UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

DEPARTAMENTO ACADÊMICO DE INFORMÁTICA

TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

CAIO SCHNEPPER

ERIC KOITI UETA

EVANDRO ALESSI CORREIA DOS SANTOS

MATHEUS BEZERRA FILHO

PROJETO FINAL DE FUNDAMENTOS DE BANCO DE DADOS

PONTA GROSSA

2018

CAIO SCHNEPPER

ERIC KOITI UETA

EVANDRO ALESSI CORREIA DOS SANTOS

MATHEUS BEZERRA FILHO

PROJETO FINAL DE FUNDAMENTOS DE BANCO DE DADOS

Projeto final de Fundamentos de Banco de Dados apresentado como requisito parcial para obtenção da nota e conclusão da matéria, Departamento Acadêmico de Informática da Universidade Tecnológica Federal do Paraná.

Professora: Simone de Almeida

PONTA GROSSA

2018

SUMÁRIO

1 INTRODUÇÃO

Este projeto representa o processo que leva à criação de um banco de dados para a administração de um banco de sangue, partindo da criação do modelo entidade e relacionamento e finalizando no banco físico seguindo o que foi descrito no enunciado do projeto, utilizando views para realizar a consulta de dados mais avançados e stored procedure para manipulação desses dados. Este projeto será utilizado como método avaliativo da equipe a fim de obter nota parcial para a disciplina de Fundamentos de Banco de Dados, demandando a aplicação de conhecimentos e conceitos aprendidos durante o semestre.

A estrutura do projeto é: modelo de entidade e relacionamento, modelo relacional, scripts de criação do banco de dados, scripts de criação das stored procedures, scripts de criação dos índices e script de criação das views.

2 DESENVOLVIMENTO

O objetivo almejado com a criação do banco de dados foi de além do armazenamento dos dados referentes às doações de sangue dos doadores e detecção de possíveis doenças presentes do sangue, também gerenciar esses dados de maneira eficiente, e dessa forma viabilizar o armazenamento de doações ou exames passados, mantendo então a base de dados a mais conservadora possível em relação aos dados contidos nela. A modelagem foi desenvolvida com foco na optimização da base de dados, tendo o seguinte funcionamento:

- Para iniciar o processo de doação, o doador deve primeiramente fazer o exame de anemia, caso o nível de hemoglobina do doador esteja a baixo do estabelecido para a doação, o processo de doação é interrompido.

- Para casos de doadores frequentes, o sistema irá fazer a busca da última data da última doação realizada, e para realizar uma nova doação o período não poderá ser inferior a 90 dias para homens ou 120 dias para mulheres.

