Universidade do Minho
Mestrado Integrado em Engenharia Informática
Licenciatura em Ciências da Computação

Unidade Curricular de Bases de Dados

Ano Letivo de 2017/2018

Troca-Turnos da UM

Diana Costa A78985

Marcos Pereira A79116

Sérgio Oliveira A77730

Vítor Castro A77870

Novembro de 2017

BD

Data de Recepção	
Responsável	
Avaliação	
Observações	

Troca-Turnos da UM

Diana Costa A78985

Marcos Pereira A79116

Sérgio Oliveira A77730

Vítor Castro A77870

Novembro de 2017

Este trabalho é, carinhosamente, dedicado a tudo o que nos faz reconhecer o valor da vida.

Mães, pais, padrinhos e avós, obrigado porque nos suportais, dia após dia, alimentando-nos com proteicos bifes de boi, pelas saladas de atum refrescantes e pelo melhor arroz seco com panados. Desta feita, dedico também aos SASUM, para que o esforço implicado neste trabalho lhes sirva de inspiração e que, em breve, venha à sua rede peixe melhor que aquele que é servido na cantina.

Não podemos deixar de vos agradecer, professores do YouTube, por ensinarem matérias que nem vós sabeis, criando interrogações que, ao serem respondidas, nos fazem ascender na escala quântica de bits do conhecimento.

Ao ambiente universitário, por nos ensinar a olhar as pessoas com bons olhos (optometria, vocês são importantes), por nos mostrares a beleza da vida pelas pessoas que por nós passam (vocês merecem, seres fabulosos gerados por derivação da costela de Adão, as entradas mais baratas no Keimódrumo) e por nos empurrares pelas portas do conhecimento (umas vezes com mais jeito, outras nem tanto).

Também ao Elon Musk queremos agradecer, porque um dia a nossa febre por carros se vai revelar na compra de um veículo elétrico de elevado torque, pela criação de tamanha obra de arte, o Tesla Roadster.

Ao condutor da camioneta da AAUM que, todos os dias de manhã, pelas 8:30h, acelera vigorosamente pelo trânsito teimando em passar, repetidamente, cerca de 200m de fila, diariamente.

A Marcelo Rebelo de Sousa, atual Presidente da República, pelo carinho prestado aos portugueses e que, despretensiosamente, nos abrace nos difíceis momentos de avaliação que aí se avizinham.

À CMTV, por conseguir encontrar espécimes humanos de pertinente exploração, que se distinguem pelas mais diversas formas (tiques, sotaques, interpretação) dos outros mundanos.

Aos docentes que nos lecionam certas cadeiras, por nos mostrarem pela experiência quais os caminhos que não devemos seguir.

Ao Criador do Mundo, para que apareça e nos mostre o caminho da Salvação.

Por tudo aquilo em que acreditamos.

May the force be with us.

Resumo

Perante a proposta de criar uma base de dados de raiz, surgiu desde logo a ideia de aliar as Unidades Curriculares de Bases de Dados e o trabalho prático de Desenvolvimento de Sistemas de Software. Assim, o projeto atual baseia-se na implementação de uma base de dados para um sistema de gestão de turnos práticos dos alunos de MIEI, o "Troca-Turnos da UM".

Posto isto, e depois de escolhido o tema, decorreu o levantamento e análise de requisitos, que permitiu a criação de entidades, relacionamentos, cardinalidades e atributos, necessários à realização do Modelo Concetual.

Numa segunda etapa, o trabalho do grupo incidiu sobre converter o Modelo anterior para Modelo Lógico, derivando um conjunto de relações do Modelo Concetual, obedecendo às regras de normalização, e validando o mesmo através das transações com o utilizador.

Como terceira instância, foi necessário transformar o Modelo Lógico em Modelo Físico, através do *MySQL*, sistema de gestão de bases de dados (SGBD) que utiliza a linguagem *SQL* (*Structured Query Language*) como interface. Assim, depois de elaborado o *design* das relações e de transações com o utilizador, foi preciso conjeturar o tamanho necessário à representação dos dados em disco, seguido do povoamento efetivo da base de dados.

Por fim, bastou definir restringir as vistas dos utilizadores e o acesso aos dados. Foi conseguida, depois de algum trabalho e tempo, uma base de dados bem estruturada, manipulável, segura e passível de ser usada em contexto real.

Área de Aplicação: Desenho e arquitetura de Sistemas de Bases de Dados para controlo de turnos do DI-UM.

Palavras-Chave: Turno, UC, Aluno, Docente, Entidade, Atributo, Relacionamento, Base de Dados.

Índice

Índice de Figuras

Índice de Tabelas

1. Definição do Sistema

1.1. Contexto de aplicação do sistema

O curso de Engenharia Informática, na Universidade do Minho, recebe anualmente cerca de 150 pessoas. A licenciatura tem a duração de 3 anos e, todos os anos, há a necessidade de atribuir horários aos alunos inscritos.

Nos últimos anos, o docente responsável de cada unidade curricular abria inscrições para os turnos da sua disciplina em data a definir e os alunos procediam, então, à inscrição no turno que desejavam. No entanto, este processo tem originado múltiplos problemas organizacionais a alguns estudantes uma vez que não se conseguem inscrever em determinado turno a tempo e são, posteriormente, confrontados com sobreposições com outras aulas. Este processo de atribuição de horários tende a tornar-se progressivamente mais complicado, uma vez que as vagas abertas têm vindo a crescer e os alunos retidos anualmente aumentam de ano para ano.

Com a entrada do novo Diretor de Curso, em colaboração com o CESIUM, o centro de estudantes de informática, foi desenvolvida uma aplicação que gere a distribuição dos alunos pelos turnos, de forma a que possam ir a todas as cadeiras em que estão inscritos. Depois de desenvolvida esta aplicação, o Departamento de Informática decidiu que seria importante reunir os dados relativos aos docentes, cadeiras e turnos por eles lecionados e também quais os alunos inscritos em cada um deles.

Assim, o DI contactou-nos, tendo feito uma proposta de 5000€ pela implementação da base de dados, valor este que subiu para 7000€ depois de negociação.

1.2. Fundamentação da implementação da base de dados

O principal objetivo da implementação da base de dados é a manutenção de dados, para que seja possível ao Departamento de Informática fazer o controlo dos alunos em cada ano, bem como conseguir obter dados rapidamente para fazer estatísticas de qualidade.

Esta preocupação com a qualidade do ensino tem sido crescente, principalmente depois da mudança para o modelo Bolonha, em que se registou uma aparente redução da qualidade do curso. Para além disso, têm surgido várias queixas por parte dos alunos de desadequação das UCs lecionadas às necessidades reais no mundo do trabalho. Ainda mais, a percentagem de repetentes a certas cadeiras tem sido muito elevada e a Direção admitiu que necessita de mais dados estatísticos fiáveis e mais específicos do que os fornecidos pelos relatórios internos da Universidade.

Surge, então, a necessidade de implementar um SBD forte e eficiente que possibilite o armazenamento de toda a informação relativa aos estudantes, cadeiras e UCs, semestre após semestre. É de extrema importância construir um SBD compreensível, permitindo assim uma estrutura que facilite a consulta e atualização rápida da base de dados. Como este sistema vai ser usado em importantes estatísticas, é mandatório que a solução a implementar seja segura, garantindo a integridade dos dados e, obviamente, confidencialidade de informação.

1.3. Análise da viabilidade do processo

Para que o SBD do projeto cumpra os requisitos identificados depois do contacto com o cliente, decidimos optar pela implementação de um SBDR (Sistema de Base de Dados Relacional), destacando-se algumas das principais vantagens, relacionadas com o sistema necessário:

Simplicidade – A estruturação de dados de um SBDR evita a complexidade, organizando intuitivamente os dados em tabelas, o que permite um desenvolvimento rápido e a relativo baixo custo, como pedido pelo cliente.

Facilidade de exposição – A capacidade de efetuar manobras de consulta e junção de tabelas, entre outras capacidades, permite a apresentação de dados filtrados por virtualmente qualquer requisito (presente quer em linhas ou colunas), por diferentes ordens e apenas com a informação necessária. Deste modo, os dados apresentados serão apenas os pedidos e relevantes para o trabalho de avaliação de qualidade, sendo esta uma necessidade especificada.

Integridade – A integridade dos dados é essencial e garantida pelos tipos e integridade referencial entre tabelas (através das chaves estrangeiras), sendo de extrema importância para

que as estatísticas obtidas sejam fiáveis. Há então a garantia de precisão e consistência de dados.

Flexibilidade – A capacidade de adaptação de um SBDR é essencial para que a resposta à mudança de requisitos e aumento de dados seja fácil. Futuras alterações também podem ser implementadas facilmente, sem que o sistema e as relações de entre os dados sejam alteradas.

Normalização – É garantido com a normalização que o projeto livra a base de dados de redundância, fomentando a precisão dos dados, sendo este SBDR robusto e confiável.

2. Levantamento e Análise de Requisitos

2.1. Requisitos Levantados

Em conversa com o Diretor do Departamento de Informática, recolhemos informações sobre o que era preciso ser feito (Ver entrevista em anexo 1).

2.1.1 Requisitos de descrição

Deve ser guardada informação quanto a Alunos, UCs e Docentes.

Relativamente a cada Aluno, este tem um Número mecanográfico (único), Nome e Curso.

Para cada UC deve ser armazenado um Código (da universidade, único), Nome, Ano (em que é lecionada, segundo o plano de estudos) e ETCS (valor da UC).

No que toca a cada Docente, deve ser registado o seu Número Mecanográfico (único), Nome e Escola (a que está associado).

Um aluno deve estar associado a um ou mais turnos, referentes às UCs em que está inscrito, em determinado ano. Os docentes lecionam as UCs.

2.1.2 Requisitos de exploração

Deve ser possível:

Obter a lista de cadeiras a que um aluno está inscrito, bem como os respetivos turnos.

Obter a lista de alunos de uma cadeira que determinado docente leciona.

Obter a lista de alunos inscritos no curso.

Obter a lista de alunos por UC.

Obter a lista de docentes que leciona determinada cadeira.

Contagem dos pontos anteriores.

Obter lista de docentes do curso.

Obter o TOP 3 de UCs com mais alunos inscritos e respetivos docentes.

2.1.3 Requisitos de controlo

Vai ser estabelecido o perfil de *admin*, para o Diretor de Curso, que vai ter acesso a toda a informação e operações sobre a base de dados. Poderá inserir Docentes e UCs, dando a sua informação e fazendo as associações estabelecidas na descrição.

Vai também ser estabelecido o perfil de *aluno* que apenas vai poder consultar informações referentes a turnos e UCs, não podendo fazer alterações à base de dados. Já os *docentes* poderão aceder a toda a informação, mas apenas manipular a correspondente aos turnos.

2.2. Análise geral dos requisitos

O aluno é um dos principais pontos de interesse do sistema sendo que, a partir do momento em que se regista, fica identificado pelo seu número mecanográfico. Ele é associado a diversos turnos (um ou mais por cadeira), de acordo com o resultado providenciado pela plataforma SWAP, após seleção de cadeiras.

As cadeiras são lecionadas por um ou mais docentes, que também podem lecionar uma ou mais cadeiras.

Estes docentes podem estar associados a apenas uma escola.

Mais tarde podem acontecer trocas de turno relativamente a determinado aluno.

