Bases de Dados

Escola Secundária

Ângela Cardoso, Rui Costa e Ricardo Lopes

29 de Maio de 2015

Conteúdo

Capítulo 1

Introdução

No âmbito da unidade curricular Bases de Dados, do Mestrado Integrado em Engenharia Informática e Computação, decidimos modelar e implementar uma base de dados para gestão de uma escola secundária.

O sistema deve conter informação sobre alunos, docentes, encarregados de educação, disciplinas, turmas e áreas, permitindo a adição de novos elementos, assim como a consulta dos elementos previamente adicionados e das relações existentes entre eles.

Ao longo deste documento, além do modelo UML, considerado na versão anterior, descrevem-se o modelo relacional e a implementação em SQL que desenvolvemos para a referida base de dados. Tal como na primeira parte do trabalho, voltamos a expor as várias restrições consideradas, o tipo de situações que estão enquadradas neste modelo e aquelas que estão fora do âmbito da solução que propomos. Além disso, descrevemos as várias alterações que foram feitas relativamente à abordagem inicial.

Capítulo 2

Descrição do Contexto

Numa escola secundária há 3 grupos de pessoas fundamentais: os professores, os alunos e os encarregados de educação. Cada um destes grupos tem um papel distinto, com o objetivo comum de formar os alunos no âmbito das respetivas áreas escolhidas. Além destas pessoas, numa escola também podemos encontrar funcionários e diretores. No entanto, como veremos mais adiante, estes não são contemplados no modelo que construímos.

Além de algumas tarefas de estilo mais administrativo, como reuniões ou elaboração de relatórios, as principais responsabilidades dos professores são dar aulas e avaliar os alunos. Um professor que seja diretor de turma, tem ainda que reunir com os encarregados de educação dos alunos dessa turma, consoante as necessidades do aluno.

Os alunos têm como principais tarefas a frequência das aulas, o estudo fora destas e a realização das diversas provas de avaliação.

O encarregado de educação deve acompanhar o aluno nos seus estudos e reunir com o diretor de turma, para ajudar a optimizar o aproveitamento do aluno.

Os alunos estão organizados em turmas, consoante o ano que frequentam e a área dos seus estudos. Para cada turma há um professor responsável (o diretor de turma), um conjunto de disciplinas e um professor responsável por cada disciplina.

A escola providencia várias áreas de formação, consoante o ramo e o tipo de curso. Por exemplo, no ramo científico poderá haver áreas de carácter geral e de caráter tecnológico ou mesmo profissional. Para cada área de formação existe um professor que coordena essa área.

Um ano letivo é constituído por um ou mais períodos de avaliação. Atualmente, o ano letivo está dividido em 3 períodos, mas é possível que isso seja alterado no futuro. Em cada período os alunos são avaliados nas disciplinas que frequentam obtendo uma classificação que fica registada.

Capítulo 3

Descrição da Solução Implementada

No modelo desenvolvido, consideramos as seguintes classes principais:

- **Pessoa** - superclasse que contém a informação comum aos docentes, alunos e encarregados de educação, como o número de identificação na escola, o nome, a morada, a data de nascimento, etc;
- **Docente** - tem atributos próprios, como o ano de admissão e as qualificações;
- **Aluno** - tal como o docente, tem um ano de entrada na escola, tendo ainda um campo de observações, como por exemplo uma suspensão ou uma condição médica que seja do interesse da escola, e os campos derivados média e nota máxima, cujo cálculo é atualizado automaticamente consoante o aluno vai completando as disciplinas;
- **Encarregado_Educação** - tem um horário preferencial para ser contactado pela escola, sendo esse contacto normalmente efetuado pelo diretor de turma de um dos alunos pelo qual é responsável;
- **Área** - contém informação sobre o tipo de curso, nomeadamente, se se trata de um curso de carácter geral, tecnológico ou profissional, assim como o ramo a que pertence;
- **Turma** - é identificada por um número e pelo ano escolar a que pertence, além disso está associada a um ano letivo específico;
- **Disciplina** - possui um código de identificação da escola, assim como o nome e a descrição;
- **Ano_Letivo** - tem uma data de início e uma data de fim, sendo constituído por vários períodos letivos;

- **Período** - é identificado por um número, contendo ainda informação sobre as datas em que inicia e termina.

