U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Rede de Centros de Vacinação

Entrega III

Bases de Dados 2020/2021

Grupo 709

Adriano Soares **up201904873**@up.pt
Francisco Cerqueira **up201905337**@up.pt
Guilherme Calassi **up201800157**@up.pt

Índice

Índice **2**

Contexto **3**

Diagrama UML (Revisto) **4**

Esquema Relacional **5**

Dependências Funcionais e Formas Normais **6**

Restrições **6**

Interrogações **11**

Gatilhos **13**

Dificuldades Encontradas **14**

Contexto

Uma rede de centros de vacinação pretende armazenar informação relativamente aos mesmos, onde os enfermeiros irão vacinar um conjunto de utentes.

Ambos são **pessoas**, descritos pelo seu nome, idade, morada, código postal, telefone, email, data de nascimento, gênero, data em que foi registrado no sistema e ainda um identificador global que servirá de identificação no sistema.

Cada **centro** é caracterizado por um identificador único, uma morada, telefone, email e possivelmente um website.

Para fins estatísticos, uma **morada** deve conter informação sobre o distrito, concelho e código postal e cada pessoa deve estar associada a exatamente uma morada.

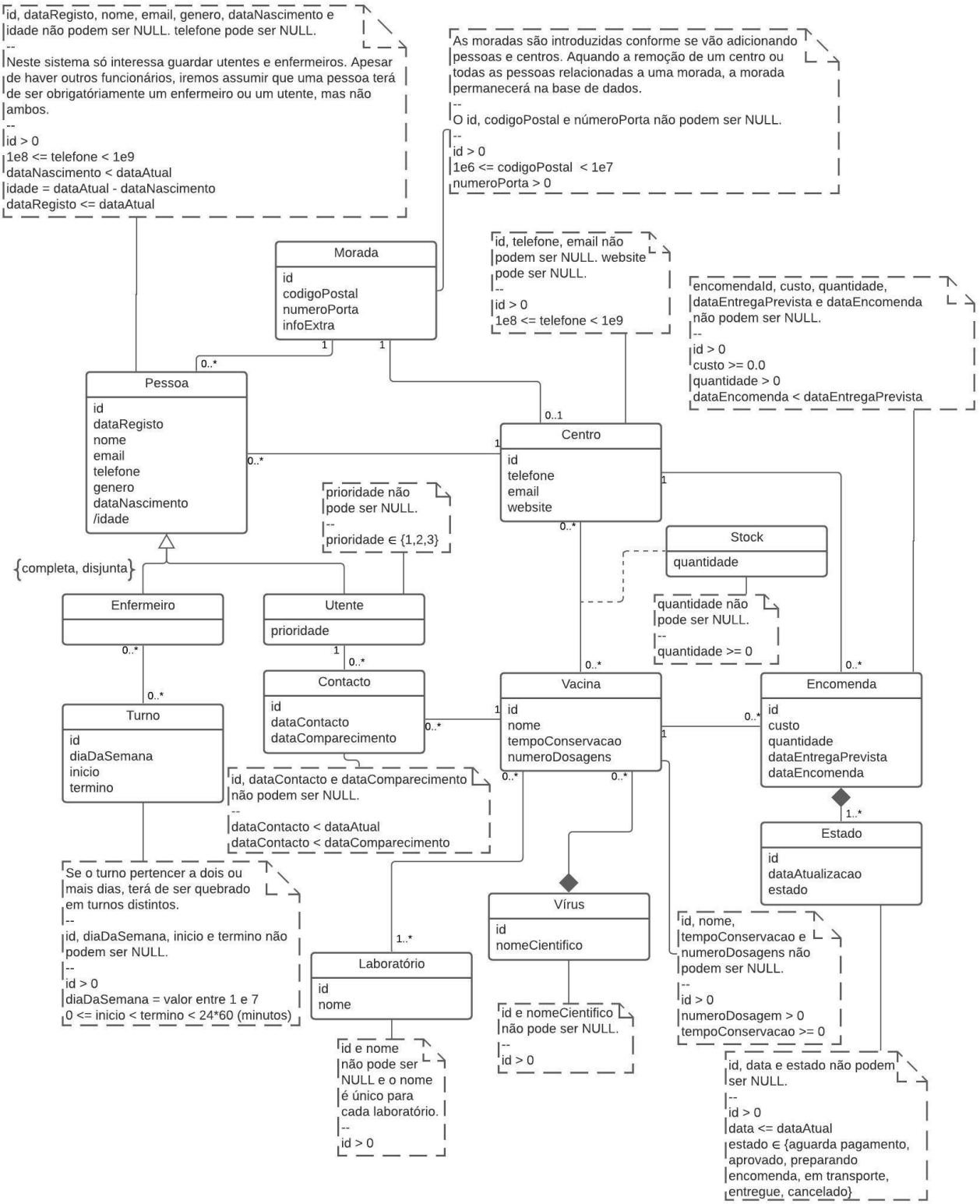
Os **utentes** poderão ser assinalados como prioritários, consoante a sua situação, e poderão ser **contactados** pelo centro, pretendendo-se guardar a respetiva data em que foi contactado e a data de comparecimento.