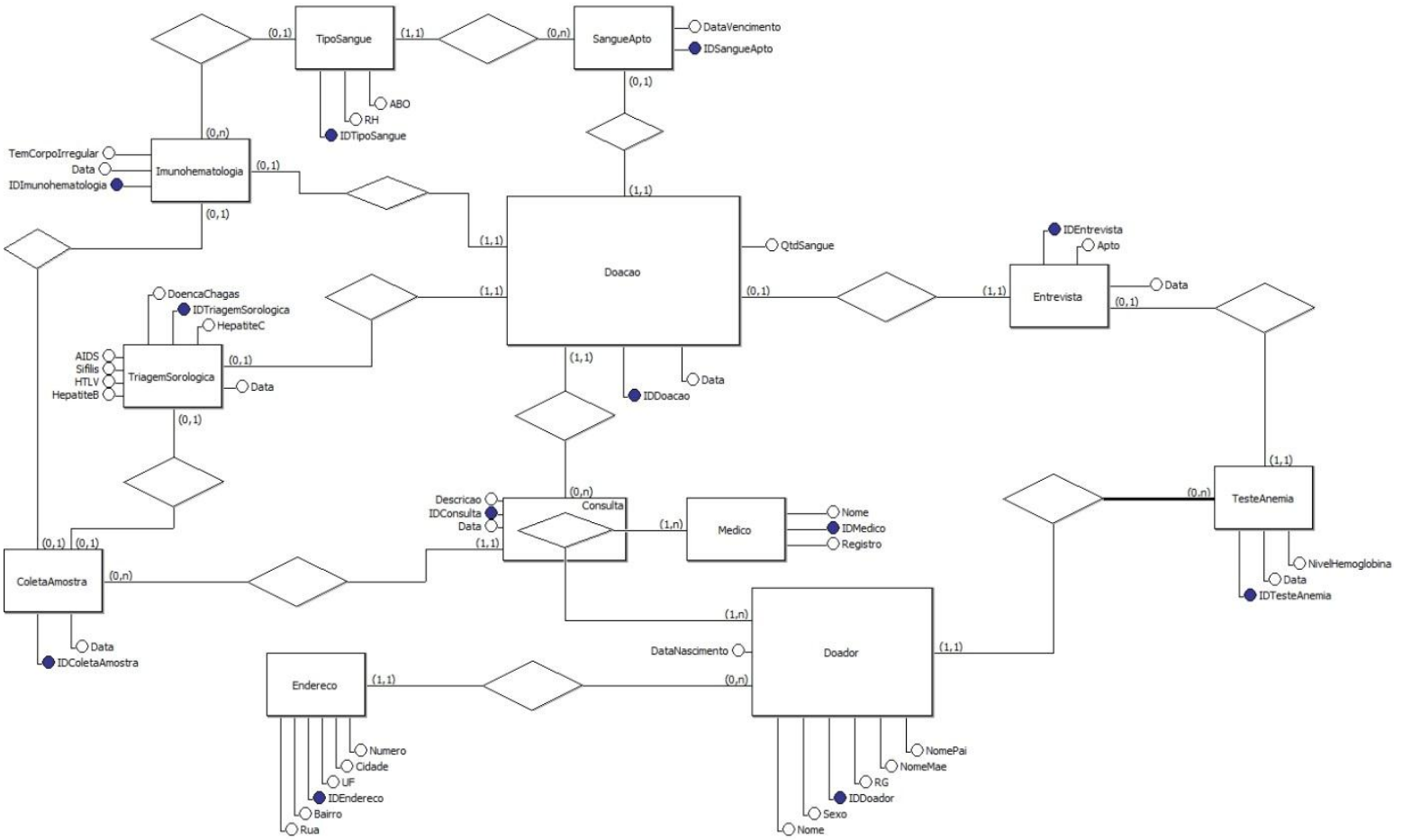
- Caso seja feita a constatação de que o doador é inapto a realizar a doação, seus dados ficam salvos na base de dados.

- Para realmente concluir uma doação, é necessário que o doador atenda a todos os requisitos da fundação pró-sangue, tanto em exames quanto na triagem clínica.

- A exclusão de dados apenas é permitida, se houver casos de tuplas apenas com chaves.