Além destas, abstraímos também os conceitos de **Código_Postal** e **Localidade**, tendo surgido ainda classes de relação entre algumas das classes vistas no ponto anterior, que descreveremos mais adiante.

Em cada instante, para qualquer par turma e disciplina, existe um único professor associado. No entanto, este professor poderá ser substituído, por motivos de doença, por exemplo. Além disso, um dado professor pode estar apto a lecionar outras disciplinas além destas, estando essa informação contida na base de dados da escola. Há ainda uma relação entre cada turma e um docente específico: o diretor de turma. Uma dada turma tem um e um só diretor, que à partida será responsável pela turma durante todo o ano letivo. No entanto, em situações excepcionais, poderá ser necessária a substituição do diretor de turma, sendo isso contemplado no modelo que desenhamos.

Os alunos são classificados a todas as disciplinas que frequentam consoante o período do ano letivo. Tal como foi mencionado acima, com a adição de classificações são recalculadas a média e a nota máxima do aluno. As disciplinas frequentadas por um dado aluno são determinadas consoante a turma em que ele está inserido. Sendo que a turma, por sua vez, depende do ano escolar e da área de formação dos seus alunos.

À semelhança daquilo que acontece com os diretores de turma, cada área possui um único coordenador em cada momento. No entanto, esse coordenador poderá variar ao longo do tempo. Um dado docente não é coordenador de mais do que uma área ao mesmo tempo.

Os encarregados de educação e os diretores de turma podem marcar reuniões por iniciativa de qualquer um deles, ficando registadas a data, a hora e a sala da reunião. Excepcionalmente, um encarregado de educação poderá reunir com outro professor do aluno pelo qual é responsável, ficando essa reunião registada da mesma forma. Cada aluno possui um único encarregado de educação (que pode ser responsável por vários alunos), sendo registada a relação de parentesco do encarregado com o aluno. Em alguns casos essa relação poderá não ser familiar, no entanto é sempre usada a palavra parentesco, ainda que no sentido lato.

