1 2  9 0 FACULDADE DE CIÊNCIAS E TECNOLOGIA UNIVERSIDADE D COIMBRA

Sistema de gestão de um hospital

Projeto no âmbito da disciplina Bases de Dados

- Entrega intercalar -

Licenciatura em engenharia informática - 2º ano - 2023/2024

Desenvolvido por:
Miguel Curto Castela - uc2022212972 - micas.castela@gmail.com
Francisco Lapa Silva - uc2022213583 - francisco.lapamsilva@gmail.com
Débora Alves - uc2022213498 - debora.alves.004@gmail.com

Conteúdo

Plano de Desenvolvimento (após a meta 1) 3
Introdução 4
Descrição de entidades, relações, atributos e regras de integridade 5
Principais decisões do E-R 10
Transições do modelo relacional para o físico 11
Funções não representadas no ER 14
Transações Relevantes 15
Potenciais conflitos e como evitar 17
Casos específicos de problemas de concorrência: 17
1- Ao criar um novo utilizador 17
2- Ao marcar uma consulta ou cirurgia 17
3- Adicionar prescrição 18
4- Lista de top 3 pacientes do mês e relatório mensal de cirurgias 18

Plano de Desenvolvimento (após a meta 1)

Semana 1 (25 março - 31 março):
- Fazer Tabelas. ->TODOS
- Geração de dados iniciais ->TODOS

Semana 2 (1 abril - 7 abril):
- Fazer API:
  - Receber e tratar Requests. ->Francisco
  - User Application. ->Miguel
  - User Authentication. ->Débora

Semana 3 (8 abril - 14 abril):
- Fazer API:
  - Schedule Appointment. ->Miguel
  - See Appointment. ->Débora
  - Schedule Surgery. ->Débora
  - Get Prescriptions. ->Miguel
  - Add Prescriptions. ->Francisco

Semana 4 (15 abril - 21 abril):
- Reunião intermédia. -> TODOS
- Tempo para imprevistos e reposições. -> TODOS

Semana 5 (22 abril - 28 abril):
- Fazer API:
  - Execute Payment. ->Miguel
  - List Top 3 Patients. ->Francisco

Semana 6 (30 abril - 5 maio):
- Fazer API:
  - Daily Summary. -> Miguel
  - Generate monthly report. ->Débora

Semana 7 (6 maio - 12 maio):
- Escrever manual de instalação. -> Miguel
- Escrever manual de utilizador. ->Débora

Semana 8 (13 maio -19 maio):
- Rever manuais. -> TODOS
- Finalização do trabalho. -> TODOS
- Tempo para imprevistos e reposições. -> TODOS

Semana 9 (20 maio - 23 maio) :
- Finalização e entrega.

Introdução

Este trabalho tem como objetivo criar um sistema de gestão da informação de um hospital, através de uma base de dados, onde será armazenada toda a informação relevante para tal. Vai ser possível gerir a informação do hospital (criar, consultar, atualizar e apagar) através de uma aplicação e *web server*, que será criada com a ajuda de um *REST API* que garante a ligação à base de dados. Servirá também de plataforma para permitir aos utilizadores a marcação de consultas, assim como permitirá consultar todas as suas prescrições médicas e as informações relacionadas. Do ponto de vista do *staff* do hospital, a aplicação permitirá associar os médicos e enfermeiras às cirurgias, consultas ou hospitalizações, para além de permitir a consulta às prescrições associadas a certo paciente. Estes serviços médicos irão todos ser registados no sistema, assim como as faturas associadas a estes.
Primeiramente, foi desenvolvido um diagrama E-R para modelar conceptualmente o sistema, através da definição de entidades e relações entre elas.
Com isto concluímos que, ao utilizar a aplicação, um Paciente pode agendar uma consulta, consultar as suas consultas, consultar as suas prescrições e consultar e pagar a sua fatura. Um Assistente poderá consultar as consultas de um paciente, assim como associar um médico e enfermeiras a certa cirurgia, consulta ou hospitalização. Poderá também consultar os 3 pacientes com maior despesa no hospital e consultar o histórico geral do funcionamento do hospital e da prática de cada médico ou enfermeira e os detalhes deste. Um médico pode adicionar prescrições a uma hospitalização ou consulta de um cliente, ou apenas consultá-las. um enfermeiro poderá também consultar as marcações.

