Universidade Federal de São Carlos - Campus Sorocaba
Bacharelado em Ciência da Computação
Banco de Dados

Grupo 05 – Plataforma online de cursos

Professora: Sahudy Montenegro González

Gabriel Goulart Homem - RA: 771011
Maurício Marques da Silva Junior - RA: 771053
Renan Oliveira de Barros Lima - RA: 771061

Fase Final
Data Entrega: 21/06/2021

Índice

1 - Descrição do problema

O banco de dados sobre a plataforma online de cursos tem como objetivo conter informações sobre os usuários da mesma, sendo eles alunos e professores, além de conter informações sobre os cursos desenvolvidos pelos professores.

O usuário da plataforma tem como dados seu nome, email, sua senha de acesso - que deve possuir pelo menos 8 caracteres, data de nascimento e sua idade calculada a partir da data de nascimento. O mesmo pode ser aluno e professor ao mesmo tempo ou não. Como aluno ele pode cursar os cursos presentes no banco de dados desde que ele tenha comprado eles, e tem direito a um desconto no preço do curso dependendo de quando a compra do curso foi executada. Já o professor tem que cadastrar suas formações acadêmicas, e ele pode desenvolver vários cursos na plataforma sozinho ou em parceria com outros professores.

Sobre o curso devem ser guardadas sua descrição, código do curso, nome, quantidade de módulos e preço, sendo que o preço sempre será maior ou igual a zero. O curso também possui tanto sua categoria, que detém sua categoria principal e seu glossário (outros significados para a mesma categoria, exemplo “TI” e Tecnologia da informação), quanto vários módulos, que por sua vez contém suas seções, o nome do módulo e um ou vários vídeos contidos no módulo do curso.

1.1 - Consultas

1. Quantos alunos cursaram cada curso?
    a. retorna o número de alunos inscritos em cada curso
2. Quantos usuários tem na plataforma?
    a. retorna o valor numérico
3. Listar os cursos de uma determinada categoria
    a. recebe o nome da categoria
    b. retorna a lista de Cursos
4. Listar os professores que são alunos
    a. retorna todos os professores que são alunos
5. Listar os módulos de um curso
    a. recebe um curso
    b. retorna uma lista com os módulos.
6. Pegar o conteúdo de um módulo de um curso específico
    a. recebe um curso, e o módulo que se quer obter as vídeo-aulas
    b. recebe os valores do módulo
7. Pesquisar curso pelo nome do curso
    a. recebe o nome do curso
    b. retorna o preço e descrição
8. Calcular o preço do curso com desconto
    a. recebe um curso
    b. retorna o valor do curso
9. Quantidade de vídeos por módulo de um curso
    a. recebe um curso
    b. retornar a quantidade de vídeos para cada módulo do curso

2 - Projeto conceitual

A figura 1 e a tabela 1 apresentam o projeto do modelo de entidade de relacionamento obtido a partir da descrição do problema.

