U.PORTO

FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

SERVIÇO DE URGÊNCIA

Relatório da base de dados

Grupo 504

Diogo Serra Duque up201406274@fe.up.pt

José Aleixo Cruz up201403526@fe.up.pt

Renato Sampaio de Abreu up201403377@fe.up.pt

Índice

Descrição do contexto

A base de dados implementada terá como objetivo a gestão de um serviço de urgência de um hospital. Para isso, decidimos basearmo-nos no funcionamento (de forma muito simplificada) do serviço de urgência do Hospital Santo António.

Assim, no contexto da nossa base de dados, o serviço de urgência é composto por uma equipa de urgência que inclui médicos, enfermeiros e pessoal administrativo. Esta equipa tratará de todos os aspetos referentes ao utente.

Primeiramente é realizada a admissão nas urgências, que está a cargo do pessoal administrativo, e respetiva identificação da urgência médica do utente. Após isto, e de forma a representar o real funcionamento deste serviço, é feita a triagem de Manchester, pelos enfermeiros, atribuindo assim uma prioridade (emergente, muito urgente, urgente, pouco urgente e não urgente) e uma área de ação (médica, cirúrgica, ortopedia, clínica geral), conforme a condição atual do paciente. Com isto o utente estará sempre associado a uma urgência médica desde a admissão inicial, a qual será atendida, mais ou menos rapidamente, de acordo com os princípios da triagem.

Desta forma, e de acordo com a informação das urgências médicas existentes, os utentes associados a estas receberão o diagnóstico conforme a sua prioridade e por um médico que possa exercer funções na área de ação. Após o diagnóstico, e dependendo deste, é realizado o tratamento (cirurgia em casos mais graves, prescrições nos restantes), o qual está a cargo do médico. Uma vez que todos os tratamentos tenham sido realizados, o utente é autorizado, pelo médico, a ter alta.

Principais conceitos

Para gestão dos recursos humanos, há uma superclasse *Pessoa* caracterizada pela idade, morada, nome, sexo e nif. Esta terá as seguintes classes derivadas, que herdam os atributos anteriormente referidos:

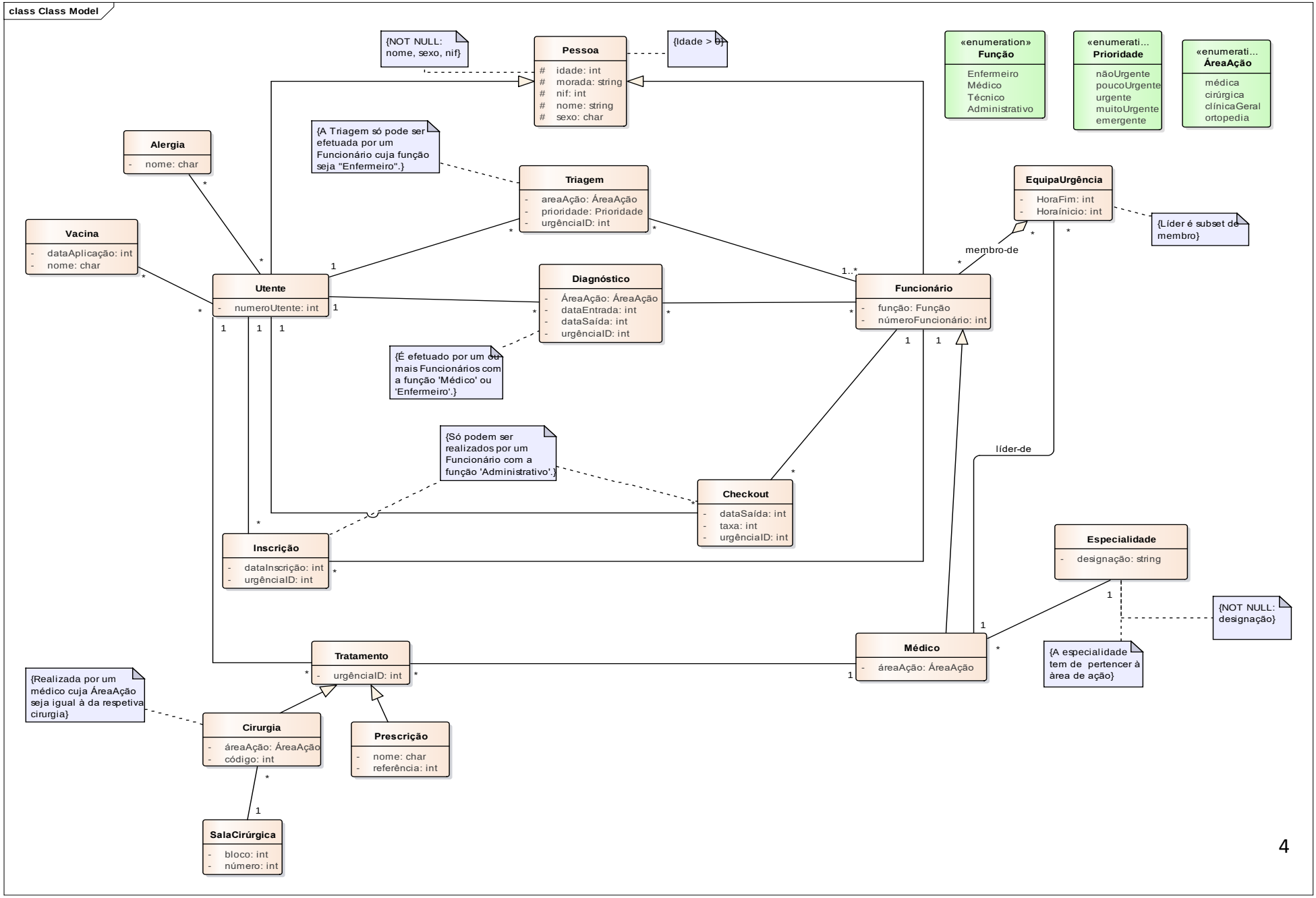
- *Utente*, definido pelo seu registo médico (alergias e vacinação) e pelo seu número de utente. Esta classe estará sempre associada aos vários procedimentos existentes numa urgência médica real (inscrição, diagnóstico, tratamento, etc), os quais estarão a cargo da equipa de urgência.
- *Funcionário* definido por um ID (identificação única dentro do hospital) e pela sua função (médico, enfermeiro, técnico, administrativo):
    - O funcionário com função Administrativo é responsável por proceder à Inscrição do Utente, isto é, o registo da hora de entrada e a identificação única da urgência médica em questão bem como Checkout do utente (pagamento de taxas e restantes burocracias).
    - O enfermeiro, tem como funções aplicar a *Triagem* a cada urgência médica. Isto implica definir a área de ação da urgência consoante a condição do utente. Para além disso, também é definida a *Prioridade* e o tempo-alvo para ser atendido.

- O médico que é caracterizado pela sua área de ação e consequentemente pela especialidade. Este faz parte da *EquipaUrgência*, podendo ser o líder da equipa e/ou fazer parte, e a sua função é, após a inscrição e triagem, realizar o *Diagnóstico*, o qual pode ser auxiliado por enfermeiros. O diagnóstico permite então saber o *Tratamento* necessário para cada urgência médica, o qual também é efetuado pelo médico.

Em relação à gestão dos recursos logísticos existem algumas classes que foram mencionadas acima mas não devidamente explicadas:

- *EquipaUrgência*, corresponde à agregação das várias instâncias de *Funcionário*, constituindo assim a equipa de urgência existente num determinado momento no hospital, liderada por um médico.
- *Diagnóstico*, que tem como objetivo determinar o Tratamento que se deve aplicar ao caso médico em questão sendo este realizado por um médico com o auxílio, se necessário de enfermeiros. Por seu lado, o *Tratamento* pode ser uma *Cirurgia* ou uma *Prescrição* de fármacos. A *Cirurgia* é sempre realizada por um médico cuja área de ação seja idêntica à área de ação da urgência médica e que tenha a especialidade necessária para realizar essa cirurgia.