Descrição de entidades, relações, atributos e regras de integridade

User

User define o utilizador que vai usar o sistema. Este pode ser um **Patient** ou um **Employee** (definido através de herança)

- **_user_CC [primary key]**: int - not null – unique - tem de ter 9 dígitos
- **Birthday**: Date - not null - não pode aceitar uma data superior à atual
- **Nationality:** varchar - not null - nacionalidade existente
- **Name**: varchar - not null - apenas letras, acentos e espaços
- **Password**: varchar - not null - unique - não pode ter caracteres especiais.
- **Mail**: varchar - not null - unique - tem de ser um tipo de email válido (ex: @gmail, @hotmail, @outlook)

Patient e **Employee** são tipos de **user** (relação de herança) que herdam as suas propriedades (atributos, incluindo a primary key), o que permite aceder a **user** através de **employee** e de **patient:**

Patient

Patient define o **user** paciente, este pode marcar um **appointment** ou ser submetido a uma **hospitalization**.

Employee

Esta entidade é de onde descendem todos os empregados que trabalham no hospital: **Doctor**, **Nurse** ou **Assistant** (relação de herança para patient), logo herdam as suas propriedades, permitindo a associação. Estes têm os seus contratos (**contract**) associados.

shift_start: Timestamp – not null – tem de ser antes de shift_end
shift_end: Timestamp - not null – tem de ser depois de shift_start

Contract

Um **employee** pode ter vários **contract** ou não. Estes estão ligados a employee através da sua primary key mas também têm o seu próprio ID_contract.
Foi escolhida uma relação de 1 … 1 **employees** para 1 … n **contracts** porque se assume que a base de dados irá guardar todos os contratos de um **employee**, quer estejam ativos ou não (e um contrato tem sempre associado um **employee**), daí ser necessários os atributos:

- **ID_contract [primary key]**: Bigint - not null - unique
- **Start_date**: date - not null - antes da data de fim
- **End_date**: date - not null - após a data de início

Doctor

Representa os médicos, que vão estar associados à sua **specialization.** Assim como podem ser responsáveis por um **appointment** ou fazer uma **surgery.**

- **Medical_license_ID**: varchar - not null - unique
- **University:** varchar - not null - tem de ser uma universidade existente, apenas letras, acentos e espaços.

Specialization

É-nos especificado que cada médico pode ter apenas uma **specialization** (então a ligação é feita 0 … n **doctor** por cada 1 **specialization**, assim define-se que cada doutor tem apenas um especialização, mas varias especializações podem estar associadas a diferentes doutores) e que existe uma relação hierárquica entre elas. Por consequência, tomámos a decisão de ter uma entidade que pudesse guardar e representar as várias especializações existentes e a sua hierarquia, para isto ligamos **specialization** a ela própria (criando uma segunda tabela que funciona como uma árvore, associando as especializações em níveis de hierarquia, com especializações que podem estrar e “cima” ou em “baixo). Esta ligação foi então feita de 0 … 1 para 0 … n.

- **Expertise [primary key]**: varchar - not null - unique - apenas letras, acentos e espaços

Nurse

Estas estão ligadas à sua categoria e têm uma relação 1…1 para 0…n **hospitalization** porque uma **nurse** pode ser responsável por várias hospitalizações, mas cada hospitalização só tem uma **nurse** responsável.

Category

Categoria de cada **nurse**, que são organizadas hierarquicamente. Estas têm uma relação 0 … n para 1 ... 1 categoria porque uma **nurse** pode apenas ter uma categoria, mas cada categoria na lista pode ter 0 a N **nurses** associadas. Da mesma maneira que foi explicado em Specialization, esta tabela está ligada a ela mesma para definir a hierarquia.

- **Category name**: varchar - not null - unique - apenas letras, acentos e espaços

Assistant

Define as assistentes que vão ter certos controlos sobre o funcionamento do hospital, como mencionado, fazem parte da "categoria" **employee** (herança).

