs) implementados na base de dados. Seu intuito é, essencialmente, ora simplificar certos tipos de consultas — conforme será visto nas funções —, ora automatizar atualizações de dados cascadeadas entre diferentes tabelas.

Antes de prosseguir, vale ressaltar que, como a maior parte das IESs é composta de universidades, para fins de nomenclatura das funções, trataram-se *universidade* e *IES* como sinônimos.

### 11.1. Funções

#### 11.1.1. cursos\_oferecidos\_universidade

##### Definição

Retorna a tabela de cursos oferecidos pela instituição de ensino superior especificada via o parâmetro `universidade TEXT`, a ser identificada pelo nome ou pela sigla. Utilizando a linguagem PL/PgSQL, a função realiza uma consulta que combina as tabelas **Curso** e **IES**. O resultado é ordenado alfabeticamente. É definida por:

```
CREATE OR REPLACE FUNCTION cursos_oferecidos_universidade(universidade TEXT)
RETURNS TABLE(cursos_oferecidos TEXT) AS
$$
BEGIN
    RETURN QUERY
    SELECT C.nome::TEXT AS cursos_oferecidos -- Aqui convertemos
explicitamente para 'text'
    FROM curso AS C
    INNER JOIN
        ies AS I
    ON C.id_ies_campus = I.id_emec
    WHERE I.nome = universidade
    OR I.sigla = universidade
    ORDER BY C.nome ASC;
END;
$$ LANGUAGE plpgsql;
```

##### Teste

Foi escolhida a *Faculdade de Botucatu*, que oferece os cursos de *Engenharia têxtil*, *Gestão estratégica* e *Programas interdisciplinares abrangendo negócios, administração e direito*. Conforme mostra a saída, o teste, dado pela *query* abaixo, obteve êxito:

```
SELECT * FROM cursos_oferecidos_universidade('Faculdade de Botucatu (FDB)')
```

Saída

nome
Engenharia têxtil
Gestão estratégica
Programas interdisciplinares abrangendo negócios, administração e direito

3 linha(s)

Figura 16 Cursos oferecidos pela FDB.

### 11.1.2. qtd\_universidades\_municipio

Definição

Calcula a quantidade de universidades presentes em um município específico dentro de um estado (UF). Utilizando PL/PgSQL, a função realiza uma contagem dos registros na tabela **Campus** que correspondem ao município e UF fornecidos como parâmetros. Essa função é útil para consultas rápidas sobre a distribuição de instituições de ensino por região. É definida por:

```
CREATE OR REPLACE FUNCTION qtd_universidades_municipio(municipio TEXT, uf TEXT)
RETURNS INTEGER AS
$$
    DECLARE total_universidades INTEGER;
    BEGIN
        SELECT COUNT(*) INTO total_universidades
        FROM campus
        WHERE uf = uf_local
        AND municipio = municipio_local;
        RETURN total_universidades;
    END;
$$ LANGUAGE plpgsql;
```

Teste

Foi escolhida a cidade do *Rio de Janeiro*, localizada na UF *RJ*. De acordo com a tabela **Campus**, o resultado esperado era 18. Conforme mostra a saída, o teste, dado pela *query* abaixo, obteve êxito:

```
SELECT * FROM qtd_universidades_municipio('Rio de Janeiro', 'RJ')
```

PostgreSQL 15.3 (Ubuntu 15.3-1.pgdg18.04+1) rodando em localhost:5433 -- Você está logado como usuário "bd124201"

SQL | Histórico | Encontrar | Sair

phpPgAdmin: PostgreSQL?: bd124201?

Resultados da consulta

qtd_universidades_municipio
18

1 linha(s)

Tempo de execução total: 5.186 ms

SQL executado.

Editar SQL | Download

Figura 17 Cursos oferecidos pela FDB.