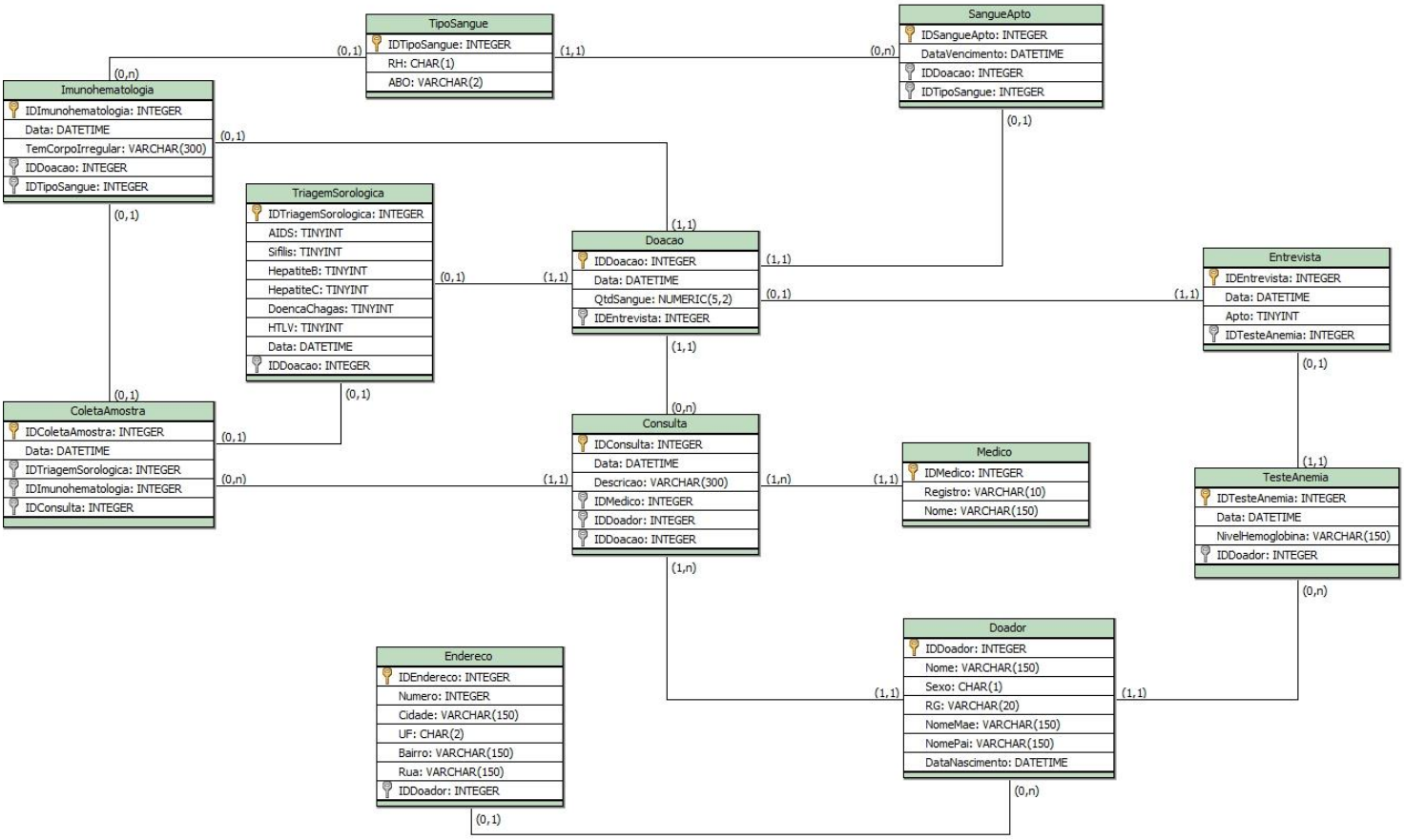
A análise do enunciado que resultou no modelo apresentado baseou-se no fato de que para finalizar o processo de doação de sangue, o doador deve passar por uma série de exames necessários que são feitos individualmente entre si para liberação de sua efetiva doação. Como também o controle de futuras novas doações feitas por um mesmo doador. Portanto, não há necessidade de preocupação caso o doador seja convocado para uma consulta médica ou para repetição de exame.