Role

Esta tabela representa a lista de roles que uma **nurse** pode ter. Estas **roles** vão ser os papéis que ela executa em determinada **surgery** ou **appointment.**

- **Role_id [primary key]**: Bigint – not null – unique
- **Role_name**: varchar - not null - unique - apenas letras, acentos e espaços

Enrollment_surgery

A role que determinada **nurse** vai exercer numa **surgery.** Uma **nurse** só pode ter uma **role**, mas uma **role** pode estar associada a várias nurses**.**
Enrollment surgery é weak para com **role**. É também para **nurse** e **surgery** (só existe e pode ser identificada com recurso aos atributos destas, como as primary keys de **role**, **surgery** e **nurse**), o que permite ligar uma **nurse** com uma certa **role** (vinda da lista de roles em **role**) a uma surgery. Todas estas ligações são de 1 para 0 … n pois **nurse** pode ter diferentes **roles** para cirurgias diferentes. O mesmo acontece para **Enrollment appointment**

Enrolment_appointment

A role que determinada **nurse** ou **nurses** vão exercer num **appointment.** Weak para **Nurse**, **Role** e **appointment** pois só existe e pode ser identificada usando os atributos destas, nomeadamente as Primary keys de **role**, **appointment** e **nurse**.

Appointment

Define as consultas que vão estar associadas ao **patient** que a marcou e ao **doctor** responsável. Ambas 1 para 0 … n pois um **appointment** tem obrigatoriamente um único doutor e paciente, mas estes podem estar associados a diferentes **appointments**. Vai estar também associada à **bill**, que vai ter o fixed cost estipulado para as consultas.

- **Appointment_ID [primary key]:** Bigint - not null - unique
- **Appointment_date**: timestamp - not null – após a data atual

Type

Vai definir que tipo de **appointment** uma consulta é. Foi decidido fazer uma tabela com os tipos possíveis de consulta quando um paciente a vai marcar. Tem uma ligação 1 para 0… n **appointment** pois varios appointments podem ter o mesmo tipo, mas um **appointment** tem sempre obrigatoriamente um tipo.

- **Type [primary key]:** varchar - not null - unique - apenas letras, acentos e espaços

Hospitalization

Está associada com o **patient**, com uma relação 1 para 0...n (isto porque um **patient** pode ter associadas 0 ou mais **hospitalizations)**. Da mesma maneira está também associada com a **nurse** responsável, isto porque uma hospitalização só tem uma **nurse** responsável, mas uma **nurse** pode estar responsável por 0 a N hospitalizações. Está associada às **surgeries** que pode "causar" (1 hospitalização pode causar 0 a N **surgeries**). Como o **appointment,** vai estar também associada à **bill**, que vai ter o fixed cost estipulado para a **Hospitalization** (tendo em conta as cirurgias).

- **Hosptitalization_ID [primary key]**: Bigint - not null - unique varchar
- **Room**: int - not null
- **Type**: varchar - not null - apenas letras, acentos e espaços
- **Start_date**: datetime - not null - antes da data final
- **End_date**: datetime - not null - após a data de início

Surgery

É sempre associada a uma hospitalização, pois sempre que é criada uma nova surgery sem estar associada a uma hospitalização, é criada uma hospitalização, que tem **nurses** com **roles** (presentes através de **enrolement_surgery**) assim como um **doctor** que faz a cirurgia (1 para 0 … n, pois um doutor pode ser estar associado a 0 a N cirurgias, que têm um doutor). Unicamente identificável através de ID_surgery.

- **ID_surgery [primary key]: Bint - not null - unique**
- **Type: varchar - not null - apenas letras, acentos e espaços**
- **Surgery_date: timestamp - not null**
- **Surgery_duration: float - not null - formato horas, minutos e segundos**

Billing

Como referido anteriormente está associada a **hospitalization** e **appointment**, que têm fixed cost. **Billing** tem uma relação 1 para 1 com **hospitalização** e **appointment**, pois cada hospitalização e consulta que tem preço físico tem uma bill associada**.**

- **Bill_ID [primary key]**: Bigint - not null - unique
- **Cost**: number (2 casas decimais) - not null

Payment

Só existe se existir **bill** a que esteja associado (weak pois só é possível identificar payment com a primary key de billing, usada como foreign key). É usado para dividir o preço de uma bill em prestações, daí a relação 1 para N.