## 11.2. Procedimento armazenado

### 11.2.1. Definição

Foi implementada uma *procedure*, denominada *bota\_AC*, que atualiza a coluna *modalidade\_vaga* de **candidata** para o valor 'AC', desde que a modalidade original da vaga esteja entre 'PD', 'RA' ou 'RE', e o curso esteja associado a uma IES privada. Isso se deve à regra de negócio de que IESs privadas apenas oferecem vagas em ampla concorrência. É definida por:

```
CREATE OR REPLACE PROCEDURE bota_AC()
AS $$
BEGIN
    -- Atualizar a modalidade_vaga para 'AC' se o curso pertence a uma IES
    privada
    UPDATE candidata AS CA
    SET     modalidade_vaga = 'AC'
    WHERE   CA.modalidade_vaga IN ('PD', 'RA', 'RE') -- Modalidades que
    devem ser ajustadas
    AND     CA.cod_curso IN
    (
        SELECT C.cod_emec
        FROM   curso AS C
        JOIN   ies AS I
        ON     C.id_ies_campus = I.id_emec
        WHERE  I.categ_adm IN ('Privada com fins lucrativos', 'Privada sem
    fins lucrativos')
    );
END;
$$ LANGUAGE plpgsql;
```

### 11.2.2. Teste

Para testarmos o procedimento, definimos uma *view* auxiliar denominada *candidaturas\_cursos\_privados*, que continha todas as tuplas de **candidata** com

curso oferecidos por IESs privadas. Primeiramente, verificamos que as tuplas da *view* apresentavam-se coerentes. Em seguida, com **UPDATE**, atualizamos todas as modalidades para 'RE', e verificamos que a **view** tinha sido modificada. Por fim, executamos **bota\_AC()**, e constatamos que todas as tuplas da *view* possuíam agora modalidade 'AC', conforme esperado. Logo, o teste, composto pelas *queries* abaixo, foi exitoso.

```
CREATE OR REPLACE VIEW candidaturas_cursos_privados AS
(
    SELECT  *
    FROM    candidata
    WHERE   cod_curso IN
    (
        SELECT  cod_emec
        FROM    cursos_privados
    )
);

UPDATE candidaturas_cursos_privados
SET     modalidade_vaga = 'RE';

CALL bota_AC();
```

### 11.2.3. Saída

id_discente	cod_curso	vest_status	periodo	bolsa	modalidade_vaga	vestibular
49630236754	0011A01A	matricula	2021.2	ProUni	RE	proprio
09768714563	0011A08A	desistencia	2023.2	ProUni	RE	proprio
08444251370	0011A09A	espera	2023.1	institucional	RE	proprio
80116686852	0111C01A	reprovacao	2023.1	ProUni	RE	ENEM
11187410481	0113E02A	espera	2016.2	ProUni	RE	ENEM
20944613081	0114B01A	desistencia	2016.1	FIES	RE	proprio
85490262532	0114C03A	reprovacao	2020.2	institucional	RE	ENEM
61861705831	0114C05A	matricula	2018.1	ProUni	RE	ENEM
77142744344	0114E02A	reprovacao	2020.1	FIES	RE	proprio
76462098223	0114F01A	reprovacao	2015.2	nenhum	RE	ENEM
05827584245	0114H01A	desistencia	2016.1	ProUni	RE	proprio
35606721979	0114T01A	desistencia	2021.2	FIES	RE	ENEM
67094108517	0115L05A	reprovacao	2018.2	institucional	RE	ENEM
48419989959	0115L07A	desistencia	2018.2	ProUni	RE	ENEM
20002197886	0115L14A	reprovacao	2018.1	institucional	RE	proprio
29521805950	0321J01A	reprovacao	2016.1	ProUni	RE	ENEM
83938095725	0321P01A	desistencia	2020.1	nenhum	RE	ENEM
83016483087	0321R01A	espera	2019.2	institucional	RE	ENEM
74475899190	0322A01A	desistencia	2018.1	ProUni	RE	proprio
27318019504	0322R01A	desistencia	2015.1	FIES	RE	ENEM

Figura 18 candidaturas\_cursos\_privados antes de bota\_AC.



