UNIVERSIDADE D
COIMBRA

Faculdade de Ciências e Tecnologias

BASES DE DADOS - LICENCIATURA EM ENGENHARIA INFORMÁTICA

SISTEMA DE GESTÃO DE HOSPITAIS

Trabalho Realizado por:
Nuno Batista e Miguel Martins

Março, 2024

1 Introdução..... 2
1.1 Membros e contactos.....**2**
1.2 Descrição breve do projeto..... 2
2 Funcionamento.....3
2.1 Tabelas E-R..... 3
2.2 Decisões principais E-R.....**7**
2.3 Transações.....**8**
2.4 Problemas de concorrência..... 10
3 Plano de desenvolvimento..... 11
3.1 Tarefas planeadas.....**11**
3.2 Linha de tempo..... **12**

1 Introdução

1.1 - Membros e contactos

Nuno Batista 👍 - nunomarquesbatista@gmail.com
Miguel Martins 👍 - uc2022213951@student.uc.pt

1.2 - Descrição breve do projeto

O objetivo deste projeto é criar um sistema eficiente e intuitivo para consultar, processar e gerir informação nas bases de dados de hospitais.

Adicionalmente, o projeto servirá de plataforma para os pacientes poderem agendar e pagar serviços médicos. Tendo em conta que podem haver vários utilizadores a aceder ao sistema em simultâneo, podem surgir problemas de concorrência, que serão discutidos mais para a frente.

Nessa base de dados, deverá ser guardada informação sobre médicos, enfermeiros, assistentes, medicamentos e os seus efeitos secundários, pacientes, consultas, hospitalizações, faturas a pagar, especializações médicas, níveis de hierarquia dos enfermeiros, possíveis funções desempenhadas pelos enfermeiros e os contratos dos empregados.

Vão existir 4 tipos de utilizador, cada um com diferentes funcionalidades associadas, sendo essas funcionalidades:

- Paciente
    - Agendar consulta.
    - Consultar as suas próprias consultas.
    - Consultar as suas próprias prescrições.
    - Pagar fatura (pode ser paga por partes)
- Assistente
    - Consultar as consultas de qualquer paciente.
    - Agendar cirurgias associadas a uma hospitalização (ou criar nova hospitalização).
    - Consultar os 3 pacientes que gastaram mais dinheiro no mês atual.
    - Consultar um sumário diário contendo detalhes sobre as cirurgias, pagamentos e prescrições das hospitalizações de um dado dia.
    - Consultar uma lista dos médicos com mais cirurgias de cada mês dos últimos 12 meses.
- Médico
    - Consultar as prescrições de qualquer paciente.
    - Adicionar prescrição a uma consulta ou hospitalização.
- Enfermeiro

- Consultar as prescrições de qualquer paciente.

2 Funcionamento

2.1 - Tabelas E-R

➔ service_user

- **user_id** : *BigInt - Primary key* | Identificador único de cada user
- **name** : *VarChar - Not Null* | Nome do utilizador
- **nationality** : *VarChar - Not Null* | Nacionalidade do utilizador
- **phone** : *Int - Not Null, Unique* | Contacto telefónico do utilizador
- **birthday** : *Date - Not Null* | Data de nascimento do utilizador
- **email** : *VarChar - Not Null, Unique* | Contacto de email do utilizador
- **password**: *VarChar - Not Null* | Password do utilizador

```
-- 'user_id' must have 10 digits and be non-negative
LENGTH(user_id) = 10
--  'name' field only contains letters and spaces
name ~ '^[a-zA-Z ]+$';
--  phone - Phone number has to comprise of 9 digits
LENGTH(phone) != 9 OR phone
-- birthday - Birthday can't be earlier than 1904
birthday < '1904-01-01'
```

➔ patient

- **patient_id** : *BigInt - Not Null, Unique* | Identificador único de cada paciente

```
-- 'patient_id' must have 10 digits and be non-negative
LENGTH(patient_id) = 10
```