Os **enfermeiros** deverão estar associados a **turnos**, que são caracterizados por uma data de início, término e dia da semana, podendo excepcionalmente estar de baixa ou de férias, não tendo algum turno atribuído.

Acerca das **vacinas**, é necessário saber a data de validade, nome, modo de administração, **laboratório(s)**, número de dosagens e o **vírus** ao qual está associada, interessando apenas saber o nome científico do mesmo. Também será necessário guardar o **stock** de cada uma das vacinas em cada centro, para que este seja renovado quando necessário.

Finalmente, pretende-se guardar informações acerca das **encomendas** realizadas por cada centro, sendo necessário saber o respetivo o id, que dependerá do centro associado, custo, quantidade de cada tipo de vacina, o **estado** da encomenda, interessando guardar a data de cada estado pelo qual a encomenda passa, a data de entrega prevista e a data da encomenda.

Diagrama UML (Revisto)



Esquema Relacional

1. **Morada** (<u>id</u>, codigoPostal, numeroPorta, infoExtra)
    i. {id} -> {codigoPostal, numeroPorta, infoExtra}
2. **Centro** (<u>id</u>, telefone, email, website, moradaID -> Morada)
    i. {id} -> {telefone, email, website, moradaID}
    ii. {moradaID} -> {id, telefone, email, website}
3. **Enfermeiro** (<u>id</u>, dataRegisto, nome, email, telefone, genero, dataNascimento, idade, moradaID -> Morada, centroID -> Centro)
    i. {id} -> {dataRegisto, nome, email, telefone, genero, dataNascimento, idade, moradaID, centroID}
4. **Utente** (<u>id</u>, dataRegisto, nome, email, telefone, genero, dataNascimento, idade, moradaID -> Morada, prioridade, centroID -> Centro)
    i. {id} -> {dataRegisto, nome, email, telefone, genero, dataNascimento, idade, moradaID, centroID, prioridade}
5. **Turno** (<u>id</u>, diaDaSemana, inicio, termino)
    i. {id} -> {diaDaSemana, inicio, termino}
    ii. {diaDaSemana, inicio, termino} -> {id}
6. **Horario** (<u>enfermeiroID</u> -> Enfermeiro, <u>turnoID</u> -> Turno)
7. **Contacto** (<u>id</u>, dataContacto, dataComparecimento, utenteID -> Utente, vacinaID -> Vacina)
    i. {id} -> {dataContacto, dataComparecimento, utenteID, vacinaID}
8. **Virus** (<u>id</u>, nomeCientifico)
    i. {id} -> {nomeCientifico}
    ii. {nomeCientifico} -> {id}
9. **Laboratorio** (<u>id</u>, nome)
    i. {id} -> {nome}
    ii. {nome} -> {id}
10. **Vacina** (<u>id</u>, nome, dataValidade, numeroDosagens, virusID -> Virus)
    i. {id} -> {nome, dataValidade, numeroDosagens, virusID}
11. **Fabricante** (<u>vacinaID</u> -> Vacina, <u>laboratorioID</u> -> Laboratorio)
12. **Stock** (<u>vacinaID</u> -> Vacina, <u>centroID</u> -> Centro, quantidade)
    i. {vacinaID, centroID} -> {quantidade}
13. **Encomenda** (<u>id</u>, centroID -> Centro, vacinaID -> Vacina, custo, quantidade, dataEntregaPrevista, dataEncomenda)
    i. {id} -> {centroID, vacinaID, custo, quantidade, dataEntregaPrevista, dataEncomenda}
14. **Estado** (<u>id</u>, encomendaID -> Encomenda, estado, dataAtualização)
    i. {id} -> {encomendaID, estado, dataAtualização}

