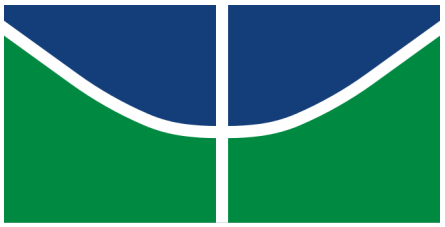

UnB

Universidade de Brasília

CIC0097 - Bancos de Dados

Projeto Final
UnBoard

Aluno:
Arthur H S Carvalho

Matrícula:
211026673

10 de julho de 2023

Conteúdo

1 Introdução

1.1 O que é o UnBoard?

O UnBoard é uma plataforma online projetada para revolucionar a forma como os estudantes avaliam e fornecem feedback sobre os professores. Com foco em aprimorar a experiência de aprendizado, o UnBoard funciona como um centro dinâmico onde os estudantes podem compartilhar suas percepções valiosas e experiências com a comunidade acadêmica.

1.2 Repositório do projeto

O repositório público do UnBoard pode ser encontrado em https://github.com/akaTsunemori/unboard. O arquivo *database_setup/mysql-setup.txt* contém os comandos básicos para criação de usuário e database. O script *SQL* de geração do banco de dados localiza-se em *database_setup/unboard.sql*, bem como os scripts em *Python* que populam o banco de dados.

> *Deve ser fornecido código para inserção de pelo menos 3 linhas em cada uma das tabelas criadas*

Dado o requisito acima, o arquivo *database_setup/populate.sql* cumpre essa tarefa. Note que, para popular o banco de dados com o conteúdo das tabelas *csv*, as instruções estão localizadas no *README.md*.

Finalmente, é importante ressaltar que o arquivo *README.md* contém instruções gerais sobre o projeto, refira-se a ele para executar os scripts acima e popular o banco de dados, e para fazer o setup das dependências e executar o programa de interface com o usuário.

1.3 Vídeo de apresentação

Além das instruções para o setup do projeto um vídeo apresentando e descrevendo o projeto foi elaborado e pode ser assistido através do link: https://youtu.be/LbsL7P1IktQ.

2 Tecnologias

2.1 Sistema de Gerenciamento de Bancos de Dados

Para o Sistema de Gerenciamento de Bancos de Dados, usou-se MySQL. As principais razões para essa decisão são:

- Popularidade e comunidade ativa: O MySQL é um dos SGBDs mais populares do mundo.
- Código aberto: O MySQL é um software de código aberto, o que significa que você pode baixá-lo, usá-lo e modificá-lo gratuitamente.
- Facilidade de uso: O MySQL é conhecido por sua facilidade de instalação, configuração e uso.
- Desempenho: O MySQL é otimizado para oferecer um bom desempenho em diversas situações. Ele é capaz de lidar com grandes quantidades de dados e suporta operações rápidas de leitura e gravação. Além disso, o MySQL possui recursos avançados de indexação e otimização de consultas para melhorar ainda mais o desempenho do sistema.
- Escalabilidade: O MySQL é altamente escalável, o que significa que ele pode lidar com um aumento no número de usuários, tráfego e tamanho do banco de dados sem comprometer o desempenho.
- Compatibilidade: O MySQL é compatível com várias plataformas e sistemas operacionais, incluindo Windows, Linux e macOS.
- Segurança: O MySQL possui recursos robustos de segurança, incluindo criptografia de dados em trânsito e em repouso, controle de acesso granular e auditoria de eventos. Ele também tem uma comunidade ativa que monitora e corrige regularmente quaisquer vulnerabilidades de segurança.

2.2 População do banco de dados

Para popular o banco de dados, foram escritos scripts em Python, que coletam e trabalham com os dados usando o módulo pandas.

2.3 Interface de usuário

A interface de usuário foi construída usando *HTML* e *CSS*, tecnologias que são fáceis de se trabalhar e permitem a construção de interfaces modernas e funcionais, com ótimo desempenho.

2.4 Programação

Tanto o *front-end* quanto o *back-end* foram programados em *Python*, usando o módulo Flask para a funcionalidade das páginas do aplicativo e o módulo mysql-connector-python para a comunicação entre o *Python* e o *MySQL*.