Diagrama de classes UML



Modelo relacional

No modelo relacional da nossa base de dados, além de traduzirmos as associações do diagrama de classes em relações, decidimos interligar objetos relacionados à mesma urgência, usando "*urgênciaID*" como atributo de certas relações a de que uma urgência está dependente. Desta forma, escusamos a criação de uma nova classe e podemos aceder a todos os objetos relacionados com uma determinada urgência, realizando uma pesquisa restrita a um ID.

O texto abaixo representa o nosso modelo relacional, sendo que os atributos a negrito são as chaves primárias de cada relação.

- *Pessoa* (**idPessoa**, nome, idade, sexo, morada, nif)
- *Utente* (**númeroUtente**, idPessoa → Pessoa)
- *EquipaUrgência* (**idEquipa**, dataInício, dataFim, idLíder -> Médico)
- *Funcionário* (**númeroFuncionário**, idPessoa → Pessoa, função, idEquipa→Equipa)
- *Médico* (**númeroFuncionário → Funcionário**, idEspecialidade → Especialidade, áreaAção)
- *Alergia* (**idAlergia**, nome)
- *Vacina* (**idVacina**, nome, dataAplicação)
- *Especialidade* (**idEspecialidade**, designação)
- *Inscrição* (**urgênciaID**, númeroUtente → Utente, númeroFuncionário→Funcionário, horaEntrada)
- *Checkout* (**urgênciaID**, númeroUtente →Utente, númeroFuncionário →Funcionário, taxa, horaSaída)
- *Triagem* (**urgênciaID**, númeroUtente → Utente, númeroFuncionário→Funcionário, áreaAção, prioridade)
- *Diagnóstico* (**urgênciaID, dataEntrada, dataSaída**, númeroUtente → Utente, númeroFuncionário → Funcionário, áreaAção)
- *Tratamento* (**urgênciaID**, númeroUtente → Utente, númeroFuncionário → Funcionário)
- *Cirurgia* (**código, urgênciaID**, ÁreaAção, idSalaCirúrgica → SalaCirúrgica)
- *Prescrição* **(referência, urgênciaID**, nome)
- *SalaCirúrgica* (**idSalaCirúrgica**, bloco, número)
- *UtenteAlergia* (**idAlergia → Alergia, númeroUtente → Utente**)
- *UtenteVacina* (**idVacina →Vacina, númeroUtente → Utente**)

