Faculdade de Engenharia da Universidade do Porto

Modelação Conceptual - Clube de Canoagem

Bases de Dados

Grupo 606:

Miguel Fernandes up202207547@fe.up.pt Tomás Vinhas up202208437@fe.up.pt

Xavier Silva up202207183@fe.up.pt

Índice

Lista de figuras

Utilidade da Base de Dados

Eis alguns pontos de vista pelos quais esta base de dados é do interesse da direção de qualquer clube:

1. Saber que sócios é que pagaram as quotas para continuarem a frequentar o clube e a manter o estatuto de sócio.
2. Perceber quantos e quais atletas realizam cada treino para que se consiga perceber a evolução destes e o grau de empenho de cada um.
3. Ter em conta o escalão de um atleta para se ter uma ideia da sua margem de progressão (quanto mais velhos, menos conseguem evoluir, geralmente).
4. Apontar o número de medalhas, pontos que obtém nas provas e a posição final de um atleta, para observar com facilidade o seu sucesso (ou insucesso) desportivo.
5. Registar a quantidade de material disponível, para que em caso de carência face ao número de atletas, se possa adquirir mais.
6. Anotar o ano de fabrico do material para que este possa ser substituído caso seja necessário (geralmente, quanto mais velho o material, em pior estado se encontra).
7. Entender o motivo de um atleta não terminar a prova para identificar um padrão caso que se torne recorrente.
8. Saber a que atleta está associado cada remo/embarcação para identificar quem o utiliza em caso de danos no material.

Lista de Restrições

- *CHECK* (dataNascimento < '2005-01-01') – "PessoaSocio" e "PessoaTreinador"
  Para se tornar sócio ou treinador do clube, é necessária a maioridade (é incrementado anualmente).

- *UNIQUE* - "telemovel", "nif", "numSocio", "numFederado"
  O número de telemóvel de cada pessoa é único exceto o dos atletas, pois alguns atletas mais novos podem não ter telemóvel, deixando o número do encarregado de educação na base de dados (que também pode fazer parte do clube, havendo dois iguais); NIF, número de sócio e número de federado têm que ser únicos por motivos óbvios.

- *DEFAULT NULL* - "numPista"
  O número da pista só existe nas provas de velocidade, pelo que deve permanecer *null* para os restantes tipos de prova.

- *NOT NULL*
  Todos os atributos destacados como *NOT NULL* são essenciais na base de dados, quer para o correto funcionamento da mesma, quer por ser informação que de nenhuma forma pode ser descartada.

Alterações ao UML da primeira entrega

1. Removemos a *constraint* “mandato=2” relativa à entidade “Presidente” por não termos nenhum atributo mandato pois seria pouco relevante para a base de dados em questão e tornaria o esquema confuso.
2. Eliminamos a relação “elege” entre as entidades “Sócio” e “Presidente”, e “escolhe” entre esta última classe e “Treinador” por se tratar de informação inútil - desta iriam surgir tabelas com todos os sócios/treinadores ligados ao mesmo presidente.
3. Eliminamos o atributo que diz os clubes representados pelo treinador, por acharmos que esta informação não é útil para gerir um clube, mas sim no momento anterior à contratação do treinador.
4. Adicionamos o atributo “telemovel” a “Pessoa” para ter um contacto associado a cada elemento do clube.
5. Retiramos a associação múltipla entre “Atleta”, “Treinador”, “Treino” e “Material” devido à sua dificuldade de leitura e problemas futuros que com ela iriam provavelmente surgir, devido à sua complexidade.
6. Removemos o atributo “quantidade” de “Material” pois, tendo acesso ao material existente é-nos possível determinar esse número. Adicionalmente, “quantidade” não resume bem a funcionalidade deste atributo dado que queremos saber a quantidade de remos e de embarcações, não de um remo/embarcação específico.
7. Acrescentamos um “idMaterial” para que, na ligação “possui”, um atleta esteja relacionado a um material específico.
8. De forma semelhante à alínea acima, adicionamos um “idProva” para que um “Atleta” seja ligado a uma “Prova” específica.
9. “Treino” passa a ser classe associativa da relação entre “Treinador” e “Atleta”, fruto da remoção da associação múltipla e por acharmos que faz mais sentido, devido a resultar diretamente da relação entre estas duas entidades.
10. Retiramos a *constraint* a limitar o número de atletas em função do material fruto da remoção da associação múltipla.
11. Acrescentamos o nome à relação entre “Atleta” e “Prova” destinada a quando o atleta não conclui a prova (“desiste”).
12. Adicionamos o atributo “horaInicio” à entidade “Prova” para melhor distinção, uma vez que ocorrem muitas provas diferentes num só dia.
13. Adicionamos o atributo “data" à classe associativa “Treino” para haver uma melhor distinção entre treinos.