3. Modelação Concetual

3.1. Apresentação da abordagem de modelação realizada

Para a modelação concetual foi necessário, por ordem de execução, identificar entidades e relacionamentos entre as mesmas, seguido da associação de atributos, tanto às entidades, como às relações. De seguida, seria importante determinar o domínio dos atributos e determinar as chaves candidatas, primárias e alternativas. Depois, era requerido que fosse verificada a existência de redundâncias no modelo, e que o mesmo fosse validado de modo a apoiar as transações requeridas. Por fim, num contexto real, faltava apenas rever com o utilizador o modelo, de forma a garantir que este fosse uma representação “verdadeira” da informação pretendida do empreendimento.

3.2. Identificação e caracterização das entidades

Para o grupo conseguir identificar as entidades, e após a análise dos requisitos, foi necessária a elaboração da tabela seguinte, em que se distinguem as entidades do sistema (Aluno, UC e Docente), descrevem-se as mesmas e analisam-se as suas ocorrências.

Nome da Entidade	Descrição	Sinónimos	Ocorrências
Aluno	Cliente que pretende que lhe seja atribuído um horário para assistir às UCs.	Cliente, Estudante	Um aluno está inscrito a uma ou mais UCs.
UC	Disciplinas que um aluno tem que assistir.	Unidade curricular, Disciplina	Uma UC é assistida por vários alunos e lecionada por vários docentes.
Docente	Quem leciona as Unidades Curriculares.	Professor	Um docente leciona uma ou várias disciplinas.

Tabela 1 – Entidades e caracterizações

3.3. Identificação e caracterização dos relacionamentos

Entre as entidades apresentadas acima, foram estabelecidos relacionamentos e as respetivas cardinalidades, após a leitura da análise de requisitos e a representação dos dados numa tabela, como a seguinte.

Nome da entidade	Multiplicidade	Relacionamento	Multiplicidade	Nome da entidade
Docente	N	Leciona	N	UC
UC	N	Tem inscrito	N	Aluno

Tabela 2 - Tabela de relacionamentos

3.4. Identificação e caracterização das associações dos atributos com as entidades e relacionamentos

Para esta etapa do projeto foi necessário associar, apropriadamente, atributos com entidades e relacionamentos. Da mesma maneira que foi procedido para as entidades, deve-se procurar na análise de requisitos por frases que sejam indicativas de possíveis identificadores, características ou propriedades de uma entidade, que resultam, consequentemente e dependendo da sua importância, no conjunto dos seus atributos.

3.4.1 Atributos simples/compostos

Tal como dito anteriormente, após a análise dos requisitos, foi possível identificar os atributos referentes a cada entidade. Assim, chegou-se à conclusão que apenas existem atributos simples no sistema atual.

3.4.2 Atributos derivados

Após a análise dos requisitos atuais, verifica-se que não são necessários quaisquer atributos derivados no sistema.

3.4.3 Atributos multivalor

Depois de efetuada a análise de requisitos, conclui-se que não existem atributos multivalor no sistema.

3.4.4 Atributos de relacionamento

Após a observação do levantamento de requisitos, verifica-se a existência de dois atributos de relacionamento, nomeadamente, o “Turno” e o “Ano”, uma vez que um aluno frequenta uma unidade curricular num determinado ano letivo e num determinado turno, apenas.

3.4.5 Associação entre atributos e entidades

Com os atributos identificados, é necessário detalhá-los, por exemplo, atribuindo-lhes um nome que seja significativo para o utilizador. Segue-se uma tabela, referente a toda a informação detalhada de todos os atributos presentes no sistema.

Nome da Entidade	Atributos	Descrição	Tipo e Tamanho	Nulo	Multivalor	Derivado	Composto
Docente	Número	Número de identificação do docente	Inteiro positivo	Não	Não	Não	Não
	Nome	Nome do docente	45 Caracteres Variáveis	Sim	Não	Não	Não
	Escola	Escola a que o docente pertence	45 Caracteres Variáveis	Não	Não	Não	Não
UC	Código	Código da UC	Inteiro positivo	Não	Não	Não	Não
	Nome	Nome da UC	45 Caracteres Variáveis	Sim	Não	Não	Não
	Ano	Ano em que a UC é lecionada no curso	Inteiro positivo	Não	Não	Não	Não
	ECTS	Número de ECTS que a UC vale.	Inteiro positivo	Não	Não	Não	Não
Aluno	Número	Número identificativo do aluno	Inteiro positivo	Não	Não	Não	Não
	Nome	Nome do aluno	45 Caracteres Variáveis	Sim	Não	Não	Não
	Curso	Curso (sigla) que o aluno frequenta	10 Caracteres Variáveis	Não	Não	Não	Não

Tabela 3 - Tabela de associação entre atributos e entidades

3.5. Determinação do domínio dos atributos

O objetivo deste passo é determinar o domínio (conjunto de valores de um determinado tipo) de todos os atributos no modelo. Assim, encontram-se a seguir detalhados todos grupos de valores para cada atributo.

Entidade	Atributo	Domínio e Explicação
Docente	Número	É um inteiro positivo, dado que é um número que identifica um docente (Exemplo: 32998).
	Nome	É uma string com 45 caracteres variáveis, já que explicita o nome do docente (Exemplo: Maria Gomes).
	Escola	É uma string com 45 caracteres variáveis, uma vez que identifica a escola a que o docente pertence (Exemplo: Escola de Engenharia).
UC	Código	É um inteiro positivo, pois apenas identifica uma UC (Exemplo: 435).
	Nome	É uma string com 45 caracteres variáveis, já que explicita o nome da UC (Exemplo: Base de Dados).
	Ano	É um inteiro positivo, de 1 a 3, pois indica o ano em que uma UC é lecionada na licenciatura (Exemplo: 3).
	ECTS	É um inteiro positivo, porque indica os créditos correspondente a uma UC (Exemplo: 5).
Aluno	Número	É um inteiro positivo, dado que é um número que identifica um aluno (Exemplo: 78985).
	Nome	É uma string com 45 caracteres variáveis, dado que explicita o nome do aluno (Exemplo: Joel Santos).
	Curso	É uma string com 10 caracteres variáveis, porque indica a sigla do curso do aluno (Exemplo: MIEI).
Relacionamento	**Atributo**	**Domínio e Explicação**
UC-Aluno	Turno	É uma string com 3 caracteres, onde os dois primeiros caracteres indicam o tipo de turno, e o terceiro o número do turno (dígito inteiro positivo) (Exemplo: PL5).
UC-Aluno	AnoLetivo	É um inteiro positivo, que indica o ano letivo presente (Exemplo: 1718).

Tabela 4 - Domínio dos atributos

3.6. Chaves primárias, secundárias e alternativas

Para determinar as diversas chaves, o raciocínio será o mesmo para cada entidade. Assim, será elaborada uma explicação em detalhe para a primeira entidade anunciada e, para as restantes, a explicação apenas será detalhada caso apresente alguma diferença.

Relativamente ao **Aluno**, seria necessário escolher, em primeiro lugar, as chaves candidatas a serem identificadoras **únicas** desta entidade. Por esta razão, de entre os atributos "Número", "Nome" e "Curso", o grupo decidiu escolher "Número" como chave candidata ("Curso" e "Nome" são más opções, uma vez que vários alunos têm o mesmo curso, e um mesmo nome pode pertencer a vários alunos). Como o "Número" é um inteiro, e como se garante sempre singularidade através de números, decidiu-se que esta seria a chave primária. Desta maneira, e como apenas havia uma chave candidata, não sobram chaves alternativas para esta entidade.

No que toca à **UC**, escolheram-se (entre "Código", "Nome", "Ano" e "ECTS") "Código" e "Nome" para chaves candidatas, já que ambas oferecem singularidade à entidade. Os fatores de peso na decisão da chave primária foram o facto de "Código" ser um número inteiro, ser menos provável de ser alterado, ter com menos caracteres e ser mais fácil de usar do ponto de vista do utilizador. Assim, sobra, para chave alternativa, o "Nome" da UC.

Já para **Docente**, de entre "Número", "Nome" e "Escola", escolheu-se, analogamente e pelas mesmas razões da entidade Aluno, "Número" como chave candidata e primária. Deste modo, não restam chaves alternativas.

Por último, para o **Relacionamento UC-Aluno**, existem duas chaves, ambas candidatas, que são "AnoLetivo" e "Turno". Decidiu-se que seriam ambas primárias, porque uma UC tem inscrito um aluno apenas num determinado turno, num determinado ano letivo. Não bastaria o atributo "AnoLetivo" pois pode acontecer de um aluno querer repetir a cadeira em anos diferentes. Assim, não sobram chaves alternativas para esta relacionamento.

3.7. Verificar o modelo para a ocorrência de redundâncias

Este passo do projeto teve como objetivo verificar a ocorrência de redundâncias no modelo e, em caso positivo, removê-las. Existem três atividades neste passo:

- ✓ Reexaminar os relacionamentos (1:1)
- ✓ Remover relacionamentos redundantes
- ✓ Considerar a dimensão do tempo

Como primeira instância, foram verificados os relacionamentos (1:1), como forma a identificar duas entidades que pudessem representar o mesmo objeto, e assim serem

convertidas numa só entidade. Uma vez que não existem relacionamentos com esta cardinalidade, garante-se que não existe redundância a este nível.

Como segunda instância, foi verificada a existência de relacionamentos redundantes, isto é, se existiam mais do que um caminho entre duas entidades que fosse desnecessário. Assim, depois de analisado o modelo em busca desta anomalia, verificou-se que não existia, e, portanto, não existe redundância a este nível.

Por fim, faltava considerar a dimensão de um relacionamento, ou seja, examinar o significado de cada relacionamento, já que estes podem variar ao longo do tempo. Depois de analisado o modelo, verifica-se que não existem redundâncias a este nível (em função do tempo).

3.8. Detalhe ou generalização de entidades

Após a revisão de todo o modelo concetual, e com a verificação do modelo para a ocorrência de redundâncias da secção anterior, conclui-se que não foi necessário generalizar nem detalhar nenhuma entidade.

3.9. Apresentação e explicação do diagrama ER

Apresenta-se, em baixo, o diagrama ER, com o objetivo de representar concetualmente as entidades e os relacionamentos entre as entidades, na base de dados. Depois de todas as secções anteriores e de todas as explicações, resta apenas fazer um resumo visual do modelo concetual, de modo a assimilar toda a informação.

Assim, pode-se ver no desenho, o modelo final do grupo, mais concretamente o relacionamento entre Aluno e UC, e entre UC e Docente. Ambos os relacionamentos são N:N, indicando que um Aluno está inscrito a várias UC's, e uma UC tem vários Alunos inscritos. Este relacionamento tem dois atributos, que permitem lidar com informação relativa a turnos num determinado ano letivo. O outro relacionamento indica que um Docente leciona várias UC's, e uma UC é lecionada por vários Docentes.