id_discente	cod_curso	vest_status	periodo	bolsa	modalidade_vaga	vestibular
49630236754	0011A01A	matricula	2021.2	ProUni	AC	proprio
09768714563	0011A08A	desistencia	2023.2	ProUni	AC	proprio
08444251370	0011A09A	espera	2023.1	institucional	AC	proprio
80116686852	0111C01A	reprovacao	2023.1	ProUni	AC	ENEM
11187410481	0113E02A	espera	2016.2	ProUni	AC	ENEM
20944613081	0114B01A	desistencia	2016.1	FIES	AC	proprio
85490262532	0114C03A	reprovacao	2020.2	institucional	AC	ENEM
61861705831	0114C05A	matricula	2018.1	ProUni	AC	ENEM
77142744344	0114E02A	reprovacao	2020.1	FIES	AC	proprio
76462098223	0114F01A	reprovacao	2015.2	nenhum	AC	ENEM
05827584245	0114H01A	desistencia	2016.1	ProUni	AC	proprio
35606721979	0114T01A	desistencia	2021.2	FIES	AC	ENEM
67094108517	0115L05A	reprovacao	2018.2	institucional	AC	ENEM
4841998959	0115L07A	desistencia	2018.2	ProUni	AC	ENEM
20002197886	0115L14A	reprovacao	2018.1	institucional	AC	proprio
29521805950	0321J01A	reprovacao	2016.1	ProUni	AC	ENEM
83938095725	0321P01A	desistencia	2020.1	nenhum	AC	ENEM
83016483087	0321R01A	espera	2019.2	institucional	AC	ENEM
74475899190	0322A01A	desistencia	2018.1	ProUni	AC	proprio
27348618504	0322B01A	desistencia	2015.1	FIES	AC	ENEM

Figura 19 candidaturas\_cursos\_privados depois de bota\_AC.

## 11.3. Gatilho

### 11.3.1. Definição

Inicialmente, criou-se a função `verifica_e_insere_local`, com retorno tipo **TRIGGER** — sendo, assim, associável a um gatilho —, para garantir que uma combinação de UF e município seja registrada em **Local** antes de associada a um registro em **Campus**, caso ainda não exista em **Local**. É definida por:

```
CREATE OR REPLACE FUNCTION verifica_e_insere_local()
RETURNS TRIGGER AS
$$
BEGIN
    -- Verifica se a combinação de UF e município já existe
    IF NOT EXISTS
    (
        SELECT *
        FROM local
        WHERE uf = NEW.uf_local
        AND      municipio = NEW.municipio_local
    ) THEN
        -- Insere o local na tabela local
        INSERT INTO local (uf, municipio)
        VALUES (NEW.uf_local, NEW.municipio_local);
```

```

        END IF;
        RETURN NEW; -- Continua a operação de inserção em Campus
    END;
$$ LANGUAGE plpgsql;

```

Em seguida, um *trigger*, denominado `trg_verifica_e_insere_local`, foi associado à função, configurado para executar a verificação antes de cada operação de inserção em **Campus**. Note, portanto, que o gatilho relaciona duas tabelas do esquema relacional: **Campus** e **Local**. É definido por:

```

CREATE OR REPLACE TRIGGER trg_verifica_e_insere_local
    BEFORE INSERT ON campus
    FOR EACH ROW
        EXECUTE FUNCTION verifica_e_insere_local();

```

### 11.3.2. Teste

Em um primeiro teste, o comando **INSERT INTO campus** foi executado com os valores (528, 'RJ', 'Milmandia'), em que 528 é o identificador da PUC-Rio. Após a execução, foi confirmado que o município *Milmandia*, que antes não existia na base, foi devidamente adicionado a **Local**, permitindo o término da inserção em **Campus**, que também se mostrou exitosa.

Em um segundo, tentou-se inserir um *campus* da UFRJ em *Milmandia* com (586, 'RJ', 'Milmandia'), a fim de verificar se a função tentaria reinserir o município em **Local**, o que provocaria um erro de duplicidade. Como não foi apontado qualquer erro durante a execução e como a tupla foi inserida corretamente em **Campus**, concluímos que o teste obteve êxito.