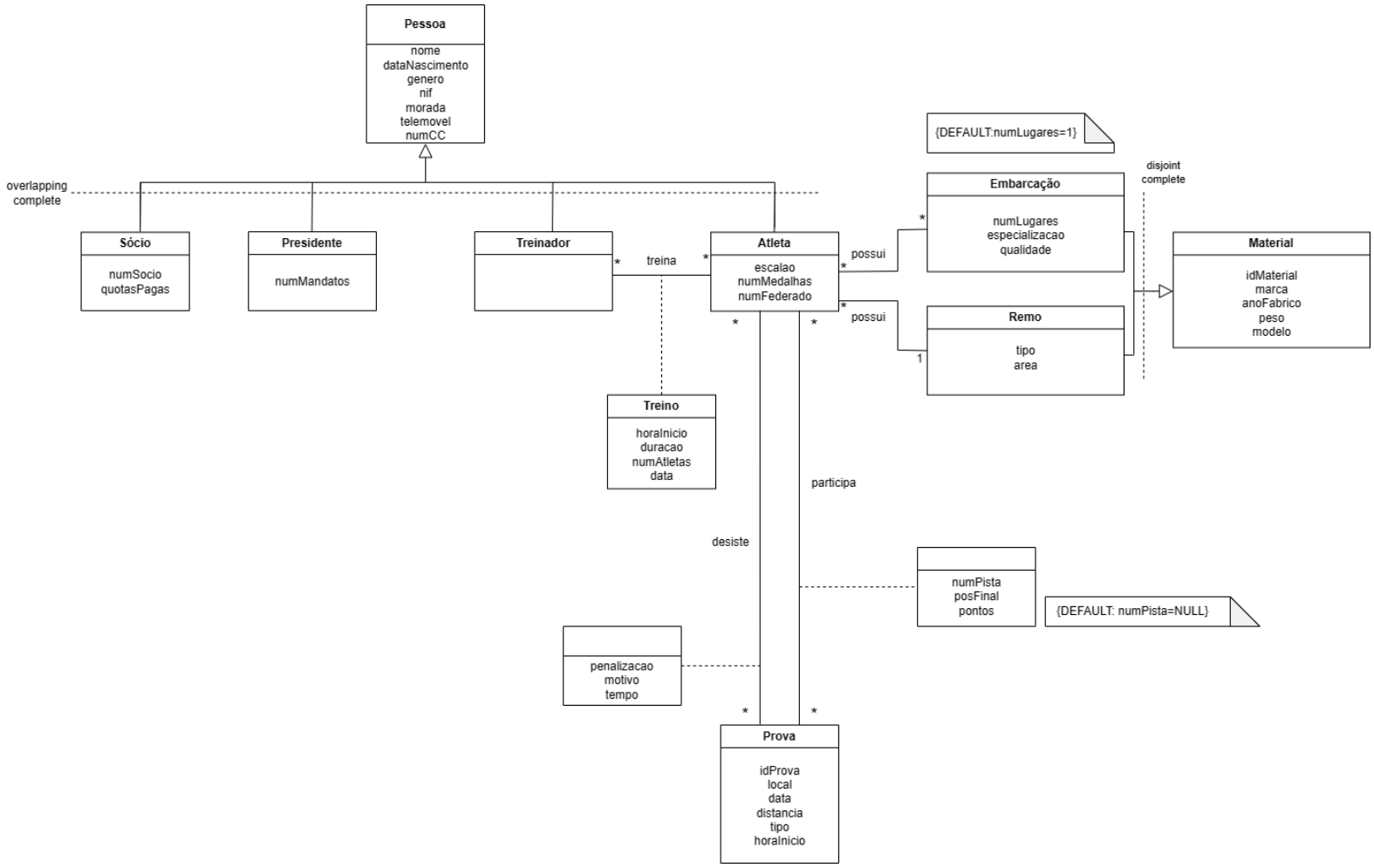

Figura 1- Esquema relacional melhorado

Esquema Relacional e Dependências Funcionais

PessoaSocio(numCC, nome, dataNascimento, genero, nif, morada, telemovel, numSocio, quotasPagas)
Dependências Funcionais:
{numCC} —> {nome, dataNascimento, genero, nif, morada, numSocio, telemovel, quotasPagas}
{nif}->{numCC,nome, dataNascimento, genero, morada, numSocio, telemovel, quotasPagas}
{numSocio} —>{numCC, nome, dataNascimento, genero, nif, morada, telemovel, quotasPagas}
{telemovel} —>{numCC, nome, dataNascimento, genero, morada, nif, numSocio, quotasPagas}
{nome, morada, dataNascimento} —> {numCC, genero, nif, numSocio, telemovel, quotasPagas}
BCNF: Sim | **3NF**: Sim