Nota: A estratégia escolhida para converter a generalização foi *object-oriented*, visto que é ideal para generalizações disjuntas.

Dependências Funcionais e Formas Normais

Todas as relações cumprem com os requisitos da Forma Normal de Boyce-Codd, na medida em que o lado esquerdo de cada dependência enumerada na página anterior é uma super-key da respetiva relação - condição suficiente, encontrando-se consequentemente também na 3ª Forma Normal.

Restrições

PK: Primary Key
UK: Unique Key
CK: Check
NN: Not Null

Morada

Não podem haver duas moradas com o mesmo id. Id é positivo, código postal tem que ter obrigatoriamente 7 dígitos e o número de porta tem que ser positivo. Todas as moradas devem ter um código postal, e número de porta associado, sendo a informação extra opcional.

id: ***PK, CHECK (id > 0)***
codigoPostal: ***NN, CHECK (1e6 <= codigoPostal && codigoPostal < 1e7)***
numeroPorta: ***NN, CHECK (numeroPorta > 0)***
infoExtra: ***--***

Centro

Não podem haver dois centros com o mesmo id nem com a mesma morada. Id é positivo e o número de telefone, se existente, terá 9 dígitos. Todos os centros devem ter um email e morada associados, sendo o website e o número de telefone opcionais.

id: ***PK, CHECK (id > 0)***
telefone: ***NN, CHECK (1e8 <= telefone && telefone < 1e9)***
email: ***NN***
moradaID: ***NN, UK, REFERENCES Morada***
website: ***--***

Enfermeiro

Não pode haver dois enfermeiros com o mesmo id. Id é positivo e o número de telefone, se existente, terá 9 dígitos. Todos os enfermeiros devem ter uma data de registo, nome, email, género, data de nascimento, morada e centro associados.

id: ***PK, CHECK (id > 0)***
dataRegisto: ***NN, CHECK (dataRegisto < NOW())***
nome: ***NN***
email: ***NN***
telefone: ***CHECK (1e8 <= telefone && telefone < 1e9)***
genero: ***NN***
dataNascimento: ***NN, CHECK (dataNascimento < NOW())***
moradaID: ***NN, REFERENCES Morada***
centroID: ***NN, REFERENCES Centro***

Utente

Não pode haver dois utentes com o mesmo id. Id é positivo, o número de telefone, se existente, terá 9 dígitos e a prioridade é um número entre 1 e 3. Todos os utentes devem ter uma data de registo, nome, email, género, data de nascimento, morada e centro associados.