2.5 Connector

Como já mencionado, usou-se o mysql-connector-python para a comunicação entre o *Python* e o *MySQL*. Esse módulo permite definirmos uma *connection* e um *cursor*, onde o cursor pode ser usado para realizar *queries* explícitas para o banco de dados, e a conexão é usada para realizar o *commit* de alterações no SGBD. A principal decisão para usar esse módulo e não o *SQLAlchemy* foi o fato de que o *Alchemy* não permite *queries* explícitas, usá-lo seria contrário aos requisitos do projeto.

2.6 JavaScript

Foi usado JavaScript em um grau minúsculo no projeto, apenas para pequenos scripts que ficariam convenientes de se escrever no próprio template *HTML*.

3 Modelagem do banco de dados

A criação do Modelo Entidade-Relacionamento (MER) e do Modelo Relacional é uma prática fundamental antes de montar um banco de dados, pois esses modelos fornecem uma representação visual e conceitual dos dados e suas relações.

3.1 Modelo Entidade-Relacionamento

As principais vantagens do Modelo Entidade-Relacionamento são:

- Representação visual dos dados: O MER fornece uma representação visual dos dados e suas relações.
- Compreensão dos requisitos do sistema: O processo de criação do MER ajuda a analisar e compreender os requisitos do sistema. Ele permite identificar as entidades principais envolvidas, os atributos que descrevem essas entidades e como elas se relacionam entre si.
- Identificação de entidades e atributos: O MER ajuda a identificar as entidades-chave (objetos do mundo real) e seus atributos (características dessas entidades).
- Estabelecimento de relacionamentos: O MER permite definir os relacionamentos entre as entidades. Esses relacionamentos descrevem como as entidades se conectam e interagem entre si.
- Comunicação entre as partes interessadas: O MER é uma ferramenta de comunicação eficaz entre os analistas, projetistas, desenvolvedores e demais partes interessadas envolvidas no projeto do banco de dados.

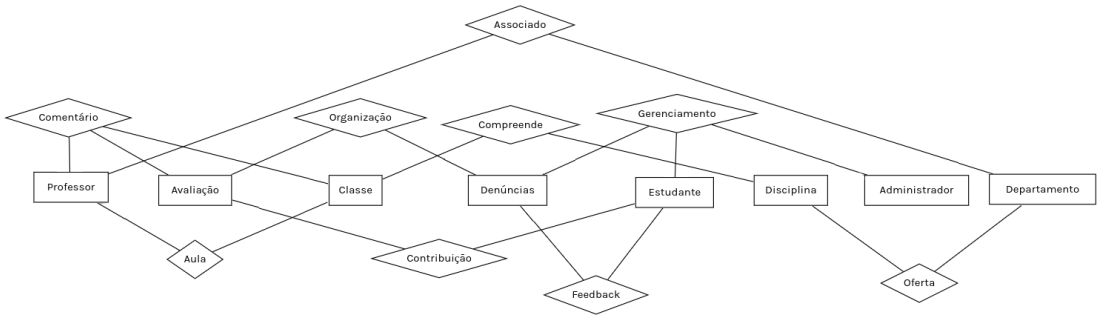

Figura 1: Modelo Entidade-Relacionamento.

A Figura 1 também pode ser encontrada no diretório *database_setup/Relationship-Entity-Model*, para melhor visibilidade da mesma. Os atributos não foram anexados

ao diagrama para que haja uma melhor visibilidade das entidades e seus relacionamentos. A descrição entidade-atributo para cada entidade é:

- Estudante: email, nome, curso, matrícula;
- Departamento: código, nome;
- Disciplina: código, nome;
- Classe: código, semestre, horário;
- Avaliação: avaliação, nota;
- Denúncia: avaliação;

3.2 Modelo Relacional

A criação do Modelo Relacional é uma prática importante no projeto de banco de dados. Algumas razões para se fazer o modelo relacional:

- Organização dos dados: O Modelo Relacional permite organizar os dados em tabelas. Essa organização facilita o armazenamento, a manipulação e a recuperação dos dados de forma eficiente.
- Integridade dos dados: O Modelo Relacional define restrições de integridade que garantem a consistência dos dados. Por exemplo, as chaves primárias garantem que cada registro seja único, enquanto as chaves estrangeiras estabelecem relacionamentos entre tabelas, mantendo a integridade referencial.
- Flexibilidade e extensibilidade: O Modelo Relacional oferece flexibilidade e extensibilidade para lidar com mudanças nos requisitos do sistema.
- Normalização dos dados: O Modelo Relacional incentiva a aplicação de técnicas de normalização, que visam eliminar redundâncias e anomalias nos dados.