PessoaPresidente(numCC, nome, dataNascimento, genero, nif, morada, telemovel, numMandatos)
Dependências Funcionais:
{numCC} —> {nome, dataNascimento, genero, nif, morada, telemovel, numMandatos}
{nif} —>{numCC, nome, dataNascimento, genero, morada, telemovel, numMandatos}
{telemovel} —>{numCC, nome, dataNascimento, genero, nif, morada, numMandatos}
{nome, morada, dataNascimento} —> {numCC, genero, nif, numSocio, telemovel, numMandatos}
BCNF: Sim | **3NF**: Sim

PessoaAtleta(numCC, nome, dataNascimento, genero, nif, morada, telemovel, numFederado, numMedalhas, escalao, idMaterial—>MaterialRemo)
Dependências Funcionais:
{numCC} —> {nome, dataNascimento, genero, nif, morada, telemovel, numMedalhas, escalao, numFederado,idMaterial—>MaterialRemo}
{nif} —>{numCC, nome, dataNascimento, genero, morada, telemovel, numMedalhas, escalao, numFederado, idMaterial—>MaterialRemo}
{numFederado} —>{numCC, nome, dataNascimento, genero, nif, morada, telemovel, numMedalhas, escalao, idMaterial—>MaterialRemo}
{nome, morada, dataNascimento} —> {numCC, genero, nif, numSocio, telemovel, numMedalhas, escalao, numFederado,idMaterial—>MaterialRemo}
BCNF: Sim | **3NF**: Sim

PessoaTreinador(numCC, nome, dataNascimento, genero, nif, morada, telemovel)
Dependências Funcionais:
{numCC} —> {nome, dataNascimento, genero, nif, telemovel, morada}
{nif} —>{numCC, nome, dataNascimento, genero, telemovel, morada}
{telemovel} —>{numCC, nome, dataNascimento, genero, nif, morada}
{nome, morada, dataNascimento} —> {numCC, genero, nif, numSocio, telemovel}
BCNF: Sim | **3NF**: Sim

Em todas as *child-classes* que surgem de "Pessoa", podemos ver nas suas dependências funcionais que para qualquer relação R não trivial A —> B, A é chave, dado que tanto o número de cartão de cidadão, como o NIF, o número de federado e o nome em conjunto com a morada e a data de nascimento nos permitem chegar a apenas um atleta. Assim sendo, podemos concluir que "PessoaAtleta", "PessoaTreinador", "PessoaSocio" e "PessoaPresidente" se encontram na *Boyce-Codd Normal Form* (BCNF) e consequentemente na *Third Normal Form* (3NF) - como esta forma normal diz que A tem sempre que ser uma chave/superchave ou B tem que ser composto apenas por atributos que fazem parte de pelo menos uma chave da relação, então qualquer relação que esteja normalizada em BCNF, está também escrita em 3NF.

MaterialEmbarcacao(idMaterial, marca, modelo, anoFabrico, peso, numLugares, especializacao, qualidade)
Dependências Funcionais:
{idMaterial} —> {marca, anoFabrico, peso, modelo, numLugares, especialização, qualidade}
BCNF: Sim | **3NF**: Sim

MaterialRemo(idMaterial, marca, modelo, anoFabrico, peso, tipo, área)
Dependências Funcionais:
{idMaterial} —> {marca, anoFabrico, peso, modelo, tipo, área}
BCNF: Sim | **3NF**: Sim

Prova(idProva, local, data, distancia, horaInício, tipo)
Dependências Funcionais:
{idProva} —> {local, data, distancia, horaInicio, tipo}
BCNF: Sim | **3NF**: Sim

Treina(idTreino, numCC->PessoaTreinador, numCC->PessoaAtleta, data, horaInicio, duracao, numAtletas)
Dependências Funcionais:
{idTreino} —>{numCC—>PessoaTreinador, numCC—>PessoaAtleta, data, horaInicio, duracao, numAtletas}
BCNF: Sim | **3NF**: Sim

Quanto à “Prova”, “Treina” e às relações respetivas ao “Material”, foi criado um “id” para cada uma destas relações, sendo esta a sua única chave. Como em todas estas dependências funcionais, A é sempre uma chave, encontra-se em BCNF e em 3NF.