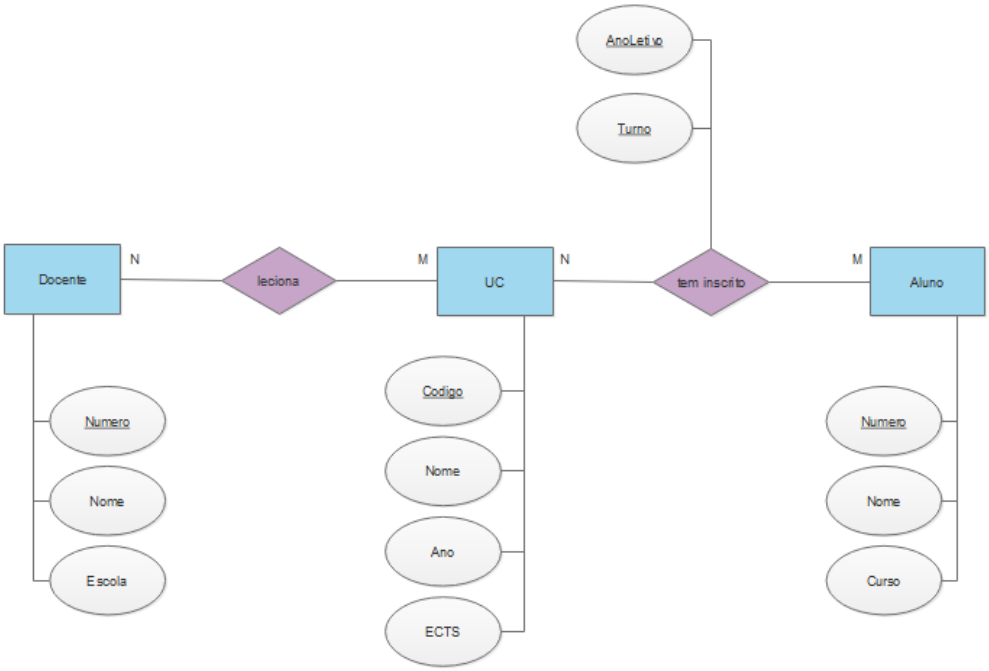

Figura 1 - Modelação Concetual

3.10. Validação do modelo de dados e *queries* com o utilizador

Neste ponto, já se garante um modelo concetual que representa a informação requerida pelo o utilizador. É agora necessário verificar o modelo relativamente às transações e requisitos do utilizador, isto é, se consegue responder às interrogações exigidas pelo utilizador. Em caso negativo, significa que houve algum erro na modelação, e alguns passos da modelação terão de ser reanalisados.

Desta forma, as interrogações retiradas da análise de requisitos, que foram implementadas pelo grupo, seguem-se em baixo, juntamente com uma explicação que garante a resposta às *queries*.

- ✓ **Quais as cadeiras a que um aluno está inscrito?**
  Para saber as cadeiras atuais de um aluno específico, são necessárias as entidades aluno, UC e os respetivos números identificativos e código, e consegue-se assim responder com sucesso a esta interrogação, retornando as cadeiras pretendidas.

- ✓ **Quais são os turnos a que um aluno está inscrito?**
  Para saber os turnos atuais de um aluno específico, são necessárias as entidades aluno, UC e os respetivos números identificativos e código, e consegue-se assim responder com sucesso a esta interrogação, retornando os turnos requeridos.

- ✓ **Quais são os alunos, por UC, que um determinado docente leciona?**
  Para responder a esta *query*, serão necessárias as três entidades do modelo, e consegue-se executar com sucesso através da comparação entre número de aluno, código de UC e número do docente pretendido.

- ✓ **Quais os alunos inscritos no curso?**
  Com o objetivo de saber quais são os alunos inscritos (nome e número) no curso, basta analisar a tabela da entidade aluno, e é assim conseguida executar com sucesso a interrogação.

- ✓ **Quais os alunos inscritos por UC?**
  Com o objetivo de saber quais são os alunos inscritos (nome e número) por UC, basta analisar a entidade aluno e UC, e é assim conseguida uma resposta válida à interrogação.

- ✓ **Qual o total de UCs que estão a ser lecionadas no curso?**
  Para saber quantas são as cadeiras a ser lecionadas, de momento, no curso, basta analisar a tabela da entidade UC, contando as mesmas. Consegue-se assim responder com sucesso a esta interrogação.

- ✓ **Quantos alunos estão inscritos no curso?**
  Para conseguir saber quantos são os alunos inscritos no curso, basta analisar a tabela da entidade aluno, contando os mesmos. É assim conseguida executar com sucesso a interrogação.

- ✓ **Qual o Top 3 de UCs com mais alunos inscritos, e quais os docentes que as lecionam?**
  Esta interrogação tem fins, acima de tudo, estatísticos, uma vez que permite identificar as cadeiras com um maior número de alunos inscritos (diretamente proporcional à taxa de reprovação), com o objetivo de tomar medidas que beneficiem tanto os alunos, como os docentes, e também a taxa de sucesso académico. Assim, através da informação (atributos) de todas as entidades, como o código da UC, número do aluno e número do docente foi possível responder a esta interrogação com sucesso.

Conclui-se assim que o modelo responde a todas as *queries*, e após a revisão do utilizador, nenhum problema foi detetado, o que indica que o modelo foi aceite.

4. Modelação Lógica

4.1. Construção e validação do modelo de dados lógico

Nesta parte vai-se reproduzir o modelo lógico com base no modelo concetual anteriormente estabelecido. Por isso, derivam-se os relacionamentos para o modelo lógico criando tabelas ou relações a fim de se representar as entidades, os atributos e os relacionamentos que foram identificados.

4.1.1 Entidades Fortes

Na modelação concetual realizada estão presentes três entidades: Docente, UC e Aluno. Como tal, por serem independentes umas das outras, são então também entidades fortes. Por isso, para cada uma delas, vai ser criada uma tabela, no modelo lógico, com os seus atributos.

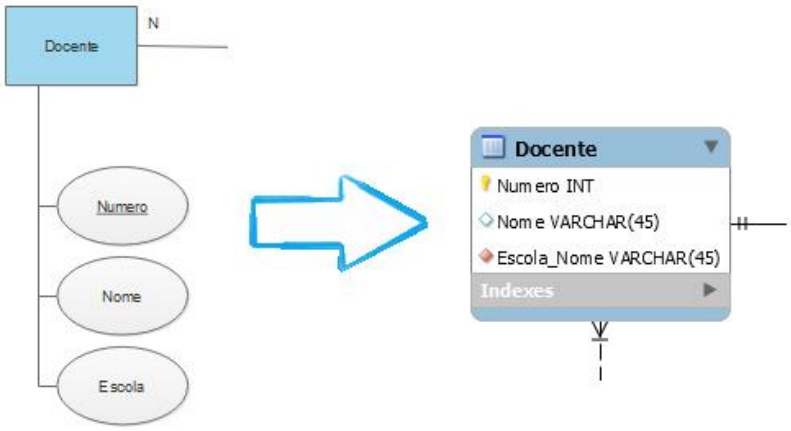

Figura 2 - concetual para lógico (Entidade Docente)

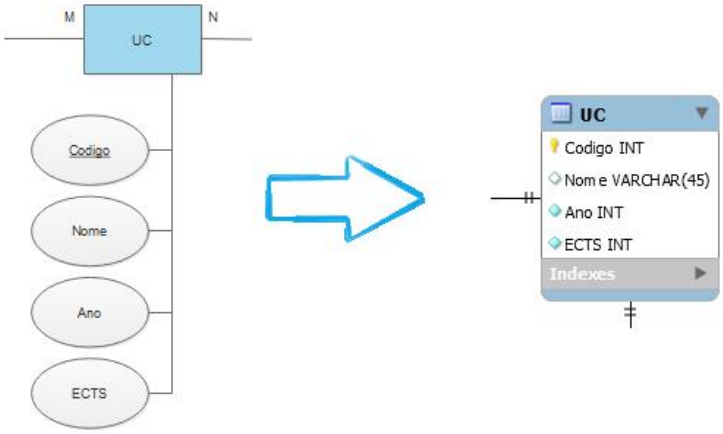

Figura 3 –concetual para lógico (Entidade UC)

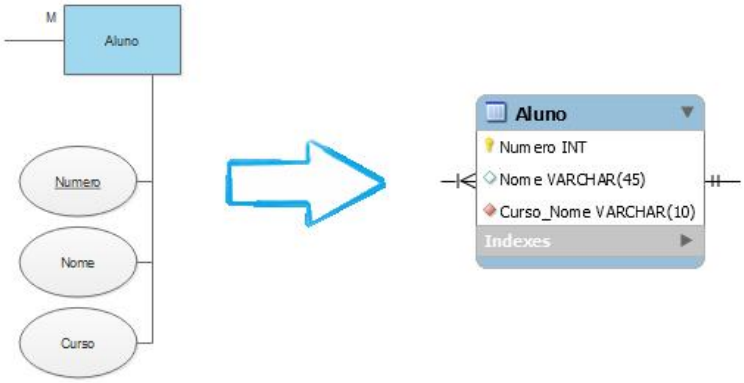

Figura 4 – concetual para lógico (Entidade Aluno)

4.1.2 Entidades Fracas

Neste modelo concetual não existem entidades fracas.

4.1.3 Relacionamentos binários 1:N

Apesar de, no modelo concetual, não existirem relacionamentos binários 1:N, surgiu a ideia de ser possível evitar que, por erro de introdução, surjam cursos que são o mesmo mas que estejam escritos de maneira diferente. Ex: MIEI vs MiEI. A ocorrência de algo do género criaria dois cursos diferentes na base de dados, quando correspondem ao mesmo. A mesma situação acontece com as escolas a que os docentes pertencem.

Assim, as entidades Docente e Aluno vão possuir uma chave estrangeira com a referência a duas novas tabelas que vão ter o nome da Escola e o nome do Curso, respetivamente.

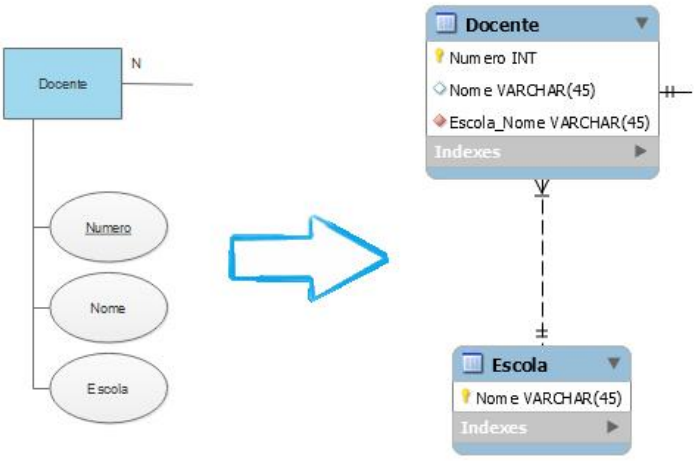

Figura 5 – concetual para lógico (atributo Escola para tabela)

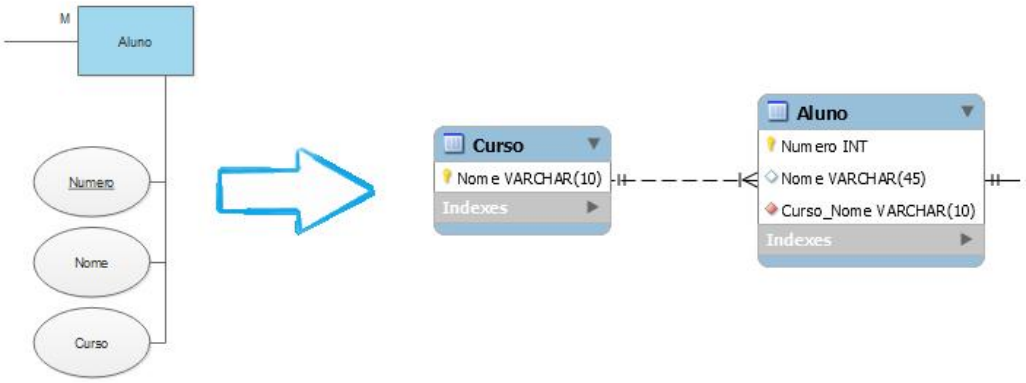

Figura 6 – concetual para lógico (atributo Curso para tabela)

4.1.4 Relacionamentos N:N

Neste modelo concetual existem relacionamentos N:N. Assim, para cada um deles, criou-se uma tabela com uma cópia dos atributos que são chave primária das entidades que participam no relacionamento, mas que nesta nova tabela serão chaves estrangeiras. Além disso, pelo menos uma das chaves estrangeiras terá de fazer parte de chave primária da nova tabela em combinação com os atributos que fazem parte da relação.