Instruções de Linguagem de Definição de Dados SQL

De seguida encontram-se explícitas as instruções que usamos para criar a estrutura da nossa base de dados, gerando as tabelas necessárias, com os respetivos atributos.

```
CREATE TABLE Pessoa(
idPessoa INT NOT NULL PRIMARY KEY,
nome CHAR(30) NOT NULL,
idade INT NOT NULL,
sexo CHAR(20) NOT NULL,
morada VARCHAR(50),
NIF INT NOT NULL UNIQUE);

CREATE TABLE EquipaUrgencia(
idEquipa INT NOT NULL PRIMARY KEY,
dataInicio DATETIME NOT NULL,
dataFim DATETIME,
idLider INT NOT NULL,
FOREIGN KEY (idLider) REFERENCES Medico(idLider)
CHECK (dataFim > dataInicio));

CREATE TABLE Utente(
numeroUtente INT NOT NULL PRIMARY KEY,
idPessoa INT NOT NULL,
FOREIGN KEY (idPessoa) REFERENCES Pessoa(idPessoa));

CREATE TABLE Funcionario(
numeroFuncionario INT NOT NULL PRIMARY KEY,
idPessoa INT NOT NULL,
funcao VARCHAR(50) NOT NULL,
idEquipa INT,
FOREIGN KEY (idPessoa) REFERENCES Pessoa(idPessoa),
FOREIGN KEY (idEquipa) REFERENCES EquipaUrgencia(idEquipa));

CREATE TABLE Medico(
numeroFuncionario INT NOT NULL PRIMARY KEY,
idEspecialidade INT,
areaAcao VARCHAR(50) NOT NULL,
FOREIGN KEY (numeroFuncionario) REFERENCES
Funcionario(numeroFuncionario),
FOREIGN KEY (idEspecialidade) REFERENCES
Especialidade(idEspecialidade));

CREATE TABLE Alergia(
idAlergia INT NOT NULL PRIMARY KEY,
nome VARCHAR(50) NOT NULL);

CREATE TABLE Vacina(
idVacina INT NOT NULL PRIMARY KEY,
nome VARCHAR(50) NOT NULL);

CREATE TABLE Especialidade(
idEspecialidade INT NOT NULL PRIMARY KEY,
designacao VARCHAR(50));

CREATE TABLE SalaCirurgica(
idSalaCirurgica INT NOT NULL PRIMARY KEY,
bloco CHAR(20) NOT NULL,
numero INT NOT NULL);
```

```
CREATE TABLE UtenteAlergia(
idAlergia INT NOT NULL REFERENCES Alergia(idAlergia),
numeroUtente INT NOT NULL REFERENCES Utente(numeroUtente),
PRIMARY KEY(idAlergia, numeroUtente));

CREATE TABLE UtenteVacina(
idVacina INT NOT NULL REFERENCES Vacina(idVacina),
numeroUtente INT NOT NULL REFERENCES Utente(numeroUtente),
dataAplicacao DATE,
PRIMARY KEY(idVacina, numeroUtente, dataAplicacao));

CREATE TABLE Inscricao
(urgenciaID INT PRIMARY KEY NOT NULL,
numeroUtente INT REFERENCES Utente(numeroUtente) NOT NULL,
numeroFuncionario INT REFERENCES Funcionario(númeroFuncionario) NOT
NULL,
horaEntrada DATETIME NOT NULL);

CREATE TABLE Checkout
(urgenciaID INT PRIMARY KEY NOT NULL,
numeroUtente INT REFERENCES Utente(numeroUtente) NOT NULL,
numeroFuncionario INT REFERENCES Funcionario(númeroFuncionario) NOT
NULL,
taxa INT NOT NULL,
dataSaida DATETIME NOT NULL);

CREATE TABLE Triagem
(urgenciaID INT PRIMARY KEY NOT NULL,
numeroUtente INT REFERENCES Utente(numeroUtente) NOT NULL,
numeroFuncionario INT REFERENCES Funcionario(númeroFuncionario) NOT
NULL,
areaAcao TEXT NOT NULL,
prioridade TEXT NOT NULL);

CREATE TABLE Diagnostico
(urgenciaID INT NOT NULL,
numeroUtente INT REFERENCES Utente(numeroUtente) NOT NULL,
numeroFuncionario INT REFERENCES Funcionario(numeroFuncionario) NOT
NULL,
dataEntrada DATETIME NOT NULL,
dataSaida DATETIME,
PRIMARY KEY(urgenciaID, dataEntrada, dataSaida),
CHECK (dataSaida > dataEntrada));

CREATE TABLE Tratamento
(urgenciaID INT PRIMARY KEY NOT NULL,
numeroUtente INT REFERENCES Utente(numeroUtente) NOT NULL,
numeroFuncionario INT REFERENCES Funcionario(númeroFuncionario) NOT
NULL);

CREATE TABLE Cirurgia(
urgenciaID INT NOT NULL REFERENCES Tratamento(urgenciaID),
codigo INT NOT NULL UNIQUE,
areaAcao TEXT NOT NULL,
idSalaCirurgica INT NOT NULL UNIQUE REFERENCES
SalaCirurgica(idSalaCirurgica),
PRIMARY KEY(urgenciaID, codigo));

CREATE TABLE Prescricao(
urgenciaID INT NOT NULL REFERENCES Tratamento(urgenciaID),
referencia INT NOT NULL,
```

```
nome CHAR(20) NOT NULL,
PRIMARY KEY (urgenciaID, referencia));
```

Instruções de Linguagem de Manipulação de Dados SQL

Os próximos comandos de SQL são responsáveis por inserir objetos de exemplo nas tabelas anteriormente geradas. Para cada possibilidade de inserção que há na base de dados, replicamos um exemplo. Todos os comandos que escrevemos encontram-se no ficheiro SQL anexo "inserts.sql".

