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

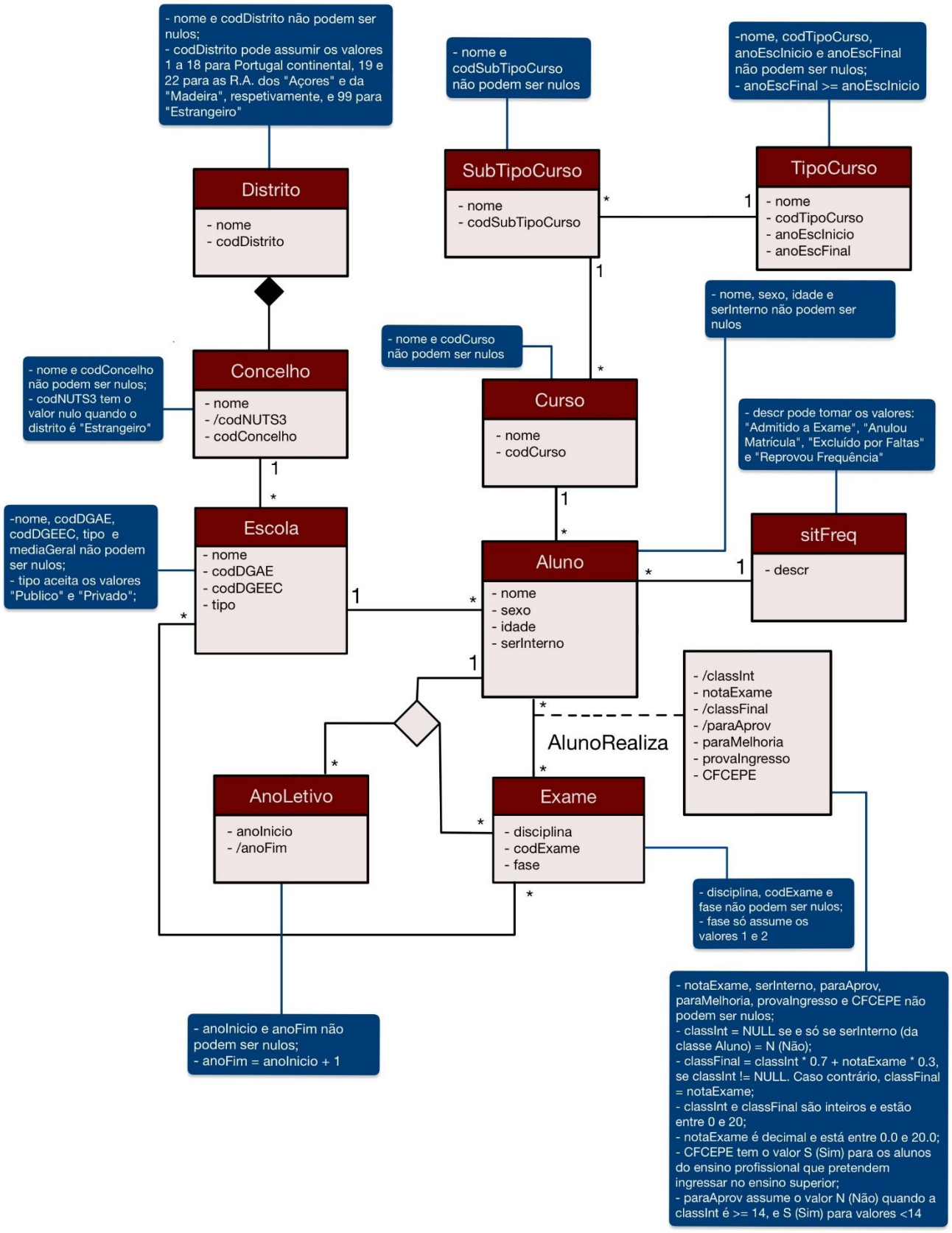
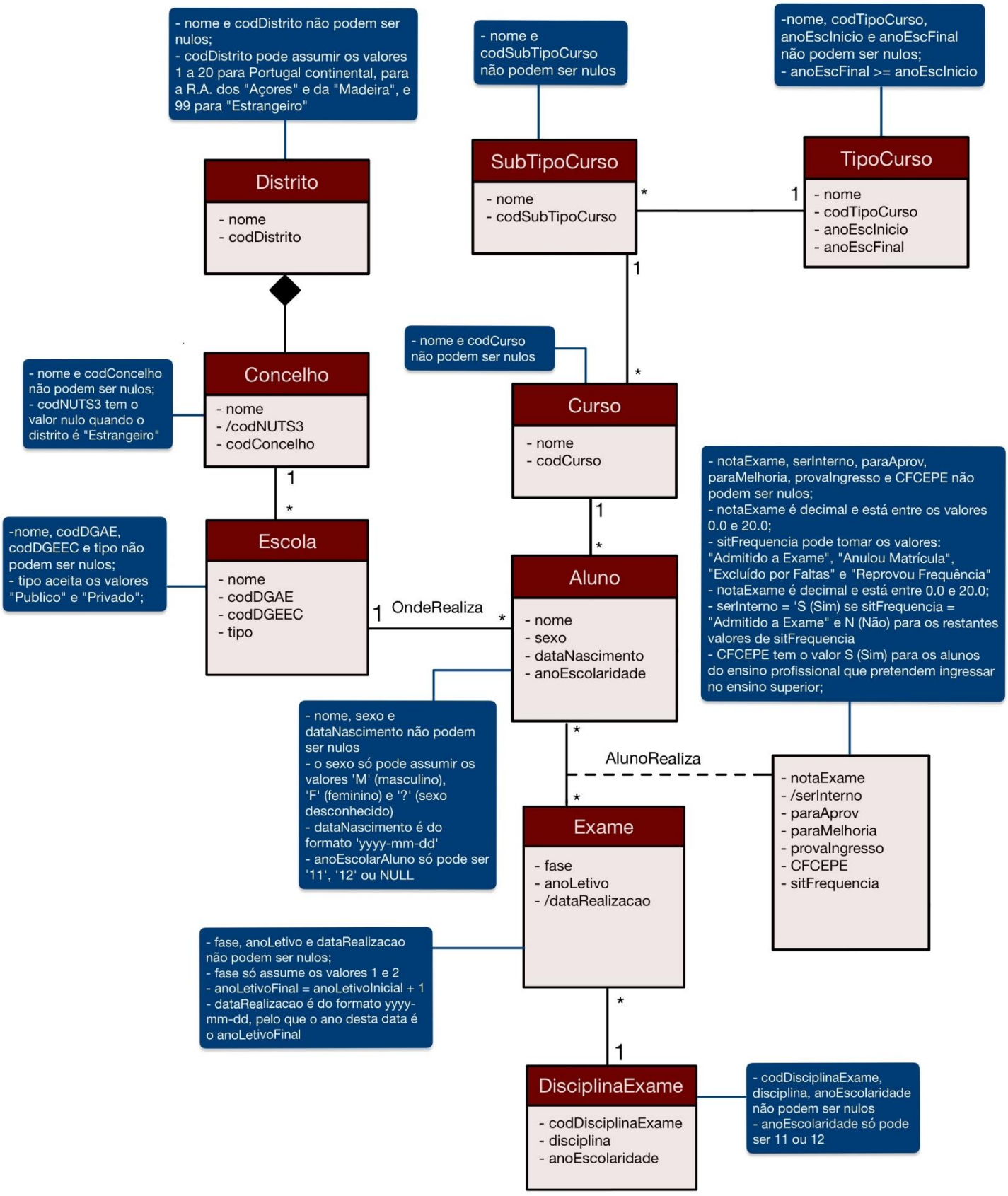