➔ contract

- **contract_id** : *BigInt - Not Null, Unique* | Identificador único do contrato
- **start_date** : *Date - Not Null* | Data de inicio do contrato
- **end_date** : *Date - Not Null* | Data de fim de contrato

```
-- start_date must be previous to end_date
start_date >= end_date;
-- 'contract_id' must have 10 digits and be non-negative
LENGTH(contract_id) = 10
```

➔ employee

➔ **nurse**

➔ **rank**

- **rank_id** : *BigInt - Primary Key* | Identificador único do nível hierárquico
- **rank_name** : *VarChar - Not Null, Unique* | Nome do nível hierárquico

```
-- 'rank_name' field only contains letters and spaces
rank_name ~ '^[a-zA-Z ]+$';
-- 'rank_id' must have 10 digits and be non-negative
LENGTH(rank_id) = 10
```

➔ **assistant**

➔ **doctor**

- **university**: *VarChar - Not Null* | Universidade onde se formou
- **license_id** : *VarChar - Not Null, Unique* | Identificador da licença médica
- **graduation_date** : *Date - Not Null* | Data em que o médico se formou

```
-- 'university' field only contains letters and spaces
university~ '^[a-zA-Z ]+$';
-- 'license_id' must have 10 digits and be non-negative
LENGTH(license_id) = 10
```

➔ **specialization**

- **spec_id**: *BigInt - Primary Key* | Identificador único da especialização
- **spec_name** : *VarChar - Not Null, Unique* | Nome da especialização

```
-- 'spec_name' field only contains letters and spaces
spec_name~ '^[a-zA-Z ]+$';
-- 'spec_id' must have 10 digits and be non-negative
LENGTH(spec_id) = 10
```

➔ **appointment**

- **app_id** : *BigInt - Primary Key* | Identificador da consulta
- **app_date** : *Timestamp - Not Null* | Data da consulta

```
-- 'app_id' must have 10 digits and be non-negative
LENGTH(app_id) = 10
```

➔ **app_type**

- **type_id** : *BigInt - Primary Key* | Identificador único do tipo de consulta
- **type_name** : *VarChar - Not Null, Unique* | Tipo de consulta

-- 'type_id' must have 10 digits and be non-negative
LENGTH(type_id) = 10
-- 'type_name' field only contains letters and spaces
type_name ~ '^[a-zA-Z]+$';

➔ bill

- **bill_id** : *BigInt - Primary Key* | Identificador da fatura
- **cost** : *BigInt - Not Null* | Valor da fatura
- **is_payed**: *Bool - Not Null* | Já foi paga ou não

  -- 'cost' must be positive
  cost> 0
  -- 'payment_id' must have 10 digits and be non-negative
  LENGTH(payment_id) = 10

➔ payment

- **payment_id** : *BigInt - Not Null, Unique* | Identificador único do pagamento
- **amount** : *BigInt - Not Null* | Valor do pagamento
- **payment_date** : *Timestamp - Not Null* | Data do pagamento

  -- 'amount' must be positive
  amount > 0
  -- 'payment_id' must have 10 digits and be non-negative
  LENGTH(payment_id) = 10

➔ surgery

- **surgery_id** : BigInt - *Primary Key* | Identificador único da cirurgia
- **type** : *VarChar - Not Null* | Tipo de cirurgia
- **surg_date** : *Date - Not Null* | Data de realização da cirurgia

  -- 'surgery_id' must have 10 digits and be non-negative
  LENGTH(surgery_id) = 10
  -- 'type' field only contains letters and spaces
  type~ '^[a-zA-Z]+$';

➔ role

- **role_id**: *BigInt - Primary Key* | Identificador único da função
- **name** - *VarChar - Not Null, Unique* | Nome da função

  -- 'role_name' field only contains letters and spaces
  role_name ~ '^[a-zA-Z]+$';
  -- 'role_id' must have 10 digits and be non-negative
  LENGTH(role_id) = 10

➔ **enrolment_surgery**

➔ **enrolment_appointment**

➔ **hospitalization**

- **hosp_id** : *BigInt - Primary Key* | Identificador único a hospitalização
- **start_date** : *Timestamp - Not Null* | Data de inicio da hospitalização
- **end_date** : *Timestamp - Not Null* | Data do fim da hospitalização
- **type** : *VarChar - Not Null* | Tipo de hospitalização

```
-- 'type' field only contains letters and spaces
type~ '^[a-zA-Z ]+$';
-- start_date must be previous to end_date
start_date >= end_date;
-- 'hospital_id' must have 10 digits and be non-negative
LENGTH(hospital_id) = 10
```