Pessoa (com morada)

```
INSERT INTO Pessoa(idPessoa, nome, idade, sexo, morada, NIF) VALUES
(1, 'Renato Abreu', 19, 'Masculino', 'Esposende', 123456789);
```

Pessoa (sem morada)

```
INSERT INTO Pessoa(idPessoa, nome, idade, sexo, NIF) VALUES (,
'Liliana', 27, 'Feminino', 456789123);
```

Utente

```
INSERT INTO Utente(numeroUtente, idPessoa) VALUES (1000, 1);
```

Funcionário (enfermeiro, administrativo, técnico ou médico)

```
INSERT INTO Funcionario(numeroFuncionario, idPessoa, funcao, idEquipa)
VALUES (500, 11, 'Enfermeiro', 1);
INSERT INTO Funcionario(numeroFuncionario, idPessoa, funcao, idEquipa)
VALUES (503, 24, 'Administrativo', 1);
INSERT INTO Funcionario(numeroFuncionario, idPessoa, funcao, idEquipa)
VALUES (514, 20, 'Tecnico', 3);
INSERT INTO Funcionario(numeroFuncionario, idPessoa, funcao, idEquipa)
VALUES (515, 26, 'Medico', 1);
```

Médico (com especialidade)

```
INSERT INTO Medico(numeroFuncionario, idEspecialidade, areaAcao)
VALUES (515, 1, 'Ortopedia');
INSERT INTO Medico(numeroFuncionario, idEspecialidade, areaAcao)
VALUES (516, 2, 'Cirurgica');
```

Alergia

```
INSERT INTO Alergia VALUES(1, 'Polen');
```

Vacina

```
INSERT INTO Vacina VALUES(1, 'Tetano');
```

Especialidade

```
INSERT INTO Especialidade VALUES(1, 'Ortopedica');
```

Sala cirúrgica

```
INSERT INTO SalaCirurgica VALUES(2120,'B', 120);
```

Relação entre utente e alergia

```
INSERT INTO UtenteAlergia VALUES(1, 1003);
```

Relação entre utente e vacina

```
INSERT INTO UtenteVacina VALUES(1, 1004, '1996-07-01');
```

Equipa de urgência

```
INSERT INTO EquipaUrgencia VALUES(1, '2016-04-22 15:00', '2016-04-23
12:00', 515);
```

Inscrição de um utente

```
INSERT INTO Inscricao(urgenciaID, numeroUtente, numeroFuncionario,
horaEntrada) VALUES(1, 1000, 502, '2016-04-22 17:00');
```

Triagem de um utente

```
INSERT INTO Triagem(urgenciaID, numeroUtente, numeroFuncionario,
areaAcao, prioridade) VALUES(1, 1000, 501, 'Medica', 'poucoUrgente');
INSERT INTO Triagem(urgenciaID, numeroUtente, numeroFuncionario,
areaAcao, prioridade) VALUES(2, 1001, 500, 'Cirurgica', 'emergente');
```

Diagnóstico de um utente

```
INSERT INTO Diagnostico(urgenciaID, numeroUtente, numeroFuncionario,
dataEntrada, dataSaida) VALUES(1, 1000, 501, '2016-04-22 17:23',
'2016-04-22 18:23');
INSERT INTO Diagnostico(urgenciaID, numeroUtente, numeroFuncionario,
dataEntrada, dataSaida) VALUES(2, 1001, 500, '2016-04-22 22:51',
'2016-04-22 23:51');
```

Cirurgia a um utente

```
INSERT INTO Cirurgia(urgenciaID, codigo, areaAcao, idSalaCirurgica)
VALUES(2, 3, 'Cirurgica', 2120);
```

Prescrição de um utente

```
INSERT INTO Prescricao(referencia, urgenciaID, nome) VALUES(12349, 1,
'Ben-u-ron');
```

Checkout de um utente

```
INSERT INTO Checkout(urgenciaID, numeroUtente, numeroFuncionario,
dataSaida, taxa) VALUES(1, 1000, 502, '2016-04-22 21:23', 100);
```

Tratamento a um utente

```
INSERT INTO Tratamento(urgenciaID, numeroUtente, numeroFuncionario)
VALUES(1, 1000, 515);
```

