Faculdade de Engenharia da Universidade do Porto

Projeto de BD 21/22 – 2ª entrega

Definição do Esquema Relacional
Análise de Dependências Funcionais e Formas Normais
Criação da Base de Dados em SQLite
Adição de Restrições à Base de Dados
Carregamento de Dados

L.EIC | Base de Dados 2021/2022

Carla Teixeira Lopes & Michel Ferreira

Turma 2LEIC06 (grupo 601):

António Ferreira – up202004735@edu.fe.up.pt
João Maldonado – up202004244@edu.fe.up.pt
Tomás Gomes – up202004393@edu.fe.up.pt

Índice

I. Contexto 4
II. Diagrama UML 5
III. Diagrama UML (Revisto) 6
IV. Esquema Relacional 7
V. Análise Dependências Funcionais e Formas Normais 8-9

Contexto

Pretende-se guardar todas as informações referentes aos exames nacionais realizados no ensino secundário.

Com isto em consideração, para um **Exame**, é necessário armazenar a disciplina, e respetivo código, e a fase de cada exame, pelo que, no máximo, um **Aluno** pode ir a duas fases (1ª fase e 2ª fase), sendo também possível este realizar um exame da mesma disciplina em diferentes **Anos Letivos**.

Quanto a cada **Aluno,** é necessário saber o nome, sexo, idade e se é um aluno interno ou não. Também é importante ter conhecimento da sua **Situação de Frequência**, isto é, se um aluno está admitido a exame, se anulou a matrícula, se foi excluído por faltas ou se reprovou por não ter conseguido frequência. Tendo em conta estas informações, é de referir a relevância de armazenar os objetivos com que o aluno realiza o exame (se é para aprovação, para melhoria, como prova de ingresso ou CFCEPE, ou seja, como prova de prosseguimento de estudos, para alunos do ensino profissional, recorrente, vocacional e outros). Após a realização de um exame, o aluno obterá uma nota que, juntamente com a sua classificação interna da disciplina, determinará a classificação final. Caso o aluno não seja interno, a classificação final será a nota do exame. Além disso, deve ser tida em conta a **Escola** do aluno e o **Curso** que frequenta.

Por um lado, uma **Escola** possui um nome, código DGAE (código de agrupamento), código DGEEC (código da Direção-Geral de Estatísticas da Educação e Ciência, que identifica cada escola) e tipo (escola privada ou pública). Cada escola está localizada num **Concelho** (que tem nome, código de concelho e código NUTS3) e cada concelho pertence a um, e um só, determinado **Distrito** (que é definido pelo nome e código do distrito).

Por outro lado, para qualquer **Curso,** é importante armazenar informação relativa ao seu nome e código identificador. Todavia, a todos os cursos é atribuído um **Subtipo de Curso** (para o qual é preciso saber o nome e código específico), que, por sua vez, tem um único **Tipo de Curso** (que possui um nome, código e ano de escolaridade inicial e final).