Com ambos os testes, definidos abaixo em SQL, validou-se o correto funcionamento do *trigger* e da função associada.

```

/* VIEWS AUXILIARES */
-- Municípios do RJ que começam com 'M'
CREATE OR REPLACE VIEW municipios_rj_m AS
(
    SELECT  *
    FROM    local
    WHERE   municipio LIKE 'M%'
    AND     uf = 'RJ'
    ORDER BY municipio
);

-- Campi localizados no estado RJ
CREATE OR REPLACE VIEW campi_rj AS
(
    SELECT  *

```

```

        FROM      campus
        WHERE      uf_local = 'RJ'
        ORDER BY  id_ies
    );

/* TESTE 1 */
-- Exibir estado de Local e de Campus antes da inserção, de modo a apontar que
-- não existe o município "Milmandia"
SELECT * FROM municipios_rj_m;
SELECT * FROM campi_rj;

-- Inserir valor em Campus
INSERT INTO Campus (id_ies, uf_local, municipio_local)
VALUES (528, 'RJ', 'Milmandia'); -- 528 é o id_emec da PUC-Rio

-- Reverificar Local e Campus
SELECT * FROM municipios_rj_m;
SELECT * FROM campi_rj;

/* TESTE 2 */
-- Inserir novamente outro campus nesse município e verificar que
-- não há duplicidade na inserção
INSERT INTO Campus (id_ies, uf_local, municipio_local)
VALUES (586, 'RJ', 'Milmandia'); -- 586 é o id_emec da UFRJ

-- Reverificar Local e Campus
SELECT * FROM municipios_rj_m;
SELECT * FROM campi_rj;

```

### 11.3.3. Saída

PostgreSQL 15.3 (Ubuntu 15.3-1.pgdg18.04+1) rodando em localhost:5433 -- Você está logado como usuário "bd124201"
SQL | Histórico  
| Encontrar | Sair

phpPgAdmin: PostgreSQL?: bd124201?:

**Resultados da consulta**

uf	municipio
RJ	Macaé
RJ	Maricá
RJ	Miguel Pereira

3 linha(s)

Tempo de execução total: 7.117 ms

SQL executado.

[Editar SQL](#) | [Download](#)

Figura 20 Municípios do Rio de Janeiro que começam com *M* antes da inserção.

PostgreSQL 15.3 (Ubuntu 15.3-1.pgdg18.04+1) rodando em localhost:5433 -- Você está logado como usuário "bd124201"

phpPgAdmin: PostgreSQL?: bd124201?: trabalho\_bd?: campi\_rj?:

**Navegar**

id_ies	uf_local	municipio_local
26	RJ	Rio de Janeiro
77	RJ	Cabo Frio
202	RJ	Rio de Janeiro
278	RJ	Rio de Janeiro
516	RJ	Rio de Janeiro
593	RJ	Rio de Janeiro
633	RJ	Rio de Janeiro
677	RJ	Rio de Janeiro
991	RJ	Rio de Janeiro
1928	RJ	Rio de Janeiro
1961	RJ	Campos dos Goytacazes
2091	RJ	Rio de Janeiro
2126	RJ	Rio de Janeiro
2571	RJ	Itaperuna
3525	RJ	Rio de Janeiro
4428	RJ	Macaé
5016	RJ	Rio de Janeiro
5019	RJ	Rio de Janeiro
5051	RJ	Bom Jesus do Itabapoana
16967	RJ	Rio de Janeiro
21503	RJ	Rio de Janeiro
21885	RJ	Rio de Janeiro
22121	RJ	Três Rios
23818	RJ	Niterói
23820	RJ	Rio de Janeiro

25 linha(s)

[Voltar](#) | [Expandir](#) | [Atualizar](#)

Figura 21 *Campi* localizados no estado do Rio de Janeiro antes da inserção.

PostgreSQL 15.3 (Ubuntu 15.3-1.pgdg18.04+1) rodando em localhost:5433 -- Você está logado como usuário "bd124201"

SQL | Histórico  
Encontrar | Sair

phpPgAdmin: PostgreSQL?: bd124201?: trabalho\_bd?: municipios\_rj\_m?:

**Navegar**

uf	municipio
RJ	Macaé
RJ	Maricá
RJ	Miguel Pereira
RJ	Milmandia

4 linha(s)

[Voltar](#) | [Expandir](#) | [Atualizar](#)

Figura 22 Municípios do Rio de Janeiro que começam com *M* após ambas as inserções.

