universidade de aveiro

Relatório do Trabalho Final

Bases de Dados 42532

Mestrado integrado em Engenharia de Computadores e Telemática

2019/2020

Realizado por

Jacinto Lufilakio 89162, Joaquim Ramos 88812.

Secções

Análise de Requisitos

Surgiu a necessidade de uma escola primária de ter acesso mais rápido à informação dos alunos. Essa escola tem a informação dos alunos armazenada de forma física com documentos em papel, arrumados em múltiplas capas. Para uma funcionária poder afirmar que certa pessoa está autorizada a levantar um aluno, é necessário procurar a capa onde está arrumada a ficha de inscrição do aluno, procurar a ficha de inscrição do aluno e agora que tem a ficha já pode confirmar se a pessoa está autorizada a recolher ou não o aluno. Este processo de ir à procura nas capas para obter informação dos alunos é demorado e ineficiente, porque pode ocorrer que o documento que tenha a informação necessária esteja na posse de alguém e não esteja na capa.

Então é necessário passar os dados dos alunos para um só local que permita o acesso rápido a eles. Este é o objetivo da base de dados que desenvolvemos.

Para desenvolvermos a base de dados foi necessário perceber que informação é que precisava de ser guardada.

A escola tem 4 turmas todos os anos letivos, cada turma é definida pelo grau de escolaridade (1ª à 4ª classe), ano letivo e professor. Num ano letivo existe uma turma para cada grau de escolaridade. Existem atividades extracurriculares que são independentes do ano letivo e grau de escolaridade, são definidas pelo nome e orientadas por um professor.

Ao longo do ano letivo são realizadas visitas de estudo. No mesmo dia só acontece uma visita de estudo, tem um nome, hora de chegada e hora de partida. Os alunos participantes não têm de ser necessariamente da mesma turma, isto é, pode ser visita de estudo para vários graus de escolaridade.

Sobre os professores sabe-se o nome, telemóvel e email.

Para os alunos é necessário guardar o nome, data de nascimento, morada, problemas de saúde, medicação que o

aluno toma, restrições alimentares e se é admitido o uso de antipiréticos para febres/outras.

Os alunos têm um encarregado de educação, que é um adulto com alguma relação de parentesco com o aluno e pode levantar o aluno. Mas podem existir outros adultos com relações de parentesco que podem levantar um aluno.

Sobre os encarregados de educação é necessário guardar o nome, morada, telemóvel, telefone, telefone de trabalho, email, local de trabalho e profissão. Telefone e telefone de trabalho não são obrigatórios. Local de trabalho não existe caso o encarregado de educação seja desempregado.

Sobre os adultos que podem levantar um ou mais alunos só é necessário guardar o nome e número de telemóvel.

Um aluno pode pertencer a várias turmas, mas só uma por ano letivo porque não pode estar em graus de escolaridade diferentes no mesmo ano letivo. Um aluno pode pertencer a várias atividades extracurriculares.

Modelo Entidade/Relação

Perante os resultados da análise de requisitos chegou-se à conclusão que era necessário definir as seguintes entidades e relações:

- Entidade “Grau de escolaridade”, que contém o seguinte atributo “grau”;

- Entidade “Ano letivo”, que contém o seguinte atributo “ano letivo”;

- Entidade “Professor”, que contém os seguintes atributos “Nome”, ”Nº Telemóvel”, “Email” e “id prof” que é um número gerado automaticamente;

- Entidade “Turma”, que contém o seguinte atributo “Número” que é um número gerado automaticamente;

-Relação “Grau” entre a entidade “Turma” e a entidade “Grau de escolaridade”, uma turma tem um só grau de escolaridade, mas um grau de escolaridade pode sê-lo de várias turmas;

-Relação “Ano” entre a entidade “Turma” e a entidade “Ano Letivo”, uma turma existe num só ano letivo, mas num ano letivo podem existir várias turmas de graus diferentes;

-Relação “Lecionada por” entre a entidade “Turma” e a entidade “Professor”, uma turma tem um só professor, mas um professor pode lecionar turmas de diferentes anos letivos;

-Entidade “Atividade Extracurricular” que contém o seguinte atributo “Nome”;

-Relação “Orientada por” entre a entidade “Atividade Extracurricular” e a entidade “Professor”, um professor pode orientar várias atividades extracurriculares, e uma atividade extracurricular é orientada por um ou mais professores.