2.1 MODELO DE ENTIDADE E RELACIONAMENTO



Modelo Conceitual

2.2 MODELO RELACIONAL



Modelo lógico

2.3 SCRIPT SQL DE CRIAÇÃO DA BASE DE DADOS

```
CREATE DATABASE BANCOSANGUE;
USE BANCOSANGUE;

CREATE TABLE Doador (
 IDDoador INTEGER NOT NULL AUTO_INCREMENT,
 Nome VARCHAR(150),
 Sexo CHAR(1),
 RG VARCHAR(20),
 NomeMae VARCHAR(150),
 NomePai VARCHAR(150),
 DataNascimento DATETIME,
   PRIMARY KEY(IDDoador)
);

CREATE TABLE Endereco (
 IDEndereco INTEGER NOT NULL AUTO_INCREMENT,
 Numero INTEGER,
 Cidade VARCHAR(150),
 UF CHAR(2),
 Bairro VARCHAR(150),
 Rua VARCHAR(150),
   IDDoador INTEGER NOT NULL,
 PRIMARY KEY(IDEndereco),
   FOREIGN KEY(IDDoador) REFERENCES Doador (IDDoador)
         ON UPDATE CASCADE
      ON DELETE CASCADE
);

CREATE TABLE TesteAnemia (
 IDTesteAnemia INTEGER NOT NULL AUTO_INCREMENT,
 Data DATETIME,
 NivelHemoglobina VARCHAR(150),
 IDDoador INTEGER NOT NULL,
   PRIMARY KEY(IDTesteAnemia),
   FOREIGN KEY(IDDoador) REFERENCES Doador (IDDoador)
         ON UPDATE CASCADE
      ON DELETE CASCADE
```

```
);

CREATE TABLE Entrevista (
 IDEntrevista INTEGER NOT NULL AUTO_INCREMENT,
 Data DATETIME,
 Apto TINYINT,
 IDTesteAnemia INTEGER NOT NULL,
   PRIMARY KEY(IDEntrevista),
 FOREIGN KEY(IDTesteAnemia) REFERENCES TesteAnemia(IDTesteAnemia)
           ON UPDATE RESTRICT
       ON DELETE RESTRICT
);

CREATE TABLE Doacao (
 IDDoacao INTEGER NOT NULL AUTO_INCREMENT,
 Data DATETIME,
 QtdSangue NUMERIC(5,2),
 IDEntrevista INTEGER NOT NULL,
   PRIMARY KEY(IDDoacao),
 FOREIGN KEY(IDEntrevista) REFERENCES Entrevista(IDEntrevista)
           ON UPDATE RESTRICT
       ON DELETE RESTRICT
);

CREATE TABLE TriagemSorologica (
 IDTriagemSorologica INTEGER NOT NULL AUTO_INCREMENT,
 AIDS TINYINT,
 Sifilis TINYINT,
 HepatiteB TINYINT,
 HepatiteC TINYINT,
 DoencaChagas TINYINT,
 HTLV TINYINT,
 Data DATETIME,
 IDDoacao INTEGER NOT NULL,
   PRIMARY KEY(IDTriagemSorologica),
   FOREIGN KEY(IDDoacao) REFERENCES Doacao (IDDoacao)
      ON UPDATE CASCADE
```

```
        ON DELETE CASCADE
);

CREATE TABLE TipoSangue (
 IDTipoSangue INTEGER NOT NULL AUTO_INCREMENT,
 RH CHAR(1),
 ABO VARCHAR(2),
   PRIMARY KEY(IDTipoSangue)
);

CREATE TABLE Imunohematologia (
 IDImunohematologia INTEGER NOT NULL AUTO_INCREMENT,
 Data DATETIME,
 TemCorpoIrregular VARCHAR(300),
 IDDoacao INTEGER NOT NULL,
 IDTipoSangue INTEGER NOT NULL,
   PRIMARY KEY(IDImunohematologia),
 FOREIGN KEY(IDTipoSangue) REFERENCES TipoSangue(IDTipoSangue)
            ON UPDATE RESTRICT
       ON DELETE RESTRICT,
   FOREIGN KEY(IDDoacao) REFERENCES Doacao (IDDoacao)
       ON UPDATE CASCADE
       ON DELETE CASCADE
);

CREATE TABLE SangueApto (
 IDSangueApto INTEGER NOT NULL AUTO_INCREMENT,
 DataVencimento DATETIME,
 IDDoacao INTEGER NOT NULL,
 IDTipoSangue INTEGER NOT NULL,
   PRIMARY KEY(IDSangueApto),
 FOREIGN KEY(IDTipoSangue) REFERENCES TipoSangue(IDTipoSangue)
            ON UPDATE RESTRICT
       ON DELETE RESTRICT,
   FOREIGN KEY(IDDoacao) REFERENCES Doacao (IDDoacao)
       ON UPDATE CASCADE
       ON DELETE CASCADE
```

```
);

CREATE TABLE Medico (
 IDMedico INTEGER NOT NULL AUTO_INCREMENT,
 Registro VARCHAR(10),
 Nome VARCHAR(150),
   PRIMARY KEY(IDMedico)
);

CREATE TABLE Consulta (
 IDConsulta INTEGER NOT NULL AUTO_INCREMENT,
 Data DATETIME,
 Descricao VARCHAR(300),
 IDMedico INTEGER NOT NULL,
 IDDoador INTEGER NOT NULL,
 IDDoacao INTEGER NOT NULL,
   PRIMARY KEY(IDConsulta),
 FOREIGN KEY(IDMedico) REFERENCES Medico(IDMedico)
          ON UPDATE RESTRICT
      ON DELETE RESTRICT,
 FOREIGN KEY(IDDoador) REFERENCES Doador(IDDoador)
          ON UPDATE RESTRICT
      ON DELETE RESTRICT,
   FOREIGN KEY(IDDoacao) REFERENCES Doacao (IDDoacao)
      ON UPDATE RESTRICT
      ON DELETE RESTRICT
);

CREATE TABLE ColetaAmostra (
 IDColetaAmostra INTEGER NOT NULL AUTO_INCREMENT,
 Data DATETIME,
 IDTriagemSorologica INTEGER,
 IDImunohematologia INTEGER,
 IDConsulta INTEGER NOT NULL,
   PRIMARY KEY(IDColetaAmostra),
 FOREIGN KEY(IDImunohematologia) REFERENCES
Imunohematologia(IDImunohematologia)
```

```
        ON UPDATE RESTRICT
    ON DELETE RESTRICT,
  FOREIGN KEY(IDTriagemSorologica) REFERENCES TriagemSorologica (IDTriagemSorologica)
    ON UPDATE RESTRICT
    ON DELETE RESTRICT,
  FOREIGN KEY(IDConsulta) REFERENCES Consulta (IDConsulta)
    ON UPDATE CASCADE
    ON DELETE CASCADE
);
```