PostgreSQL 15.3 (Ubuntu 15.3-1.pgdg18.04+1) rodando em localhost:5433 -- Você está logado como usuário "bd124201"

phpPgAdmin: PostgreSQL?: bd124201?: trabalho\_bd?: campi\_rj?:

## Navegar

id_ies	uf_local	municipio_local
26	RJ	Rio de Janeiro
77	RJ	Cabo Frio
202	RJ	Rio de Janeiro
278	RJ	Rio de Janeiro
516	RJ	Rio de Janeiro
528	RJ	Milmandia
586	RJ	Milmandia
593	RJ	Rio de Janeiro
633	RJ	Rio de Janeiro
677	RJ	Rio de Janeiro
991	RJ	Rio de Janeiro
1928	RJ	Rio de Janeiro
1961	RJ	Campos dos Goytacazes
2091	RJ	Rio de Janeiro
2126	RJ	Rio de Janeiro
2571	RJ	Itaperuna
3525	RJ	Rio de Janeiro
4428	RJ	Macaé
5016	RJ	Rio de Janeiro
5019	RJ	Rio de Janeiro
5051	RJ	Bom Jesus do Itabapoana
16967	RJ	Rio de Janeiro
21503	RJ	Rio de Janeiro
21885	RJ	Rio de Janeiro
22121	RJ	Três Rios
23818	RJ	Niterói
23820	RJ	Rio de Janeiro

27 linha(s)

[Voltar](#) | [Expandir](#) | [Atualizar](#)

Figura 23 *Campi* localizados no estado do Rio de Janeiro após ambas as inserções.

## 12. Índice

Um índice é uma estrutura auxiliar associada a uma tabela que pode proporcionar ganho de *performance* em consultas SQL, porque, dado um valor que se deseja buscar, consegue mapeá-lo à página ou até mesmo ao registro onde se encontra. Para tanto, são utilizados ponteiros.

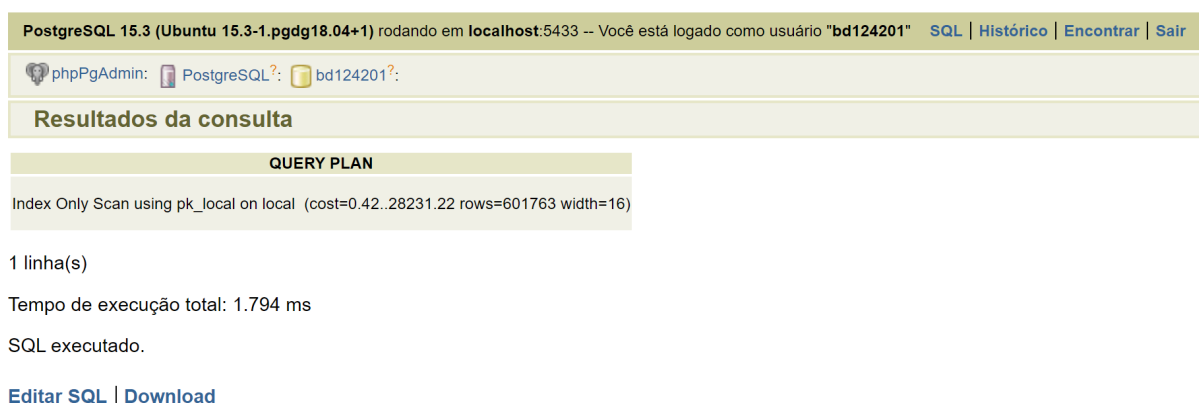
### 12.1. Índice Primário Existente

Nesse sentido, o índice primário se caracteriza por ponteiros que referenciam, de maneira segmentada e ordenada, registros de um arquivo necessariamente ordenado por alguma de suas chaves, quase sempre a primária. Mais claramente, ele permite acesso em disco da página de um registro específico mediante sua chave. Em geral, toda tabela contendo definição de chave primária será ordenada por esse atributo e, conseqüentemente, terá um índice primário automaticamente criado pelo SGBD.