Instruções de interrogação

1. Qual é a especialidade do médico chamado "Fábio"?

```
SELECT designacao
FROM
    (Pessoa
    INNER JOIN
    Funcionario
    ON Pessoa.idPessoa = Funcionario.idPessoa
    INNER JOIN
    Medico
    ON Funcionario.numeroFuncionario = Medico.numeroFuncionario
    INNER JOIN
    Especialidade
    ON Medico.idEspecialidade = Especialidade.idEspecialidade)
WHERE Pessoa.nome = 'Fábio';
```

2. Qual é o conjunto de IDs dos utentes do sexo masculino com idade inferior a 30 anos que foram às urgência entre o dia 22 e 26 de Abril de 2016?

```
SELECT DISTINCT Utente.numeroUtente
FROM
    (Pessoa
    INNER JOIN
    Utente
    ON Pessoa.idPessoa = Utente.idPessoa
    INNER JOIN
    Inscricao
    ON Utente.numeroUtente = Inscricao.numeroUtente)
WHERE
((Inscricao.horaEntrada >= '2016-04-22' AND Inscricao.horaEntrada <=
'2016-04-26')
AND
(Pessoa.sexo == 'Masculino' AND Pessoa.idade < 30))
ORDER BY Utente.numeroUtente;
```

3. Quais são os números dos utentes que foram atendidos pelo médico "Duque" ou que têm a vacina contra o "Tetanto"?

```
SELECT DISTINCT Utente.numeroUtente
FROM
    (Pessoa
    INNER JOIN
    Funcionario
    ON Pessoa.idPessoa = Funcionario.idPessoa
    INNER JOIN
    Tratamento
    ON Tratamento.numeroFuncionario = Funcionario.numeroFuncionario
    INNER JOIN
    Utente
    ON Utente.numeroUtente = Tratamento.numeroUtente)
WHERE Pessoa.nome = 'Duque'

UNION

SELECT Utente.numeroUtente
FROM
    (Utente
    INNER JOIN
```

```
    UtenteVacina
    ON Utente.numeroUtente = UtenteVacina.numeroUtente
    INNER JOIN
    Vacina
    ON Vacina.idVacina = UtenteVacina.idVacina
    )
WHERE Vacina.nome = 'Tetano';
```

4. Quantas pessoas já tomaram a vacina contra o "Tetano"?

```
SELECT COUNT(*) numeroUtente FROM UtenteVacina
INNER JOIN Vacina
ON Vacina.idVacina=UtenteVacina.idVacina
WHERE Vacina.nome='Tetano';
```

5. Quais os IDs dos médicos de área de ação "Cirurgica" que estiveram na equipa de urgência ativa no dia 26 de Abril de 2016 às 13:00?

```
SELECT Medico.numeroFuncionario FROM Medico
INNER JOIN Funcionario
ON(Medico.numeroFuncionario=Funcionario.numeroFuncionario)
INNER JOIN EquipaUrgencia
ON(EquipaUrgencia.dataInicio<='2016-04-26 13:00' AND
EquipaUrgencia.dataFim >'2016-04-26 13:00')
WHERE (Medico.areaAcao='Cirurgica' AND
Funcionario.idEquipa=EquipaUrgencia.idEquipa);
```

6. Quais os IDs dos utentes cuja triagem resultou num nível de prioridade "emergente" ou "muito urgente"?

```
SELECT numeroUtente FROM Triagem
WHERE Triagem.prioridade IN('emergente','muitoUrgente');
```

7. Quais os nomes, idades e sexo dos utentes que ficaram mais do que 1 dia nas urgências?

```
SELECT DISTINCT nome, idade, sexo
FROM
    (Pessoa
    INNER JOIN
    Utente
    ON Pessoa.idPessoa = Utente.idPessoa
    INNER JOIN
    Inscricao
    ON Utente.numeroUtente = Inscricao.numeroUtente
    INNER JOIN
    Checkout
    ON Utente.numeroUtente = Checkout.numeroUtente)
GROUP BY Pessoa.nome
HAVING (julianday(Checkout.dataSaida) -
julianday(Inscricao.horaEntrada) > 1);
```