2.4 SCRIPT SQL DE CRIAÇÃO DAS PROCEDURES

-- Insere Doador

```
call SP_INSERTDOADOR('4', 'm', '2', '1', '1', '2018-12-01', 1,'4', 'PR', 'vila', 'manoel');

drop procedure SP_INSERTDOADOR;
DELIMITER &&
CREATE PROCEDURE SP_INSERTDOADOR
(IN pNome VARCHAR(150), pSexo CHAR(1), pRG VARCHAR(20), pNomeMae VARCHAR(150), pNomePai VARCHAR(150), pDataNascimento DATETIME, pNumero INTEGER, pCidade VARCHAR(150), pUF CHAR(2), pBairro VARCHAR(150), pRua VARCHAR(150))
BEGIN
 IF (NOT EXISTS(SELECT IDDoador
                FROM Doador
```

```
                    WHERE RG = pRG)) THEN
        BEGIN

                INSERT INTO Doador (Nome, Sexo, RG, NomeMae, NomePai, DataNascimento)
                    VALUES (pNome, pSexo, pRG, pNomeMae, pNomePai, pDataNascimento);

                INSERT INTO Endereco(Numero, Cidade, UF, Bairro, Rua, IDDoador)
                    VALUES (pNumero, pCidade, pUF, pBairro, pRua, LAST_INSERT_ID());

         END;
           END IF;
        END&&
        DELIMITER ;
```