id: ***PK, CHECK (id > 0)***
dataRegisto: ***NN, CHECK (dataRegisto < NOW())***
nome: ***NN***
email: ***NN***
telefone: ***CHECK (1e8 <= telefone && telefone < 1e9)***
genero: ***NN***
dataNascimento: ***NN, CHECK (dataNascimento < NOW())***
prioridade: ***NN, CHECK (prioridade > 0 && prioridade < 4)***
moradaID: ***NN, REFERENCES Morada***
centroID: ***NN, REFERENCES Centro***

Turno

Não podem haver dois turnos com o mesmo id. Id é positivo, o dia da semana está compreendido entre 1 e 7, a hora de início é superior a 0 e inferior à hora de término, sendo esta inferior a 24*60 (minutos). Todos os turnos devem ter um dia da semana, hora de início e término associado. Não podem haver dois turnos com as mesmas horas de início e término iguais.

id: ***PK, CHECK (id > 0)***
diaDaSemana: ***NN, CHECK (diaDaSemana > 0 && diaDaSemana <= 7)***
inicio: ***NN, CHECK (inicio >= 0)***
termino: ***NN, CHECK (termino < 24*60)***
CHECK (inicio < termino)
UNIQUE (diaDaSemana, inicio, termino)

Horário

Não podem haver dois horários com o mesmo enfermeiro e turno associado. Todos os horários devem ter um enfermeiro e turno associados.

enfermeiroID: ***REFERENCES Enfermeiro***
turnoID: ***REFERENCES Turno***
PK (enfermeiroID, turnoID)

Contacto

Não podem haver dois contactos com o mesmo id. Id é positivo, a data de contacto é inferior à data de comparecimento. Todos os contactos devem ter uma data de contacto, data de comparecimento, utente e vacina associados.

id: ***PK, CHECK (id > 0)***
dataContacto: ***NN, CHECK (dataContacto <= NOW())***
dataComparecimento: ***NN***
utenteID: ***NN, REFERENCES Utente***
vacinaID: ***NN, REFERENCES Vacina***
CHECK (dataContacto < dataComparecimento)

Virus

Não podem haver dois vírus com o mesmo id, positivo, nem com o mesmo nome científico.

id: ***PK, CHECK (id > 0)***
nomeCientifico: ***NN, UK***

Laboratorio

Não podem haver laboratórios com o mesmo id nem com o mesmo nome.

id: ***PK, CHECK (id > 0)***
nome: ***NN, UK***

Vacina

Não podem haver duas vacinas com o mesmo id. Id e o número de dosagens têm de ser positivos. Todas as vacinas devem ter um vírus, nome, data de validade e número de dosagens associado.

id: ***PK, CHECK (id > 0)***
virusID: ***NN, REFERENCES Virus***
nome: ***NN***
tempoConservacao: ***NN***
numeroDosagens: ***NN, CHECK (numeroDosagens > 0)***

Fabricante

Não podem haver dois fabricantes com a mesma vacina e laboratório associado. Todos os fabricantes devem ter uma vacina e laboratório associados.

vacinaID: ***REFERENCES Vacina***
laboratorioID: ***REFERENCES Laboratorio***
PK (vacinaID, laboratorioID)

Stock

Não podem haver dois registos com o mesmo par de vacinaID e centroID. A quantidade é um atributo obrigatório, não podendo ser negativa. VacinaID referencia Vacina, bem como centroID referencia Centro.

vacinaID: ***REFERENCES Vacina***
centroID: ***REFERENCES Centro***
quantidade: ***NN, CHECK (quantidade >= 0)***
PK (vacidaID, centroID)

Encomenda

Não podem haver duas encomendas com o mesmo id. O id é positivo, o custo é não negativo, a quantidade é superior ou igual a 1, e data de encomenda é inferior à data de entrega. CentroID referencia a Centro, assim como VacinaID referencia a Vacina. Nenhum atributo pode ser null.

