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

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

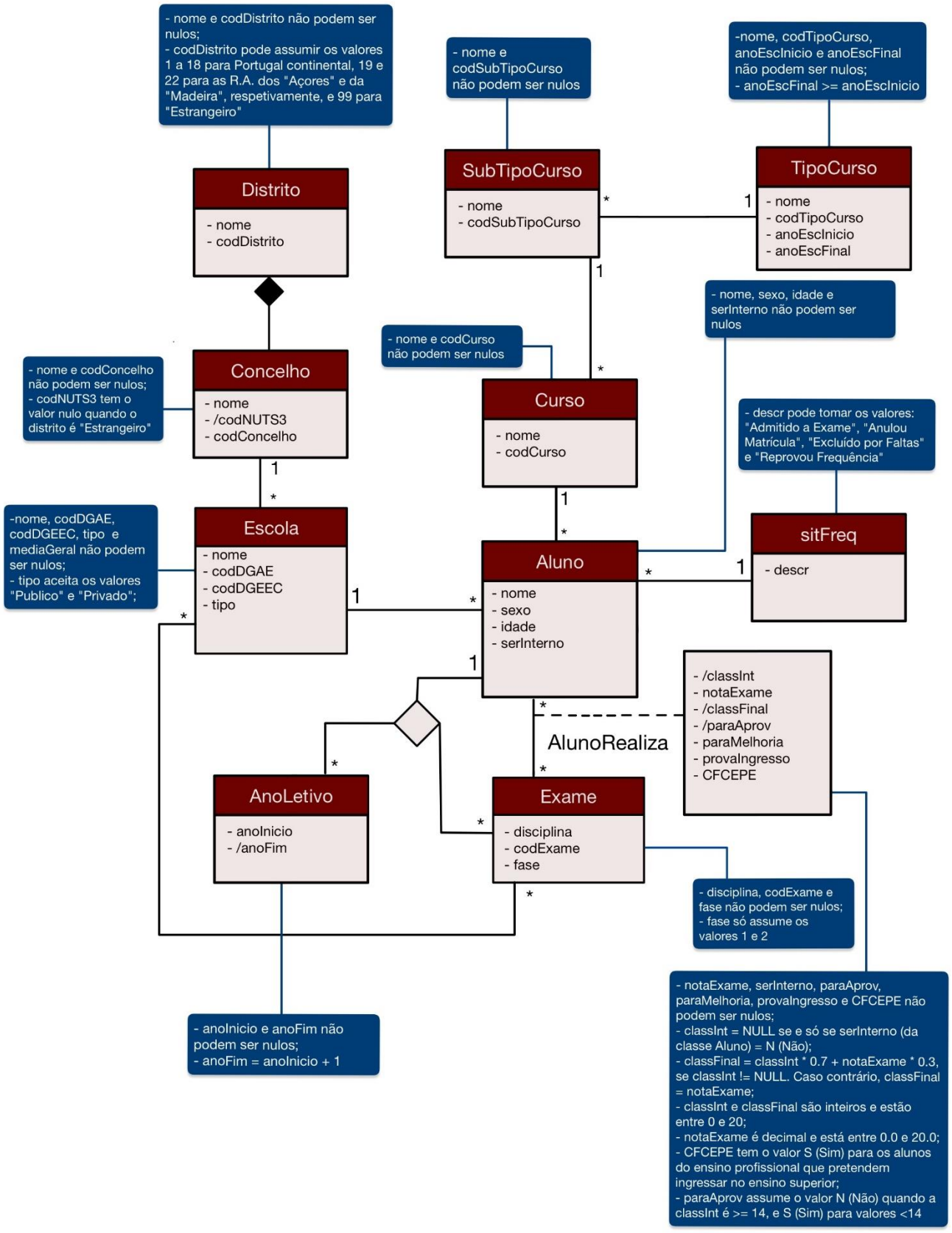
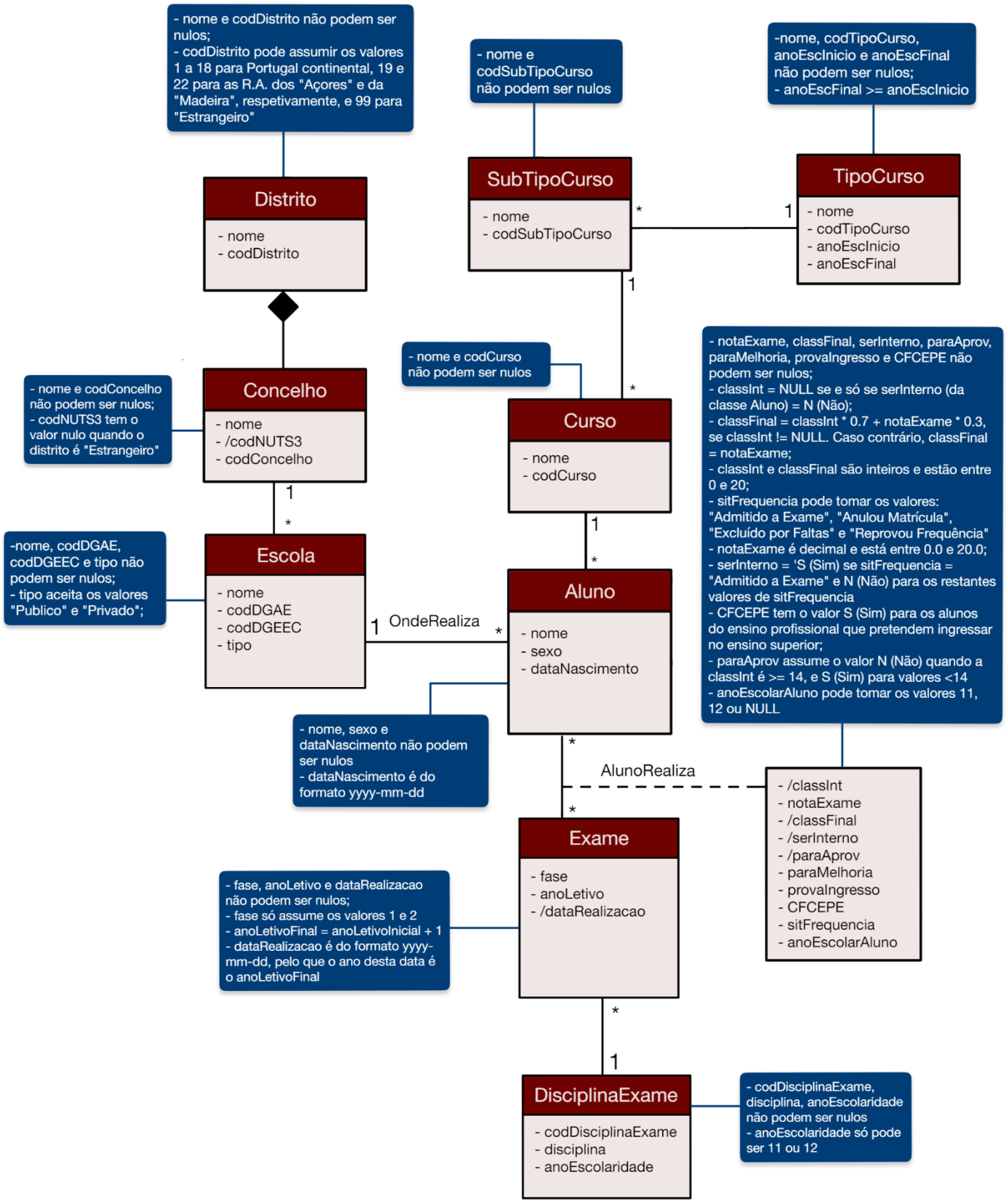