--Altera Doador

```
        DELIMITER &&
        CREATE PROCEDURE SP_ALTERDOADOR
        (IN pNome VARCHAR(150), pSexo CHAR(1), pRG VARCHAR(20), pNomeMae VARCHAR(150), pNomePai VARCHAR(150), pDataNascimento DATETIME)
        BEGIN
         IF (EXISTS(SELECT IDDoador
                    FROM Doador
                    WHERE IDDoador = pIDDoador)) THEN
         BEGIN
                    UPDATE Endereco
                        SET  Nome = pNome, Sexo = pSexo, RG = pRG, NomeMae = pNomeMae, NomePai = pNomePai, DataNascimento = pDataNascimento
                        WHERE IDDoador = pIDDoador;
```

```
 END;
 END IF;
END&&
DELIMITER ;
```

--Altera Endereco

```
select * FROM DOADOR;


call SP_ALTERENDERECO(1, 1,'4', 'PR', 'vila', 'manoel');

DELIMITER &&
CREATE PROCEDURE SP_ALTERENDERECO
(IN pIDEndereco INTEGER, pNumero INTEGER, pCidade  VARCHAR(150), pUF  CHAR(2), pBairro  VARCHAR(150), pRua  VARCHAR(150), pIDDoador INTEGER)
BEGIN
 IF (EXISTS(SELECT IDENDERCO
                FROM ENDERECO
                WHERE IDEndereco = pIDEndereco and IDDoador = pIDDoador)) THEN
 BEGIN
        UPDATE Endereco
                SET  Numero = pNumero, Cidade = pCidade, UF = pUF, Bairro = pBairro, Rua = pRua, IDDoador = pIDDoador
        WHERE IDEndereco = pIDEndereco;
 END;
 END IF;
END&&
DELIMITER ;


select * FROM ENDERECO;
```

```
--Insere Medico
call SP_INSERTMEDICO('Evandro', '12135');

DELIMITER &&
CREATE PROCEDURE SP_INSERTMEDICO
(IN pNome VARCHAR(150), pRegistro VARCHAR(10))
BEGIN
 IF (NOT EXISTS(SELECT IDMedico
                FROM Medico
                WHERE Registro = pRegistro)) THEN
 BEGIN
        INSERT INTO Medico (Nome, Registro)
                VALUES (pNome, pRegistro);
 END;
   END IF;
END&&
DELIMITER ;

call SP_INSERTMEDICO(1, 'Evandro', '12135');

DELIMITER &&
CREATE PROCEDURE SP_ALTERMEDICO
(IN pIDMedico INTEGER, pNome VARCHAR(150), pRegistro VARCHAR(10))
BEGIN
 IF (EXISTS(SELECT IDMedico
                FROM Medico
                WHERE IDMedico = pIDMedico and Registro = pRegistro))
THEN
 BEGIN
        UPDATE Medico
                SET  Nome = pNome, Registro = pRegistro
       WHERE IDMedico = pIDMedico;
 END;
```

```
 END IF;
END&&
DELIMITER ;
select * FROM MEDICO;
```

--Insere TesteAnemia

```
DELIMITER &&
CREATE PROCEDURE SP_INSERTTESTEANEMIA
(IN pData datetime, pNivelHemoglobina VARCHAR(150), pIDDoador INTEGER)
BEGIN
 IF (pIDDoador <> NULL
   AND EXISTS(SELECT IDDoador
         FROM Doador
      WHERE IDDoador = pIDDoador)) THEN
 BEGIN
         INSERT INTO TesteAnemia(Data, NivelHemoglobina, IDDoador)
               VALUES (pData, pNivelHemoglobina, pIDDoador);
 END;
   END IF;
END&&
DELIMITER ;
```