O Modelo Relacional foi construído a partir do Modelo Entidade-Relacionamento, mas adaptações consideradas necessárias foram acrescentadas. Além disso, atributos foram acrescentados de modo que o modelo descreva também os atributos relacionados ao sistema, não somente os atributos físicos de cada entidade.

- Estudantes(**email**, nome, senha, admin_status, curso, matricula, foto_perfil);
- Professores(**numero_id**, nome);
- Departamentos(**codigo**, nome);
- Disciplinas(**codigo**, nome, depto_codigo_fk);
- Turmas(**id**, codigo, disciplina_cod_fk, semestre, prof_id_fk, horario);
- AvaliacoesClasses(**email_estudante_fk**, **id_classe_fk**, avaliacao, nota);
- AvaliacoesProfessores(**email_estudante_fk**, **id_professor_fk**, avaliacao, nota);
- DenunciasAvaliacoesClasse(**email_estudante_fk**, **id_classe_fk**);
- DenunciasAvaliacoesProfessor(**email_estudante_fk**, **id_professor_fk**);

Onde o sublinhado representa uma chave estrangeira e o negrito representa a chave primária.

4 Avaliação de formas normais

A avaliação das formas normais em uma tabela envolve analisar a estrutura da tabela e verificar se ela atende aos requisitos de cada forma normal. As principais formas normais são a Primeira Forma Normal (1NF), a Segunda Forma Normal (2NF) e a Terceira Forma Normal (3NF). Analisaremos se as tabelas *Estudantes*, *Departamentos*, *Turmas* atendem às 3 formas normais mencionadas antes. Para que fiquem claros os critérios, descrevemos as 3 formas normais como:

1. Primeira Forma Normal (1NF):
    - Verificar se não há repetições de grupos de valores em uma única célula;
    - Cada coluna deve conter apenas um valor, evitando listas ou conjuntos de valores em uma única coluna;
    - Garantir que cada registro seja único e tenha uma chave primária.
2. Segunda Forma Normal (2NF):
    - Verificar se a tabela está na 1NF;
    - Identificar as dependências funcionais entre as colunas. Uma dependência funcional ocorre quando o valor de uma coluna depende do valor de outra coluna;
    - Se houver dependências parciais, separar as colunas em diferentes tabelas para eliminar redundâncias.
3. Terceira Forma Normal (3NF):
    - Verificar se a tabela está na 2NF;
    - Identificar as dependências transitivas, onde uma coluna depende de outra através de uma terceira coluna;
    - Se houver dependências transitivas, normalizar a tabela separando as colunas dependentes em outra tabela.

Tendo os critérios acima em perspectiva, avaliaremos as formas normais de cada tabela mencionada:

1. Tabela *Estudantes*:
    - Primeira Forma Normal (1NF): A tabela possui uma chave primária (email) e todas as colunas contêm apenas valores atômicos. Portanto, está na 1NF.

- Segunda Forma Normal (2NF): Não existem dependências parciais na tabela porque cada coluna depende totalmente da chave primária (email). Cada atributo (nome, senha, admin_status, foto_perfil) é diretamente relacionado ao email de um estudante e não é possível identificar uma dependência parcial;
- Terceira Forma Normal (3NF): Não há dependências transitivas na tabela porque não existem colunas que dependam indiretamente de outra coluna através de uma terceira coluna. Todos os atributos estão diretamente relacionados à chave primária (email).