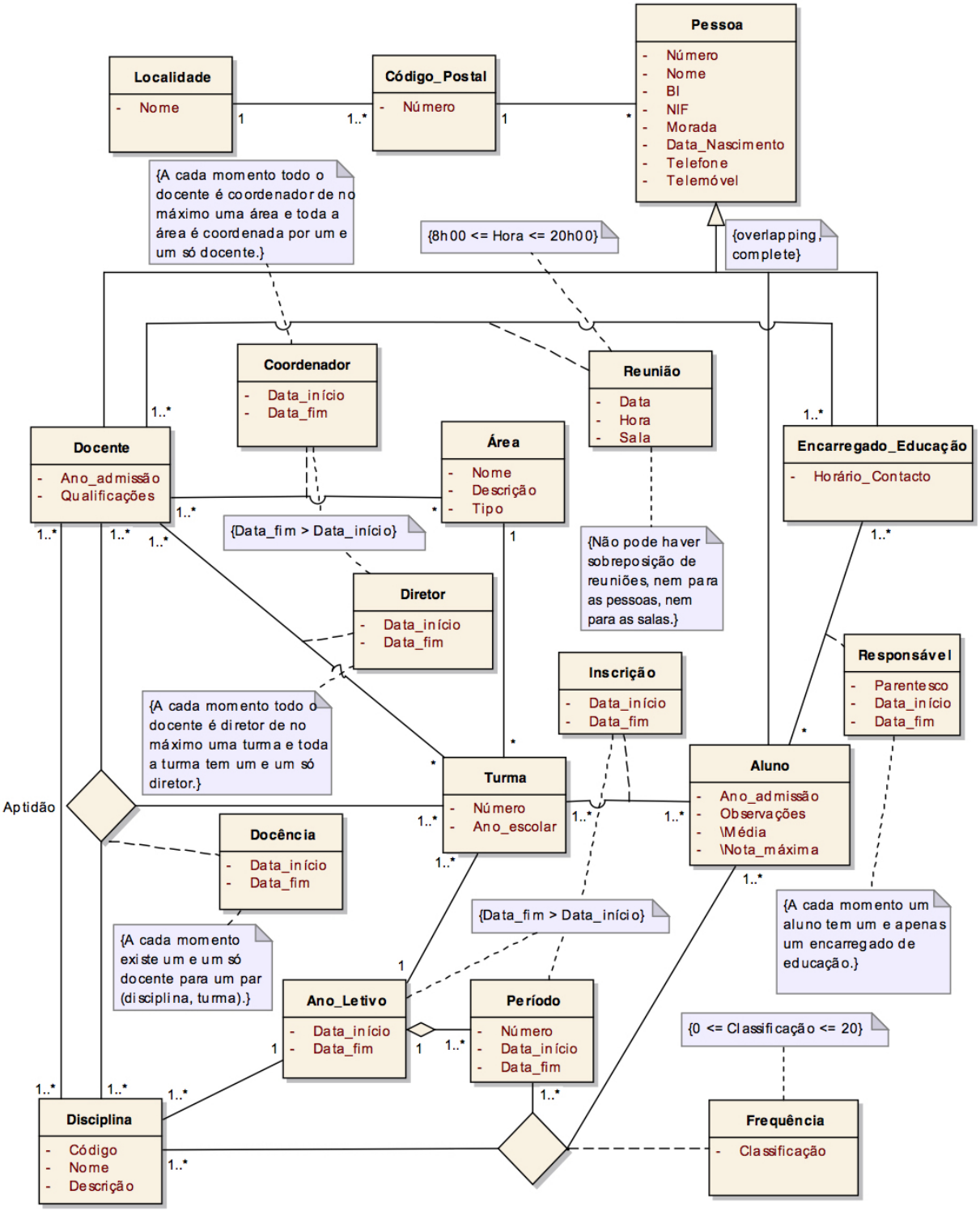
Consideramos que é possível uma mesma pessoa ser simultaneamente docente da escola e encarregado de educação. Além disso, embora apenas em casos muitos especiais, um

determinado aluno pode ser também encarregado de educação (de si próprio ou de outro aluno), nomeadamente se esse aluno for maior de idade ou tiver sido emancipado.

No modelo que desenhamos, não são contemplados os horários das várias aulas. Sendo assim, não é possível determinar o horário de um determinado aluno ou de um professor. Também não é incluída qualquer informação sobre os funcionários da escola, sobre os seus diretores ou sobre eventuais enfermeiros ou psicólogos. Numa versão mais alargada, poderiam e deveriam ser consideradas estas classes. Também seria interessante considerar informação mais detalhada sobre o percurso do aluno na escola, como por exemplo alterações significativas na sua vida pessoal, registos de faltas, visitas à enfermagem ou ao gabinete de psicologia, testes psicotécnicos que tenha efetuado, projetos e grupos extracurriculares em que tenha estado envolvido, etc.

Capítulo 4

Diagrama de Classes UML



Capítulo 5

Modelo Relacional

- **localidade** (id_localidade, nome)
- **codigo_postal** (id_cod_postal, numero, id_localidade → localidade)
- **pessoa** (id_pessoa, numero, nome, bi, nif, morada, data_nasc, telefone, telemovel, id_cod_postal → codigo_postal)
- **docente** (id_docente, ano_admissao, qualificacoes, id_pessoa → pessoa)
- **aluno** (id_aluno, ano_admissao, observacoes, \media, \max_nota**,** id_pessoa → pessoa)
- **encarregado** (id_encarregado, horario_contacto, id_pessoa → pessoa)
- **area** (id_area, nome, descricao, tipo)
- **coordenador** (id_coordenador, data_ini, data_fim, id_docente → docente, id_area → area)
- **turma** (id_turma, numero, ano_escolar, id_area → area, id_ano_letivo → ano_letivo)
- **diretor** (id_diretor, data_ini, data_fim, id_docente → docente, id_turma → turma)
- **inscricao** (id_inscricao, data_ini, data_fim, id_aluno → aluno, id_turma → turma)
- **responsavel** (id_responsavel, parentesco, data_ini, data_fim, id_aluno → aluno, id_encarregado → encarregado)
- **disciplina** (id_disciplina, codigo, nome, descricao, id_ano_letivo → ano_letivo)
- **ano_letivo** (id_ano_letivo, data_ini, data_fim)
- **periodo** (id_periodo, numero, data_ini, data_fim, id_ano_letivo → ano_letivo)