id: ***PK, CHECK (id > 0)***
centroID: ***NN, REFERENCES Centro***
vacinaID: ***NN, REFERENCES Vacina***
custo: ***NN, CHECK (custo >= 0.0)***
quantidade: ***NN, CHECK (quantidade > 0)***
dataEntregaPrevista: ***NN***
dataEncomenda: ***NN***
CHECK (dataEncomenda < dataEntregaPrevista)

Estado

Não podem existir dois estados com o mesmo id. EncomendaID, data e estado são atributos obrigatórios.

id: ***PK, CHECK (id > 0)***
encomendaID: ***NN, REFERENCES Encomenda***
dataAtualização: ***NN, CHECK (data <= NOW())***
estado: ***NN, IN (“aguarda pagamento”, “aprovado”, “preparando encomenda”, “em transporte”, “entregue”, “cancelado”)***

Nota: Foram aplicados a todos os atributos que são chaves estrangeiras as restrições ON DELETE RESTRICT ON UPDATE CASCADE, à exceção das composições que têm ON DELETE CASCADE ON UPDATE CASCADE.

Interrogações

Apresentamos de seguida uma lista de interrogações, onde tentamos conciliar a sua diversidade, pertinência e complexidade.

1. Número de vacinas produzidas pelo laboratório '*APRLabs*':
   Verificação das interligações entre **Laboratório** e **Fabricante**, utilizando a diretiva **COUNT**. A estratégia baseou-se em determinar o número de tuplos onde o nome do laboratório é o desejado.

2. Número de pessoas por morada:
   Dado que a generalização **Pessoa/Utente/enfermeiro** é disjunta foi efetuada a diretiva **UNION ALL** entre as tabelas **Enfermeiro** e **Utente** com o atributo **moradaID**, determinando o conjunto de moradas de ambas as tabelas. Posteriormente utilizou-se a diretiva **COUNT** no produto cruzado entre **Morada** e o resultado da união anteriormente referida de modo a obter a contagem e as respectivas informações das diversas moradas. Utilizou-se também o operador **GROUP BY** de modo a agrupar o **COUNT** por morada.

3. Vacinas vencidas:
   Consulta das vacinas que estão vencidas. Verifica se o tempo de conservação da vacina foi ultrapassado desde o dia que ela foi entregue ao centro, através das tabelas **Vacina**, **Encomenda**, **Estado** e **Centro**. Para a comparação de datas foi utilizada a função **strftime('%s', ...)** (segundos desde 01/01/1970).

4. Média das vacinas em stock por centro:
   Utilizou-se o operador **GROUP BY**, de modo a agrupar os tuplos por centro e a diretiva **AVG**, que, em conjunto com o operador mencionado, calcula a média das quantidades para cada centro. Foi apenas necessário utilizar a tabela **Stock**.

5. Vacinas aplicadas:
   Verificação dos tuplos com datas de comparecimento anteriores à atual no produto cruzado das tabelas **Contacto**, **Utente**, **Vacina** e **Centro** (foram utilizadas mais tabelas para a exibição de informação relevante).

6. Encomendas entregues mais recentes (todos os centros):
   Foi utilizada a diretiva **ORDER BY** descendente por data de atualização no produto cruzado das tabelas **Estado**, **Encomenda**, **Vacina** e **Centro**, filtrando os tuplos em que a encomenda está entregue.

7. Vacinas mais próximas de vencer:
   Utilizou-se o operador **ORDER BY** por ordem ascendente de tempo até o final do prazo de validade da vacina no produto cruzado entre **Vacina**, **Encomenda** e **Estado**. Para a comparação de datas foi utilizada a função **strftime('%s', ...)** (segundos desde 01/01/1970).

8. Laboratórios que não produzem vacinas para todos os vírus:
   A estratégia baseou-se em percorrer a tabela **Laboratório** e verificar, com o auxílio do operador **NOT EXISTS**, que a diferença (**EXCEPT**) entre as tabelas **Virus** e os vírus para os quais as respectivas vacinas são efetivas resulta numa tabela vazia (são iguais). A segunda tabela é obtida através da junção condicional de **Fabricante** e **Vacina**.