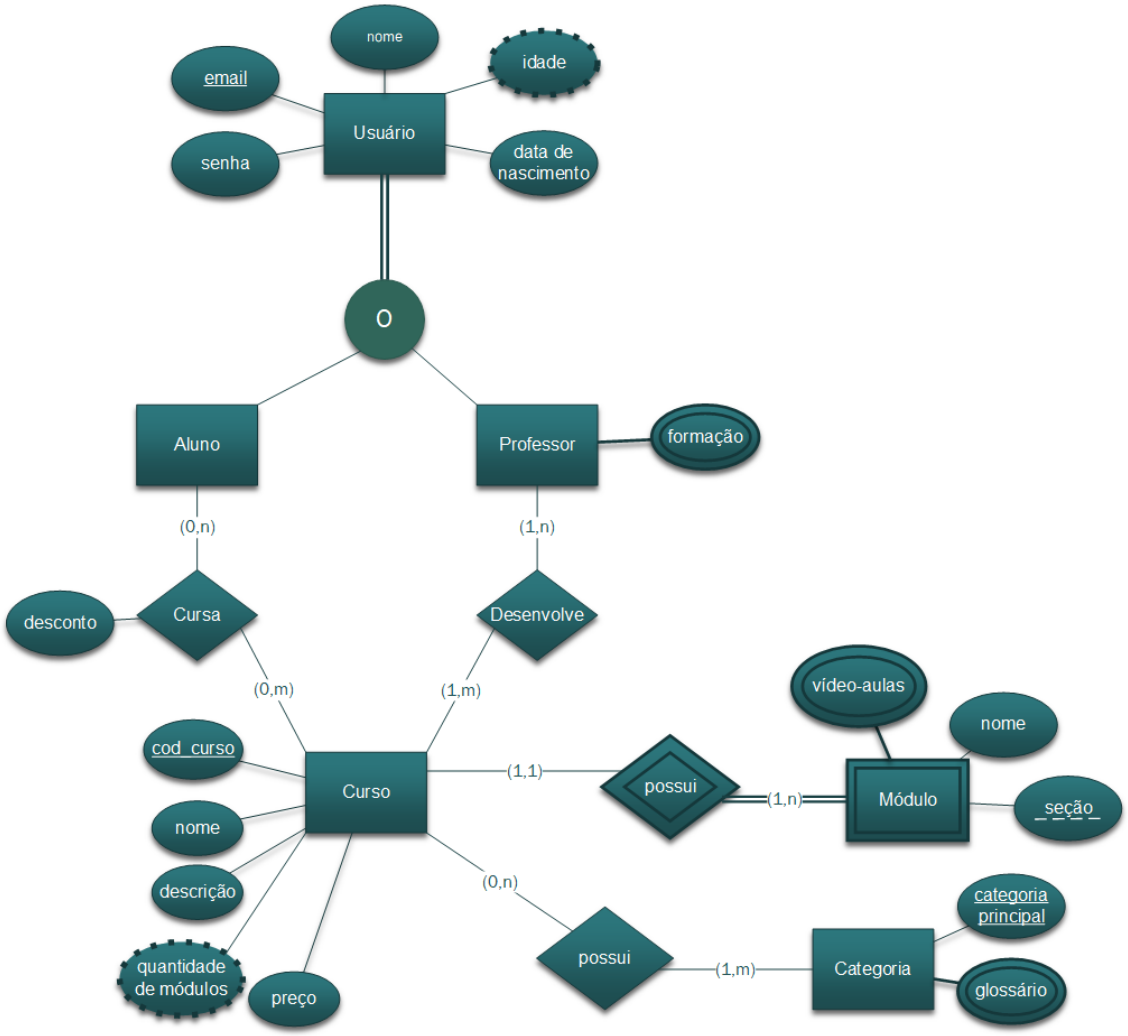

Figura 1: Diagrama de Entidade Relacionamento

Fonte: Autoria própria

Tipo-Entidade	Atributo	Tipo	Restrição
Usuário	email	identificador	Obrigatório
	nome	monovalorado	Obrigatório
	senha	monovalorado	Obrigatório, tamanho >= 8
	data de nascimento	monovalorado	Obrigatório
	idade	calculado	Obrigatório
Usuário: Aluno			
Usuário: Professor	formação	multivalorado	Obrigatório
Curso	cod curso	identificador	Obrigatório
	nome	monovalorado	Obrigatório
	descrição	monovalorado	Obrigatório
	preço	monovalorado	Obrigatório, valor >= 0
	quantidade de módulos	calculado	Obrigatório
Módulo	seção	identificador parcial	Obrigatório
	nome	monovalorado	Obrigatório
	video-aulas	multivalorado	Obrigatório
Categoria	categoria principal	identificador	Obrigatório
	glossário	multivalorado	Obrigatório
Cursa	desconto	monovalorado	Opcional, valor entre 0 e 1

Tabela 1: Tabela de metadados

Fonte: Autoria própria

3 - Projeto Lógico

A transformação do diagrama de entidade relacionamento para o modelo relacional apresentado na figura 2 ocorreu de modo manual e o modelo final atende tanto a 3ª forma normal (3FN) quanto a forma normal de Boyce-Codd(FNBC). Porque respeita a 1ª forma normal, onde todos os atributos são monovalorados e os que eram multivalorados viram tabelas ou relações próprias, atende também a 2ª forma normal, já que os atributos só dependem da chave primária tendo apenas dependência total, depois atende a 3ª por não ter dependência transitiva, tendo somente dependência funcional total em todas as relações, e por último a forma normal de Boyce-Codd é atendida pois em todas as dependências funcionais somente superchaves determinam outros elementos da entidade.