- **frequencia** (id_frequencia, classificacao, id_aluno → aluno, id_periodo → periodo, id_disciplina → disciplina)
- **docencia** (id_docencia, data_ini, data_fim, id_disciplina → disciplina, id_turma → turma, id_docente → docente)
- **aptidao** (id_docente → docente, id_disciplina → disciplina)
- **reuniao** (id_reuniao, data, hora, sala, id_docente → docente, id_encarregado → encarregado)

Capítulo 6

Implementação em SQLite

```
.mode column
.headers on

DROP TABLE IF EXISTS localidade;
CREATE TABLE localidade (
id_localidade INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
nome VARCHAR NOT NULL);

DROP TABLE IF EXISTS codigo_postal;
CREATE TABLE codigo_postal (
id_cod_postal INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
numero VARCHAR NOT NULL,
id_localidade INTEGER NOT NULL,
FOREIGN KEY(id_localidade) REFERENCES localidade(id_localidade));

DROP TABLE IF EXISTS pessoa;
CREATE TABLE pessoa (
id_pessoa INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
numero VARCHAR,
nome VARCHAR NOT NULL,
bi INTEGER,
nif INTEGER,
morada VARCHAR NOT NULL,
data_nasc DATETIME NOT NULL,
telefone VARCHAR,
telemovel VARCHAR,
```

```
id_cod_postal INTEGER NOT NULL,
FOREIGN KEY(id_cod_postal) REFERENCES codigo_postal(id_cod_postal));

DROP TABLE IF EXISTS docente;
CREATE TABLE docente (
id_docente INTEGER PRIMARY KEY NOT NULL,
ano_admissao INTEGER NOT NULL,
qualificacoes BLOB NOT NULL,
FOREIGN KEY(id_docente) REFERENCES pessoa(id_pessoa));

DROP TABLE IF EXISTS aluno;
CREATE TABLE aluno (
id_aluno INTEGER PRIMARY KEY NOT NULL,
ano_admissao INTEGER NOT NULL,
observacoes BLOB,
media FLOAT,
max_nota FLOAT,
FOREIGN KEY(id_aluno) REFERENCES pessoa(id_pessoa));

DROP TABLE IF EXISTS encarregado;
CREATE TABLE encarregado (
id_encarregado INTEGER PRIMARY KEY NOT NULL,
horario_contato VARCHAR ,
FOREIGN KEY(id_encarregado) REFERENCES pessoa(id_pessoa));

DROP TABLE IF EXISTS area;
CREATE TABLE area (
id_area INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
nome VARCHAR NOT NULL,
descricao VARCHAR NOT NULL,
tipo VARCHAR NOT NULL);
```

```
DROP TABLE IF EXISTS coordenador;
CREATE TABLE coordenador (
id_coordenador INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
data_ini DATETIME DEFAULT CURRENT_DATE,
data_fim DATETIME DEFAULT NULL,
id_docente INTEGER NOT NULL,
id_area INTEGER NOT NULL,
FOREIGN KEY(id_docente) REFERENCES docente(id_docente),
FOREIGN KEY(id_area) REFERENCES area(id_area),
CHECK (data_ini <= data_fim));

DROP TABLE IF EXISTS turma;
CREATE TABLE turma (
id_turma INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
numero VARCHAR NOT NULL,
ano_escolar VARCHAR NOT NULL,
id_area INTEGER NOT NULL,
id_ano_letivo INTEGER NOT NULL,
FOREIGN KEY(id_area) REFERENCES area(id_area),
FOREIGN KEY(id_ano_letivo) REFERENCES ano_letivo(id_ano_letivo));

DROP TABLE IF EXISTS diretor;
CREATE TABLE diretor (
id_diretor INTEGER PRIMARY KEY NOT NULL,
data_ini DATETIME DEFAULT CURRENT_DATE,
data_fim DATETIME DEFAULT NULL,
id_docente INTEGER NOT NULL,
id_turma INTEGER NOT NULL,
FOREIGN KEY(id_docente) REFERENCES docente(id_docente),
FOREIGN KEY(id_turma) REFERENCES turma(id_turma),
CHECK (data_ini <= data_fim));
```

```
DROP TABLE IF EXISTS inscricao;