Diagrama UML (Revisto)



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
      Key: {idAluno, idExame}

      Formas: BCNF? Sim
      3NF? Sim

- 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

Ao analisar as dependências funcionais consideradas, pode-se afirmar que todas as relações se encontram em Forma Normal de Boyce-Codd e em 3ª Forma Normal.

Seja R -> S uma FD não trivial de uma relação A, por um lado, esta última encontra-se em BCNF se e só se R for uma superkey/key. Por outro lado, A está em 3ª Forma Normal se R for uma superkey/key ou se S for um atributo primo, ou seja, um atributo que pertence a pelo menos uma key da relação.

Sendo assim, como se consegue determinar, a partir dos elementos da esquerda de todas as FDs apresentadas (R), os restantes atributos de cada relação, podemos então concluir que R é uma superkey/key, o que por sua vez implica que todas as relações estão em Forma Normal de Boyce-Codd e em 3ª Forma Normal.

Restrições

Distrito –

- Não pode haver dois distritos com o mesmo código:
  - codDistrito PRIMARY KEY

- O código tem de ser um número entre 1 e 18 para os distritos de Portugal continental, 19 e 22 para as R.A. dos Açores e da Madeira, respetivamente, e 99 para o Estrangeiro:
  - CONSTRAINT check_codDistrito CHECK ((codDistrito > 0) AND (codDistrito <= 19 OR codDistrito = 22 OR codDistrito = 99))

- Não pode haver dois distritos com o mesmo nome:
  - CONSTRAINT nome_distrito_unique UNIQUE

- Todos os distritos têm de ter um nome:
  - CONSTRAINT notnull_nome_distrito NOT NULL

Concelho –

- Não pode haver dois concelhos com o mesmo ID:
  - idConcelho PRIMARY KEY

- Todos os concelhos têm de ter um código
  - CONSTRAINT notnull_codConcelho NOT NULL

- Se o código de distrito for '99' (Estrangeiro), então o código do concelho também terá que ser '99':
  - CONSTRAINT check_codConcelho CHECK (CASE WHEN codDistrito = 99 THEN codConcelho = 99 WHEN codDistrito <> 99 THEN codConcelho <> 99 END)

- Não pode haver dois concelhos com o mesmo nome:
  - CONSTRAINT nome_concelho_unique UNIQUE

- Todos os concelhos têm de ter um nome:
  - CONSTRAINT notnull_nome_concelho NOT NULL

- Todos os concelhos têm de ter um código de distrito:
  - CONSTRAINT notnull_codDistrito NOT NULL

- O código de distrito de um concelho tem de corresponder ao código de distrito de um distrito:
  - CONSTRAINT foreignkey_codDistrito REFERENCES Distrito(codDistrito)

Escola –

- Não pode haver duas escolas com o mesmo ID:
  - idEscola PRIMARY KEY

- Não pode haver nem códigos DGAE nem códigos DGEEC repetidos:
  - CONSTRAINT codDGAE_unique UNIQUE
  - CONSTRAINT codDGEEC_unique UNIQUE

- Todas as escolas têm de ter um nome, um código DGAE, um código DGEEC, um tipo e um ID de concelho:
    - CONSTRAINT notnull_nome_escola NOT NULL
    - CONSTRAINT notnull_codDGAE NOT NULL
    - CONSTRAINT notnull_codDGEEC NOT NULL
    - CONSTRAINT notnull_tipo_escola NOT NULL
    - CONSTRAINT notnull_idConcelho NOT NULL

- Toda as escolas têm públicas ou privadas:
    - CONSTRAINT check_tipo_escola CHECK (tipo = 'PÚBLICO' OR tipo = 'PRIVADO')

- O ID de concelho de uma escola tem de corresponder ao ID de concelho de um concelho:
    - CONSTRAINT foreignkey_idConcelho REFERENCES Concelho(idConcelho)

TipoCurso –

- Não pode haver dois tipos de curso com o mesmo código:
    - codTipoCurso PRIMARY KEY

- O código de um tipo de curso só pode ser: C', 'E', 'N', 'P', 'Q', 'R', 'T', 'U', 'V' e 'W':
    - CONSTRAINT check_codTipoCurso CHECK (codTipoCurso IN ('C', 'E', 'N', 'P', 'Q', 'R', 'T', 'U', 'V' e 'W'))

- Não pode haver nomes de tipos de curso repetidos:
    - CONSTRAINT nome_unique UNIQUE

- Todos os tipos de curso têm de ter um nome, um ano escolaridade de início e de fim:
    - CONSTRAINT notnull_nome NOT NULL
    - CONSTRAINT notnull_anoEscInicio NOT NULL
    - CONSTRAINT notnull_anoEscFinal NOT NULL

- O ano de escolaridade de início e de fim de um tipo de curso têm de ser iguais ou superiores ao 10º ano e iguais ou inferiores ao 12º ano, pelo que o ano de escolaridade final tem de ser superior ou igual ao ano de escolaridade inicial:
    - CONSTRAINT check_anoEscInicio CHECK (anoEscInicio >= 10 AND anoEscInicio <=12)
    - CONSTRAINT check_anoEscFinal CHECK (anoEscFinal >= 10 AND anoEscFinal <=12)
    - CONSTRAINT anoEscFinal_maior_anoEscInicio CHECK (anoEscFinal >= anoEscInicio)

SubTipoCurso –

- Não pode haver dois subtipos de curso com o mesmo código:
    - codSubTipoCurso PRIMARY KEY