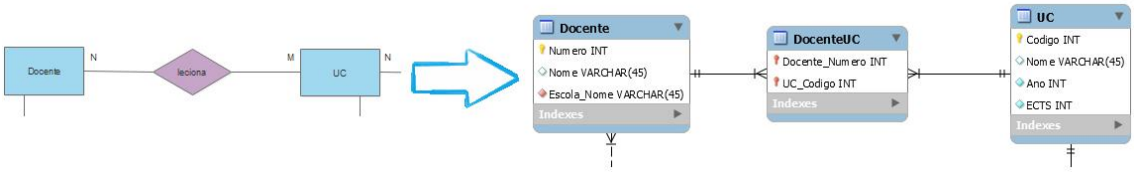

Figura 7 – concetual para lógico (relacionamento N:N entre Docente e UC)

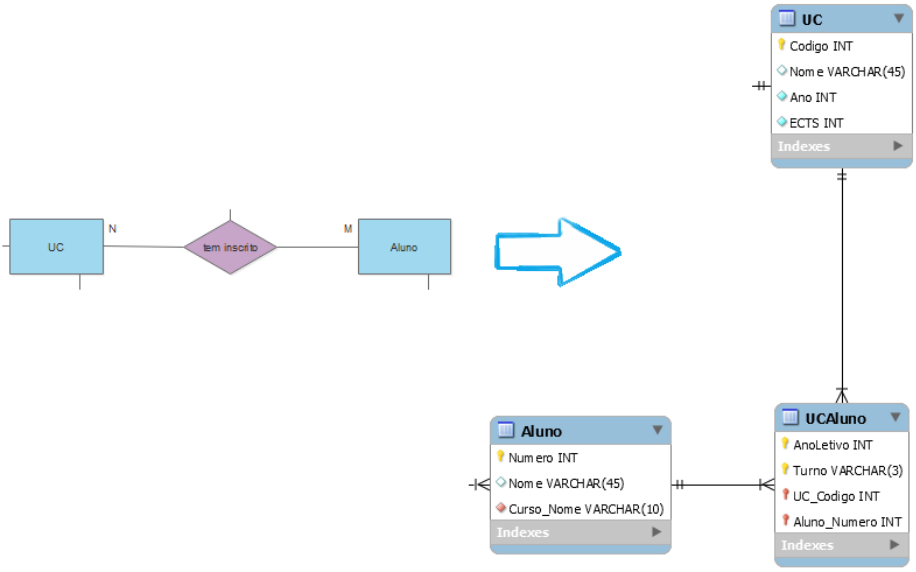

Figura 8 – concetual para lógico (relacionamento N:N entre Aluno e UC)

4.1.5 Relacionamentos 1:1

No modelo concetual não existem relacionamentos 1:1.

4.1.6 Relacionamentos recursivos 1:1

No presente modelo concetual não existem relacionamentos recursivos 1:1.

4.1.7 Relacionamentos superclasse/subclasse

No modelo concetual existente não existem relacionamentos superclasse/subclasse.

4.1.8 Relacionamentos complexos

Neste modelo concetual não existem relacionamentos complexos.

4.1.9 Atributos multivalor

No modelo concetual não existem atributos multivalor.

4.2. Desenho do modelo lógico

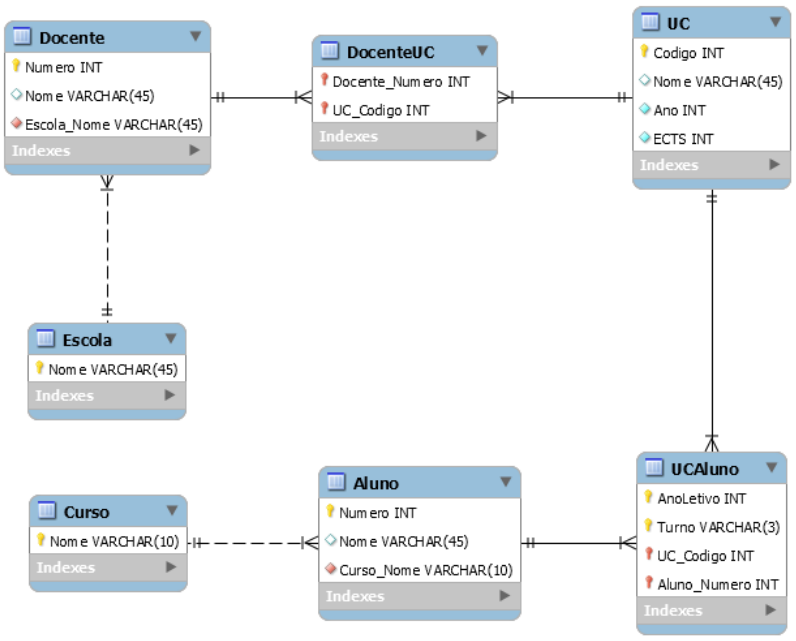

Figura 9 – Modelação Lógica

4.3. Validação do modelo com através da normalização

Neste momento, e com vista a eliminar redundâncias, urgindo aumentar o desempenho e aumentar a integridade de dados, procede-se à normalização dos dados até à terceira forma normal, inclusive.

Primeira forma normal – no modelo lógico não existem entidades com atributos repetidos e multivalorados, ou seja, têm um e um só valor. Assim todas as tabelas estão na 1FN.

Segunda forma normal – uma vez que todas as tabelas com um só atributo como chave primária estão, automaticamente, na 2FN (desde que na 1FN), e que temos tabelas com chaves primárias compostas, então é apenas para estas que é necessária análise. Além disso todas as tabelas estão na 1FN.

A tabela DocenteUC não tem atributos não chave por isso está na 2FN;

A tabela UCAluno não tem atributos não chave por isso está na 2FN;

A tabela Aluno tem atributos não chave. Nome depende de Número e Curso_Nome. Pensar acerca da segunda forma normal permitiu encontrar um erro de modelação lógica da tabela Aluno uma vez que, inicialmente, o atributo Curso_Nome estava definido como chave estrangeira e como chave primária e percebeu-se que se poderia ter dependências funcionais pelo que deixou, então, de ser primária. É também percetível que, se Curso_Nome e Numero

pertencessem à chave primária composta, seria possível inscrever o mesmo aluno (com o mesmo ID) em dois cursos diferentes e isso estaria errado.

Terceira forma normal – todas as tabelas estão na 2FN e não há como haver dependências transitivas em colunas com menos de dois atributos não chave. Por isso, podemos, desde já, afirmar que as tabelas Curso, UCAluno e DocenteUC estão na 3FN. Analisando as restantes que têm dois ou mais atributos não chave, não há nenhuma em que existam atributos não chave dependentes de outros atributos não chave. Logo, todas as tabelas estão na 3FN.

4.4. Validação do modelo com interrogações do utilizador

Procedeu-se à verificação das queries que podem ser satisfeitas de acordo com o modelo lógico. id_aluno e id_docente são, respetivamente, o número do aluno e docente para o qual se faz as interrogações.

1. Cadeiras em que um aluno está inscrito.

∏UC.Nome (∂Aluno_Numero = 'id_aluno' ((UC) ∞Codigo=UC_Codigo (UCAluno)))

2. Turnos em que um aluno está inscrito.

∏UC.Nome,UCAluno.Turno (∂Aluno_Numero = 'id_aluno' ((UCAluno) ∞UC_Codigo=Codigo (UC)))

3. Alunos inscritos à cadeira que um docente leciona.

∏Aluno.Nome,UC.Nome (∂Docente_Numero = 'id_docente' (
(Aluno) ∞Numero=Aluno_Numero (UCAluno) ∞UC_Codigo=Codigo (UC) ∞Codigo=UC_Codigo (DocenteUC)))

4. Todos os alunos do curso.

∏Numero,Nome (Aluno)

5. Alunos e respetivas cadeiras.

∏Aluno.Numero,Aluno.Nome,UC.Nome (
(Aluno) ∞Numero=Aluno_Numero (UCAluno) ∞UC_Codigo=Codigo (UC))

6. Total de UCs lecionadas no curso.

⌡COUNT Codigo (UC)

7. Total de alunos inscritos no curso.

⌡COUNT Numero (Aluno)

8. Número de cadeiras com mais inscritos e respetivos docentes, por ordem.

∏UC.Nome,COUNT(*),Docente.Nome (⌡ORDER COUNT(*) DESC (⌡GROUP UC.Nome (
(Docente) ∞Numero=Docente_Numero (DocenteUC) ∞UC_Codigo=Codigo (UC)
∞Codigo=UC_Codigo (UCAluno) ∞Aluno_Numero=Numero (Aluno))))

4.5. Validação do modelo com as transações estabelecidas

Em relação a este tópico, decidiu-se implementar seis transações que são importantes e que são suportadas e concretizáveis a partir do modelo lógico.

- ✓ **criação de um novo aluno** – insere-se um novo docente na tabela Aluno incluindo o seu número, nome e nome do curso;
- ✓ **criação um novo docente que leciona UCs** – insere-se um novo docente na tabela Docente incluindo o seu número, nome e nome da escola;
- ✓ **criação de UCs** – insere-se uma nova unidade curricular na tabela UC incluindo o seu código, nome, ano e ECTS;
- ✓ **mudar o nome de uma UC já existente no sistema** – altera-se o nome da UC na tabela UC incluindo o seu novo nome;
- ✓ **mudar o nome do curso do aluno** – altera-se o nome do curso na tabela Curso incluindo o seu novo curso, se ainda não existir;
- ✓ **mudar o turno do aluno numa UC** – altera-se o turno do aluno na tabela UCAluno incluindo o seu novo turno.

4.6. Revisão do modelo lógico com o utilizador

Efetuou-se a verificação com o utilizador e verificou-se que o problema em questão estava bem representado no modelo lógico elaborado.

5. Implementação Física

5.1. Seleção do sistema em gestão de bases de dados

O SGBD usado foi o MySQL, e as razões para isso prendem-se pelos seguintes tópicos:

- ✓ Grátis ⟶ investimento inicial reduzido
- ✓ Open-source
- ✓ Multiplataforma
- ✓ Alto desempenho
- ✓ Fácil encontrar ajuda ⟶ comunidade enorme
- ✓ Usado por grandes empresas
- ✓ Fácil de manipular

5.2. Tradução do esquema lógico para o sistema de gestão de base de dados escolhido em SQL

Nesta fase, fez-se a tradução do esquema lógico para o SGBD através da funcionalidade “Forward Engineer” disponibilizada pelo MySQL Workbench (ver anexo 2).