CREATE TABLE inscricao (
id_inscricao INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
data_ini DATETIME NOT NULL,
data_fim DATETIME NOT NULL,
id_aluno INTEGER NOT NULL,
id_turma INTEGER NOT NULL,
FOREIGN KEY(id_aluno) REFERENCES aluno(id_aluno),
FOREIGN KEY(id_turma) REFERENCES turma(id_turma),
CHECK (data_ini <= data_fim));

DROP TABLE IF EXISTS responsavel;
CREATE TABLE responsavel (
id_responsavel INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
parentesco VARCHAR NOT NULL,
data_ini DATETIME DEFAULT CURRENT_DATE,
data_fim DATETIME DEFAULT NULL,
id_aluno INTEGER NOT NULL,
id_encarregado INTEGER NOT NULL,
FOREIGN KEY(id_aluno) REFERENCES aluno(id_aluno),
FOREIGN KEY(id_encarregado) REFERENCES encarregado(id_encarregado),
CHECK (data_ini <= data_fim));

DROP TABLE IF EXISTS disciplina;
CREATE TABLE disciplina (
id_disciplina INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
codigo VARCHAR NOT NULL,
nome VARCHAR NOT NULL,
descricao TEXT,
id_ano_letivo INTEGER NOT NULL,
FOREIGN KEY(id_ano_letivo) REFERENCES ano_letivo(id_ano_letivo));
```

```
DROP TABLE IF EXISTS ano_letivo;
CREATE TABLE ano_letivo (
id_ano_letivo INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
data_ini DATETIME NOT NULL,
data_fim DATETIME NOT NULL,
CHECK (data_ini < data_fim));

DROP TABLE IF EXISTS periodo;
CREATE TABLE periodo (
id_periodo INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
numero INTEGER NOT NULL,
data_ini DATETIME NOT NULL,
data_fim DATETIME NOT NULL,
id_ano_letivo INTEGER NOT NULL,
FOREIGN KEY(id_ano_letivo) REFERENCES ano_letivo(id_ano_letivo),
CHECK (data_ini < data_fim));

DROP TABLE IF EXISTS frequencia;
CREATE TABLE frequencia (
id_aluno INTEGER NOT NULL,
id_periodo INTEGER NOT NULL,
id_disciplina INTEGER NOT NULL,
classificacao FLOAT NOT NULL,
PRIMARY KEY(id_aluno, id_periodo, id_disciplina),
FOREIGN KEY(id_aluno) REFERENCES aluno(id_aluno),
FOREIGN KEY(id_periodo) REFERENCES periodo(id_periodo),
FOREIGN KEY(id_disciplina) REFERENCES disciplina(id_disciplina),
CHECK (classificacao > 0 AND classificacao <= 20));

DROP TABLE IF EXISTS docencia;
CREATE TABLE docencia (
id_docencia INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
```

```
data_ini DATETIME DEFAULT CURRENT_DATE,
data_fim DATETIME DEFAULT NULL,
id_disciplina INTEGER NOT NULL,
id_turma INTEGER NOT NULL,
id_docente INTEGER NOT NULL,
FOREIGN KEY(id_disciplina) REFERENCES disciplina(id_disciplina),
FOREIGN KEY(id_turma) REFERENCES turma(id_turma),
FOREIGN KEY(id_docente) REFERENCES docente(id_docente),
CHECK (data_ini <= data_fim));

DROP TABLE IF EXISTS aptidao;
CREATE TABLE aptidao (
id_docente INTEGER NOT NULL,
id_disciplina INTEGER NOT NULL,
PRIMARY KEY (id_docente, id_disciplina),
FOREIGN KEY(id_docente) REFERENCES docente(id_docente),
FOREIGN KEY(id_disciplina) REFERENCES disciplina(id_disciplina));

DROP TABLE IF EXISTS reuniao;
CREATE TABLE reuniao (
id_reuniao INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
data DATETIME NOT NULL,
hora DOUBLE NOT NULL,
sala VARCHAR NOT NULL,
id_docente INTEGER NOT NULL,
id_encarregado INTEGER NOT NULL,
FOREIGN KEY(id_docente) REFERENCES docente(id_docente),
FOREIGN KEY(id_encarregado) REFERENCES encarregado(id_encarregado));
```