O modelo relacional da plataforma teve sua herança projetada utilizando atributos discriminadores para determinar se o usuário é um aluno e/ou professor (opção D dos slides de aula). A razão pela qual tal modelo foi selecionado é devido ao pequeno número de subclasses da herança do nosso projeto, aos poucos atributos nessas subclasses, fazendo com que haja poucos atributos adicionais, e pela característica desse modelo de evitar muita repetição de chaves primárias e junções desnecessárias.

O problema ao utilizar o modelo escolhido, seria que os respectivos atributos poderiam ser nulos ou falsos, já que são controlados a partir da aplicação. No trabalho o problema foi resolvido: não sendo possível deixar os campos discriminatórios nulos, utilizando uma trigger para fazer com que um dos dois atributos discriminatórios seja verdadeiro - já que são booleanos e discriminatórios é proibido que os dois sejam falsos - que é o atributo de tipo aluno e além disso no trabalho em questão os usuários podem ser professor e aluno ao mesmo tempo

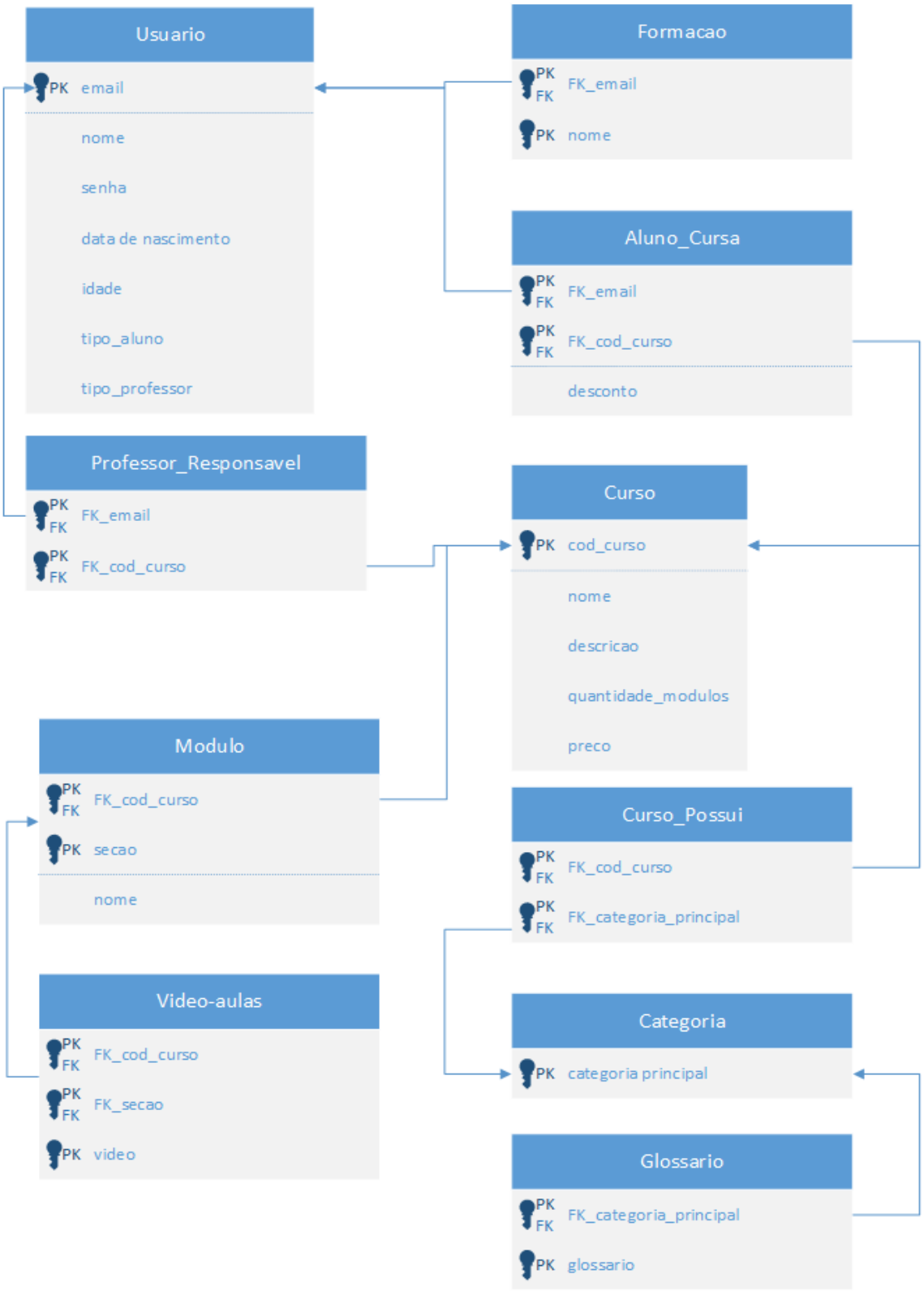

Figura 2: Projeto do modelo relacional

Fonte: Autoria própria

Usuario(email, nome, senha, data de nascimento, idade, tipo_aluno, tipo_professor)

Formação (FK_email, nome)

FK_email referencia Usuario

Curso(cod_curso, nome, descrição, quantidade_modulos, preço)

Aluno_Cursa(FK_email, FK_cod_curso, desconto)

FK_email referencia Usuario

FK_cod_curso referência Curso

Professor_Responsavel(FK_email, FK_cod_curso)

FK_email referencia Usuario

FK_cod_curso referência Curso

Modulo(FK_cod_curso, secao, nome, quantidade_modulos)

FK_cod_curso referência Curso

Video-aulas(FK_cod_curso, FK_secao, video)

{FK_cod_curso, seção} referência Modulo

Categoria(categoria principal)

Glossario(FK_categoria principal, glossario)

FK_categoria_principal referência Categoria

Curso_Possui(FK_cod_curso, FK_categoria_principal)

FK_cod_curso referência Curso

FK_categoria_principal referência Categoria

4 - SQL