Diagrama UML

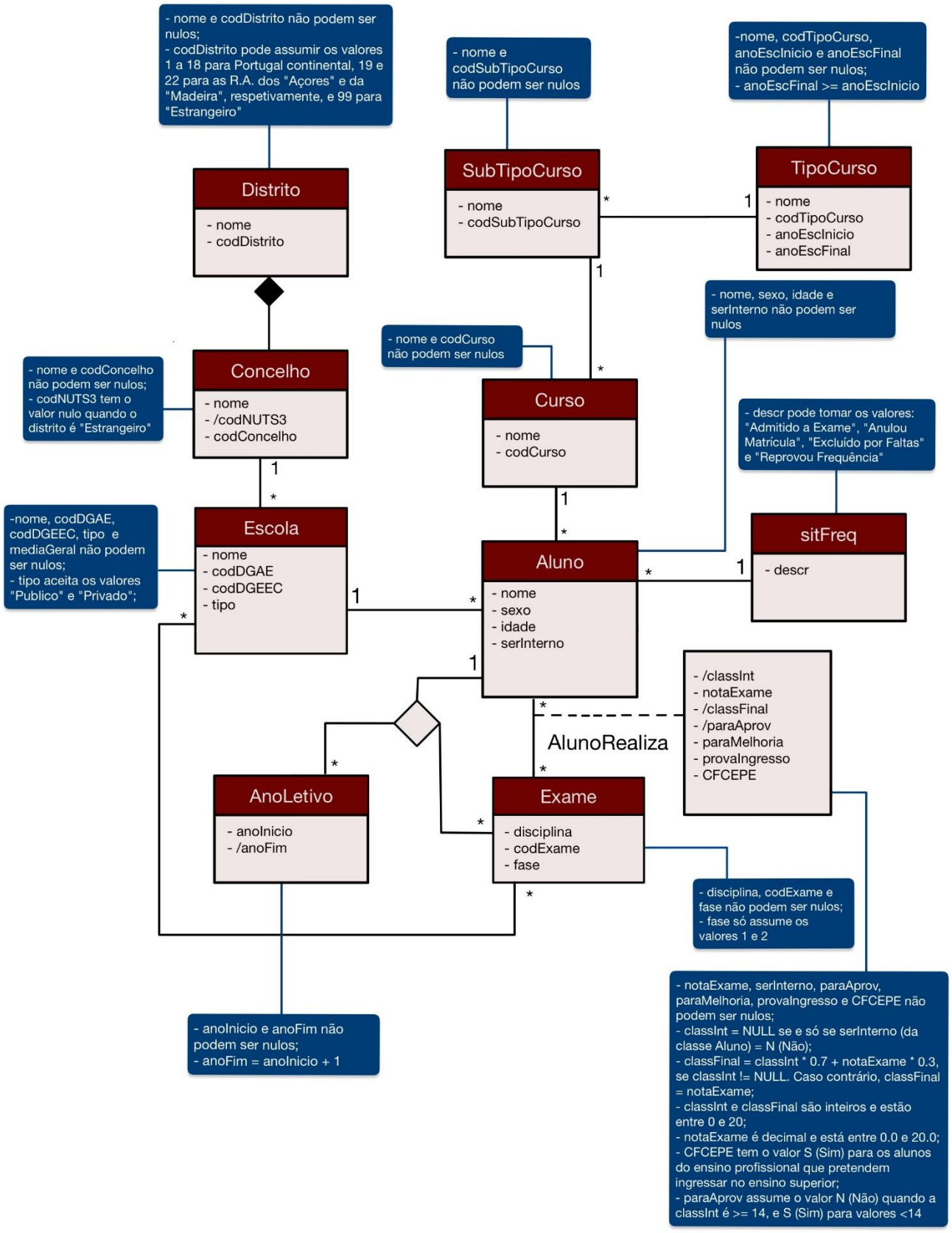
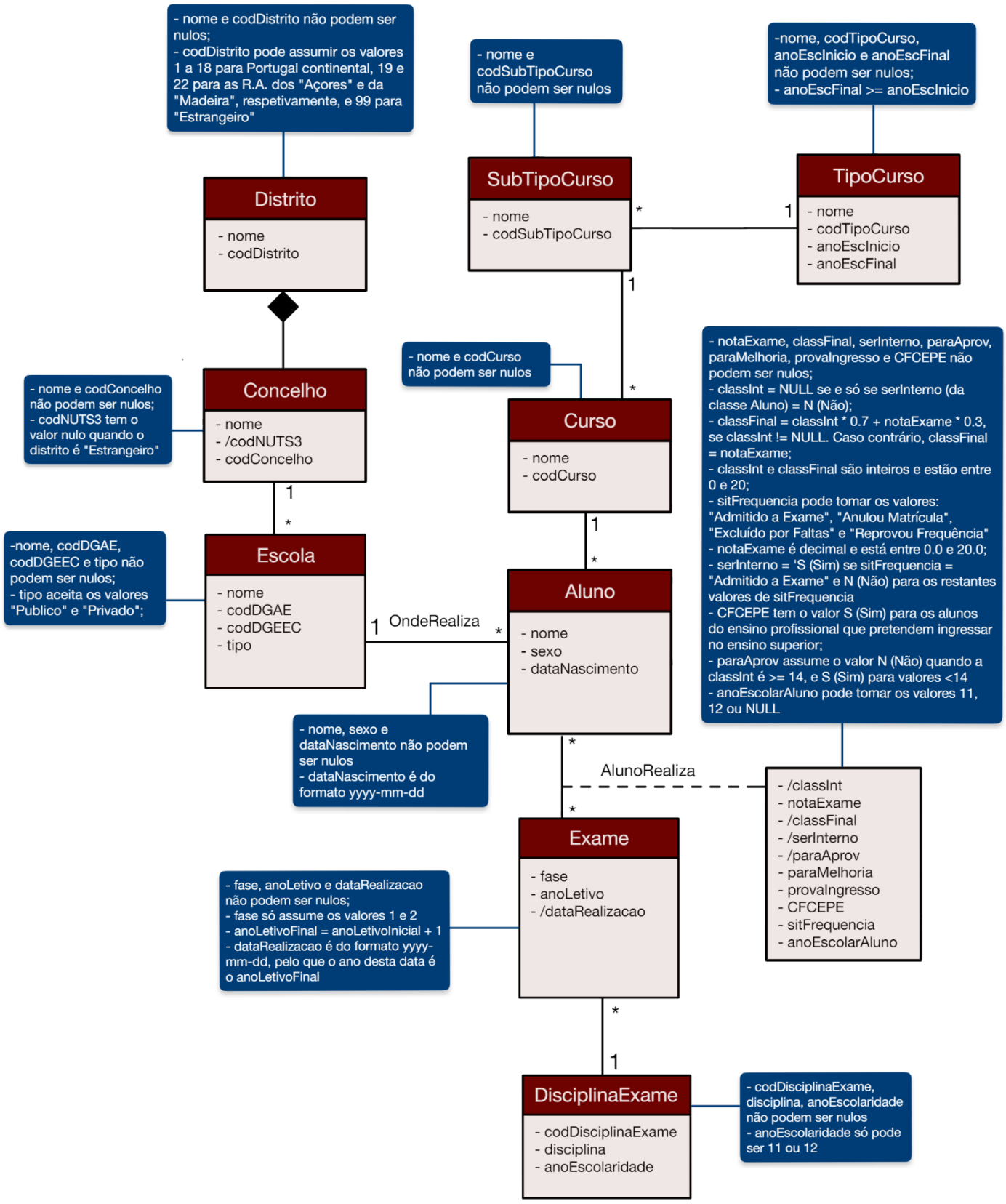