➔ **prescription**

- **presc_id** : *BigInt - Primary Key* | Identificador único da prescrição
- **presc_date** : *Timestamp - Not Null* | Data da prescrição

```
-- 'presc_id' must have 10 digits and be non-negative
LENGTH(presc_id) = 10
```

➔ **medication**

- **med_id** : *BigInt - Primary Key* | Identificador único da medicação
- **med_name** : *VarChar - Not Null, Unique* | Nome da medicação

```
-- 'med_name' field only contains letters and spaces
med_name ~ '^[a-zA-Z ]+$';
-- 'med_id' must have 10 digits and be non-negative
LENGTH(med_id) = 10
```

➔ **dose**

- **amount** : *Integer - Not Null* | Valor de cada dose
- **time_of_day** : *VarChar - Primary Key* | Define a hora do dia para tomar o medicamento
- **duration** : *SmallInt - Not Null* | Define durante quantos dias o paciente deve tomar o medicamento

```
-- 'amount' must be positive
amount > 0
-- 'duration' must be positive
duration> 0
-- 'time_of_the_day' has the domain morning, afternoon or evening
time_of_the_day IN ('morning', 'afternoon', 'evening');
```

➔ **effect_properties**

- **probability** : *Float - Not Null* | Probabilidade de ocorrência de sortir efeito
- **severity** : *Number - Not Null* | Grau de severidade dos efeitos

```
-- 'severity' must be positive
severity> 0
-- 'probability' must be between 0 and 1
probability> 0 AND probability < 1
```

➔ **side_effects**

- **effect_id** : *BigInt - Primary Key* | Identificador único do efeito
- **symptom** : *VarChar - Not Null, Unique* | Sintomas do efeito

```
--  'symptom' field only contains letters and spaces
symptom~ '^[a-zA-Z ]+$';
-- 'effect_id' must have 10 digits and be non-negative
LENGTH(effect_id) = 10
```

2.2 - Decisões principais E-R

Esta secção tem como objetivo explicar algumas das decisões menos óbvias tomadas durante a criação do E-R.

1 - Razão do set de entidades service_user

Para efeitos de categorização e como os **employees** e os **patients** têm diversos atributos em comum, foi criado o set de entidades **service_user**, tornando **employees** e **patients** especializações de **service_user** que herdam os seus atributos.

2 - Associação de nurses com diferentes roles a appointments e surgery

De modo a associar **nurse** a uma **surgery** ou **appointment** com um dado **role**, o set de entidades **nurse** está ligado a **enrolment_surgery** e **enrolment_appointment**, que, por sua vez, estão ligados a **role** (**role** contém entradas para todas as possíveis funções de **nurse**) bem como a **surgery** e **appointment**. Por exemplo, se uma entidade **nurse N** tiver o **role R** numa **surgery S**, então, a entrada na tabela **enrolment_surgery** vai ter uma das foreign keys a apontar para **N**, outra a apontar para **S** e a terceira a apontar para **R**.

3 - Razão para usar primary keys “forçadas” em certas tabelas

Tomando como exemplo o set de entidades **specialization**, esta tem o atributo **spec_name** denotando o nome da especialização, que, evidentemente, nunca pode ser repetido, no entanto, este atributo é uma string de caracteres, o que torna as queries de SQL mais lentas do que com inteiros, portanto, em nome de eficiência, foi adicionado o atributo

spec_id, sendo este um número inteiro. Situações análogas são verificadas nos sets de entidades **medication**, **side_effect**, **role** e **app_type**.

