**Introdução**

A base de dados preocupa-se apenas com a vertente do serviço ao domicílio e foi criada com o objetivo de armazenar todos os dados necessários para a realização deste serviço. Além disso, foi otimizada para garantir facilidade de uso e conformidade com as regras e restrições propostas, de forma a evitar o armazenamento de informações desnecessárias ou irrelevantes. Também considerámos a possibilidade de que alguns dados possam vir a ser necessários no futuro para fins estatísticos ou legais, como informações de pagamento, mas isso será esclarecido em mais detalhe mais à frente.

**Descrição da Clínica**

Vamos assumir o cenário de um cliente novo que precisa agendar um conjunto de vacinas trianuais para o seu gato. Este cliente deve entrar em contato com a clínica através de um telemóvel ou email e fornecer uma série de dados necessários. Para adicioná-lo à base de dados, é indispensável dispor de pelo mebase um método de contato futuro (email ou telemóvel), o seu nome, o seu número de Cartão de Cidadão (CC) e as informações de pelo menos um animal, pois um cliente só é relevante para a base de dados se tiver ao menos um animal associado. No futuro, será possível associar mais formas de contato ou animais ao mesmo cliente, mas, inicialmente, basta o mencionado.

Após o cliente e o animal estarem registados na base de dados, ele pode finalmente agendar um serviço para o seu domicílio. Para isso, será necessário associar uma localização (para os médicos saberem para onde se deslocar), uma data e hora, o animal em questão (apenas um animal por agendamento) e, por fim, o serviço a ser realizado, neste caso, vacinas trianuais.

Com todas essas informações definidas, podemos associar os médicos veterinários necessários ao agendamento. Um agendamento precisa ter pelo menos um médico veterinário associado para que faça sentido. O serviço fornecido pela clínica é bastante simples, por isso, todos os médicos são veterinários. A base de dados também garante que, ao associar um veterinário a um agendamento, este não tenha outro agendamento com horário sobreposto.

**Funcionamento da Base de Dados**

**Tabela "Serviço"**

Um serviço na base de dados é definido como um conjunto de vacinas ou consultas predefinidas. Essa decisão foi tomada para simplificar a reutilização de serviços comuns, como, por exemplo, o caso das vacinas trianuais. Estas consistem, essencialmente, em um conjunto de vacinas. Para evitar ter que adicionar todas as vacinas uma a uma em cada agendamento, criámos um *preset*. No entanto, caso nenhum dos *presets* existentes atenda à necessidade do cliente, é possível criar um novo serviço ou modificar os antigos.

**Pagamentos**

Por fim, é necessário compreender o processo de pagamento. Sempre que um agendamento é definido, é criada uma tabela à parte para o pagamento associado a esse agendamento. Esse pagamento ainda não é efetuado, pois, na clínica, o cliente realiza o pagamento apenas após o serviço ser realizado. Durante esse período, o cliente pode alterar o serviço ou a localização, o que implica na atualização automática do preço até a hora do serviço. Quando o pagamento é finalizado, ele não pode mais ser alterado. Mesmo que o agendamento seja posteriormente eliminado da base de dados, os dados referentes ao pagamento são mantidos por motivos legais, para garantir o histórico de todas as transações realizadas na clínica.

Com as informações acima, temos uma ideia superficial do funcionamento da clínica e do seu sistema de base de dados. Nos próximos passos, resumiremos todas as restrições originadas desta descrição e como lidamos com elas no sistema.

**Restrições e Implementações**

1. O cliente precisa de ter pelo menos um número de telemóvel ou email associado.
2. Um cliente precisa de ter pelo menos um animal inicialmente para ser relevante.
3. Um agendamento precisa de ter pelo menos um veterinário.
4. Um serviço precisa de ter pelo menos uma consulta ou vacina.
5. Um veterinário não pode ter horários sobrepostos.
6. A taxa de deslocamento é uma função que depende da distância do domicílio até à clínica.
7. O valor total do serviço é a soma do valor de todas as vacinas e consultas associadas.
8. O valor total do pagamento é a soma do valor total do serviço e da taxa de deslocamento.
9. Um animal não pode ter dois agendamentos em horários sobrepostos.

A 1ª e a 2ª restrição são garantidas utilizando a função definirNovoCliente. Em vez de utilizarmos manualmente *Insert Into Pessoa* e depois *Insert Into Cliente*, *Insert Into Animal*, *Insert Into Telemóvel* e *Insert Into Email*, podemos fazer tudo isso com uma única função. Esta verifica se nos seus parâmetros foram mencionados pelo menos um telemóvel ou email e um animal e a sua espécie, garantindo as duas restrições. Essa função não só garante as restrições mencionadas, como também facilita muito a sintaxe para o pessoal que vai futuramente mexer nesta base de dados.