5.3. Tradução das interrogações do utilizador para SQL

```sql
/*
 Os alunos devem ser capazes de obter a lista de cadeiras
 a que estão inscritos.
*/
DROP PROCEDURE IF EXISTS lista_cadeiras;
DELIMITER $$
CREATE PROCEDURE lista_cadeiras(id_aluno INT)
BEGIN
    SELECT UC.Nome FROM UC
        INNER JOIN UCAluno ON UCAluno.UC_Codigo = UC.Codigo
        WHERE UCAluno.Aluno_Numero = id_aluno;
END
$$
DELIMITER ;

CALL lista_cadeiras(78985);
```

Figura 10 – MySQL Query – lista de cadeiras

```sql
/*
 Os alunos devem ser capazes de obter a listagem de turnos
 a que estão inscritos.
*/

DROP PROCEDURE IF EXISTS lista_turnos;
DELIMITER $$
CREATE PROCEDURE lista_turnos(id_aluno INT)
BEGIN
    SELECT UC.Nome, UCAluno.Turno FROM UCAluno
        INNER JOIN UC ON UC.Codigo = UCAluno.UC_Codigo
        WHERE UCAluno.Aluno_Numero = id_aluno;
END
$$
DELIMITER ;

CALL lista_turnos(78985);
```

Figura 11 – MySQL Query – lista de turnos

```
/*
 Os docentes devem ser capazes de obter a lista dos alunos das UC's
 que lecionam.
*/
DROP PROCEDURE IF EXISTS lista_alunos;
DELIMITER $$
CREATE PROCEDURE lista_alunos(id_docente INT)
BEGIN
    SELECT Aluno.Nome, UC.Nome FROM Aluno
    INNER JOIN UCAluno ON UCAluno.Aluno_Numero = Aluno.Numero
    INNER JOIN UC ON UC.Codigo = UCAluno.UC_Codigo
    INNER JOIN DocenteUC ON DocenteUC.UC_Codigo = UC.Codigo
    WHERE DocenteUC.Docente_Numero = id_docente;
END
$$
DELIMITER ;

CALL lista_alunos(79194);
```

Figura 12 – MySQL Query – lista de alunos que lecionam

```
/*
 O diretor de curso deve ser capaz de obter a listagem de todos os
 alunos do curso.
*/
SELECT Aluno.Numero, Aluno.Nome FROM Aluno;
```

Figura 13 – MySQL Query – lista de alunos

```
/*
 O diretor de curso deve ser capaz de aceder à lista
 de alunos e respectivas cadeiras.
*/
SELECT Aluno.Numero, Aluno.Nome, UC.Nome FROM Aluno
    INNER JOIN UCAluno ON UCAluno.Aluno_Numero = Aluno.Numero
    INNER JOIN UC ON UC.Codigo = UCAluno.UC_Codigo;
```

Figura 14 – MySQL Query – lista de alunos e UCs

```
/*
 O diretor de curso deve conseguir saber o total de UC's que
 estão a ser lecionadas no curso.
*/
SELECT COUNT(Codigo) AS numero_ucs FROM UC;
```

Figura 15 – MySQL Query - número de UCs do curso

```
/*
 O diretor de curso deve conseguir saber o total de Alunos
 que o curso tem atualmente.
*/
SELECT COUNT(Numero) AS numero_alunos FROM Aluno;
```

Figura 16 – MySQL Query - número de alunos no curso

```sql
/*
 O diretor de curso deve ser capaz de saber qual o top 3
 de cadeiras com mais alunos inscritos, e respetivos docentes.
*/
SELECT UC.Nome, COUNT(*), Docente.Nome FROM UC
    INNER JOIN UCAluno ON UCAluno.UC_Codigo = UC.Codigo
    INNER JOIN Aluno ON Aluno.Numero = UCAluno.Aluno_Numero
    INNER JOIN DocenteUC ON DocenteUC.UC_Codigo = UC.Codigo
    INNER JOIN Docente ON Docente.Numero = DocenteUC.Docente_Numero
    GROUP BY UC.Nome
    ORDER BY COUNT(*) DESC
    LIMIT 3;
```

Figura 17 – MySQL Query – top 3 de UCs com mais alunos

5.4. Tradução das transações estabelecidas para SQL

```sql
/*
 Adicionar uma nova UC.
*/
DROP PROCEDURE IF EXISTS inserir_uc;
DELIMITER $$
CREATE PROCEDURE inserir_uc(IN codigo INT,
                            IN nome VARCHAR(45),
                            IN ano INT,
                            IN ects INT)
BEGIN
    DECLARE erro BOOL DEFAULT 0;
    DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET erro = 1;
    START TRANSACTION;

    INSERT INTO UC VALUES (codigo, nome, ano, ects);

    IF erro
        THEN ROLLBACK;
        ELSE COMMIT;
    END IF;
END
$$
DELIMITER ;
```

Figura 18 – Transação de criação de nova UC

```sql
/*
 Adicionar um novo docente.
*/
DROP PROCEDURE IF EXISTS inserir_docente;
DELIMITER $$
CREATE PROCEDURE inserir_docente(IN numero INT,
                                 IN nome VARCHAR(45),
                                 IN escola VARCHAR(45))
BEGIN
    DECLARE erro BOOL DEFAULT 0;
    DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET erro = 1;
    START TRANSACTION;

    INSERT INTO Docente VALUES (numero, nome, escola);

    IF erro
        THEN ROLLBACK;
        ELSE COMMIT;
    END IF;
END
$$
DELIMITER ;
```

Figura 19 – Transação de inserção de novo docente

```sql
/*
 Modificar o curso de um aluno.
*/
DROP PROCEDURE IF EXISTS modificar_curso_aluno;
DELIMITER $$
CREATE PROCEDURE modificar_curso_aluno(IN curso_aluno VARCHAR(10),
                                       IN numero_aluno INT)
BEGIN
    DECLARE erro BOOL DEFAULT 0;
    DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET erro = 1;
    START TRANSACTION;

    UPDATE Aluno SET Curso = curso_aluno WHERE Aluno.Numero = numero_aluno;

    IF erro
        THEN ROLLBACK;
        ELSE COMMIT;
    END IF;
END
$$
DELIMITER ;
```

Figura 20 – Transação para alterar curso de aluno

```sql
/*
 Modificar o nome de uma UC.
*/
DROP PROCEDURE IF EXISTS modificar_uc_nome;
DELIMITER $$
CREATE PROCEDURE modificar_uc_nome(IN nome_uc VARCHAR(10),
                                   IN codigo_uc INT)
BEGIN
    DECLARE erro BOOL DEFAULT 0;
    DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET erro = 1;
    START TRANSACTION;

    UPDATE UC SET Nome = nome_uc WHERE UC.Codigo = codigo_uc;

    IF erro
        THEN ROLLBACK;
        ELSE COMMIT;
    END IF;
END
$$
DELIMITER ;
```

Figura 21 – Transação para modificar o nome de UC

```sql
/*
 Modificar o turno de um aluno a uma UC.
*/
DROP PROCEDURE IF EXISTS modificar_turno_aluno_uc;
DELIMITER $$
CREATE PROCEDURE modificar_turno_aluno_uc(IN turno_new VARCHAR(3),
                                          IN codigo_uc INT,
                                          IN numero_aluno INT)
BEGIN
    DECLARE erro BOOL DEFAULT 0;
    DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET erro = 1;
    START TRANSACTION;

    UPDATE UCAluno SET Turno = turno_new WHERE UCAluno.UC_Codigo = codigo_uc
                                           AND UCAluno.Aluno_Numero = numero_aluno;

    IF erro
        THEN ROLLBACK;
        ELSE COMMIT;
    END IF;
END
$$
DELIMITER ;
```

Figura 22 – Transação para mudar turno de aluno

```
/*
 Adicionar um novo aluno.
*/
DROP PROCEDURE IF EXISTS inserir_aluno;
DELIMITER $$
CREATE PROCEDURE inserir_aluno(IN numero INT,
                               IN nome VARCHAR(45),
                               IN curso VARCHAR(10))
BEGIN
    DECLARE erro BOOL DEFAULT 0;
    DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET erro = 1;
    START TRANSACTION;

    INSERT INTO Aluno VALUES (numero, nome, curso);

    IF erro
        THEN ROLLBACK;
        ELSE COMMIT;
    END IF;
END
$$
DELIMITER ;
```

Figura 23 – Transação para inserção de novo aluno

5.5. Escolha, definição e caracterização de índices em SQL

Dado que a base de dados presente não é significativamente grande, não é necessário fazer este tipo de otimizações. Por isso, na opinião do grupo, não há necessidade de usar tabelas de índices.

5.6. Estimativa do espaço em disco da base de dados e taxa de crescimento anual

Nesta fase, faz-se uma estimativa do espaço necessário para a fase inicial da base de dados, pós povoamento.

Assim, em primeiro lugar, é necessário saber qual é o espaço que cada atributo ocupa em cada tabela e, com base neste valor, procede-se à multiplicação pelo número de dados que foram inseridos no povoamento.

Tabela Docente

Atributos	Tipo	Tamanho (Bytes)
Número	INT	4
Nome	VARCHAR(45)	46

Tabela 5 – Tamanho dos atributos de Docente

Para cada entrada da tabela Docente são ocupados 50 bytes. Como existem 12 registos iniciais então são ocupados, no máximo, 600 bytes.

Tabela DocenteUC

Atributos	Tipo	Tamanho (Bytes)
Docente_Número	INT	4
UC_Código	INT	4

Tabela 6 – Tamanho dos atributos de DocenteUC

Para cada entrada da tabela DocenteUC são ocupados 8 bytes. Como existem 30 registos iniciais então são ocupados, no máximo, 240 bytes.

Tabela UC

Atributos	Tipo	Tamanho (Bytes)
Código	INT	4
Nome	VARCHAR(45)	46
Ano	INT	4
ECTS	INT	4

Tabela 7 – Tamanho dos atributos de UC

Para cada entrada da tabela UC são ocupados 58 bytes. Como existem 27 registos iniciais então ocupam-se, no máximo, 1566 bytes.

Tabela Escola

Atributos	Tipo	Tamanho (Bytes)
Nome	VARCHAR(45)	46

Tabela 8 – Tamanho dos atributos de Escola

Para cada entrada da tabela UC são ocupados 46 bytes. Como existem 2 registos iniciais então ocupam-se, no máximo, 92 bytes.

Tabela Curso

Atributos	Tipo	Tamanho (Bytes)
Nome	VARCHAR(45)	46

Tabela 9 – Tamanho dos atributos de Curso

Para cada entrada da tabela UC são ocupados 46 bytes. Como há 1 registo inicial então são ocupados, no máximo, 46 bytes.

Tabela Aluno

Atributos	Tipo	Tamanho (Bytes)
Número	INT	4
Nome	VARCHAR(45)	46

Tabela 10 – Tamanho dos atributos de Aluno

Para cada entrada da tabela UC são ocupados 50 bytes. Como existem 41 registos iniciais então ocupam-se, no máximo, 2050 bytes.