Os scripts de criação do banco de dados se localizam no arquivo `ddl_grupo5.sql`, enquanto que os códigos de alimentação do banco de dados estão no arquivo `dados_grupo5.sql`, as consultas se encontram no arquivo `consultas_grupo5.sql` e as triggers se encontram no arquivo `trigger_grupo5.sql`.

Foi utilizada a cláusula DDL de “check” nas tabelas de Usuário, Aluno_Cursa e Curso. Que são usadas para verificar se a senha é igual a ou maior que 8 caracteres, verificar se o desconto está entre 0 e 1 e verificar se o preço do curso é maior que 0 respectivamente.

Sobre as tabelas foram usadas a cláusula de “delete cascade”, já que se viu necessário deletar em cascata as tabelas de relacionamento e a tabela de módulo e formação que tem uma dependência de existência, exceto na tabela Professor_Responsavel que utiliza “restrict” para que não seja possível deletar o professor sem deletar o curso, já que o curso não pode ficar sem responsável. E por último foram utilizadas as cláusulas “not null” na maioria dos atributos exceto no atributo desconto na tabela Aluno_Cursa, já que ele é opcional.

4.1 - Consultas

Nessa seção será realizada a descrição das seleções realizadas no banco.

Consulta 1: Quantos alunos cursaram cada curso? (Gabriel)

AR: $T1_{(cod\_curso,\ quantidade)} \leftarrow {}_{FK\_cod\_curso}\mathcal{F}_{COUNT(FK\_email)}(Aluno\_Cursa)$

$T2 \leftarrow \pi_{cod\_curso,\ nome}(Curso)$

$T1 \bowtie T2$

```
SQL: SELECT c1.cod_curso, c1.nome, count(c2.FK_email)
     FROM Curso c1 NATURAL JOIN Aluno_Cursa c2
     WHERE c1.cod_curso = c2.FK_cod_curso
     GROUP BY c1.cod_curso;
```

A consulta tem o objetivo de calcular o total de alunos inscritos em determinado curso, poderá ser usado de diversas formas, seja para ter um controle sobre o número de inscrições ou calcular o total valor gerado pelo curso.

Consulta 2: Quantos usuários tem na plataforma? (Renan)

AR: $T1_{(count\_usuarios)} \leftarrow \mathcal{F}_{COUNT(email)}(Usuario)$

```
SQL:  SELECT count(email)
      FROM Usuario;
```

O objetivo dessa consulta é obter o número total de usuários da plataforma, tal valor será usado para que seja possível uma melhor administração da plataforma.