Diagrama UML (Revisto)



NOTA: Para além das correções feitas pelo professor (como a eliminação da classe AnoLetivo e a necessidade de acrescentar o ano de escolaridade em que um exame é realizado), foram introduzidas novas alterações no diagrama UML, no sentido de o tornar mais completo e simples, sendo de destacar: a adição da classe DisciplinaExame, a remoção das classificações internas e finais de cada aluno, a introdução do ano de escolaridade de um aluno e a conversão da classe sitFreq para um atributo da AlunoRealiza).

Esquema Relacional

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

Análise de Dependências Funcionais e de Formas Normais

Uma relação R está em BCNF (Boyce-Codd Normal Form) se, para cada FD A -> B:

- A -> B é trivial;
- A é uma (super)key

Por outro lado, R está em 3NF (3ª forma normal) se, para todas as FDs da relação A -> B:

- A é uma (super)key da relação
- B é apenas formado por atributos primos (atributos pertencentes a uma key)

NOTA: Se uma relação está em BCNF, então também está em 3NF.

Tendo isto em consideração, pode-se afirmar que as únicas FDs encontradas para cada relação da base de dados são FDs triviais, em que a primary key determina os restantes atributos da relação em questão. Sendo assim, conclui-se que todas as FDs da base de dados considerada estão em BCNF e, consequentemente, em 3NF.

Restrições

Distrito –

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

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

- O código tem de ser um número entre ‘1’ e ‘20’ para os distritos de Portugal continental, para a R.A. dos Açores e da Madeira e ‘99’ para o Estrangeiro:
  - CONSTRAINT check_codDistrito CHECK ((codDistrito > 0) AND (codDistrito <= 20 OR codDistrito = 99))

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

- Quando o código do distrito é ‘99’, o nome do distrito tem de ser ‘Estrangeiro’:
  - CONSTRAINT check_nomeEstrangeiro CHECK(CASE WHEN codDistrito = '99' THEN nome = 'Estrangeiro' END)

Concelho –

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

- Todos os concelhos têm de ter um código, um nome e um código de distrito:
  - codConcelho NOT NULL
  - nome NOT NULL
  - codDistrito NOT NULL

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

- 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)

- O código de distrito de um concelho tem de corresponder ao código de distrito de um distrito:
  - CONSTRAINT foreignkey_codDistrito FOREIGN KEY (codDistrito) REFERENCES Distrito(codDistrito) ON DELETE RESTRICT ON UPDATE CASCADE