9. Fabricante com vacinas mais vendidas e respetiva quantidade:
   Primeiramente, subentende-se que o fabricante com vacinas mais vendidas corresponde ao(s) fabricante(s) da vacina que mais encomendas tem registadas. Deste modo, utilizou-se várias sub-queries de modo a obter as interligações entre **Encomenda**, **Fabricante** e **Laboratório**.
   Inicialmente obteve-se a quantidade total de doses encomendadas para cada vacina, usando o operador **GROUP BY**. Posteriormente, selecionou-se a vacina com o maior número destas, através da diretiva **MAX**, obtendo-se também a quantidade. Finalmente, juntou-se o resultado obtido com as tabelas mencionadas acima, de modo a obter o nome do laboratório.

10. Utentes nunca vacinados:
    Utilização do operador **ORDER BY** para retornar os utentes que nunca foram vacinados em forma ordenada por prioridade e data de nascimento para desempate. Foi utilizado um **LEFT JOIN** entre as tabelas **Utente** e **Contacto**.

Gatilhos

Finalmente, foram definidos três gatilhos utilizados para a manutenção e monitorização da base de dados.

1. **AssertDisjointGeneralization**
   O objetivo deste gatilho é garantir que a generalização **Pessoa** é disjunta, não permitindo que um **Utente** também seja **Enfermeiro** e vice-versa. Para assegurar estas condições, seriam necessários dois gatilhos: um para impedir a inserção de um utente que já é enfermeiro e outro para impedir a inserção de um enfermeiro que é utente.
   Tal como é referido no enunciado, devemos implementar apenas um deles, deste modo, optamos por escolher o gatilho que garante que não é criado nenhum utente que já se encontra como enfermeiro na base de dados. A implementação do outro gatilho seria muito semelhante a este e sofreria poucas alterações.
   A estratégia utilizada baseia-se na verificação dos dados que são usados para considerar duas pessoas iguais (nome, email, telefone, gênero e data de nascimento) e, em caso afirmativo, um **levantamento de um erro** ocorre, impedindo a inserção.

2. **CreateDefaultStateForOrder**
   Este gatilho estabelece que cada encomenda introduzida na base de dados seja associada com um estado padrão "aguarda pagamento". Quando é inserida uma **Encomenda** o gatilho insere também um tuplo com o tempo atual e estado "aguarda pagamento" em **Estado**.

3. **AddStockWhenDeliveryArrives**
   Este gatilho tem como objetivo fazer **UPDATE** à tabela **Stock** caso seja inserido na tabela **Estado** um tuplo com o estado 'entregue' simbolizando a chegada de uma encomenda. Deste modo, quando isto ocorre é chamada a diretiva **UPDATE** na tabela **Stock** na linha respectiva ao centro e à vacina que é entregue. Nesta a quantidade em stock é somada à quantidade da encomenda (acesso à tabela **Encomenda**).

Nota: a verificação de gatilhos tem que ser feita posteriormente à criação e povoação da base de dados, isto é, após cada teste/verificação tem que ser feita novamente a criação de tabelas e respectiva povoação para não ocorrerem erros na tentativa de escrita de tuplos com id's repetidos. Os gatilhos devem ser ativos após a criação e povoação, verificados e desativados logo de seguida e proceder novamente com a criação e povoação dos dados.

Dificuldades Encontradas

Relativamente à verificação dos dados utilizando CHECK(), reparamos que recentemente (2021-03-17) foi implementada uma versão de SQLite que impede a utilização da função *strftime()* sem o *'now'* como argumento, pois corrigiu uma falha na deteção da mesma ser não-determinística. Mais informações em https://sqlite.org/deterministic.html (Ponto 3.1). Deste modo, optamos por utilizar a variável CURRENT_TIMESTAMP, de forma a contornar esse obstáculo.