Consulta 3: Listar os cursos de uma determinada categoria (Renan)

AR: $T1_{(cod\_curso,\ categoria)} \leftarrow \sigma_{FK\_categoria\_principal\ =\ <categoria\_principal>}(Curso\_Possui)$
$\pi_{nome}(T1 \bowtie Curso)$

```
SQL:  SELECT c.nome
      FROM Curso c NATURAL JOIN (
            SELECT FK_cod_curso AS cod_curso, FK_categoria_principal
            FROM Curso_Possui) AS p
      WHERE p.FK_categoria_principal = <categoria_principal>;
```

A responsabilidade dessa é auxiliar usuários a buscar pelo curso desejado dentro de uma categoria, como por exemplo, ele irá encontrar o curso de programação, realizando uma busca pelos cursos na categoria informática.

Consulta 4: Listar os professores que são alunos (Gabriel)

AR: $\sigma_{tipo\_professor\ =\ TRUE\ \wedge\ tipo\_aluno\ =\ TRUE}(Usuário)$

```
SQL:  SELECT *
      FROM Usuario
      WHERE tipo_professor = TRUE
            AND tipo_aluno = TRUE;
```

Essa consulta serve para recuperar uma lista de todos os professores que são alunos, essa pesquisa pode ser usada também para se obter uma ideia de quantos usuários são alunos e professores ao mesmo tempo.

Consulta 5: Listar os módulos de um curso (Renan)

AR: $\pi_{nome}(\sigma_{FK\_cod\_curso = <codigo\_do\_curso>}(Modulo))$

```
SQL:  SELECT nome
      FROM Modulo
      WHERE FK_cod_curso = <codigo_do_curso>;
```

A consulta lista os módulos de um curso para que o usuário decida se o conteúdo dos mesmo o interessa antes de comprar o curso, ou se já tiver comprado, selecionar qual módulo ele vai estudar a seguir.

Consulta 6: Pegar o conteúdo de um módulo de um curso específico (Gabriel)

AR: $T1_{(FK\_cod\_curso,\ FK\_seção,\ nome)} \leftarrow \sigma_{FK\_cod\_Curso = <cod\_Curso> \wedge seção = <seção>}(Módulo)$

$T1 \bowtie Vídeo\_aulas$

```
SQL:  SELECT v.FK_cod_curso, v.secao, m.nome, v.video
      FROM Modulo m
      NATURAL JOIN Video_aulas v(FK_cod_curso, secao, video)
      WHERE m.secao ILIKE 'Seção 2'
            AND m.FK_cod_curso = 3;
```

A consulta serve para resgatar o conteúdo do módulo que o usuário que está cursando o curso vai estudar a seguir.

Consulta 7: Pesquisar curso pelo nome do curso (Maurício)

AR: $\pi_{preço,\ descrição}\ (\sigma_{nome = <nome>}(curso))$

```
SQL:  SELECT c.preco, c.descricao
      FROM curso c
      WHERE nome ILIKE <nome>;
```

A pesquisa existe para que o usuário consiga buscar por um curso usando o nome, e resgatar as informações do preço e descrição do curso, para que o usuário possa saber do que se trata o curso e seu preço.

Consulta 8: Calcular o preço do curso com desconto (Maurício)

AR: $T1 \leftarrow \pi_{preço}(\sigma_{nome = <nome> \vee cod\_curso = <código>}(Curso))$

$\pi_{preço * <desconto>}(T1)$

```
SQL:  SELECT preco * <desconto> AS preco_desconto
      FROM curso
      WHERE nome ILIKE <nome>
            OR cod_curso = <codigo>;
```

A utilidade dessa consulta tem como objetivo que o usuário possa visualizar qual será o preço que ele irá pagar após a aplicação dos descontos disponíveis.

Consulta 9: Quantidade de vídeos por módulo de um curso (Gabriel)

AR: $\rho_{T1(FK\_cod\_curso,\ secao,\ video)}(Video\text{-}aulas)$

$T2 \leftarrow modulo \bowtie T1$

$T3 \leftarrow \sigma_{FK\_cod\_curso = <CodCurso>}(T2)$

$_{secao,\ nome}\mathcal{F}_{COUNT(video)}(T3)$

```
SQL:  SELECT m.secao, m.nome, count(v.video)
      FROM modulo m left outer JOIN Video_aulas v
            on(v.fk_cod_curso = m.fk_cod_curso
                  and v.fk_secao = m.secao)
      where m.FK_cod_curso = 1
      GROUP BY m.secao, m.nome
```

Consulta para verificar o número de aulas e em quais módulos essas aulas se localizam para um determinado curso.