- **Num_payment [primary key]: not null - unique - auto increment**
- **Amount: int - not null - tem de ser limitado a 2 casas decimais(centimos)**

Prescription

Cada **appointment** e **hospitalization** podem ter múltiplas **prescriptions** associadas (ambos 0…1 para 0…n prescriptions).

- **ID_prescription [primary key]**: varchar - not null – unique

Quantity

Só existe se existir **prescription** e **medication** associado (é weak para estas pois só existe se houver um medicamento cuja quantidade é associada e uma prescription em que se insere, e é só identificável através das PK ID_prescription do ID_medication). Define a quantidade de certo medicamento que a **prescription** tem, 1 para 0..n **prescriptions e medicamentos**, pois pode ter muitos medicamentos com diferentes quantidades.

- **Medication_ammount: int - not null**

Medication

Tem as propriedades dos medicamentos presentes em certa **prescription.**

- **Medication_ID [primary key]**: Bigint - not null - unique
- **Medication_name**: varchar - not null - apenas letras, acentos e espaços

effect_property

Define as propriedades dos side effects de certa **medication**. Só existe se existir uma **medication** e se esta tiver **side effects** associados com diferentes níveis de severidade e probabilidades de ocorrência (é weak em relação a estas pois só é identificável através das PK ID_medication e side_effect_ID). Definem a severidade e a probabilidade de ocorrência da entidade side_effect à qual está associada**.** Tem uma relação 1 para 0…n com ambas, pois pode estar presente em 0 a N medicamentos com 0 a N side effects

- **occurrences:** float - not null
- **severity:** float - not null

Side_Effect

Define os sintomas de um **side Effect** uma certa **medication** (uma medicação pode ter múltiplos **side effect**). Identifica estes side effects através do side_effect_ID.

- **Side_effect_ID [primary key]**: Bigint - not null - unique
- **Symptoms**: varchar - not null - unique - apenas letras, acentos e espaços

Principais decisões do E-R

Muitos contratos associados a cada paciente

Esta característica foi teorizada com o objetivo de associar todos os contratos (ativo e os não ativos) a cada trabalhador do hospital. Assim a base de dados armazenerá e terá disponível todos os contratos associados a certo employee, podendo ser acedidos pelo próprio.

Criação da entidade user

Como as entidades employee e patient tinham muitos atributos em comum, foi decidido criar a entidade user que os agrupa. Assim employee e patient herdam os atributos comuns de user.

Criação de role, enrollment surgery e enrollment appointment que conectam a nurse

Para associar uma enfermeira a uma cirurgia ou consulta médica com um papel específico, existem relações entre as entidades. As enfermeiras estão vinculadas às inscrições em cirurgias e consultas médicas, que, por sua vez, estão conectadas aos papéis (que contêm todas as possíveis funções de enfermeira) que podem despois associar-se às cirurgias e consultas médicas. Por exemplo, se uma enfermeira N desempenhar o papel de anestesista em uma cirurgia S, então na tabela de inscrição na cirurgia, haverá uma referência para N, outra para S e outra para a entrada da sua role na tabela de roles.

Hierarquia das categorias das nurses e das especializações dos médicos

Como foi pedido, nós definimos a especialização médica e as categorias de nurses hierarquicamente. Para isto fizemos uma relação nas categorias para elas próprias (como referido anteriormente, de 0…1 para 0…N) isto foi para simular uma hierarquia tipo “arvore” que representa as categorias ou as especializações que estão em cima ou em baixo. Pode assim ter várias classificações inferiores e/ou superiores. Nurse está ligado 0…n para a sua categoria 1…1 pois uma **nurse** está sempre associada a uma hierarquia (associada a sua categoria), mas varias **nurse** podem ter a mesma categorias. Quanto à ligação de medico à sua especialização 0…n para 1…n especializações, deste modo, medico tem de ter pelo menos uma especialização (esta, do mesmo modo, define o seu ranking de acordo com o resto delas).