Capítulo 7

Introdução de registos

```
PRAGMA foreign_keys = ON;

INSERT INTO "pessoa" VALUES(1,'AL01','Carlos',102548796,254125789,
        'Rua do Lado Esquerdo','2000-05-05','254789654','323547899',1);
INSERT INTO "pessoa" VALUES(2,'AL02','Joaquim',134548796,254125734,
        'Rua do Lado Direito','2001-06-05','954789654','923547899',2);
INSERT INTO "pessoa" VALUES(3,'AL03','Manuel',434548796,257655734,
        'Rua do Cimo Direito','2000-02-05','954234654','813547899',3);
INSERT INTO "pessoa" VALUES(4,'DC01','Alberto',434548796,257655734,
        'Rua do Cimo Direito','1978-02-05','465478965','962587451',4);
INSERT INTO "pessoa" VALUES(5,'DC02','Custodio',434548796,257655734,
        'Rua do Cimo Esquerdo','1979-02-05','4623478965','962587345',5);
INSERT INTO "pessoa" VALUES(6,'EE01','Baltazar',25478965,12547889,
        'Rua do Lado Esquerdo','1973-02-05','925656541','252587845',1);

INSERT INTO "aluno" VALUES(1,2013,NULL,NULL,NULL,1);
INSERT INTO "aluno" VALUES(2,2013,NULL,NULL,NULL,2);
INSERT INTO "aluno" VALUES(3,2013,NULL,NULL,NULL,3);

INSERT INTO "codigo_postal" VALUES(1,'4100',1);
INSERT INTO "codigo_postal" VALUES(2,'5100',2);
INSERT INTO "codigo_postal" VALUES(3,'4300',3);
INSERT INTO "codigo_postal" VALUES(4,'3600',5);
INSERT INTO "codigo_postal" VALUES(5,'3700',4);
```

```
INSERT INTO "docente" VALUES(1,2012,'Portugues, Ingles',4);
INSERT INTO "docente" VALUES(2,2011,'Matematica, Ciencias',5);

INSERT INTO "encarregado" VALUES(1,'10h-11h',6);

INSERT INTO "localidade" VALUES(1,'Porto');
INSERT INTO "localidade" VALUES(2,'Braga');
INSERT INTO "localidade" VALUES(3,'Gaia');
INSERT INTO "localidade" VALUES(4,'Espinho');
INSERT INTO "localidade" VALUES(5,'Famalicao');

INSERT INTO "responsavel" VALUES(1,'Pai','2013-04-20',NULL,1,1);
```

Capítulo 8

Principais Alterações

Com a passagem do modelo UML para o modelo relacional e para a implementação em SQL, tornou-se clara a necessidade de fazer várias alterações relativamente ao projeto inicial. Além disso, foi também necessário verter para o diagrama UML restrições que tinham sido discutidas no relatório, por forma a facilitar a implementação.

Alteramos o nome da relação entre **Docente** e **Área** para **Coordenador**, reservando a palavra **Responsável** para descrever a relação em o **Encarregado_Educação** e o **Aluno**. Além disso, todas as relações que deram origem a tabelas no modelo relacional passaram a ter um nome: **Inscrição** para a relação entre um **Aluno** e a sua **Turma**, **Frequência** para a relação ternária **Aluno** - **Disciplina** - **Período**, **Docência** para a relação ternária **Docente** - **Disciplina** - **Turma** e **Aptidão** para a relação entre o **Docente** e a **Disciplina** que pode lecionar (em lugar de *apto a lecionar*).