- Não pode haver nomes de subtipos de curso repetidos:
    - CONSTRAINT nomeSubTipoCurso_unique UNIQUE

- Todos os subtipos de curso têm de ter um nome e um código do tipo de curso:
    - CONSTRAINT notnull_nome_subTipoCurso NOT NUL

- CONSTRAINT notnull_codTipoCurso NOT NULL

- O código do tipo de um curso tem de corresponder ao código de um tipo de curso:
  - CONSTRAINT foreignkey_codTipoCurso REFERENCES TipoCurso(codTipoCurso)

- A primeira letra do código do subtipo de curso tem de responder à letra do código do tipo de curso:
  - CONSTRAINT check_codSubTipoCurso CHECK (substr(codSubTipoCurso,1,1) = codTipoCurso)

Curso –

- Não pode haver dois cursos com o mesmo código:
  - codCurso PRIMARY KEY

- Não pode haver nomes de curso repetidos:
  - CONSTRAINT nome_curso_unique UNIQUE

- Todos os cursos têm de ter um nome e um código do subtipo de curso:
  - CONSTRAINT notnull_nome_curso NOT NULL
  - CONSTRAINT notnull_codSubTipoCurso NOT NULL

- O código do subtipo de um curso tem de corresponder ao código de um subtipo de curso:
  - CONSTRAINT foreignkey_codSubTipoCurso REFERENCES SubTipoCurso(codSubTipoCurso)

Aluno –

- Não pode haver dois alunos com o mesmo ID:
  - idAluno PRIMARY KEY

- Todos os alunos têm de ter um nome, sexo, data de nascimento e código de curso:
  - CONSTRAINT notnull_nome_aluno NOT NULL
  - CONSTRAINT notnull_sexo NOT NULL
  - CONSTRAINT notnull_dataNasc NOT NULL
  - CONSTRAINT notnull_codCurso NOT NULL

- Um aluno pode ser do sexo masculino (‘M’) ou feminino (‘F’). Quando não se sabe o sexo deste, recorre-se ao valor ‘?’:
  - Sexo DEFAULT '?'
  - CONSTRAINT check_sexo CHECK (sexo = 'M' OR sexo = 'F')

- O código de curso referencia o código de um curso:
  - CONSTRAINT foreignkey_codCurso REFERENCES Curso(codCurso)

Exame –

- Não pode haver dois exames com o mesmo ID:
  - idExame PRIMARY KEY

- Todos os exames têm de ter uma fase, ano letivo, data de realização e código de exame:
  - CONSTRAINT notnull_fase NOT NULL

    - CONSTRAINT notnull_anoLetivo NOT NULL
    - CONSTRAINT notnull_dataRealizacao NOT NULL
    - CONSTRAINT notnull_codExame NOT NULL

- Um exame só pode ser 1ª ou da 2ª fase:
    - CONSTRAINT check_fase CHECK (fase = 1 OR fase = 2)

- O ano letivo em que um exame foi realizado só é do formato AnoInicial/AnoFinal, sendo que AnoFinal = AnoInicial + 1:
    - CONSTRAINT check_anoLetivo CHECK (CAST (SUBSTR (anoLetivo,6,4) AS INT) = CAST (SUBSTR (anoLetivo,1,4) AS INT) + 1 AND SUBSTR (anoLetivo,5,1) = '/')

- A data da realização do exame tem de ser igual ao AnoFinal do ano letivo:
    - CONSTRAINT check_year CHECK (strftime('%Y', dataRealizacao) = SUBSTR(anoLetivo,6,4))

- O código de exame tem de referenciar o código de exame de uma disciplina de exame:
    - CONSTRAINT foreignkey_codExame REFERENCES DisciplinaExame(codExame)

DisciplinaExame –

- Não pode haver duas disciplinas de exame com o mesmo código:
    - codExame PRIMARY KEY

- Não pode haver duas disciplinas de exame com o mesmo nome:
    - CONSTRAINT disciplina_unique UNIQUE

- Todos os exames têm de ter uma disciplina e um ano de escolaridade:
    - CONSTRAINT notnull_disciplina NOT NULL
    - CONSTRAINT notnull_anoEscolaridade NOT NULL

- O ano de escolaridade de um exame só pode ser o 11º ou o 12º ano:
    - CONSTRAINT check_anoEscolaridade CHECK (anoEscolaridade = 11 OR anoEscolaridade = 12)