4 - Hierarquia dos ranks das nurses e das especializações médicas

Estas hierarquias foram implementadas com relações recursivas, de 0..1 para 0..n, ou seja, vendo estas relações como árvores, uma entidade pode ter vários filhos bem como pode ter um pai. Deste modo, **doctor** está associado a **specialization** com 0..n para 0..n (uma vez que uma entidade **doctor** pode não ter especialização, mas também pode ter várias e diferentes entidades **doctor** podem ter a mesma especialização) e as entidades **specialization** em si estão organizadas hierarquicamente como descrito anteriormente. Por sua vez, **nurse** está associada a **rank** com 0..n para 1..1, sendo 0..n a cardinalidade de **nurse** (uma vez que uma entidade **nurse** tem obrigatoriamente um nível de hierarquia dentro do hospital, e várias entidades **nurse** podem estar no mesmo nível).

5 - Detalhes sobre o set de entidades prescription

Uma entidade **prescription** pode ter vários medicamentos e deve também definir as suas respectivas doses, para descrever a dose de uma dada medicação, foi criada o set de entidades **dose** que é weak para **prescription** e **medication**, esta vai definir a quantidade de uma entidade **medication** que contém uma entidade **prescription**, bem como a altura do dia e durante quanto tempo deve ser tomado esse medicamento.

6 - Detalhes sobre os sets de entidades medication e side_effect

Um medicamento pode ter diversos efeitos secundários com diferentes níveis de severidade e probabilidades de ocorrência, portanto, para descrever estas características para um dado medicamento, foi criado o set de entidades **effect_properties**, que é weak para **medication** e **side_effect**, este tem atributos que descrevem a severidade e a probabilidade de ocorrência da entidade **side_effect** à qual está associada, quando tomada a entidade **medication** à qual também está associada.

7 - Atributo time_of_day na tabela dose

Numa prescrição, é natural o mesmo medicamento ser tomado em alturas diferentes no dia, para tornar isto possível no modelo relacional, adicionou-se a primary key **time_of_day** que define a hora do dia a que deve ser tomada a dose definida pelo atributo **amount**. Deste modo, podemos ter o mesmo medicamento mais do que uma vez na mesma prescrição em alturas do dia diferentes.

2.3 - Definição das transações

Uma transação é uma unidade lógica de trabalho que tem de ser necessariamente concluída, caso tal não seja possível é imperativo abortar a transação.

Nesta secção vão ser descritas as principais transações mais complexas, ou seja, as transações que envolvem mais do que uma tabela.

1 - Criar um novo utilizador

Esta transação vai afetar tabelas diferentes, dependendo do tipo de utilizador registado. Em qualquer caso, é adicionada uma entrada a **service_user**.

Caso seja um empregado, vai ser adicionada uma entrada a **employee**, bem como à tabela correspondente ao tipo de empregado específico que foi adicionado, ou seja, **nurse**, **doctor** ou **assistant**. Caso seja **doctor**, ainda pode ser adicionada uma ou mais entradas à tabela que liga **doctor** a **specialization** para denotar as especializações do **doctor** adicionado. É ainda de notar que se for **nurse**, é necessário adicionar uma entrada à tabela que liga **nurse** a **rank** de modo a denotar o nível na hierarquia da **nurse** recém-adicionada.

Caso seja um paciente, é adicionada uma entrada a **patient**.

2 - Marcar cirurgia

Quando é marcada uma cirurgia, é adicionada uma entrada a **surgery** e é atualizada a entrada na tabela **bill** correspondente à hospitalização associada à cirurgia (com recurso a triggers). Quando não há hospitalização associada à cirurgia, são também adicionadas novas entradas a **hospitalization** e **bill**. Adicionalmente, podem também haver atualizações na tabela **enrolment_surgery** caso seja necessário o envolvimento de entidades **nurse**.

3 - Marcar appointment

Quando é marcada uma consulta, é adicionada uma entrada a **appointment** e à tabela **bill**. Adicionalmente, podem também haver atualizações na tabela **enrolment_appointment** caso seja necessário o envolvimento de entidades **nurse**.

4 - Adicionar prescrições

Quando é adicionada uma prescrição, para além de ser adicionada uma entrada a **prescription**, é também necessário adicionar entradas tanto à tabela que liga **prescription** a **hospitalization**, como à que liga prescription a **medication** (**dose**), de modo a descrever as doses dos medicamentos prescritos.