Tabela UCAluno

Atributos	Tipo	Tamanho (Bytes)
AnoLetivo	INT	4
Turno	VARCHAR(3)	4
UC_Codigo	INT	4
Aluno_Numero	INT	4

Tabela 11 – Tamanho dos atributos de UCAluno

Para cada entrada da tabela UC são ocupados 16 bytes. Como existem 127 registos iniciais então ocupam-se, no máximo, 2032 bytes.

Por fim, para se calcular o espaço em disco ocupado, soma-se o tamanho ocupado pelas várias tabelas. Por isso, faz-se 600 + 240 + 1566 + 92 + 46 + 2050 + 2032 = 6626 bytes. Após povoamento, o SGBD ocupa, no máximo, 6626 bytes.

5.7. Definição e caracterização das vistas de utilização em SQL

Este sistema tem dados pouco sensíveis (não existem dados pessoais confidenciais, particularmente). Por este motivo, a utilização de vistas visou apenas a restrição de acesso de certos dados. Esta restrição foi apenas aplicada ao Aluno, sendo que os restantes utilizadores podem ver todos os dados da BD.

```
/*
 View correspondente à informação que um aluno pode ver.
*/
DROP VIEW IF EXISTS viewAluno;
DELIMITER $$
CREATE VIEW viewAluno AS
    SELECT DISTINCT u.nome, u.codigo, u.ano, u.ects, a.turno
    FROM UC u, UCAluno a
    WHERE u.codigo = a.UC_Codigo;

$$
DELIMITER ;
SELECT * FROM trocaturnos.viewAluno;
```

Figura 24 – Vista para aluno

```
GRANT SELECT ON trocaturnos.viewAluno TO 'stud1'@'localhost',
                                        'stud2'@'localhost',
                                        'stud3'@'localhost';
```

Figura 25 - Permissão ao aluno para ver a view.

5.8. Definição e caracterização dos mecanismos de segurança em SQL

```
/*
 Criação de um administrador.
*/
CREATE USER 'admin'@'localhost';
    SET PASSWORD FOR 'admin'@'localhost' = 'admin1';
```

Figura 26 – Criação de administrador

```
/*
 Criação do utilizador do tipo 'docente'.
 (só criei 3 para ser representativo)
*/
CREATE USER 'doc1'@'localhost';
    SET PASSWORD FOR 'doc1'@'localhost' = 'doc1';

CREATE USER 'doc2'@'localhost';
    SET PASSWORD FOR 'doc2'@'localhost' = 'doc2';

CREATE USER 'doc3'@'localhost';
    SET PASSWORD FOR 'doc3'@'localhost' = 'doc3';
```

Figura 27 – Criação de user docente

```
/*
 Criação do utilizador do tipo 'aluno'.
 (só criei 3 para ser representativo)
*/
CREATE USER 'stud1'@'localhost';
    SET PASSWORD FOR 'stud1'@'localhost' = 'stud1';

CREATE USER 'stud2'@'localhost';
    SET PASSWORD FOR 'stud2'@'localhost' = 'stud2';

CREATE USER 'stud3'@'localhost';
    SET PASSWORD FOR 'stud3'@'localhost' = 'stud3';
```

Figura 28 – Criação de user aluno

```
/*
 Permissões do administrador.

 Pode "fazer o que quiser".
*/
GRANT SELECT, INSERT, UPDATE, DELETE ON trocaturnos.* TO 'admin'@'localhost';
```

Figura 29 – Permissões do administrador

```
/*
 Permissões dos docentes.

 Pode consultar TODAS as tabelas, mas apenas inserir, remover
 e atualizar os dados da tabela UCAluno (por causa dos turnos,
 fazer mais um turno, por exemplo)
*/

-- Permissão dos docentes em UCAluno
GRANT SELECT, INSERT, DELETE, UPDATE ON trocaturnos.UCAluno TO 'doc1'@'localhost',
                                                               'doc2'@'localhost',
                                                               'doc3'@'localhost';

-- Permissão dos docentes em Docente
GRANT SELECT ON trocaturnos.Docente TO 'doc1'@'localhost',
                                       'doc2'@'localhost',
                                       'doc3'@'localhost';

REVOKE INSERT, DELETE, UPDATE ON trocaturnos.Docente FROM 'doc1'@'localhost',
                                                          'doc2'@'localhost',
                                                          'doc3'@'localhost';

-- Permissão dos docentes em UC
GRANT SELECT ON trocaturnos.UC TO 'doc1'@'localhost',
                                  'doc2'@'localhost',
                                  'doc3'@'localhost';

REVOKE INSERT, DELETE, UPDATE ON trocaturnos.UC FROM 'doc1'@'localhost',
                                                     'doc2'@'localhost',
                                                     'doc3'@'localhost';

-- Permissão dos docentes em DocenteUC
GRANT SELECT ON trocaturnos.DocenteUC TO 'doc1'@'localhost',
                                         'doc2'@'localhost',
                                         'doc3'@'localhost';

REVOKE INSERT, DELETE, UPDATE ON trocaturnos.DocenteUC FROM 'doc1'@'localhost',
                                                            'doc2'@'localhost',
                                                            'doc3'@'localhost';
```

Figura 30 – Permissões do user docente

```
-- Permissão dos docentes em Aluno
GRANT SELECT ON trocaturnos.Aluno TO 'doc1'@'localhost',
                                     'doc2'@'localhost',
                                     'doc3'@'localhost';

REVOKE INSERT, DELETE, UPDATE ON trocaturnos.Aluno FROM 'doc1'@'localhost',
                                                        'doc2'@'localhost',
                                                        'doc3'@'localhost';

-- Permissão dos docentes em Escola
GRANT SELECT ON trocaturnos.Escola TO 'doc1'@'localhost',
                                      'doc2'@'localhost',
                                      'doc3'@'localhost';

REVOKE INSERT, DELETE, UPDATE ON trocaturnos.Curso FROM 'doc1'@'localhost',
                                                        'doc2'@'localhost',
                                                        'doc3'@'localhost';

-- Permissão dos docentes em Curso
GRANT SELECT ON trocaturnos.Curso TO 'doc1'@'localhost',
                                     'doc2'@'localhost',
                                     'doc3'@'localhost';

REVOKE INSERT, DELETE, UPDATE ON trocaturnos.Curso FROM 'doc1'@'localhost',
                                                        'doc2'@'localhost',
                                                        'doc3'@'localhost';
```

Figura 31 – Permissões do user docente - continuação

```
/*
 Permissões dos alunos.
 Podem apenas consultar as tabelas UCAluno e UC.
*/

-- Permissão dos alunos em UC
GRANT SELECT ON trocaturnos.UC TO 'stud1'@'localhost',
                                  'stud2'@'localhost',
                                  'stud3'@'localhost';

REVOKE INSERT, DELETE, UPDATE ON trocaturnos.UC FROM 'stud1'@'localhost',
                                                     'stud2'@'localhost',
                                                     'stud3'@'localhost';
-- Permissão dos alunos em UCAluno
GRANT SELECT ON trocaturnos.UCAluno TO 'stud1'@'localhost',
                                       'stud2'@'localhost',
                                       'stud3'@'localhost';

REVOKE INSERT, DELETE, UPDATE ON trocaturnos.UCAluno FROM 'stud1'@'localhost',
                                                          'stud2'@'localhost',
                                                          'stud3'@'localhost';

-- Permissão dos alunos nas restantes tabelas (revoke de tudo)
REVOKE SELECT, INSERT, DELETE, UPDATE ON trocaturnos.DocenteUC FROM 'stud1'@'localhost',
                                                                    'stud2'@'localhost',
                                                                    'stud3'@'localhost';

REVOKE INSERT, DELETE, UPDATE ON trocaturnos.Docente FROM 'stud1'@'localhost',
                                                          'stud2'@'localhost',
                                                          'stud3'@'localhost';

REVOKE INSERT, DELETE, UPDATE ON trocaturnos.Escola FROM 'stud1'@'localhost',
                                                         'stud2'@'localhost',
                                                         'stud3'@'localhost';

REVOKE INSERT, DELETE, UPDATE ON trocaturnos.Aluno FROM 'stud1'@'localhost',
                                                        'stud2'@'localhost',
                                                        'stud3'@'localhost';

REVOKE INSERT, DELETE, UPDATE ON trocaturnos.Curso FROM 'stud1'@'localhost',
                                                        'stud2'@'localhost',
                                                        'stud3'@'localhost';
```

Figura 32 – Permissões do user aluno

Foi também implementado um trigger que permite negar a criação de duas UCs com o mesmo nome. De seguida encontram-se o trigger e o teste no Workbench, que, como já existe uma UC com o nome ‘Álgebra Linear’ (ver anexo 3 – Povoamento da BD), indica que já existe uma UC com o mesmo nome.

```
/*
 Duas UC's diferentes não podem ter o mesmo nome
*/
DROP TRIGGER IF EXISTS nomeUC;
DELIMITER $$
CREATE TRIGGER nomeUC -- nome trigger
BEFORE INSERT ON UC -- tabela e ação (insert)
FOR EACH ROW
BEGIN

DECLARE message VARCHAR(75);

IF (new.nome IN
    (SELECT nome FROM uc WHERE
            uc.nome = new.nome))
THEN
    SET message = 'Já existe uma UC com o mesmo nome!';
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = message;
END IF;
END
$$
DELIMITER ;

-- TESTE DO TRIGGER
-- já existe a cadeira 'Álgebra Linear', vai dar erro
 INSERT INTO UC VALUES (841, 'Álgebra Linear', 3, 5);
```

Figura 33 - Trigger

Figura 34 - Erro recebido

5.9. Revisão do sistema implementado com o utilizador

O sistema implementado foi revisto com utilizador, que aceitou a Base de Dados implementada e, após algumas semanas de uso, demonstrou o seu contentamento.

O sistema respondeu a todas as questões expostas e está, atualmente, ao serviço do Departamento de Informática.

Conclusões e Trabalho Futuro

Este foi um projeto desafiante. No início deste trabalho pensou-se que a dificuldade era pouca e, por isso, pensou-se em fazer algo com uma dimensão razoável, replicando uma situação de necessidade real. À medida que se deram os primeiros passos na preparação do trabalho e depois do grupo estudar com mais atenção todos os aspetos da construção de uma BD, concluiu-se que se devia optar pela resolução de um problema simples, mas que permitisse uma aprendizagem segura.

Com isto em mente, fez-se a aproximação a um sistema que já se conhecia bem e que está atualmente em execução, a plataforma SWAP. Começando pelo modelo concetual, rapidamente se deu conta da necessidade de fixar requisitos e cumpri-los. Havia a constante tentação de mudar requisitos, mas isso só levava ao reinício constante do processo de desenvolvimento da BD. A partir deste momento, foi muito mais fácil começar a desenvolver a passos firmes o projeto.

O grupo apercebeu-se, aquando da passagem do modelo concetual para o modelo lógico, da importância dos relacionamentos e da forma como se constroem as chaves primárias. Surgiram várias dúvidas que, em confronto com os professores e colegas, se julgam esclarecidas e houve assim uma grande aprendizagem com este processo.

Enquanto grupo, desenvolveu-se a capacidade de comunicação e de divisão de tarefas, apesar de todos acompanharem o desenvolvimento dos outros. Durante este processo foi possível verificar, mais uma vez, a implicância que certas decisões tomadas no desenho inicial da solução teriam nos seus passos intermédios e, por isso, fizeram-se algumas iterações do ciclo de desenvolvimento, até que se verificasse a correção dos modelos apresentados.