Diagrama UML (Revisto)



Esquema Relacional

- Distrito(codDistrito, nome);
- Concelho(idConcelho, codConcelho, nome, codNUTS3, codDistrito -> Distrito);
- Escola(idEscola, nome, codDGAE, codDGEEC, tipo, idConcelho -> Concelho);
- Aluno(idAluno, nome, sexo, dataNascimento, anoEscolaridade, codCurso -> Curso);
- OndeRealiza(idEscola -> Escola, idAluno -> Aluno);
- Exame(idExame, fase, anoLetivo, dataRealizacao, codDisciplinaExame - > DisciplinaExame);
- DisciplinaExame(codDisciplonaExame, disciplina, anoEscolaridade)
- AlunoRealiza(idAluno -> Aluno, idExame -> Exame, classInt, notaExame, classFinal, serInterno, paraAprov, paraMelhoria, provaIngresso, CFCEPE, sitFrequencia, anoEscolarAluno);
- Curso(codCurso, nome, codSubTipoCurso -> SubTipoCurso);
- SubTipoCurso(codSubTipoCurso, nome, codTipoCurso-> TipoCurso);
- TipoCurso(codTipoCurso, nome, anoEscInicio, anoEscFinal).

Análise de Dependências Funcionais e de Formas Normais

- Distrito(codDistrito, nome):
  - **FDs:**
    codDistrito -> nome
    nome -> codDistrito
    Keys: {codDistrito}, {nome}

    Formas: BCNF? Sim
    3NF? Sim

- Concelho(idConcelho, codConcelho, nome, codNUTS3, codDistrito -> Distrito):
  - **FDs:**
    idConcelho -> codConcelho, nome, codDistrito, codNUTS3
    nome -> idConcelho, codConcelho, codDistrito, codNUTS3
    Keys: {idConcelho}, {nome}

    Formas: BCNF? Sim
    3NF? Sim

- Escola(idEscola, nome, codDGAE, codDGEEC, tipo, idConcelho-> Concelho):
  - **FDs**:
    idEscola -> nome, codDGAE, codDGEEC, tipo, idConcelho
    codDGAE -> idEscola, nome, codDGEEC, tipo, idConcelho
    codDGEEC -> idEscola, nome, codDGAE, tipo, idConcelho
    Keys: {idEscola}, {codDGAE}, {codDGEEC}

    Formas: BCNF? Sim
    3NF? Sim

- Aluno(idAluno, nome, sexo, dataNascimento, codCurso -> Curso):
  - **FDs:**
    idAluno -> nome, sexo, dataNascimento, serInterno, anoEscolaridade, codCurso
    Key: {idAluno}

