BCC - IME USP

MAC0439 - Laboratório de Banco de Dados

Projeto Conceitual de Banco de Dados - Fase 2

Profª: Kelly Rosa Braghetto

1. Definição dos integrantes do grupo e escolha do sistema a ser implementado

Integrantes:

Eduardo Bier - 8536148
Maurício Luiz Abreu Cardoso - 6796479
Ricardo de Oliveira - 3683165
Taís Pinheiro - 7580421

Tema:

Sistema Web para simulados - BrainPin

Descrição:

Nesse sistema, os alunos que cursam o ensino fundamental e médio podem estudar e fazer exercícios dispostos no sistema com o intuito de serem aprovados em exames de vestibular ou de vestibulinho. Esses exercícios serão categorizados e gerenciados por usuários classificados como Administradores ou Professores. Os exercícios podem ser apoiados em vídeos explicativos, em comentários ou em mensagens deixadas por outros usuários.

O sistema basicamente detecta o desempenho e evolução do usuário (estudante) com base no número de questões acertadas, na dificuldade e no tema dessas questões. A cada nível atingido pelo estudante, um Avatar diferente pode ser desbloqueado. Os usuários do sistema podem interagir entre si.

Banco de Dados Relacional: Armazena os dados de usuários, de questões, temas, disciplinas e a relação entre eles.

Banco de Dados não relacional: Armazena a estrutura de comentários, posts de professores e interação dos usuários.

2. Definição dos requisitos funcionais e de dados do sistema

Usuários

Professor

- Adiciona, exclui e altera questões, videos, tópicos e subtópicos das disciplinas que leciona.
- Responde a dúvidas de alunos que forem postadas nos exercícios.
- Publica avisos no _feed_.
- Cada professor está ligado a uma ou mais disciplinas.
- Pode escolher qualquer avatar

Aluno

- Resolve exercícios.
- Assiste vídeos.
- Tem um nível relacionado a cada disciplina, tópico e subtópico
- Tem um avatar personalizável:
    - Avatares são disponibilizados conforme o aluno atinge um determinado nível em um conjunto de tópicos/disciplinas. Por exemplo, para abrir o avatar de Arquimedes, poderia ser necessário atingir nível 5 em Geometria Plana e nível 3 em Ótica.
- Dá nota a questões e material didático.
- Comenta questões, no intuito de discutir métodos de resolução, tirar dúvidas, etc.

Administrador

- Adiciona, exclui e altera questões, videos, tópicos e subtópicos de qualquer disciplina.
- Cadastra professores.
- Pode escolher qualquer avatar

Disciplina, Tópico e Subtópico

- Cada disciplina é classificada em tópicos e cada tópico em subtópicos. Por exemplo, podemos ter uma disciplina 'Matemática', subdividida em um tópico 'Geometria' e subdividido em um tópico 'Triângulos Isósceles'.
- O nível de um aluno em uma disciplina depende do seu nível nos tópicos e subtópicos dela.
- Conforme o nível do aluno cresce em uma disciplina, tópico ou subtópico, novas questões, tópicos e subtópicos são abertos. Assim, um aluno pode ter que atingir nível 3 no tópico 'Triângulos Retângulos' para abrir o tópico 'Triângulos Quaisquer' e suas questões.
- Além de questões, vídeos, livros e outros materiais didáticos também são disponibilizados para os alunos.

Questão

- Tem uma ou mais resoluções associadas a ela.
- Pode pertencer a mais de uma disciplina, tópico e subtópico.
- Tem um nível de dificuldade, que será usado para determinar se um aluno deve ou não ter acesso a ela de acordo com o seu nível em um determinado tópico.
- Ao acertar uma questão, o aluno ganha pontos para cada tópico relacionado à questão. A quantidade de pontos ganha pode ser maior para um determinado tópico.
- Cada questão tem uma seção de comentários de alunos, onde alunos poderão tirar dúvidas, discutir métodos de resolução e fazer perguntas. Tanto alunos quanto professores podem postar neste espaço.
- A resolução de uma questão e seus comentários não devem estar visíveis por padrão (só aparecem depois que o aluno já respondeu a questão ou se clicar para aparecer).
- As alternativas de uma questão podem mudar de ordem e de valor. Assim, as questões também podem ficar mais difíceis colocando alternativas incorretas com valores mais plausíveis.

Nível

- Para avançar um nível o aluno deve conseguir uma certa quantidade de pontos (através de questões).
- A quantidade de pontos para avançar um nível não é constante. Ou seja, avançar do nível 1 para o nível 2 pode ser mais fácil do que avançar do nível 9 para o nível 10.
- Como mencionado anteriormente, níveis estão associados a disciplinas, tópicos, subtópicos e alunos.