A 1ª restrição também e reforçada por um trigger validarRemocaoContactos que corre sempre que haja algum delete nas tabelas do telemóvel ou email que apenas verifica se por acaso não estamos a remover o último contacto dum cliente, porque sempre convém termos alguma maneira de contactar os clientes.

A 3ª, a 4ª e a 9ª restrição são garantidas a partir da *stored procedure* definirAgendamento. Esta *procedure* é o coração da base de dados. Requer todos os dados necessários para definir o agendamento em si, mas também requer um veterinário inicial e depois precisa de um serviço. Verifica também logo se o animal associado já tem algum agendamento com um horário sobreposto se tiver manda o utilizador repensar as suas escolhas na vida. Se já tivermos o *preset* do serviço necessário criado, basta apenas passarmos o id do serviço existente como parâmetro. Mas, se o serviço não existir ainda, temos de passar o nome do serviço e depois uma consulta, vacina ou ambos, e os seus valores respetivos. Depois disso, são criados o agendamento, o serviço, a consulta e a vacina de uma vez, evitando, mais uma vez, uma série desnecessária de *Inserts Into*.

A 5ª restrição é garantida pelo definirAgendamento também, porque, quando criamos o agendamento com um veterinário inicial, verificamos a disponibilidade dele através de uma outra função, "validarSobreposicao". Esta restrição também é garantida pela função AssociarVeterinarioAoAgendamento. Mais uma vez, em vez de permitir associar veterinários manualmente ao agendamento, fazemos isso através da função, que não só faz o *insert* necessário como verifica se o veterinário em questão tem disponibilidade ou não.

A 6ª restrição é garantida através de um *trigger* atualizarTaxaDeslocamento, que é chamado antes de qualquer *insert* ou *update* na tabela localização. O cliente pode sempre mudar o endereço e, consequentemente, a distância, daí atualizamos a taxa de deslocamento. A fórmula de cálculo é simples: distância \* 2 se a distância for menor que 1 km, distância \* 3 se for menor que 5 km e distância \* 4 nos restantes casos.

A 7ª restrição é garantida através de um outro *trigger*, atualizarValorTotalServico. Este passa por todas as consultas e vacinas associadas e depois soma tudo logo após qualquer *insert*, *update* ou *delete* nas tabelas serviço\_consulta ou serviço\_vacina.

A 8ª restrição é garantida através de um conjunto de *triggers* e funções: atualizarPagamentoTotalPeloServico, atualizarPagamentoTotalPelaLocalizacao e atualizarPagamentoPorRefLocal. Estas três funções são chamadas sempre que haja algum *update* no serviço, na localização ou quando trocamos o ref\_local no agendamento, respetivamente. As três têm a mesma ideia de somar os valores da tabela serviço e da tabela localização. Contudo, como podem haver vários agendamentos para o mesmo serviço e mais algumas complicações técnicas, decidiu-se que o código ficaria muito mais percetível e compreensível se a implementação fosse separada em duas funções semelhantes, mas que começam a sua procura de tabelas de partida diferentes.

Por fim, temos ainda algumas funções que não são indispensáveis para a base de dados e foram criadas apenas por questões de estética e simplicidade de leitura e utilização futura. Estas funções têm funcionalidades bastante óbvias e simples:

* A função criarVeterinario serve para evitar fazer *INSERT INTO pessoa* e depois *INSERT INTO veterenario*.
* A função atualizarPagamento serve para facilitar a finalização do pagamento e faz apenas um *UPDATE* com os valores fornecidos, melhorando a sintaxe.
* As funções adicionarConsultasAoServico e adicionarVacinaAoServico fazem apenas *inserts* nas tabelas serviço\_consulta e serviço\_vacina, respetivamente, e, mais uma vez, servem apenas a um propósito estético.

Nota: Estas não foram todas as restrições de integridade, apenas aquelas que tivemos que resolver utilizando funções, stored procedures ou triggers. O resto das restrições são mencionadas no modelo relacional.

Ainda temos duas vistas, a infoParaVeterinarios e a infoParaClientes estas vistas tem um propósito autoexplicativo pelo seu nome. Mas se formos a detalhe. A primeira serve para veterinários juniors que ainda não são muito de confiança ou que por alguma outra razão não convem terem acesso a toda informação na base de dados. Dai esta vista dá-lhes toda a informação necessária para realizar o serviço sem informações privadas como os dados do cliente ou então dados do pagamento. Apenas disponibiliza a hora em que precisam de trabalhar, onde trabalhar, com quem trabalhar e o que fazer.

Depois a segunda vista e feita para a informação disponível para os clientes que podemos disponibilizar para o cliente a qualquer altura por exemplo na área cliente do mesmo se tivermos um website ou algo do género. Esta informação consiste em dados do próprio cliente como o nr cliente, o seu nome o nome dos seus animais e os agendamentos que tem com os seus respetivos horários e preços e o serviço prestado.