Participa(numCC->PessoaAtleta, idProva->Prova, numPista, posFinal, pontos)
Dependências Funcionais:
{numCC—>PessoaAtleta, idProva—>Prova} —> {numPista, posFinal, pontos}
BCNF: Sim | **3NF**: Sim

Desiste(numCC->PessoaAtleta, idProva->Prova, penalizacao, motivo, tempo)
Dependências Funcionais:
{numCC—>PessoaAtleta, idProva—>Prova} —> {penalizacao, motivo, tempo}
BCNF: Sim | **3NF**: Sim

Para cada participação numa prova ou desistência da mesma, um atleta apenas pode participar e desistir uma vez da mesma, sendo o número do cartão de cidadão e o “idProva” as chaves estrangeiras necessárias para chegar aos restantes atributos.

Possui(numCC->PessoaAtleta, idMaterial->MaterialEmbarcacao)
Dependências Funcionais:
{numCC—>PessoaAtleta, idMaterial} —> {}
BCNF: Sim | **3NF**: Sim

Neste caso, apenas guardamos os atributos número de cartão de cidadão e “idMaterial”, que implicam que não haja atributos não chave - é uma relação trivial, logo está em BCNF e também em 3NF.

Após escrevermos todas as dependências funcionais, a IA *PopAi* referiu que estas pareciam estar corretas mas que não é capaz de ter a certeza por haver a possibilidade de uma má interpretação dos dados, devido ao pensamento lógico necessário para a sua identificação. Também pedimos para que esta analisasse se todas as relações se encontravam na 3NF e na BCNF, tendo obtido a afirmação que todas as relações se encontravam em ambas as formas, não sendo necessárias quaisquer alterações.

Influência da AI na solução final (esquema relacional)

A IA *PopAi* foi bastante vaga quando lhe fornecemos o esquema UML e a nossa solução. Primeiramente só definiu cada entidade, depois (a nosso pedido) acrescentou as relações e isto tudo sem especificar qualquer *primary* ou *foreign key*. Com um último pedido nosso forneceu-nos estas informações todas e, para além dos nossos esquemas relacionais estarem iguais, a IA acrescentou uma relação “Pessoa” que atua como uma entidade independente como “Atleta” ou “Treinador” o que, devido a “Pessoa” ser uma entidade-mãe de uma generalização, está incorreto.

Deste modo, decidimos manter o nosso esquema relacional totalmente inalterado.

Reflexão sobre as sugestões da IA (“*create*” e “*populate*”)

Após o fornecimento do nosso ficheiro “*create*” e de toda a informação necessária para que a IA conseguisse “melhorar” o nosso código, esta sugeriu que tentássemos manter os tipos dos nossos dados o mais consistente possível, dando como exemplo a alteração do atributo “anoFabrico” de MaterialRemo do tipo “*YEAR*” para “*INT*”.

Adicionalmente, também nos “disse” para criar uma tabela “Pessoa” (classe mãe da nossa generalização) de modo a não estarmos sempre a escrever todas as informações de cada pessoa quando estas são referidas.

Quando pedimos para que a Inteligência Artificial nos alterasse o ficheiro “populate” está apenas nos disse que não conseguia realizar este pedido, pelo que alteramos o nosso ficheiro “*populate*” de modo a ser incorporado nas tabelas criadas pelo “*create”* da IA.

Após uma longa reflexão, optamos por não alterar nada nos nossos ficheiros “*create*” e “*populate*”. Na passagem do UML para o esquema relacional, pela generalização de “Pessoa” ser *disjoint* *complete,* houve uma omissão desta entidade e consequentemente a sua agregação a cada entidade que fazia parte dessa generalização.

Crítica Geral ao auxílio da IA no projeto

Tanto na primeira como na segunda submissão, reparamos em fragilidades e pontos fortes da inteligência artificial. É fútil recorrer à IA para qualquer exercício que envolva qualquer tipo de lógica ou de geração de dados (como a identificação das dependências funcionais, da 3NF e da BCNF no esquema relacional ou como quando lhe pedimos para gerar exemplos para cada tabela que criamos através do ficheiro “*create*”, respetivamente). Por vezes, esta poderosa ferramenta também erra, como quando incluiu no seu esquema relacional a classe de generalização “Pessoa” nem sempre sendo fidedigna.

Em contraste, permite uma fácil perceção do que é preciso fazer, expondo-nos dados e instruções ou modificando as nossas próprias declarações para o que “acha” ser mais correto. Em termos de escrever código é uma ferramenta fulcral especialmente no ficheiro “*populate*” em que a informação é muito extensa e facilmente se comete algum erro que, por sua vez, é corrigido pela IA.

Para terminar, mesmo não tendo um papel significativo nos fundamentos do projeto, foi uma grande ferramenta de auxílio que nos permitiu entender mais a fundo os nossos erros e falhas facilitando a sua resolução.