OndeRealiza –

- Não pode haver dois alunos com o mesmo ID:
    - idAluno PRIMARY KEY

- Cada aluno realiza o exame numa escola com um certo ID:
    - CONSTRAINT notnull_idEscola NOT NULL

- O ID de aluno referencia o ID de um determinado aluno e o ID da escola referencia o ID de uma escola existente:
    - CONSTRAINT foreignkey_idAluno REFERENCES Aluno(idAluno)
    - CONSTRAINT foreignkey_idEscola REFERENCES Escola(idEscola)

AlunoRealiza –

- Não pode haver dois conjuntos de IDs de aluno e de exame iguais:
  - PRIMARY KEY (idAluno, idExame)

- Um aluno quando realiza um exame vai como aluno interno (‘S’) ou externo (‘N’):
  - CONSTRAINT notnull_serInterno NOT NULL
  - CONSTRAINT check_serInterno CHECK (serInterno = 'S' OR serInterno = 'N')

- Quando um aluno vai a exame como interno, este encontra-se numa situação de ‘Admitido a Exame’, se não, a sua situação de frequência será ‘Anulou matrícula’, ‘Excluído por faltas’, ‘Reprovou frequência’ ou um valor nulo (para os alunos que já acabaram o secundário e decidiram repetir um determinado exame)
  - CONSTRAINT check_idSitFreq CHECK (sitFrequencia IN ('Admitido a exame', 'Anulou a matrícula', 'Excluído por faltas', 'Reprovou frequência')
  - CONSTRAINT check_idSitFreq_serInt CHECK (CASE WHEN serInterno = 'S' THEN sitFrequencia = 'Admitido a Exame')

- Cada aluno realiza um determinado exame com diversos objetivos: para aprovação (‘S ou ‘N’), para melhoria de nota (‘S’ ou ‘N’), para prova de ingresso na faculdade (‘S’ ou ‘N’) ou para prova de prosseguimento de estudos (‘S’ ou ‘N’):
  - CONSTRAINT notnull_paraAprov NOT NULL
  - CONSTRAINT check_paraAprov CHECK (paraAprov = 'S' OR paraAprov = 'N')
  - CONSTRAINT notnull_paraMelhoria NOT NULL
  - CONSTRAINT check_paraMelhoria CHECK (paraMelhoria = 'S' OR paraMelhoria = 'N')
  - CONSTRAINT notnull_provaIngresso NOT NULL
  - CONSTRAINT check_provaIngresso CHECK (provaIngresso = 'S' OR provaIngresso = 'N')
  - CONSTRAINT notnull_CFCEPE NOT NULL
  - CONSTRAINT check_CFCEPE CHECK (CFCEPE = 'S' OR CFCEPE = 'N')

- Quando um aluno é interno, este vai realizar um determinado exame com uma classificação interna, que vai desde 0 até 20, sendo 0 o valor por defeito:
  - classInt DEFAULT ‘0’
  - CONSTRAINT check_classInt CHECK (CASE WHEN serInterno 0 'N' THEN classInt = NULL)
  - CONSTRAINT check_classInt_values CHECK (classInt >= 0 AND classInt <= 20)

- Um aluno só faz um exame para aprovação se e só se a sua classificação interna for não nula e inferior a 14:
  - CONSTRAINT check_aluno_aprovado CHECK (paraAprov = 'N' AND classInt >= 14)
  - CONSTRAINT check_aluno_nao_aprovado CHECK (paraAprov='S' AND classInt < 14)

- Todos alunos que realizam um exame têm uma nota nesse exame, que vai desde 0.0 até 20.0, sendo 0.0 o valor por defeito:
  - notaExame DEFAULT ‘0.0’
  - CONSTRAINT notnull_notaExame NOT NULL

    - CONSTRAINT check_notaExame_values CHECK (notaExame >= 0.0 AND notaExame <= 20.0)

- Cada aluno que realiza um exame tem uma classificação final cuja forma de calcular dependerá da nota interna deste (se é nula ou não):
    - CONSTRAINT notnull_classFinal NOT NULL
    - CONSTRAINT check_classFinal CHECK (CASE WHEN classInt IS NULL THEN CAST (notaExame+0.5 AS INT) WHEN classInt IS NOT NULL THEN classFinal = CAST ((CAST (classInt AS REAL)*0.7 + notaExame*0.3) AS INT) END)