De salientar que um dos maiores desafios que poderão surgir num contexto real será mesmo o diálogo com o utilizador. Na realidade, sabe-se que muitos utilizadores pedirão uma coisa e, na verdade, precisarão de outra. Caberá aos elementos da equipa o levantamento de requisitos correto e adaptados ao cliente.

Com toda a certeza, este foi um projeto trabalhoso e interessante, que permitiu explorar uma parte da informática que o grupo não tinha ainda experienciado.

Referências Bibliográficas

[1] Thomas Connolly, Carolyn Begg 2005. Database Systems: A Practical Approach to Design, Implementation, and Management, 4ª edição, Adisson Wesley.

Lista de Siglas e Acrónimos

BD Base de Dados

SBD Sistema de Base de Dados

SGBD Sistema de Gestão de Base de Dados

ER Entidade-Relacionamento

Anexos

I. Anexo 1 - Entrevista

Foi entrevistado pela nossa equipa o Diretor do Departamento de Informática.

D – Diretor

G – Grupo

G – Boa tarde. Hoje vamos tentar perceber quais as necessidades do sistema e as funcionalidades que querem que tenha. Fale-nos um pouco do que o fez contactar-nos.

D – Boa tarde. Neste momento o nosso departamento está a passar por uma restruturação no que toca ao processo de atribuição de turnos. Imaginem o que é ter 150 alunos todos os anos a entrar e gerir os turnos todos manualmente… No caso do primeiro ano até alocávamos manualmente os alunos que se inscreviam a certo horário através do preenchimento de uma folha, o que apesar de trabalhoso não dava grandes problemas com os turnos…

G – Mas então os turnos compõem um horário, correto?

D – Sim, exatamente. No caso do primeiro ano nós dávamos já um conjunto de turnos como um horário, mas nos outros anos os alunos tinham que se inscrever cadeira a cadeira, online. O mal é que esta inscrição online originava uma enorme confusão, uma vez que certos alunos se inscreviam num turno, e depois outros em que se queriam inscrever enchiam, originando incompatibilidades.

G – Muito bem, então e os alunos que falou, há algo que os possa identificador no vosso departamento ou na Universidade?

D – De facto, todos os alunos têm um número único, o número mecanográfico, que lhes é atribuído quando fazem a inscrição na universidade. Depois têm também o nome deles…

G – Muito bem! E os turnos de que fala, como é que funcionam? Como nos pode descrever um turno?

D – Um turno é uma aula dada por um certo docente, numa sala da Universidade, em certa hora. Temos várias UCs e algumas têm mais turnos que outras. Um turno pode ser teórico ou prático também.

G – E então os docentes, como os identifica? Mudam de ano para ano, podem dar várias cadeiras, que é importante neles que também seja para si?

D – Os docentes também têm um número mecanográfico, atribuído pela universidade. Eles podem lecionar várias cadeiras de vários anos e em vários turnos.

G – E então quer um sistema que proceda à organização de alunos por cadeiras e respetivos turnos?

D – Não propriamente! Uma equipa aqui do Departamento já fez um programa que permite aos alunos se inscreverem num turno e nós queremos guardar a informação de que alunos estão inscritos nas UCs e em que turnos ficaram, quais os docentes que lecionam cada UC, porque queremos melhorar o nosso curso. Guardar esta informação irá permitir controlar melhor o curso, dando possibilidade ao Diretor de Curso, por exemplo, de obter a lista de cadeiras a que um aluno se encontra inscrito, bem como respetivos turnos, para além da lista de docentes que leciona uma cadeira.

G – E que outras funcionalidades, como essas que falou agora, lhe parecem importantes?

D – Também seria importante poder consultar a lista de alunos relativa a uma cadeira que determinado docente leciona, como a lista simplesmente por cadeira. Seria interessante também obter a lista de docentes que leciona ao curso de Informática. A listagem completa dos alunos do curso também é importante, bem como eventual contagem dos tópicos anteriores. É também muito importante que possa saber qual as cadeiras com mais alunos inscritos e respetivos inscritos, porque temos tido um número de repetentes muito grandes e queremos corrigir essa situação!

G – Então só o Diretor de Curso vai ter acesso a estas informações, correto?

D – Sim, e ele também vai ser responsável por introduzir os Docentes existentes e UCs, bem como colocar os resultados que o programa aqui desenvolvido nos der, o SWAP, no que toca aos turnos em que cada aluno fica inscrito, em cada cadeira de cada ano. Os docentes também poderão consultar a informação que pretenderem, assim como fazer alterações nos turnos.

G – Alguma funcionalidade que pareça importante, que permita mudar alguns dados?

D – Sim, os alunos podem trocar de turno! Pode também haver reajustamentos ao plano de estudos, mudança de nomenclatura e valor de cadeiras…

G – Registamos tudo o que nos disse e agora vamos fazer uma análise mais aprofundada de tudo. Por último, qual o seu orçamento?

D – Temos um teto de 5000€ reservados para este projeto.

G – 5000€ não serão suficientes. Podemos estimar com alguma precisão que o custo associado ronda os 9000€.

D – Posso, no máximo, oferecer-vos 7000€, desde que criem uma base de dados segura e funcional, porque nós precisamos mesmo disto. Na verdade, até podemos expandir este sistema para outros cursos, caso esta experiência corra bem!

G – Podemos aceitar essa proposta. Obrigado e, em breve, encontramo-nos para apresentar a solução.

II. Anexo 2 – Script de criação da BD

```
-- -----------------------------------------------------
-- Table `TrocaTurnos`.`Escola`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `TrocaTurnos`.`Escola` (
  `Nome` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`Nome`))
ENGINE = InnoDB;
```

Figura 35 – Criação da tabela Escola

```
-- -----------------------------------------------------
-- Table `TrocaTurnos`.`Docente`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `TrocaTurnos`.`Docente` (
  `Numero` INT NOT NULL,
  `Nome` VARCHAR(45) NULL,
  `Escola_Nome` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`Numero`),
  INDEX `fk_Docente_Escola1_idx` (`Escola_Nome` ASC),
  CONSTRAINT `fk_Docente_Escola1`
    FOREIGN KEY (`Escola_Nome`)
    REFERENCES `TrocaTurnos`.`Escola` (`Nome`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 36 – Criação da tabela Docente

```
-- -----------------------------------------------------
-- Table `TrocaTurnos`.`UC`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `TrocaTurnos`.`UC` (
  `Codigo` INT NOT NULL,
  `Nome` VARCHAR(45) NULL,
  `Ano` INT NOT NULL,
  `ECTS` INT NOT NULL,
  PRIMARY KEY (`Codigo`))
ENGINE = InnoDB;
```

Figura 37 – Criação da tabela UC

```
-- -----------------------------------------------------
-- Table `TrocaTurnos`.`Curso`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `TrocaTurnos`.`Curso` (
  `Nome` VARCHAR(10) NOT NULL,
  PRIMARY KEY (`Nome`))
ENGINE = InnoDB;
```

Figura 38 – Criação da tabela Curso

```
-- -----------------------------------------------------
-- Table `TrocaTurnos`.`Aluno`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `TrocaTurnos`.`Aluno` (
  `Numero` INT NOT NULL,
  `Nome` VARCHAR(45) NULL,
  `Curso_Nome` VARCHAR(10) NOT NULL,
  PRIMARY KEY (`Numero`),
  INDEX `fk_Aluno_Curso1_idx` (`Curso_Nome` ASC),
  CONSTRAINT `fk_Aluno_Curso1`
    FOREIGN KEY (`Curso_Nome`)
    REFERENCES `TrocaTurnos`.`Curso` (`Nome`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 39 – Criação da tabela Aluno

```
-- -----------------------------------------------------
-- Table `TrocaTurnos`.`DocenteUC`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `TrocaTurnos`.`DocenteUC` (
  `Docente_Numero` INT NOT NULL,
  `UC_Codigo` INT NOT NULL,
  INDEX `fk_DocenteUC_Docente_idx` (`Docente_Numero` ASC),
  INDEX `fk_DocenteUC_UC1_idx` (`UC_Codigo` ASC),
  PRIMARY KEY (`Docente_Numero`, `UC_Codigo`),
  CONSTRAINT `fk_DocenteUC_Docente`
    FOREIGN KEY (`Docente_Numero`)
    REFERENCES `TrocaTurnos`.`Docente` (`Numero`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_DocenteUC_UC1`
    FOREIGN KEY (`UC_Codigo`)
    REFERENCES `TrocaTurnos`.`UC` (`Codigo`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 40 – Criação da tabela Docente UC

```
-- -----------------------------------------------------
-- Table `TrocaTurnos`.`UCAluno`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `TrocaTurnos`.`UCAluno` (
  `AnoLetivo` INT NOT NULL,
  `Turno` VARCHAR(3) NOT NULL,
  `UC_Codigo` INT NOT NULL,
  `Aluno_Numero` INT NOT NULL,
  PRIMARY KEY (`AnoLetivo`, `Turno`, `UC_Codigo`, `Aluno_Numero`),
  INDEX `fk_UCAluno_UC1_idx` (`UC_Codigo` ASC),
  INDEX `fk_UCAluno_Aluno1_idx` (`Aluno_Numero` ASC),
  CONSTRAINT `fk_UCAluno_UC1`
    FOREIGN KEY (`UC_Codigo`)
    REFERENCES `TrocaTurnos`.`UC` (`Codigo`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_UCAluno_Aluno1`
    FOREIGN KEY (`Aluno_Numero`)
    REFERENCES `TrocaTurnos`.`Aluno` (`Numero`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 41 – Criação da tabela UCAluno

III. Anexo 3 – Povoamento da BD

```
INSERT INTO Escola
    (Nome)
    VALUES
        ('Escola de Ciências'),
        ('Escola de Engenharia')

    ;
```

Figura 42 - Povoamento da tabela "Escola"

```
INSERT INTO Curso
    (Nome)
    VALUES
        ('MIEI')

    ;
```