5 - Criar um sumário diário

Para criar um sumário diário, é necessário ler os detalhes das tabelas **prescription**, **surgery** e **payment** de um dado dia.

6 - Criar um relatório mensal

Para criar um relatório mensal, vai ser necessário ler os dados da tabela **surgery** associados a **doctor** de modo a contabilizar a quantidade de cirurgias por doutor num dado mês.

7 - Executar pagamento

Quando um paciente faz um pagamento, é adicionada uma entrada à tabela **payment**, verifica-se também se a soma dos pagamentos para uma dada **bill** é igual ao custo total dessa **bill**, nesse caso, atualiza-se o atributo **is_payed** da mesma é alterado para true.

8 - Lista de top 3 pacientes do mês

Para visualizar uma lista dos 3 clientes que mais pagaram no mês atual, terá de ser consultada a tabela **payment**, para verificar todos os pagamentos feitos nos últimos 30 dias, para saber qual paciente executou um dado pagamento, é necessário consultar a entrada de **bill** à qual o pagamento está associado, bem como a entrada de **appointment** ou **hospitalization** à qual **bill** está associada, finalmente, verifica-se qual entrada da tabela **patient** está ligada ao **appointment** ou **hospitalization**.

2.4 - Problemas de concorrência

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

1 - Marcar consulta ou cirurgia

Quando transações concorrentes tentam marcar consultas ou cirurgias, é importante garantir que o mesmo empregado não está envolvido em mais do que um serviço médico num dado momento, 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 da cirurgia não estar associada a uma hospitalização, também é preciso criar uma entrada **hospitalization**, portanto aplica-se também exclusive lock nessa tabela.

2 - Adicionar prescrição

Quando é adicionada uma prescrição, é necessário garantir que não há vários médicos a marcar prescrições conflituosas para o mesmo paciente, portanto, é feito um exclusive lock nas tabelas que associam as prescrições aos pacientes.

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

lock payment

Um aspeto importante sobre a criação da lista dos top 3 pacientes que mais pagaram no mês atual, é que, ao negligenciar o bloqueio da tabela **payment**, é possível que suceda um cenário semelhante ao seguinte:

Ordem Atual:	Lista vazia	A	A	A	B > A
Montante atual do paciente A	10€	10€	30€	30€	30€
Montante atual do paciente B	5€	5€	5€	15€	15€
Ação do programa	Início da query	A é lido	A paga 20€	B paga 10 €	B é lido
Momento atual:	T0	T1	T2	T3	T4

Neste caso, apesar do paciente B nunca ter tido um montante superior ao do A, na lista final, o do B acabou por ser considerado superior ao do A, o que está claramente errado. Para evitar este tipo de casos, é aplicado um shared lock à tabela **payment** enquanto está a ser gerada uma lista de top 3.

A mesma ideia aplica-se na transação de geração de um relatório mensal que contém médicos com o maior número de cirurgias, portanto, essa transação aplica um shared lock na tabela **surgery**.

4 - Executar pagamentos

Para que não seja possível pagar uma fatura já paga, quando se tenta fazer um pagamento, é aplicado um row-level lock na entrada da tabela **bill** correspondente ao pagamento feito.

3 Plano de desenvolvimento

3.1 - Tarefas planeadas e divisão de trabalho

1 - Adição de indivíduos - Nuno Batista - Semana 25/03-31/03
2 - Autenticação de utilizadores. - Miguel Martins - Semana 25/03-31/03
3 - Agendamento de serviços. - Nuno Batista - Semana 01/04-07/04
4 - Adição de prescrições. - Nuno Batista - Semana 08/04-14/04
5 - Consulta de serviços. - Miguel Martins - Semana 15/04-21/04
6 - Executar pagamentos. - Miguel Martins - Semana 22/04-28/04
7 - Gerar os 3 pacientes que mais gastaram. - Nuno Batista - Semana 22/04-28/04
8 - Gerar um sumário diário. - Miguel Martins - Semana 29/04-05/05
9 - Gerar o relatório mensal. - Nuno Batista - Semana 06/05-12/05
10 - Relatório final - Ambos - Semana 13/05-19/05

3.2 - Linha de tempo