Criação de type em appointment

Decidimos criar uma lista de **Type**, que liga a **appointments**, com ligação 0…n para 1 **appointment**. Isto faz com que **appointment** tenha uma lista definida de possíveis tipos. Isto faz sentido porque, enquanto uma hospitalização ou cirurgia é uma **assistant** a registar e pode estar relacionada com vários problemas do doente, uma **appointment** é o paciente que marcou, logo faz sentido haver uma triagem de tipos definidos para cada marcação

Criação de effect properties e quantity

A criação destas duas tabelas é necessária porque cada medicação de uma prescrição pode ter 1 ou mais **side effects** de uma lista. E a mesma medicação em diferentes prescrições pode ter dosagens diferentes.

Relações que são weak

Como diz a definição, tratamos as entidades que não podem ser unicamente identificadas com apenas os seus atributos (precisando assim de uma foreign key, derivada de uma primary key de uma entidade relacionada, em conjunto com os serus atributos para formar a PK). Isto acontece, como referido com quantity, effect_properties e payment (primary keys e foreign keys definifas mais a frente), assim como role, enrollment_surgery e enrollment_appointment.

Atributo shift_start e shift_end para employee

Para uma gestão da base de dados mais eficiente foi teorizado os **employees** terem um início e fim definido para os seus turnos. Isto vai facilitar a triagem por por parte do sistema ou das assistentes para decidir que medico ou enfermeira fica associado a que marcação. cirurgia ou hospitalização.

Transições do modelo relacional para o físico

Ao passar do modelo conceptual para o modelo físico, verifica-se a adição de alguns atributos em algumas tabelas, a junção de tabelas, e a criação de tabelas intermédias. Isto acontece, pois, estes elementos adicionais são necessários para criar e utilizar uma Base de Dados real, que consiga armazenar os dados e realizar as ações que pretendemos.

Patient:

- user_id_user_cc(int): A tabela “**Patient**” herda o atributo id_user_cc da tabela “**User**” como foreign key. Deste modo é possível correlacionar um paciente na tabela “**Patient**” com um utilizador na tabela “**User**”.

Employee:

- user_id_user_cc(int): A tabela “**Employee**” herda o atributo id_user_cc da tabela “**User**” como foreign key. Deste modo é possível correlacionar um funcionário na tabela “**Employee**” com um utilizador na tabela “**User**”.

Contract:

- employee_user_id_user_cc(int): A tabela “**Contract**” herda o atributo user_id_user_cc da tabela “**Employee**” como foreign key. Deste modo é possível correlacionar um contrato na tabela “**Contract**” com um empregado na tabela “**Employee**” e um utilizador na tabela “**User**”. O empregado e utilizador referenciados através do employee_user_id_user_cc correspondem à pessoa com quem o hospital tem o contrato.

Doctor:

- employee_user_id_user_cc(int): A tabela “**Doctor**” herda o atributo user_id_user_cc da tabela “**Employee**” como foreign key. Deste modo é possível correlacionar um doutor na tabela “**Doctor**” com um empregado na tabela “**Employee**” e um utilizador na tabela “**User**”.
    - Specialization_specialization: esta tabela é criada para definir a hierarquia das especializações

Assistant:

- employee_user_id_user_cc(int): A tabela “**Assistant**” herda o atributo user_id_user_cc da tabela “**Employee**” como foreign key. Deste modo é possível correlacionar um assistente na tabela “**Assistant**” com um empregado na tabela “**Employee**” e um utilizador na tabela “**User**”.

Nurse**:**

- employee_user_id_user_cc(int): A tabela “**Assistant**” herda o atributo user_id_user_cc da tabela “**Employee**” como foreign key. Deste modo é possível correlacionar um assistente na tabela “**Assistant**” com um empregado na tabela “Employee” e um utilizador na tabela “**User**”.

Enrolment_surgery:

- nurse_employee_user_id_user_cc(int): Atributo herdado employee_user_id_cc da tabela “**Nurse**” como foreign key. Permite relacionar o enfermeiro de chave primária employee_user_id_cc com um role numa cirurgia expecífica.
- role_role_id(Bint): Herdado atributo role_id da tabela “**Role**” como foreign key. Deste modo, correlaciona-se uma tarefa de um enfermeiro específico numa cirurgia específica com um tipo de tarefa.

- surgery_id_surgery(Bint): Atributo herdado id_surgery da tabela “**Surgery**” como foreign key. Permite relacionar uma cirurgia com a tarefa que um enfermeiro específico desempenha em dita cirurgia.
- surgery_hospitalization_hospitalization_id(Bint): Atributo herdado hospitalization_hospitalization_id, da tabela “**Surgery**”. Permite correlacionar a hospitalização que uma dada cirurgia requer e a tarefa de um enfermeiro nessa cirurgia e hospitalização.