Como exemplo de índice primário, podemos considerar aquele criado automaticamente pelo SGBD na tabela **Local**, quando é adicionada a chave primária (*uf*, *municipio*) com **ADD CONSTRAINT PRIMARY KEY**. Neste caso, o índice foi estruturado com base na chave primária e recebeu o mesmo nome, *pk\_local*. Para constatar seu uso pelo SGBD, foram adicionadas mais de 600 mil tuplas arbitrárias na tabela, fazendo com que o volume de dados forçasse o uso do índice. Assim, foi proposta a seguinte consulta:

```
SELECT  *  
FROM    local  
ORDER BY uf
```

A utilização do índice foi confirmada pelo plano de execução da consulta, obtido por meio do comando **EXPLAIN**. Nele, o PostgreSQL indica a realização de um *Index Scan* sobre o índice *pk\_local*, o que atesta a utilização do índice primário pelo otimizador de consultas para acelerar o acesso aos dados, como pode ser observado abaixo:



PostgreSQL 15.3 (Ubuntu 15.3-1.pgdg18.04+1) rodando em localhost:5433 -- Você está logado como usuário "bd124201" [SQL](#) | [Histórico](#) | [Encontrar](#) | [Sair](#)

phpPgAdmin: PostgreSQL? bd124201?

### Resultados da consulta

QUERY PLAN
Index Only Scan using pk_local on local (cost=0.42..28231.22 rows=601763 width=16)

1 linha(s)

Tempo de execução total: 1.794 ms

SQL executado.

[Editar SQL](#) | [Download](#)

Figura 24 Plano de execução da consulta na tabela Local.

## 12.2. Índices Secundários

Os índices secundários caracterizam-se por serem estruturados a partir de um ou mais atributos que não se encontrem ordenados no arquivo que armazena a tabela. Diante disso, diferentemente do primário, que mapeia chaves em páginas, o índice secundário mapeia valores diretamente aos registros, sendo, portanto, chamado *denso*.

Os índices secundários a seguir são usados para facilitar as consultas referentes ao CNPJ e à organização acadêmica das IESs, as quais têm uma quantidade significativa de dados inseridos no banco. Como será visto abaixo, eles foram, de fato, utilizados pelo SGBD em consultas de complexidade relativamente simples, isto é, cumprindo a função para qual foram originalmente criados, de reduzir o tempo necessário para encontrar dados pertencentes a esses atributos.

### 12.2.1. indSec\_cnpj

Definição

```
CREATE INDEX indSec_cnpj
ON ies (cnpj)
```

Exemplo de *query*

```
SELECT *
FROM ies
WHERE cnpj = '00.331.801/0001-30'
```

Plano de execução

QUERY PLAN
Bitmap Heap Scan on ies (cost=4.31..17.82 rows=4 width=184)
Recheck Cond: (cnpj = '00.331.801/0001-30'::bpchar)
-> Bitmap Index Scan on indsec_cnpj (cost=0.00..4.31 rows=4 width=0)
Index Cond: (cnpj = '00.331.801/0001-30'::bpchar)

Figura 25 Plano de execução da consulta na tabela IES com índice secundário em *cnpj*.

### 12.2.2. indSec\_org\_acad

Definição

```
CREATE INDEX indSec_org_acad  
ON ies (org_acad)
```

Exemplo de *query*

```
SELECT *  
FROM   ies  
WHERE  org_acad = 'Universidade'  
ORDER BY org_acad
```

Plano de execução

QUERY PLAN
Index Scan using indsec_org_acad on ies (cost=0.28..42.52 rows=206 width=184)
Index Cond: ((org_acad)::text = 'Universidade'::text)

Figura 26 Plano de execução da consulta na tabela IES com índice secundário em org\_acad.



## 13. Referências Bibliográficas

BRASIL. **Constituição da República Federativa do Brasil**. Art. 37, inciso XVI. Diário Oficial da União, Brasília, 5 out. 1988. Disponível em:

[https://www.planalto.gov.br/ccivil\\_03/constituicao/constituicao.htm](https://www.planalto.gov.br/ccivil_03/constituicao/constituicao.htm).