--Altera Medico

```
DELIMITER &&
CREATE PROCEDURE SP_ALTERMEDICO
(IN pIDTesteAnemia INTEGER, pData datetime, pNivelHemoglobina VARCHAR(150), pIDDoador INTEGER)
BEGIN
 IF (EXISTS(SELECT IDTesteAnemia
               FROM TesteAnemia
               WHERE pIDTesteAnemia = pIDTesteAnemia)) THEN
 BEGIN
```

```
            UPDATE TesteAnemia
                    SET  Data = pData, NivelHemoglobina = pNivelHemoglobina
                WHERE IDTesteAnemia = pIDTesteAnemia;
        END;
        END IF;
       END&&
       DELIMITER ;
```

--Insere Entrevista

```
       DELIMITER &&
       CREATE PROCEDURE SP_INSERTENTREVISTA
       (IN pData datetime, pApto TINYINT, pIDTesteAnemia INTEGER)
       BEGIN
        IF (pIDDoador <> NULL
          AND EXISTS(SELECT IDTesteAnemia
                FROM TesteAnemia
            WHERE IDTesteAnemia = pIDTesteAnemia)) THEN
        BEGIN
                INSERT INTO Entrevista(Data, NivelHemoglobina, IDTesteAnemia)
                        VALUES (pData, pApto, pIDTesteAnemia);
        END;
          END IF;
       END&&
       DELIMITER ;
```

--Altera Entrevista

```
       DELIMITER &&
       CREATE PROCEDURE SP_ALTERENTREVISTA
       (IN pIDEntrevista INTEGER, pData datetime, pApto TINYINT, pIDTesteAnemia
INTEGER)
       BEGIN
        IF (EXISTS(SELECT IDTesteAnemia
                        FROM TesteAnemia
```

```
                    WHERE pIDTesteAnemia = pIDTesteAnemia)) THEN
 BEGIN
        UPDATE Entrevista
                SET   Data = pData, Apto = pApto, IDTesteAnemia =
pIDTesteAnemia
        WHERE IDEntrevista = pIDEntrevista;
 END;
 END IF;
END&&
DELIMITER ;
```

--Insere Doacao

```
DELIMITER &&
CREATE PROCEDURE SP_INSERTDOACAO
(IN pData datetime, pQtdSangue numeric, pIDEntrevista INTEGER)
BEGIN
 IF (pIDEntrevista <> NULL
   AND EXISTS(SELECT IDEntrevista
        FROM Entrevista
     WHERE IDEntrevista = pIDEntrevista)) THEN
 BEGIN
        INSERT INTO Doacao(Data, QtdSangue, IDEntrevista)
                VALUES (pData, pQtdSangue, pIDEntrevista);
 END;
   END IF;
END&&
DELIMITER ;
```

--Altera Doacao

```
DELIMITER &&
CREATE PROCEDURE SP_ALTERDOACAO
(IN pIDEntrevista INTEGER, pData datetime, pQtdSangue numeric,
pIDEntrevista INTEGER)
BEGIN
```

```
        IF (EXISTS(SELECT IDEntrevista
                        FROM Entrevista
                        WHERE IDEntrevista = pIDEntrevista)) THEN
        BEGIN
                UPDATE Doacao
                        SET  Data = pData, QtdSangue = pQtdSangue, IDEntrevista =
pIDEntrevista
                WHERE IDEntrevista = pIDEntrevista;
        END;
        END IF;
       END&&
       DELIMITER ;
```