Category: tem Category_name como primary key

Category_Category:

Enrolment_appointment:

- nurse_employee_user_id_user_cc(int): Atributo herdado employee_user_id_cc da tabela “**Nurse**” como foreign key. Permite relacionar o enfermeiro de chave primária employee_user_id_cc com uma tarefa numa dada consulta.
- role_role_id(Bint): Herdado atributo role_id da tabela “**Role**” como foreign key. Deste modo, correlaciona-se uma tarefa de um enfermeiro específico numa consulta específica com um tipo de tarefa.
- surgery_id_appointment(Bint): Atributo herdado id_appointment da tabela “**Appointment**” como foreign key. Permite relacionar uma consulta com a tarefa que um enfermeiro específico desempenha em dita consulta.

Surgery:

- hospitalization_hospitalization_id(Bint): Atributo herdado hospitalization_id da tabela “**Hospitalization**” como foreign key. Permite relacionar uma hospitalização com a cirurgia referida.
- doctor_employee_user_id_cc(int): Atributo herdado employee_user_id_cc da tabela “**Doctor**”, referência o doutor que será responsável pela cirurgia.

Hospitalization:

- billing_bill_id(Bint): Atributo herdado bill_id da tabela “**Billing**”, usado como foreign key. Utilizado para correlacionar a hospitalização à bill correta.
- nurse_employee_user_id_cc(int): Atributo herdado employee_user_id_cc da tabela “**Employee**” como foreign key. Necessário para relacionar o enfermeiro responsável pela hospitalização.
- patient_user_id_cc(int): Atributo herdado user_id_cc da tabela “**Patient**” como foreign key. Indica o paciente que está hospitalizado.

Prescription_Hospitalization:

Tabela intermédia criada pelo modelo físico. Permite correlacionar uma prescrição e uma hospitalização, através dos atributos prescription_id_prescription e hospitalization_hospitalization_id, herdados das tabelas “**Prescription**” e “**Hospitalization**” respetivamente.

Appointment:

- billing_bill_id(Bint): Atributo herdado bill_id da tabela "**Billing**", usado como foreign key. Utilizado para correlacionar a consulta à bill correta.
- doctor_employee_user_id_cc(int): Atributo herdado employee_user_id_cc da tabela "**Employee**" como foreign key. Necessário para relacionar o doutor responsável pela consulta marcada.
- patient_user_id_cc(int): Atributo herdado user_id_cc da tabela "**Patient**" como foreign key. Indica o paciente que está a ser consultado.

Appointment_Prescription:

Tabela intermédia criada pelo modelo físico. Permite correlacionar uma prescrição e uma consulta, através dos atributos prescription_id_prescription e appointment_appointment_id, herdados das tabelas "**Prescription**" e "**Appointment**" respetivamente.

Quantity:

- medication_medication_id(Bint): Atributo herdado medication_id da tabela "**Medication**", usada como foreign key para correlacionar a medicação com a quantidade que deve ser atribuída dada a prescrição.
- prescription_id_prescription(Bint): Atributo herdado id_prescription da tabela "**Prescription**", usada como foreign key para correlacionar a prescrição com a quantidade de medicação desejada.

Effect_properties:

- sideeffect_side_effect_id(Bint): Atributo herdado side_effect_id da tabela "**SideEffect**" como foreign key para referenciar o efeito secundário causado na tabela "**SideEffect**".
- medication_medication_id(Bint): Atributo herdado medication_id da tabela "**Medication**" como foreign key para referenciar a medicação que tem os efeitos secundários da entrada.

Payment:

- billing_bill_id(Bint): Atributo herdado bill_id da tabela "**Billing**" como foreign key, permite relacionar um pagamento com a bill para o qual o pagamento foi feito.