8. Quais os numeroFuncionario dos funcionarios que participaram na equipa com urgênciaID=1?

```
SELECT Funcionario.funcao, Funcionario.numeroFuncionario
FROM
    (Funcionario
    INNER JOIN
    Inscricao
    ON Funcionario.numeroFuncionario = Inscricao.numeroFuncionario )
     WHERE(Inscricao.urgenciaID = 1)

UNION

SELECT Funcionario.funcao, Funcionario.numeroFuncionario
FROM
    (Funcionario
    INNER JOIN
    Checkout
    ON Funcionario.numeroFuncionario = Checkout.numeroFuncionario )
     WHERE(Checkout.urgenciaID = 1)

UNION

SELECT Funcionario.funcao, Funcionario.numeroFuncionario
FROM
    (Funcionario
    INNER JOIN
    Triagem
    ON Funcionario.numeroFuncionario = Triagem.numeroFuncionario )
     WHERE(Triagem.urgenciaID = 1)

UNION

SELECT Funcionario.funcao, Funcionario.numeroFuncionario
FROM
    (Funcionario
    INNER JOIN
    Diagnostico
    ON Funcionario.numeroFuncionario = Diagnostico.numeroFuncionario )
     WHERE(Diagnostico.urgenciaID = 1)

UNION

SELECT Funcionario.funcao, Funcionario.numeroFuncionario
FROM
    (Funcionario
    INNER JOIN
    Tratamento
    ON Funcionario.numeroFuncionario = Tratamento.numeroFuncionario )
     WHERE(Tratamento.urgenciaID = 1);
```

9. Qual o ID do médico que realizou mais cirurgias?

```
SELECT numeroFuncionario, MAX(cirurgias)
FROM(
SELECT Medico.numeroFuncionario, COUNT(*) as cirurgias
FROM
    (Medico
    INNER JOIN
    Tratamento
    ON Medico.numeroFuncionario = Tratamento.numeroFuncionario
    INNER JOIN
    Cirurgia
```

```
    ON Tratamento.urgenciaID = Cirurgia.urgenciaID)
GROUP BY Medico.numeroFuncionario);
```

10. Qual o nome e a idade do 1º e 2º utente mais velhos que foram às urgências e qual o ID dos médicos que os atenderam?

```
SELECT Pessoa.nome, MAX(Pessoa.idade), Tratamento.numeroFuncionario
FROM
    (Tratamento
    LEFT JOIN
    Utente
    ON Tratamento.numeroUtente = Utente.numeroUtente
    LEFT JOIN
    Pessoa
    ON Pessoa.idPessoa = Utente.idPessoa)

UNION

SELECT Pessoa.nome, MAX(Pessoa.idade), Tratamento.numeroFuncionario
FROM
    (Tratamento
    LEFT JOIN
    Utente
    ON Tratamento.numeroUtente = Utente.numeroUtente
    LEFT JOIN
    Pessoa
    ON Pessoa.idPessoa = Utente.idPessoa)
WHERE (Pessoa.idade <
        (SELECT MAX(Pessoa.idade)
        FROM
            (Tratamento
            LEFT JOIN
            Utente
            ON Tratamento.numeroUtente = Utente.numeroUtente
            LEFT JOIN
            Pessoa
            ON Pessoa.idPessoa = Utente.idPessoa)));
```

11. Quantas pessoas registadas no sistema não são médicas?

```
SELECT COUNT(*) idPessoa
FROM
    (Pessoa
    LEFT JOIN
    Funcionario
    ON Pessoa.idPessoa = Funcionario.idPessoa
    LEFT JOIN
    Medico
    ON Funcionario.numeroFuncionario = Medico.numeroFuncionario)
WHERE Medico.numeroFuncionario IS NULL;
```

Instruções com função de *triggers*

Com o objetivo de evitar a circulação de informação inválida na base de dados, desenvolvemos cinco *triggers*. Três que controlam instruções de inserção, um de atualização e outro de deleção.