A relação ternária **Docente** - **Disciplina** - **Turma** sofreu ainda uma alteração de multiplicidade. Na primeira versão, fixadas a turma e a disciplina, havia um e um só docente associado. No entanto, essa restrição não dava lugar à possibilidade de um professor ter que ser substituído durante o ano letivo. Sendo assim, a multiplicidade do lado do **Docente** passou a ser um ou mais. Para garantir que num dado instante o docente associado a um par turma e disciplina é único, adicionamos uma restrição à relação.

A relação entre **Encarregado** de educação e **Aluno** sofreu uma alteração semelhante, novamente pelo motivo de poder haver necessidade de substituir um determinado encarregado de educação. Quer para esta relação, quer para a relação **Docência**, descrita no parágrafo anterior, foram adicionadas datas de início e de fim à classe de relação, para delimitar o período de vigência de um determinado encarregado de educação ou docente, consoante o caso.

Também foi necessário fazer a mesma alteração para a relação **Inscrição** entre **Aluno** e **Turma**, de forma a que um aluno possa mudar de turma durante o ano letivo.

As relações **Coordenador** e **Diretor** têm restrições semelhantes às de **Docência**,

Responsável e **Inscrição**. Essas restrições foram incluídas no diagrama UML, juntamente com a adição de datas de início e de fim.

Em todas as classes em que existem datas de início e de fim, foi adicionada uma restrição para garantir que a data de fim ocorre após a data de início. Também foram adicionadas restrições às horas possíveis para marcação de reuniões e às classificações que os alunos podem obter. Ainda no campo das restrições, passamos a garantir que não há sobreposição de reuniões, quer a nível das pessoas que nelas participam, quer a nível da sala onde decorrem.

A relação entre **Ano_letivo** e **Período** passou de composição para agregação, por considerarmos que a relação de composição era muito forte para este caso.

Finalmente, adicionaram-se os campos derivados média e nota máxima à classe aluno. Estes campos resultam de forma natural da adição de classificações à tabela de frequências, sendo atualizados automaticamente na implementação em SQL.

Capítulo 9

Principais Dificuldades

A escolha das relações apropriadas entre as várias classes, em especial no que diz respeito à multiplicidade dessas relações continuou a gerar dificuldades. Nomeadamente, foi necessário identificar novos casos em que num momento específico existe um único elemento, mas ao longo do tempo podem existir vários.

Apesar de termos excluído à partida várias situações da base de dados da escola, o modelo obtido é relativamente complicado, com muitas classes de relação e várias restrições, que aumentam a complexidade de implementação.

A maior dificuldade desta fase foi chegar a acordo nas situações em que existem várias possibilidades de implementação, particularmente no desenho do modelo relacional. No entanto, após essa hesitação inicial, conseguimos avançar com alguma destreza.

Infelizmente, não conseguimos introduzir tantos registos como seria pretendíamos nas tabelas, por falta de colaboração de um dos elementos do grupo.

Capítulo 10

Conclusão

Apesar de inicialmente nos parecerem simples e claros vários aspetos do modelo, com a criação do modelo relacional, a implementação e a contemplação de situações concretas tornou-se necessário alterar vários aspetos da concepção original. Eventualmente, com a experiência na criação de bases de dados, será mais fácil acertar à primeira naquilo que melhor se adequa ao que o cliente pretende. De qualquer forma, cremos que é sempre importante saber reconhecer que o caminho inicial não é o mais conveniente e alterá-lo em conformidade.

Com a evolução do projeto tem-se tornado cada vez mais claro quão rapidamente um conceito simples se traduz numa base de dados extremamente complexa. No nosso exemplo concreto, para contemplar toda a realidade de uma escola, seria necessário utilizar uma base de dados consideravelmente maior e mais difícil do que a que estamos a desenvolver.

Este projeto continua a ser interessante, especialmente pela possibilidade de aplicar os conceitos da unidade curricular à medida que os mesmos são introduzidos. Além disso, apesar de dificultarem o progresso, as discussões sobre o melhor caminho a seguir também têm contribuído para a nossa aprendizagem.