-Entidade “Aluno”, que contém os seguintes atributos “Nome”, “Morada”, “Data de nasc.”, “Problemas de saúde”, “Admnistração de antipireticos”,”id_aluno” que é um número gerado automaticamente, “Medicação obrigatória” e “restrições alimentares”;

-Relação “Pertence” entre a entidade “Turma” e a entidade “Aluno”, um aluno pode pertencer a várias turmas, e uma turma pode ter vários alunos;

-Relação “Tem” entre a entidade “Atividade Extracurricular” e a entidade “Aluno”, um aluno pode ter a várias atividades extracurriculares, e uma atividade extracurricular pode ter vários alunos;

-Entidade “Visita de estudo”, que contém os seguintes atributos “Nome”, ”Data”, “Hora de partida”, “Hora de chegada”;

-Relação “Vai” entre a entidade “Visita de estudo” e a entidade “Aluno”, um aluno pode ir a várias visitas de estudo, e uma visita de estudo pode ter vários alunos.

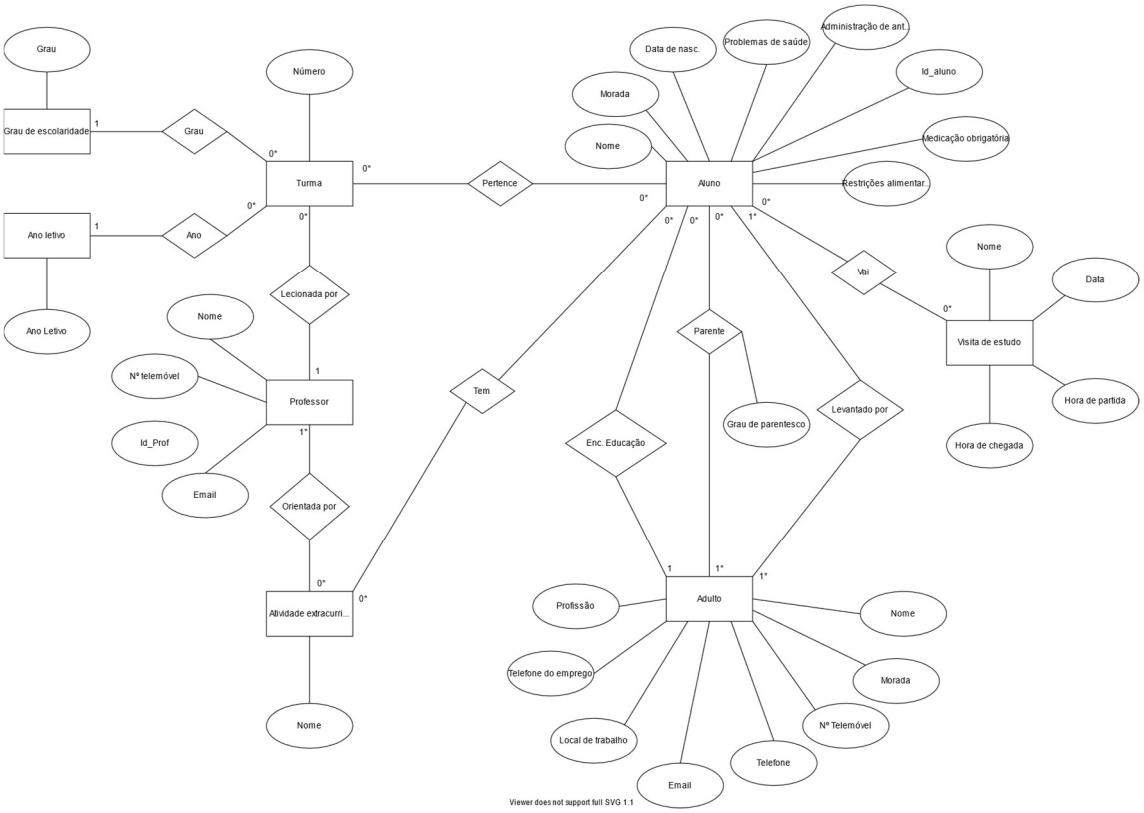
-Entidade “Adulto”, que contém os seguintes atributos “Nome”, ”Morada”, ”Nº de Telemóvel”, “Telefone”, “Email”, “Local de trabalho”, “Telefone de emprego”, “Profissão”.