4.2 Triggers

Trigger 1: Atualizar a quantidade de de módulos de um curso

```
CREATE OR REPLACE FUNCTION Atualiza_quantidade_modulos()
```

```
RETURNS trigger AS $trigger_atualiza_curso$
BEGIN
IF (TG_OP = 'DELETE') THEN
UPDATE curso SET quantidade_modulos = (select count(nome) from
modulo where
                                    FK_cod_curso =
old.fk_cod_curso)
WHERE cod_curso = OLD.fk_cod_curso;
ELSIF (TG_OP = 'INSERT') THEN
UPDATE curso SET quantidade_modulos = (select count(nome) from
modulo where
                                    FK_cod_curso =
new.fk_cod_curso)
WHERE cod_curso = NEW.fk_cod_curso;
END IF;
RETURN NULL;
END;
$trigger_atualiza_curso$ LANGUAGE plpgsql;


CREATE TRIGGER atualiza_trigger
AFTER INSERT OR DELETE ON modulo
FOR EACH ROW
EXECUTE PROCEDURE Atualiza_quantidade_modulos();
```

Trigger responsável por manter o número de módulos de cada curso sempre atualizado.

Trigger 2: Verificação da situação do usuário

```
CREATE OR REPLACE FUNCTION Atualiza_tipo_usuario()
RETURNS trigger AS $trigger_atualiza_usuario$
BEGIN
IF (TG_OP = 'UPDATE') THEN
UPDATE usuario SET tipo_aluno = true
WHERE email = new.email;
ELSIF (TG_OP = 'INSERT') THEN
UPDATE usuario SET tipo_aluno = true
WHERE email = new.email;
END IF;
RETURN NULL;
END;
$trigger_atualiza_usuario$ LANGUAGE plpgsql;
```

```
CREATE TRIGGER Verifica_Usuario
AFTER INSERT OR UPDATE
ON usuario
FOR EACH ROW
WHEN (new.tipo_professor = FALSE AND new.tipo_aluno = FALSE)
EXECUTE PROCEDURE Atualiza_tipo_usuario();
```

Trigger responsável por nunca deixar o usuário estar registrado na plataforma sem ser pelo menos aluno ou professor, se o mesmo tentar fazer isso o banco irá automaticamente mudar o tipo da conta para aluno.

Trigger 3: Atualizar idade do usuário

```
CREATE OR REPLACE FUNCTION Atualiza_idade_usuario()
RETURNS trigger AS $trigger_atualiza_idade$
BEGIN
UPDATE usuario SET idade = extract(YEAR FROM NOW()) -
extract(YEAR FROM NEW.data_nascimento)
WHERE email = new.email;
RETURN NULL;
END;
$trigger_atualiza_idade$ LANGUAGE plpgsql;


CREATE TRIGGER Atualiza_idade
AFTER INSERT OR UPDATE
ON usuario
FOR EACH ROW
WHEN (extract(YEAR FROM NOW()) - extract(YEAR FROM
new.data_nascimento) <> new.idade)
EXECUTE PROCEDURE Atualiza_idade_usuario()
```

Esse trigger é responsável por manter a idade do usuário sempre atualizada.

5 - Considerações finais

O projeto da plataforma online cursos teve como principal inspiração uma plataforma com grande atuação no mercado estudantil que é a Udemy. E além dela também houve inspiração do youtube, devido a ser uma plataforma de vídeos.

Sobre as dificuldades encontradas ao decorrer do desenvolvimento do trabalho foram: em ortografia, já que houveram vários ao decorrer das correções, muita refatoração dos diagramas - ou seja adicionando atributos e renomeando tabelas e refatoração das consultas que não estavam muito boas no início.

Já a respeito das limitações, o nosso projeto não calcula notas dos alunos, sendo que não pensamos em suportar questionários em meio aos módulos e não tem certificados salvos no nosso projeto para o aluno poder emiti-los.