    Formas: BCNF? Sim
    3NF? Sim

- OndeRealiza(idEscola -> Escola, idAluno -> Aluno):
  - **FDs:** -

Formas: BCNF? Sim
3NF? Sim

- Exame(idExame, fase, anoLetivo, dataRealizacao, codDisciplinaExame -> DisciplinaExame):
  - **FDs:**
    idExame ->fase, anoLetivo, dataRealizacao
    Key: {idExame}

    Formas: BCNF? Sim
    3NF? Sim

- DisciplinaExame(codExame, disciplina, anoEscolaridade):
  - **FDs:**
    codExame ->,disciplina,anoEscolaridade
    Key: {codExame}

    Formas: BCNF? Sim
    3NF? Sim

- AlunoRealiza(idAluno -> Aluno, idExame -> Exame, classInt, notaExame, classFinal, serInterno, paraAprov, paraMelhoria, provaIngresso, CFCEPE, sitFrequencia, anoEscolarAluno):
  - **FDs:**
    idAluno, idExame -> classInt, notaExame, classFinal, serInterno, paraAprov, paraMelhoria, provaIngresso, CFCEPE, sitFrequencia, anoEscolarAluno
    classInt, notaExame -> classFinal

    Key: {idAluno, idExame}

    Formas: BCNF? Não
    3NF? Não

    A relação viola a Forma Normal de Boyce-Codd, visto que classInt e notaExame não são uma '(super)key'. Para além disso, esta também viola a 3ª Forma Normal, uma vez que o atributo classFinal não é um atributo primo (não é um membro de nenhuma 'key').

  - **Decomposição para Forma Normal de Boyce-Codd:**

FD que viola BCFN: classInt, notaExame -> classFinal.

$\{classInt, notaExame\}^+ = \{classInt, notaExame, classFinal\}$

S1(classInt, notaExame, classFinal)
S2(classInt, notaExame, serInterno, paraAprov, paraMelhoria, provaIngresso, CFCEPE, sitFrequencia, anoEscolarAluno, idAluno, idExame)

FDs para S1:
classInt, notaExame -> classFinal
Key: {classInt, notaExame}

FDs para S2:
idAluno, idExame -> classInt, notaExame, serInterno, paraAprov, paraMelhoria, provaIngresso, CFCEPE, sitFrequencia, anoEscolarAluno
sitFrequencia -> serInterno
Key: {idAluno, idExame}

Formas: BCNF? Não
3NF? Não

FD que viola BCFN: sitFrequencia -> serInterno

$\{sitFrequencia\}^+ = \{sitFrequencia, serInterno\}$

S3(sitFrequencia, serInterno)
S4(sitFrequencia, classInt, notaExame, paraAprov, paraMelhoria, provaIngresso, CFCEPE, anoEscolarAluno, idAluno, idExame)

FDs para S3:
sitFrequencia -> serInterno
Key: {sitFrequencia}

FDs para S4:
idAluno, idExame -> classInt, notaExame, paraAprov, paraMelhoria, provaIngresso, CFCEPE, sitFrequencia, anoEscolarAluno
Key: { idAluno, idExame }

S1, S3 e S4 encontram-se então em BCNF, pois em cada dependência funcional destas duas relações, o lado esquerdo é uma 'key'.

- Curso(codCurso, nome, codSubTipoCurso -> SubTipoCurso):
  - **FDs:**
    codCurso -> nome, codSubTipoCurso
    nome -> codCurso, codSubTipoCurso

Keys: {codCurso}, {nome}

Formas: BCNF? Sim
3NF? Sim.

- SubTipoCurso(codSubTipoCurso, nome, codTipoCurso -> TipoCurso):
  - **FDs:**
    codSubTipoCurso -> nome, codTipoCurso
    nome -> codSubTipoCurso, codTipoCurso
    Keys: {codSubTipoCurso, nome}

    Formas: BCNF? Sim
    3NF? Sim

- TipoCurso(codTipoCurso, nome, anoEscInicio, anoEscFinal):
  - **FDs:**
    codTipoCurso -> nome, anoEscInicio, anoEscFinal
    nome -> codTipoCurso, anoEscInicio, anoEscFinal
    Keys: {codTipoCurso}, {nome}

    Formas: BCNF? Sim
    3NF? Sim