-Relação “Encarregado de Educação” entre a entidade “Aluno” e a entidade “Adulto”, um aluno só tem um adulto como encarregado de educação, no entanto um adulto pode ser encarregado de educação de vários alunos;

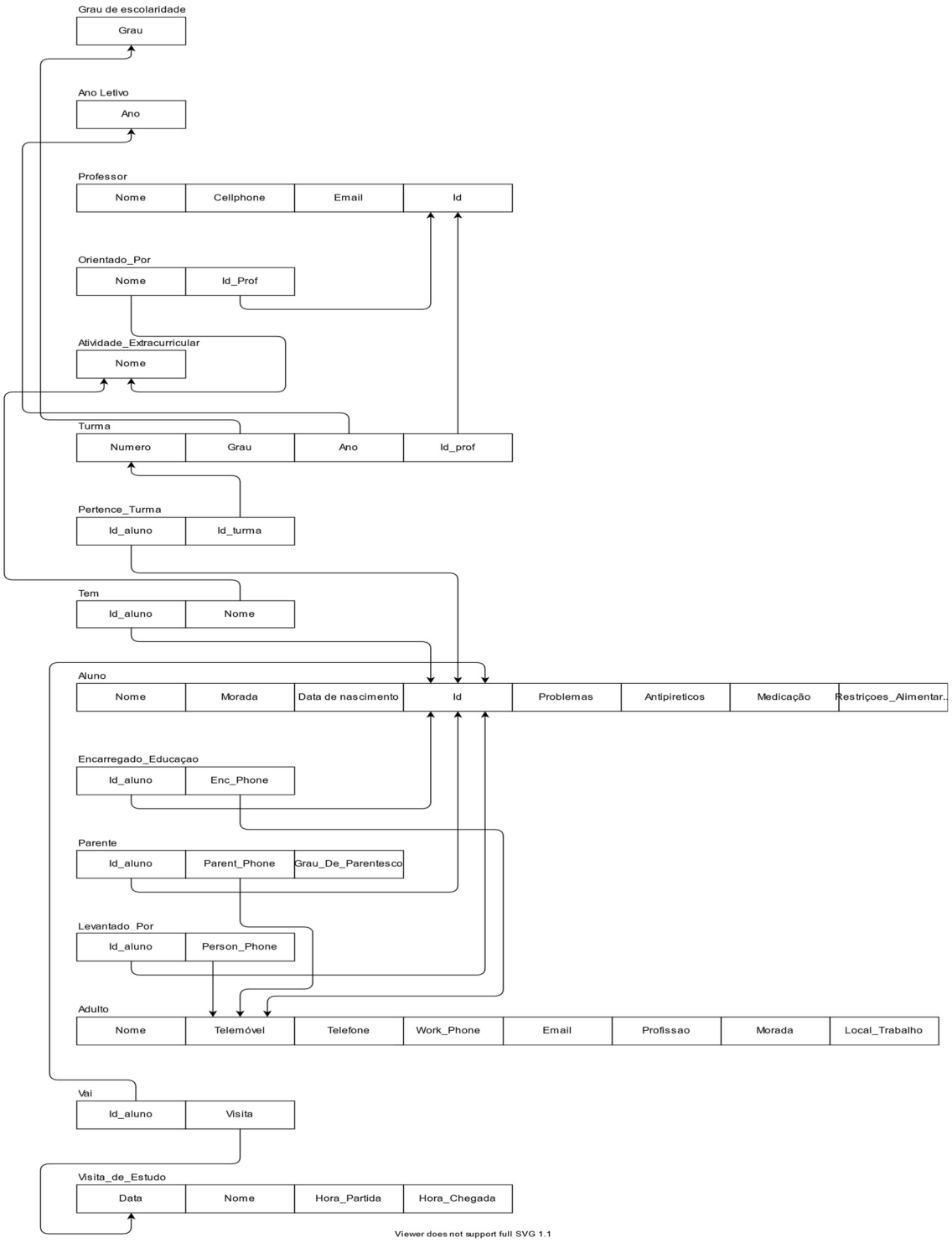
-Relação “Parente” que tem o atributo “grau de parentesco” e é entre as entidades “Aluno” e “Adulto”, que especifica a relação de parentesco entre o adulto e o aluno, um adulto pode ser parente de vários alunos, e um aluno pode ter vários adultos que lhe são parentes (no mínimo um);