Acesso em: 19 out. 2024.

BRASIL. Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira (INEP). **Exame Nacional de Desempenho dos Estudantes (ENADE)**. 2024a. Disponível em:

<https://www.gov.br/inep/pt-br/aceso-a-informacao/perguntas-frequentes/exame-nacional-de-desempenho-dos-estudantes-enade>.

Acesso em: 12 out. 2024.

BRASIL. Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira (INEP). Ministério da Educação (MEC). **Manual para classificação dos cursos de graduação e sequenciais: CINE Brasil**. Brasília: Inep, 2019.

BRASIL. Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira. **Relatório de cursos - ENADE**. 2024b. Disponível em:

<https://enade.inep.gov.br/enade/#!/relatorioCursos>.

Acesso em: 12 out. 2024.

BRASIL. **Lei nº 12.089, de 11 de novembro de 2009**. Veda que uma mesma pessoa ocupe 2 (duas) vagas simultaneamente em instituições públicas de ensino superior. Diário Oficial da União: seção 1, Brasília, DF, p. 1, 12 nov. 2009. Disponível em:

[https://www.planalto.gov.br/ccivil\\_03/\\_ato2007-2010/2009/lei/l12089.htm](https://www.planalto.gov.br/ccivil_03/_ato2007-2010/2009/lei/l12089.htm).

Acesso em: 19 out. 2024.

BRASIL. Ministério da Educação (MEC). **e-MEC: sistema eletrônico de acompanhamento de processos das instituições de ensino superior no Brasil**. 2024c. Disponível em:

<https://emec.mec.gov.br/emec/nova>.

Acesso em: 15 out. 2024.

DESAFIOS NA EDUCAÇÃO. **Conceito Institucional: importância no ensino superior**. Homepage da instituição, 2021. Disponível em:

<https://desafiosdaeducacao.com.br/conceito-institucional-importancia-no-ensino-superior/>.

Acesso em: 15 out. 2024.

HEUSER, Carlos A. **Projeto de Banco de Dados**. Porto Alegre: Sagra & Luzzatto, 1999.

INSTITUTO BRASILEIRO DE GEOGRAFIA E ESTATÍSTICA. **Cidades e Estados**. Portal IBGE, 2024a. Disponível em:

<https://cidades.ibge.gov.br/>.

Acesso em: 12 out. 2024.

INSTITUTO BRASILEIRO DE GEOGRAFIA E ESTATÍSTICA. **Cor ou raça**. Portal IBGE Educacional, 2024b. Disponível em:

<https://educa.ibge.gov.br/jovens/conheca-o-brasil/populacao/18319-cor-ou-raca.html>.

Acesso em: 19 out. 2024.

UNIVERSIDADE FEDERAL DO RIO DE JANEIRO. **Edital nº 4/2024**. Acesso ao ensino superior - THE ENEM 2024. p. 13. Rio de Janeiro, 2024. Disponível em:

[https://acessograduacao.ufjr.br/periodo-2024-1/2024-the-enem/divulgacao-dos-editais-para-o-acesso-ufjr-the-enem-2024/SEI\\_3881829\\_Edital\\_4-Geral-assinado.pdf](https://acessograduacao.ufjr.br/periodo-2024-1/2024-the-enem/divulgacao-dos-editais-para-o-acesso-ufjr-the-enem-2024/SEI_3881829_Edital_4-Geral-assinado.pdf). Acesso em: 19

out. 2024.

## **14. Agradecimentos Finais**

Expressamos enorme contentamento em finalizar o relatório desse projeto extensivo, no qual depositamos grande esforço intelectual e de tempo. Esperamos, com isso, poder ajudar a coordenadora dos cursos de graduação da PUC-Rio, que encomendou esse trabalho ao professor Sérgio da disciplina de Banco de Dados, que o repassou a nós, seus caros alunos. Que nosso empenho possa servir a uma modelagem real de base de dados em nossa universidade.