Figura 43 - Povoamento da tabela "Curso"

```sql
INSERT INTO Aluno
    (Numero, Nome, Curso_Nome)
    VALUES
        (78985, 'Maria Inês Machado', 'MIEI'),
        (77730, 'Vítor Campos', 'MIEI'),
        (79116, 'Sérgio Godinho', 'MIEI'),
        (77870, 'Diana Oliveira', 'MIEI'),
        (76111, 'Marcos Matos', 'MIEI'),
        (77321, 'Leonor Castro', 'MIEI'),
        (78404, 'Bruno Silva', 'MIEI'),
        (68503, 'Duarte Machado', 'MIEI'),
        (69545, 'Gonçalo Oliveira', 'MIEI'),
        (73435, 'Nuno Barreira', 'MIEI'),
        (74584, 'Rafael Silva', 'MIEI'),
        (81230, 'João Afonso', 'MIEI'),
        (78790, 'Rafaela Santos', 'MIEI'),
        (80828, 'Eduardo Barcelona', 'MIEI'),
        (69308, 'Alexandre Marques', 'MIEI'),
        (70920, 'André Maia', 'MIEI'),
        (70815, 'Paulo Silva', 'MIEI'),
        (76275, 'Sara Couto', 'MIEI'),
        (78633, 'Artur Oliveira', 'MIEI'),
        (73314, 'João Cruz', 'MIEI'),
        (81122, 'Catarina Araújo', 'MIEI'),
        (73941, 'Joaquim Rebelo', 'MIEI'),
        (72974, 'João Cunha', 'MIEI'),
        (80979, 'Ricardo Martins', 'MIEI'),
        (76935, 'Válter Carvalho', 'MIEI'),
        (75820, 'Miguel Cardoso', 'MIEI'),
        (70608, 'Beatriz Sousa', 'MIEI'),
        (77209, 'Juliana Pereira', 'MIEI'),
        (80885, 'Francisco Lira', 'MIEI'),
        (67848, 'João Soares', 'MIEI'),
        (72111, 'Alexandre Pedrosa', 'MIEI'),
        (78415, 'Sara Sampaio', 'MIEI'),
        (70629, 'Anderson Vítor', 'MIEI'),
        (66906, 'Hugo Gramacho', 'MIEI'),
        (78789, 'Francisco Xavier', 'MIEI'),
        (69634, 'Daniela Ribeiro', 'MIEI'),
        (71544, 'Daniel Fernandes', 'MIEI'),
        (80460, 'Daniel Macedo', 'MIEI'),
        (81682, 'Flávio Martins', 'MIEI'),
        (68892, 'Gil Sampaio', 'MIEI'),
        (51146, 'André Azevedo', 'MIEI')

    ;
```

Figura 44 - Povoamento da tabela "Aluno"

```sql
INSERT INTO UC
    (Codigo, Nome, Ano, ECTS)
    VALUES
        (365, 'Álgebra Linear', 1, 5),
        (423, 'Cálculo', 1, 5),
        (122, 'Algoritmos e Complexidade', 2, 5),
        (050, 'Programação Orientada aos Objetos', 2, 5),
        (403, 'Sistemas Operativos', 2, 5),
        (552, 'Redes de Computadores', 3, 5),
        (327, 'Cálculo de Programas', 2, 5),
        (667, 'Bases de Dados', 3, 5),
        (103, 'Laboratórios de Informática I', 1, 5),
        (210, 'Laboratórios de Informática II', 1, 5),
        (093, 'Lógica', 1, 5),
        (323, 'Programacão Funcional', 1, 5),
        (094, 'Desenvolvimento de Sistemas de Software', 3, 5),
        (039, 'Arquitetura de Computadores', 2, 5),
        (455, 'Programacão Imperativa', 1, 5),
        (312, 'Estatística Aplicada', 2, 5),
        (612, 'Sistemas de Computação', 1, 5),
        (112, 'Análise', 1, 5),
        (605, 'Engenharia Económica', 2, 5),
        (498, 'Introdução aos Sistemas Dinâmicos', 2, 5),
        (067, 'Electromagnetismo', 2, 5),
        (604, 'Computação Gráfica', 3, 5),
        (389, 'Processamento de Linguagens', 3, 5),
        (400, 'Elementos de Engenharia de Sistemas', 1, 5),
        (399, 'Métodos Numéricos e Otimização não Linear', 3, 5),
        (010, 'Laboratórios de Informática IV', 3, 5),
        (702, 'Comunicações por Computador', 3, 5)
    ;
```

Figura 45 - Povoamento da tabela "UC"

```sql
INSERT INTO Docente
    (Numero, Nome, Escola_Nome)
    VALUES

        (18054, 'Amélia Campos' ,'Escola de Ciências'),
        (59341, 'Serafim Andrade', 'Escola de Engenharia'),
        (48592, 'Carla Santos', 'Escola de Ciências'),
        (41495, 'António Macedo', 'Escola de Engenharia'),
        (59479, 'José Frade', 'Escola de Engenharia'),
        (64850, 'Dulce Oliveira', 'Escola de Engenharia'),
        (79194, 'Valério Quintas', 'Escola de Ciências'),
        (24759, 'Maria Minho', 'Escola de Ciências'),
        (74123, 'Carlos Castro', 'Escola de Engenharia'),
        (13212, 'Ivone Costa', 'Escola de Engenharia'),
        (43923, 'Cornélio Belo', 'Escola de Engenharia'),
        (26548, 'Dinis Gabriel', 'Escola de Engenharia')
    ;
```

Figura 46 - Povoamento da tabela "Docente"

```sql
INSERT INTO DocenteUC
    (Docente_Numero, UC_Codigo)
    VALUES
        (18054, 365),
        (18054, 093),
        (59341, 122),
        (59341, 604),
        (59341, 702),
        (48592, 423),
        (48592, 067),
        (41495, 403),
        (41495, 455),
        (41495, 327),
        (59479, 552),
        (64850, 323),
        (64850, 604),
        (64850, 455),
        (79194, 112),
        (79194, 093),
        (24759, 312),
        (24759, 498),
        (24759, 399),
        (74123, 389),
        (74123, 667),
        (13212, 050),
        (13212, 400),
        (43923, 103),
        (43923, 612),
        (43923, 605),
        (43923, 010),
        (26548, 210),
        (26548, 094),
        (26548, 039)

    ;
```

Figura 47 - Povoamento da tabela "DocenteUC"

```sql
INSERT INTO UCAluno
    (AnoLetivo, Turno, UC_Codigo, Aluno_Numero)
    VALUES
        (1617, 'PL1', 365, 78985),
        (1617, 'PL1', 423, 78985),
        (1617, 'PL1', 122, 78985),
        (1617, 'PL1', 050, 77730),
        (1617, 'PL1', 403, 77730),
        (1617, 'PL1', 552, 77730),
        (1617, 'PL1', 423, 77730),
        (1617, 'PL1', 050, 79116),
        (1617, 'PL1', 403, 79116),
        (1617, 'PL1', 327, 79116),
        (1617, 'PL1', 365, 79116),
        (1617, 'PL1', 122, 77870),
        (1617, 'PL1', 667, 77870),
        (1617, 'PL1', 103, 77870),
        (1617, 'PL1', 667, 76111),
        (1617, 'PL1', 103, 76111),
        (1617, 'PL1', 093, 76111),
        (1617, 'PL1', 323, 76111),
        (1617, 'PL1', 667, 77321),
        (1617, 'PL1', 094, 77321),
        (1617, 'PL1', 039, 77321),
        (1617, 'PL2', 667, 78404),
        (1617, 'PL1', 094, 78404),
        (1617, 'PL1', 039, 78404),
        (1617, 'PL2', 050, 68503),
        (1617, 'PL2', 103, 68503),
        (1617, 'PL1', 093, 68503),
        (1617, 'PL1', 323, 68503),
        (1617, 'PL2', 039, 69545),
        (1617, 'PL1', 455, 69545),
        (1617, 'PL1', 312, 69545),
        (1617, 'PL2', 039, 73435),
        (1617, 'PL2', 039, 73435),
        (1617, 'PL1', 455, 73435),
        (1617, 'PL1', 312, 73435),
        (1617, 'PL2', 667, 74584),
        (1617, 'PL2', 312, 81230),
        (1617, 'PL1', 612, 81230),
        (1617, 'PL1', 112, 81230),
        (1617, 'PL2', 050, 81230),
        (1617, 'PL1', 112, 78790),
        (1617, 'PL1', 612, 78790),
        (1617, 'PL2', 612, 80828),
        (1617, 'PL3', 667, 80828),
        (1617, 'PL2', 112, 80828),
        (1617, 'PL1', 605, 80828),
        (1617, 'PL1', 605, 69308),
        (1617, 'PL1', 498, 69308),
        (1617, 'PL1', 552, 69308),
        (1617, 'PL1', 498, 70920),
        (1617, 'PL2', 605, 70920),
        (1617, 'PL2', 498, 70815),
        (1617, 'PL1', 067, 70815),
        (1617, 'PL1', 604, 70815),
        (1617, 'PL1', 389, 70815),
        (1617, 'PL1', 604, 76275),
        (1617, 'PL1', 389, 76275),
        (1617, 'PL2', 389, 78633),
        (1617, 'PL1', 400, 78633),
        (1617, 'PL1', 399, 78633),
        (1617, 'PL1', 399, 73314),
        (1617, 'PL1', 400, 73314),
        (1617, 'PL2', 093, 73314),
        (1617, 'PL1', 010, 81122),
        (1617, 'PL1', 702, 81122),
        (1617, 'PL2', 400, 81122),
        (1617, 'PL1', 010, 73941),
        (1617, 'PL1', 010, 73941),
        (1617, 'PL1', 702, 73941),
        (1617, 'PL2', 605, 73941),
        (1617, 'PL2', 010, 72974),
        (1617, 'PL2', 702, 72974),
        (1617, 'PL2', 702, 80979),
        (1617, 'PL3', 667, 80979),
        (1617, 'PL2', 103, 80979),
        (1617, 'PL2', 093, 80979),
        (1617, 'PL3', 702, 76935),
        (1617, 'PL2', 039, 76935),
        (1617, 'PL3', 050, 76935),
        (1617, 'PL1', 210, 75820),
        (1617, 'PL2', 094, 75820),
        (1617, 'PL2', 612, 75820),
        (1617, 'PL2', 604, 70608),
        (1617, 'PL2', 112, 70608),
        (1617, 'PL2', 498, 77209),
        (1617, 'PL1', 327, 77209),
        (1617, 'PL2', 423, 77209),
        (1617, 'PL2', 365, 80885),
        (1617, 'PL2', 389, 80885),
        (1617, 'PL2', 400, 80885),
        (1617, 'PL2', 323, 80885),
        (1617, 'PL3', 667, 67848),
        (1617, 'PL3', 702, 67848),
        (1617, 'PL2', 010, 67848),
        (1617, 'PL2', 399, 67848),
        (1617, 'PL3', 010, 72111),
        (1617, 'PL2', 399, 72111),
        (1617, 'PL3', 112, 72111),
        (1617, 'PL2', 365, 78415),
        (1617, 'PL2', 423, 78415),
        (1617, 'PL1', 122, 78415),
        (1617, 'PL3', 050, 78415),
        (1617, 'PL3', 050, 78415),
        (1617, 'PL1', 403, 70629),
        (1617, 'PL1', 552, 70629),
        (1617, 'PL1', 327, 70629),
        (1617, 'PL4', 667, 66906),
        (1617, 'PL2', 103, 66906),
        (1617, 'PL1', 210, 66906),
        (1617, 'PL2', 093, 78789),
        (1617, 'PL2', 323, 78789),
        (1617, 'PL2', 094, 78789),
        (1617, 'PL1', 039, 78789),
        (1617, 'PL2', 455, 51146),
        (1617, 'PL2', 312, 51146),
        (1617, 'PL2', 612, 69634),
        (1617, 'PL3', 112, 69634),
        (1617, 'PL1', 605, 69634),
        (1617, 'PL2', 498, 71544),
        (1617, 'PL1', 067, 71544),
        (1617, 'PL2', 604, 71544),
        (1617, 'PL2', 389, 80460),
        (1617, 'PL2', 400, 80460),
        (1617, 'PL3', 399, 80460),
        (1617, 'PL3', 010, 81682),
        (1617, 'PL4', 702, 81682),
        (1617, 'PL4', 667, 81682),
        (1617, 'PL3', 010, 68892),
        (1617, 'PL4', 702, 68892),
        (1617, 'PL4', 667, 68892)

    ;
```

Figura 48 - Povoamento da tabela "UCAluno"