2. Tabela *Departamentos*:
   - Primeira Forma Normal (1NF): A tabela possui uma chave primária (codigo) e todas as colunas contêm apenas valores atômicos. Portanto, está na 1NF.
   - Segunda Forma Normal (2NF): Não existem dependências parciais na tabela porque cada coluna depende totalmente da chave primária (codigo). Cada atributo (nome) é diretamente relacionado ao código de um departamento e não há dependências parciais identificáveis.
   - Terceira Forma Normal (3NF): Não há dependências transitivas na tabela porque só existe uma coluna (name) relacionada à chave primária (id) e não há colunas dependentes dessa coluna.
3. Tabela *Turmas*:
   - Primeira Forma Normal (1NF): A tabela possui uma chave primária (id) e todas as colunas contêm apenas valores atômicos. Portanto, está na 1NF.
   - Segunda Forma Normal (2NF): Não existem dependências parciais na tabela porque cada atributo depende completamente da chave primária (id). Cada coluna (codigo, disciciplina_cod_id, semestre, professor_id_fk, horario) é diretamente relacionada ao id de uma classe, não havendo dependências parciais identificáveis.
   - Terceira Forma Normal (3NF): Não há dependências transitivas na tabela porque não há colunas que dependam indiretamente de outra coluna através de uma terceira coluna. Todos os atributos estão diretamente relacionados à chave primária (id).

5 Considerações adicionais

5.1 Tabela Disciplinas

```
CREATE TABLE Disciplines (
  id VARCHAR(20) NOT NULL,
  name VARCHAR(400) NOT NULL,
  dept_id INT NOT NULL,
  PRIMARY KEY (id, name),
  FOREIGN KEY (dept_id) REFERENCES Departments(id) ON DELETE CASCADE
);
```

Na implementação do banco de dados, a tabela *Disciplinas* acaba ferindo formas normais pelo fato de sua chave primária ser composta de *codigo* e *nome*. A ideia inicial seria ter o *codigo* como a única chave primária, pois este deveria ser único para cada disciplina. Acontece que, pelos dados fornecidos para alimentar o banco de dados, existem casos onde há duas disciplinas com o mesmo código. Como esse problema foi detectado em uma fase mais tardia do desenvolvimento, a solução simples foi simplesmente usar o nome para também diferenciar disciplinas, criando a chave primária com 2 elementos.

5.2 Uso do idioma inglês

Por decisão do aluno, todo o projeto final foi elaborado no idioma inglês, isso inclui todo o conteúdo da interface de usuário e todas as tabelas. Para que fique claro, a tabela *Turmas* recebeu o nome *Classes*. Os nomes das outras tabelas se assemelham aos nomes originais.

5.3 Construção do banco de dados

Para construir o banco de dados em *MySQL*, não foi usado nenhuma interface de usuário ou software que facilite a criação de tabelas. Todas as tabelas foram construídas manualmente e toda a lógica por trás delas foi puramente da criatividade do aluno. O principal intuito dessa postura foi o aprendizado, outro motivador foi o estímulo a não criar vícios a nenhuma ferramenta e compreender como o SQL, na prática, funciona.

6 Conclusões

O projeto final, focado em bancos de dados, foi concluído com sucesso, passando por todas as etapas essenciais do processo. Iniciamos com a modelagem do banco de dados, utilizando o Modelo Entidade-Relacionamento (MER) para representar as entidades, atributos e relacionamentos de forma clara e concisa. Essa etapa permitiu compreender os requisitos do sistema e estabelecer a estrutura lógica inicial.

Em seguida, realizamos a transformação do MER para o Modelo Relacional, organizando os dados em tabelas e definindo chaves primárias, chaves estrangeiras e restrições de integridade. Essa etapa proporcionou a base sólida para a implementação do banco de dados, garantindo a consistência e a eficiência das operações.

Na fase de implementação, criamos as tabelas, definimos as relações e importamos os dados. Além disso, desenvolvemos uma interface com o usuário, possibilitando interações intuitivas e facilitando o acesso e a manipulação dos dados armazenados no banco.

Adicionalmente, avaliamos as formas normais das tabelas do banco de dados, verificando se atendiam aos requisitos de cada forma normal. Analisamos a 1NF, 2NF e 3NF, garantindo a eliminação de redundâncias e anomalias nos dados, bem como a integridade e a eficiência do banco de dados.

Em conclusão, o projeto final abordou de maneira abrangente o desenvolvimento de um banco de dados, desde a modelagem até a implementação, incluindo a interface com o usuário e a avaliação das formas normais. Essa abordagem proporcionou uma base sólida para o gerenciamento eficiente dos dados, garantindo a qualidade, a integridade e a segurança das informações armazenadas.