Escola –

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

- Todas as escolas têm de ter um nome, um código DGAE, um código DGEEC, um tipo e um ID de concelho:
  - nome NOT NULL
  - codDGAE NOT NULL
  - codDGEEC NOT NULL
  - tipo NOT NULL
  - idConcelho int NOT NULL

- Não pode haver nem códigos DGAE nem códigos DGEEC repetidos:
  - CONSTRAINT codDGAE_unique UNIQUE (codDGAE)
  - CONSTRAINT codDGEEC_unique UNIQUE (codDGEEC)

- Toda as escolas têm de ser '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 FOREIGN KEY (idConcelho) REFERENCES Concelho(idConcelho) ON DELETE RESTRICT ON UPDATE CASCADE

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'))

- Todos os tipos de curso têm de ter um nome, um ano escolaridade de início e de fim:
  - nome NOT NULL
  - anoEscInicio NOT NULL
  - anoEscFinal NOT NULL

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

- 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

- Todos os subtipos de curso têm de ter um nome e um código do tipo de curso:
  - nome NOT NULL
  - codTipoCurso NOT NULL

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

- 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)

- O código do tipo de um curso tem de corresponder ao código de um tipo de curso:
  - CONSTRAINT foreignkey_codTipoCurso FOREIGN KEY (codTipoCurso) REFERENCES TipoCurso(codTipoCurso) ON DELETE CASCADE ON UPDATE CASCADE

Curso –

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

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

- O código do subtipo de um curso tem de corresponder ao código de um subtipo de curso:
  - CONSTRAINT foreignkey_codSubTipoCurso FOREIGN KEY (codSubTipoCurso) REFERENCES SubTipoCurso(codSubTipoCurso) ON DELETE CASCADE ON UPDATE CASCADE

Aluno –

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

- Todos os alunos têm de ter um nome, data de nascimento e código de curso:
  - nome NOT NULL
  - dataNascimento NOT NULL
  - codCurso NOT NULL

- Um aluno tem de ter um sexo, pelo que só pode ser do sexo masculino (‘M’) ou feminino (‘F’). Quando não se sabe o sexo deste, recorre-se ao valor ‘?’:
  - sexo NOT NULL ON CONFLICT REPLACE DEFAULT '?'
  - CONSTRAINT check_sexo CHECK (sexo = 'M' OR sexo = 'F')

- Um aluno tem de ser do 11º ou do 12º ano, ou já ter acabado o ensino secundário:
  - CONSTRAINT check_anoEscolaridade CHECK(anoEscolaridade = '12' OR anoEscolaridade = '11' OR anoEscolaridade = NULL)

- O código de curso referencia o código de um curso:

- CONSTRAINT foreignkey_codCurso FOREIGN KEY(codCurso) REFERENCES Curso(codCurso) ON DELETE CASCADE ON UPDATE CASCADE

DisciplinaExame –

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

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

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

- 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)

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:
  - fase NOT NULL
  - anoLetivo NOT NULL
  - dataRealizacao NOT NULL
  - 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 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 FOREIGN KEY(codExame) REFERENCES DisciplinaExame(codExame) ON DELETE CASCADE ON UPDATE CASCADE

OndeRealiza –

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

- Cada aluno realiza o exame numa escola com um certo ID:
    - 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 FOREIGN KEY(idAluno) REFERENCES Aluno(idAluno) ON DELETE CASCADE ON UPDATE CASCADE
    - CONSTRAINT foreignkey_idEscola FOREIGN KEY(idEscola) REFERENCES Escola(idEscola) ON DELETE CASCADE ON UPDATE CASCADE

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’):
    - 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 este for externo, 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_sitFrequencia CHECK (sitFrequencia IN ('Admitido a exame', 'Anulou a matrícula', 'Excluído por faltas', 'Reprovou frequência')
    - CONSTRAINT check_sitFrequencia_serInt CHECK (CASE WHEN serInterno = 'S' THEN sitFrequencia = 'Admitido a Exame' END)

- 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’):
    - paraAprov NOT NULL
    - CONSTRAINT check_paraAprov CHECK (paraAprov = 'S' OR paraAprov = 'N')
    - paraMelhoria NOT NULL
    - CONSTRAINT check_paraMelhoria CHECK (paraMelhoria = 'S' OR paraMelhoria = 'N')
    - provaIngresso NOT NULL
    - CONSTRAINT check_provaIngresso CHECK (provaIngresso = 'S' OR provaIngresso = 'N')
    - CFCEPE NOT NULL
    - CONSTRAINT check_CFCEPE CHECK (CFCEPE = 'S' OR CFCEPE = 'N')

- 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 real NOT NULL ON CONFLICT REPLACE DEFAULT '0.0'
    - CONSTRAINT check_notaExame_values CHECK (notaExame >= 0.0 AND notaExame <= 20.0)

NOTA: Foram introduzidos os parâmetros ‘ON DELETE’ e ‘ON UPDATE’ nas restrições de verificação de foreign keys, de forma a assegurar a integridade dos dados referenciados.