-Relação “Levantado por” entre as entidades “Aluno” e “Adulto”, um aluno pode ter vários adultos a levantá-lo (no mínimo um), e um adulto pode levantar vários alunos.

Destas entidades e relações resulta o seguinte diagrama:



Esquema Relacional



SQL DDL

Foram criadas as tabelas na base de dados de acordo com o esquema relacional apresentado na secção anterior. No entanto vamos mostrar a criação da tabela do professor, as outras tabelas podem ser encontradas no ficheiro “SQL_DDL_P3G2.sql” enviado em anexo com este relatório.

```
create table agenda.professor (
    id              int identity(1,1) primary key NOT NULL,
    nome            varchar(70) NOT NULL,
    phone           numeric(9) unique NOT NULL,
    email           varchar(50) unique NOT NULL,
    constraint check_emailP check(email like '%_@__%.__%')
);
```

É nesta tabela que são guardados os professores da escola, todos têm um id único que lhes é atribuído quando são inseridos na tabela, que serve de chave primária, para além disso guardamos o nome, o telemóvel e este deve ser único, e o email que também deve ser único. Criámos uma restrição para quando um tuplo é inserido ou modificado, garantirmos que o email está da forma descrita na restrição “check_email”.

SQL DML

As tabelas foram preenchidas de forma a ter um ano letivo inteiro na base de dados. Foram criadas quatro turmas cada uma com 5 a 11 alunos, e os respetivos pais dos alunos, professores das turmas. Ainda foram criadas visitas de estudo e as atividades extracurriculares junto com os professores das atividades. Estas inserções podem ser encontradas no ficheiro “SQL_DML_P3G2.sql” enviado em anexo com este relatório.

Normalização

O desenho da nossa base de dados resulta num conjunto de relações entre as entidades. Através da normalização conseguimos manter a integridade e preservação dos dados e minimizando a sua redundância. Após criar o modelo relacional da nossa base de dados reparamos que a mesma se encontrava na forma normalizada, isto é a 3ª forma normal:

- os atributos são atómicos;
- não suporta relações dentro de relações;
- não existem dependências parciais;
- não existe dependência funcionais entre atributos não chave;

Índices

Para além dos índices gerados automaticamente com as chaves primárias das tabelas, concluímos que havia necessidade de criar outros índices para diminuir o tempo de resolução das queries. Alguns exemplos:

```
--index criado para facilitar a pesquisa das turmas
create nonclustered index turma_ano_grau on agenda.turma(ano)
include(grau);
```

```
--index criado para diminuir o tempo de pesquisa quando se quer
--retornar todas as turmas de uma ano letivo
create nonclustered index turma_ano on agenda.turma(ano);
```

Stored Procedures

Criámos SPs para realizar a inserção de informação nas tabelas, para a remoção de informação, para servirem de suporte a outras SPs, e para retornar conjuntos de registos para serem mostrados na aplicação. Alguns exemplos:

```
--retorna o id de um aluno
create proc agenda.student_id (@name varchar(70), @birthday date, @ide in
t OUTPUT)
as
    select @ide = aluno.id from agenda.aluno with(index([aluno_nome_data]
))
    where agenda.aluno.nome = @name and agenda.aluno.nasc = @birthday
go
```

Esta SP serve para retornar o id do aluno através do nome e data de nascimento. É uma SP de suporte a outras como por exemplo:

```
--Atividades a que pertence certo aluno
create proc agenda.student_atividade @nome varchar(70), @nasc date
as
    declare @ide int
    exec agenda.student_id @nome, @nasc, @ide OUTPUT;
    select atividade.nome from agenda.atividade join agenda.tem on ativid
ade.nome=tem.nome
    where tem.id_aluno=@ide
go
```

Nesta SP retornamos as atividades a que pertence certo aluno, é uma SP feita para ser chamada pelo formulário em que o utilizador passa o nome e data de nascimento do aluno. Através da SP de suporte referida anteriormente conseguimos saber o id desse aluno e com ele já conseguimos saber as atividades que ele tem.