Funções não representadas no ER

Algumas medidas importantes a tomar não podem ser representadas através do ER. Estas poderão ser implementadas através de triggers.
Um dos casos é se o utilizador for um paciente a consultar a tabela receitas, as receitas que serão mostradas irão ser apenas aquelas associadas a esse paciente. Já no caso de ser um

médico a aceder à tabela serão mostradas todas as receitas passadas aos seus pacientes (quem já teve envolvido num **appointment**, **surgery** ou **hospitalization**) mesmo que não tenham sido passadas pelo mesmo. Por consequência, o mesmo acontece para as cirurgias e hospitalizações, todos os médicos e enfermeiros podem consultar as marcações nas quais estão envolvidos.
Adicionalmente, apenas as assistentes terão a opção de marcar consultas e hospitalizações, portanto também poderão consultar todas as marcações já existentes para um paciente.
Foi pensado as assistentes serem responsáveis por inserir na base de dados os enfermeiros e médicos necessários para uma cirurgia ou hospitalização.
Apenas os empregados poderão consultar contratos e só têm acesso aos seus próprios contratos, adicionados ao sistema pelas assistentes. Apenas os médicos poderão adicionar e criar receitas, associadas aos "seus" pacientes em hospitalizações/apoointments.

Transações Relevantes

- **Adicionar User (Patient, Doctor, Nurse, Assistant.):**
  Para adicionar um User, é necessário adicionar uma entrada na tabela "**Users**".
  - User-Patient:
    Se o **User** for um **Patient**, é necessário adicionar uma entrada na tabela "**Patient**".
  - User-Employee:
    Se o **User** for um **Employee**, é necessário adicionar uma entrada na tabela "**Employee**", pelo menos uma entrada na tabela "Contract".
    - Employee-Doctor:
      Se o **Employee** for **Doctor**, é necessário adicionar uma entrada na tabela "**Doctor**". Se o **Doctor** tiver especializações, deverão ser adicionadas como entradas na tabela "**Specialization**".
    - Employee-Nurse:
      Se o **Employee** for **Nurse**, é necessário adicionar uma entrada na tabela "**Nurse**".
    - Employee-Assistant:
      Se o **Employee** for **Assistant**, é necessário adicionar uma entrada na tabela "**Assistant**".

- **Marcar Consulta:**
  Para marcar uma consulta, é necessário adicionar uma entrada na tabela "**Appointment**". Se a consulta resultar numa prescrição, deve-se adicionar também uma entrada na tabela "**Prescription**".

- **Marcar Cirurgia:**
  Para marcar uma cirurgia, é necessário adicionar uma entrada à tabela "**Surgery**". Deve-se também adicionar uma entrada à tabela "**Billing**" de

acordo com o custo da operação, bem como as entradas necessárias à tabela "**Payment**", que correspondam ao pagamento da Bill associada.

- Hospitalization:

  Se ainda não estiver hospitalizado, deve-se adicionar também uma entrada na tabela "**Hospitalization**".

  - Billing:

    Ao adicionar uma hospitalização, é necessário adicionar uma entrada na tabela "**Billing**" para registar o valor que o paciente fica a dever.

- **Registar Prescrição:**

  Para adicionar uma Prescrição, é necessário primeiro verificar nas tabelas de "**Appointment**" e "**Hospitalization**" o evento (marcação ou hospitalização) referido existe, e a tabela "**Medication**" para validar que a medicação é referida, de modo a adicionar uma entrada à tabela "**Prescription**". É também necessário adicionar uma entrada na tabela "**Quantity**", de acordo com a quantidade.

- **Ver Prescrições:**

  Para ler prescrições é necessário aceder à tabela "**Users**" para validar o ID do cliente, e às tabelas "**Appointment**" e "**Hospitalization**" para por sua vez aceder à tabela "**Prescription**" para procurar as prescrições associadas a eventos associados ao paciente do ID fornecido.

- **Executar Pagamentos:**

  Para realizar um pagamento de uma Bill, é necessário verificar a validade do pagamento acedendo à tabela "**Billing**", atualizar o valor que falta pagar nessa Bill e marcar como paga se for o caso. É necessário também adicionar uma entrada na tabela "**Payment**" para registar este pagamento.

- **TOP 3:**

  Para fazer uma lista dos "TOP 3" pacientes, é necessário aceder à tabela "**Users**", para pesquisar todos os pacientes, às tabelas "**Appointment**" e "**Hospitalization**", e à tabela de "Billing**"** para contar o dinheiro pago por todos os pacientes para cada consulta e hospitalização.