1. ***Trigger* que verifica se o líder de uma equipa de urgência é um médico**

```
CREATE TRIGGER Lider
         AFTER INSERT
            ON EquipaUrgencia
      FOR EACH ROW
          WHEN (
    SELECT numeroFuncionario
      FROM Funcionario
     WHERE NOT EXISTS (
                   SELECT *
                     FROM Funcionario
                    WHERE numeroFuncionario == NEW.idLider AND
                          funcao == 'Medico'
                )
    )
BEGIN
    SELECT RAISE(ABORT, 'Lider invalido');
END;
```

Teste do trigger: o funcionário 512 não é um medico.

```
INSERT INTO EquipaUrgencia VALUES(4, '2016-04-25 10:00', '2016-04-26
19:00', 512);
```

2. ***Trigger* que verifica se não há englobamento de turnos ao inserir equipas de urgência, ou seja, que não há um turno dentro do outro.**

```
CREATE TRIGGER Turno
         AFTER INSERT
            ON EquipaUrgencia
      FOR EACH ROW
          WHEN (
    SELECT idLider
      FROM EquipaUrgencia
     WHERE EXISTS (
               SELECT *
                 FROM EquipaUrgencia
                WHERE idEquipa != NEW.idEquipa AND (
                      dataInicio > NEW.dataInicio AND
                      dataFim < NEW.dataFim) OR (
                      dataInicio < NEW.dataInicio AND
                      dataFim > NEW.dataFim)
           )
)
BEGIN
    SELECT RAISE(ABORT, 'Turno invalido.');
END;
```

Teste do trigger: já existe uma equipa que trabalha desde as 10h00 do dia 25 até às 19h00 do dia 26.

```
INSERT INTO EquipaUrgencia VALUES(5, '2016-04-25 09:00', '2016-04-26
20:00', 517);
```

3. ***Trigger* que verifica se a inscrição é feita por um funcionário que esteja na equipa de urgência ativa:**

```
CREATE TRIGGER horaEntrada
         AFTER INSERT
            ON Inscricao
      FOR EACH ROW
          WHEN (
    SELECT numeroFuncionario
      FROM Funcionario
     WHERE NOT EXISTS (
                   SELECT *
                     FROM (
                              Funcionario
                              INNER JOIN
                              EquipaUrgencia ON
(EquipaUrgencia.idEquipa = Funcionario.idEquipa AND
NEW.numeroFuncionario = Funcionario.numeroFuncionario))
WHERE EquipaUrgencia.dataInicio < NEW.horaEntrada AND
EquipaUrgencia.dataFim > NEW.horaEntrada)
)
BEGIN
    SELECT RAISE(ABORT, 'Hora de entrada invalida');
END;
```

Teste do trigger: o funcionário 507 só trabalhou na equipa ativa no dia 23 de Abril.

```
INSERT INTO Inscricao(urgenciaID, numeroUtente, numeroFuncionario,
horaEntrada) VALUES(12, 1007, 507, '2016-04-22 18:21');
```

4. ***Trigger* que, ao apagar uma especialidade da lista do hospital, coloca o valor da especialidade dos médicos correspondentes a NULL.**

```
CREATE TRIGGER EspecialidadeDelete
         AFTER DELETE
            ON Especialidade
      FOR EACH ROW
BEGIN
    UPDATE Medico
       SET idEspecialidade = NULL
     WHERE Medico.idEspecialidade = OLD.idEspecialidade;
END;
```

Teste do trigger: apagar a especialidade “Cardiologia” do hospital atualiza os valores dos médicos com essa especialidade.

```
DELETE FROM Especialidade
WHERE Especialidade.designacao = 'Cardiologia';
```

5. ***Trigger* que, ao se atualizar o ID de uma sala cirúrgica, atualiza-o também nas cirurgias que lá foram realizadas.**

```
CREATE TRIGGER SalaCirurgicaUpdate
         AFTER UPDATE OF idSalaCirurgica
            ON SalaCirurgica
      FOR EACH ROW
BEGIN
    UPDATE Cirurgia
       SET idSalaCirurgica = NEW.idSalaCirurgica
     WHERE Cirurgia.idSalaCirurgica = OLD.idSalaCirurgica;
END;
```

Teste do trigger: a sala com ID 3045 passa a ter ID 500, pelo que todas as cirurgias lá efetuadas passam a ter 'idSalaCirurgica' igual a 500.

```
UPDATE SalaCirurgica SET idSalaCirurgica = 500, bloco = 'D' WHERE
idSalaCirurgica = 3045;
```