Todas as SPs estão no ficheiro “StoredProcedures_P3G2.sql” enviado em anexo.

User Defined Functions

Foram usadas várias udfs para verificar a presença de um aluno ou de um adulto nas várias relações onde eles podem constar para ajudar à eliminação de informação. Como por exemplo:

```
--adulto pode levantar
create function agenda.adulto_pl (@nome varchar(70), @nasc date, @phone n
umeric(9)) returns int
as
    begin
        if(select count(*) from agenda.levantado join agenda.aluno on id=
id_aluno where nome=@nome and nasc=@nasc and parent_phone=@phone) = 0
            begin
                return 0
            end
        return 1
    end
go
```

Se o adulto puder levantar o aluno retorna 0 se não puder levantar retorna 1.

Todas as udfs encontram-se no ficheiro “UDFs_P3G2.sql” enviado em anexo.

Login

Para prevenir a utilização indevida do programa foi criada uma tabela de login, onde armazenado o username, password já cifrada e o tipo de utilizador. Dependendo do tipo utilizador, este pode ou não inserir/remover informação das tabelas.

```
create table agenda.logins (
    username varchar(100) not null,
    pass varchar(64) not null,
    tipo varchar(64) not null,
    primary key(username)
);
```

```
insert into agenda.logins (username, pass, tipo)
    values ('func','C340FF19831F32385E62D2F8F45068061F7362F4400F2583A64249E3333BB798','normal')
insert into agenda.logins (username, pass, tipo)
    values ('admin','AB25AF705EEDA0AAD6D61214FF69FB0A4B99FD32D86C30007BDBDD3F9BF7D5EE','mandachuva')
```

Para cifrar a password é usada a seguinte função:

```
private static string CalculateHash(string input)
        {
            //Convert the input to a byte array using specified encoding
            var InputBuffer = Encoding.Unicode.GetBytes(input);
            //Hash the input
            byte[] HashedBytes;
            using (var Hasher = new SHA256Managed())
            {
                HashedBytes = Hasher.ComputeHash(InputBuffer);
            }
            //Return
            return BitConverter.ToString(HashedBytes).Replace("-", string.Empty);
        }
```

Para verificar se o login é válido usamos a seguinte SP:

```
create proc agenda.autorization (@user varchar(100), @pas varchar(64))
as

    select logins.tipo
    from agenda.logins
    where logins.username = @user and logins.pass = @pas
go
```

Se retornar algum quer dizer que o login é válido, e que tem os privilégios do tipo que for retornado.

Existem dois tipos “normal” e “mandachuva”. O tipo “normal” só consegue visualizar informação da base de dados. O tipo “mandachuva” consegue visualizar e inserir/remover e modificar dados da base de dados.

Existem só dois utilizadores:

Username: “func”

Pass: “simple”

Que é do tipo “normal”

Username: “admin”

Pass: “verysecurepalavra”

Que é do tipo “mandachuva”

Estes são as credenciais a utilizar na página de login da aplicação.

Notas

De forma testar a aplicação com o utilizador do professor é necessário mudar a connection string nos seguintes ficheiros:

-“form1.cs” linha 104;

-“form2.cs” linha 107;

-“form3.cs” linha 54;

-“form4.cs” linha 160;

-“form5.cs” linha 112;

A script gerada que comtém o esquema da base de dados e os dados encontra-se em anexo com o seguinte nome “myproject.sql”.´

Os diagramas aqui apresentados também foram enviados em anexo, porque o viewer não suportava palavras sublinhadas ou com cores diferentes.

Percentagem de esforço

Joaquim Ramos 50%

Jacinto Lufilakio 50%