- **Criar Sumário Diário:**

  Para criar um Sumário Diário é necessário aceder às tabelas "**Billing**", "**Surgery**" e "**Prescriptions**" para contar o dinheiro gasto, número de cirurgias feitas, e número de prescrições passadas num dado dia.

- **Criar Sumário Mensal:**

  Para criar um Sumário Mensal é necessário aceder à tabela "**Users**", para pesquisar por todos os Doutores, e a tabela "**Surgery**" para pesquisar e contar todas as cirurgias feitas por um dado Doutor num determinado mês.

Potenciais conflitos e como evitar

Se não forem tomados os devidos cuidados, podemos causar vários problemas na nossa base de dados, dentro dos quais temos: demasiadas atualizações em simultâneo, perdas de dados nas tabelas envolvidas, perdas de atualizações a realizar e inconsistência de dados. Para evitar este tipo de problemas podemos recorrer a locks, como por exemplo, um row-level lock, este bloqueia apenas as linhas que são acedidas por uma transação e fornece a melhor concorrência, permitindo que transações simultâneas acedam a linhas na mesma tabela. O bloqueio em nível de linha é preferível quando há muitas transações simultâneas, cada uma operando em linhas diferentes das mesmas tabelas.
Outro exemplo seria o uso de uma exclusive lock, quando fosse necessário fazer a atualização das tabelas (ex: Share mode). No entanto, teríamos de ter em conta que o uso excessivo destes locks pode piorar o desempenho da base de dados.
Adicionalmente, temos, também, que considerar que múltiplos usuários poderão querer consultar as mesmas tabelas ao mesmo tempo, por exemplo um médico e um paciente podem querer aceder à tabela de appointments simultaneamente. Ora, seria necessário utilizar o isolamento de transações, por exemplo o Serializable.

Como é evidente, o facto de vários utilizadores poderem aceder e ler do sistema em simultâneo vai gerar transações concorrentes, como tal, podem também surgir alguns problemas de concorrência não necessariamente tratados pelo DBMS.

Casos específicos de problemas de concorrência:

1- Ao criar um novo utilizador

Para evitar que, ao adicionar um novo **user**, não sejam adicionados, concorrentemente, diferentes utilizadores com o mesmo email ou número de identificação é aplicado um row-level lock na tabela **user.**
Da mesma forma, é necessário impor bloqueio row-level na tabela correspondente ao tipo de usuário em questão. Por exemplo, se o usuário for um funcionário, é fundamental aplicar um bloqueio também na tabela dos funcionários, assim como na tabela específica do tipo de funcionário. Em caso de o usuário ser um paciente, o bloqueio deve ser aplicado à tabela dos pacientes.

2- Ao marcar uma consulta ou cirurgia

É também importante garantir que o mesmo empregado não está envolvido em mais do que um serviço médico num dado momento quando transações concorrentes tentam marcar consultas ou cirurgias, para tal, é necessário dar exclusive lock a **appointment** e **surgery** (**enrolment_appointment** e **enrolment_surgery** também, se for necessário o envolvimento de **nurse**). No caso de ser necessário criar uma nova hospitalização porque uma cirurgia não esta associada a nenhuma , é importante também que não sejam criadas duas com o

mesmo hosptialization_id, então é necessário também dar exclusive lock na tabela **hospitalization**. Para além disso, é necessário evitar que haja alterações de informação de uma bill de um paciente, em simultâneo, assim. Ao aplicar um row-level lock na informação financeira do paciente irá prevenir isto.

3- Adicionar prescrição

Sempre que um médico adiciona uma prescrição, associada a uma hospitalização ou a uma marcação, é necessário garantir que não há várias prescrições com o mesmo atributo pescription_id, portanto, é feito um exclusive row-level lock na tabela **prescription.**

4- Lista de top 3 pacientes do mês e relatório mensal de cirurgias

Para evitar erros de concorrência na geração do top 3 é aplicado um shared lock à tabela **payment** enquanto está a ser gerada a lista.
A mesma ideia aplica-se na transação de geração de um relatório mensal contendo os médicos com o maior número de cirurgias, portanto, essa transação aplica um shared lock na tabela **surgery**