2.5 SCRIPT SQL DE CRIAÇÃO DOS INDÍCES

```
CREATE INDEX Index_DoadorNome ON Doador(Nome);
CREATE INDEX Index_DoadorRG ON Doador(RG);
CREATE INDEX Index_SangueAptoDataVencimento ON SangueApto(DataVencimento);
CREATE INDEX Index_EntrevistaApto ON Entrevista(Apto);
CREATE INDEX Index_TesteAnemiaData ON TesteAnemia(Data);
CREATE INDEX Index_EnderecoBairro ON Endereco(Bairro);
CREATE INDEX Index_EnderecoRua ON Endereco(Rua);
CREATE INDEX Index_TriagemSorologica ON TriagemSorologica(Data);
CREATE INDEX Index_MedicoRegistro ON Medico(Registro);
CREATE INDEX Index_ColetaAmostraData ON ColetaAmostra(Data);
CREATE INDEX Index_ImunohematologiaData ON Imunohematologia(Data);
CREATE INDEX Index_TipoSangueABO ON TipoSangue(ABO);
```

2.6 SCRIPT SQL DE CRIAÇÃO DAS VIEWS

```
CREATE VIEW DOACOES (Nome, ABO, RH, Ultima_Doacao)
As
SELECT d.NOME, ts.ABO, ts.RH, MAX(dc.Data) AS data
FROM Doador d, TesteAnemia ta, Entrevista e, Doacao dc, Imunohematologia i,
TipoSangue ts
WHERE ta.IDDoador = d.IDDoador AND e.IDTesteAnemia = ta.IDTesteAnemia AND
dc.IDEntrevista = e.IDEntrevista AND i.IDDoacao = dc.IDDoacao
    AND ts.IDTipoSangue = i.IDTipoSangue
GROUP BY d.IDDoador
ORDER BY ts.ABO ASC, ts.RH DESC, MAX(dc.Data) ASC;

CREATE VIEW CARTEIRINHA (Nome, Data_Doacao, ABO, RH,
AntiCorpo_Irregular, HepatiteB, HepatiteC, Chagas, Sifilis, AIDS, HTLV)
As
SELECT d.NOME, dc.Data, ts.ABO, ts.RH, i.TemCorpoIrregular, t.HepatiteB,
t.HepatiteC, t.DoencaChagas, t.Sifilis, t.AIDS, t.HTLV,
FROM Doador d, TesteAnemia ta, Entrevista e, Doacao dc, Imunohematologia i,
TipoSangue ts, TriagemSorologica t,
WHERE ta.IDDoador = d.IDDoador AND e.IDTesteAnemia = ta.IDTesteAnemia AND
dc.IDEntrevista = e.IDEntrevista AND i.IDDoacao = dc.IDDoacao
    AND ts.IDTipoSangue = i.IDTipoSangue AND t.IDDoacao = dc.IDDoacao,
GROUP BY ts.ABO, ts.RH, d.NOME, d.Data DESC;


CREATE VIEW DOACOES_GERAIS
SELECT ts.ABO, ts.RH, COUNT(*)  AS quantidade
FROM Doador d, TesteAnemia ta, Entrevista e, Doacao dc, Imunohematologia i,
TipoSangue ts
WHERE ta.IDDoador = d.IDDoador AND e.IDTesteAnemia = ta.IDTesteAnemia AND
dc.IDEntrevista = e.IDEntrevista AND i.IDDoacao = dc.IDDoacao
    AND ts.IDTipoSangue = i.IDTipoSangue AND dc.data > (CURDATE() - INTERVAL
5 YEAR)
GROUP BY CONCAT(ts.ABO, ts.RH)
ORDER BY COUNT(*) ASC;
```

3 CONCLUSÃO

A criação de projetos finais em quaisquer disciplinas que necessitam de experiência, tanto teórica quanto prática é de extrema importância para o aprendizado do aluno, verificação e fixação de conteúdos aprendidos durante o semestre letivo.

O trabalho feito foi de extrema importância, não só pelo conhecimento adquirido, mas também pela possibilidade de colocar em pratica tudo o que foi ministrado durante as aulas.

A resolução do problema que foi apresentada neste documento foi feita em conjunto, levando em conta a análise de cada membro do grupo, chegando a um consentimento pela equipe, tendo então, inúmeras maneiras de se resolver o mesmo problema.