Outras informações

- Qualquer um pode se cadastrar como aluno no sistema. Um professor que se cadastrar como aluno teria duas contas diferentes, uma de professor e outra de aluno.
- Um usuário deve estar logado para ter acesso às disciplinas.
- Ao entrar no sistema, um aluno tem acesso a um conjunto de questões e vídeos separados por disciplina.

Página Inicial

- Tem uma lista das disciplinas e o progresso (nível) do aluno em cada uma delas.

Contém um _feed_ com:

  - Alunos em destaque (o aluno pode escolher aparecer). Esse destaque pode ser avaliado por semana.
  - Publicações de alunos
  - Avisos de professores (em destaque)

- A página inicial mostra imagens e mensagens motivacionais de pessoas que são referência para os alunos.

3. Projeto conceitual do banco de dados relacional

Para além do que já está representado no diagrama, é importante ressaltar alguns pontos que nosso modelo deve atender. Primeiramente, uma série de permissões terão que ser criadas para que se garanta algumas funcionalidades do sistema. Por exemplo, professores só poderão criar, editar e apagar questões relacionadas a disciplinas que lecionam. Administradores, por sua vez, terão liberdade para fazer isso com qualquer questão, mesmo que não estejam relacionados a uma disciplina específica. Além disso, a quantidade de pontos necessárias para alcançar um nível também não foi especificada no modelo.

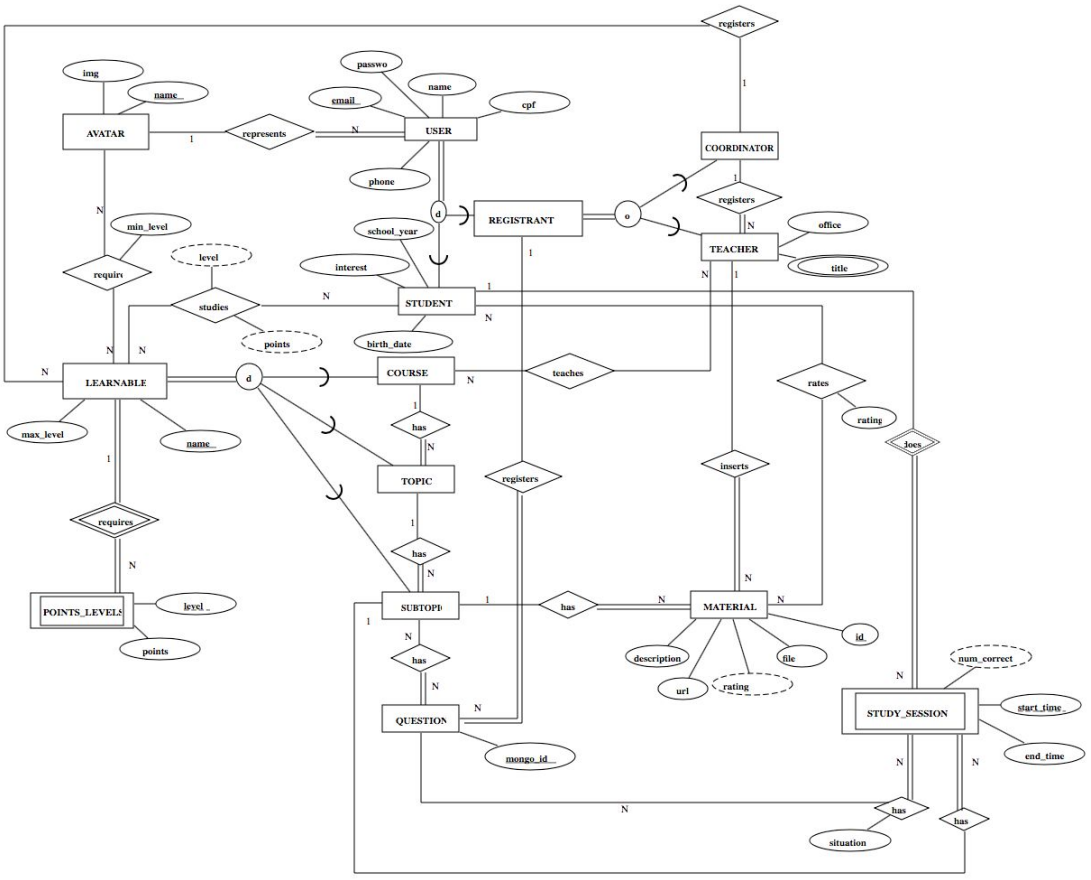

Figura 1: Modelo Relacional

4. Modelo semiestruturado do banco de dados NoSQL

A interação entre os usuários é o principal requisito a ser apoiado no banco de dados não relacional. Nessa funcionalidade, queremos deixar a interação o mais natural e livre possível.

Cada aluno poderá dar uma nota para cada questão, ou fazer comentários nos posts referentes às questões. As resoluções das questões também estará no banco nosql, sendo que cada questão pode possuir mais de uma solução.

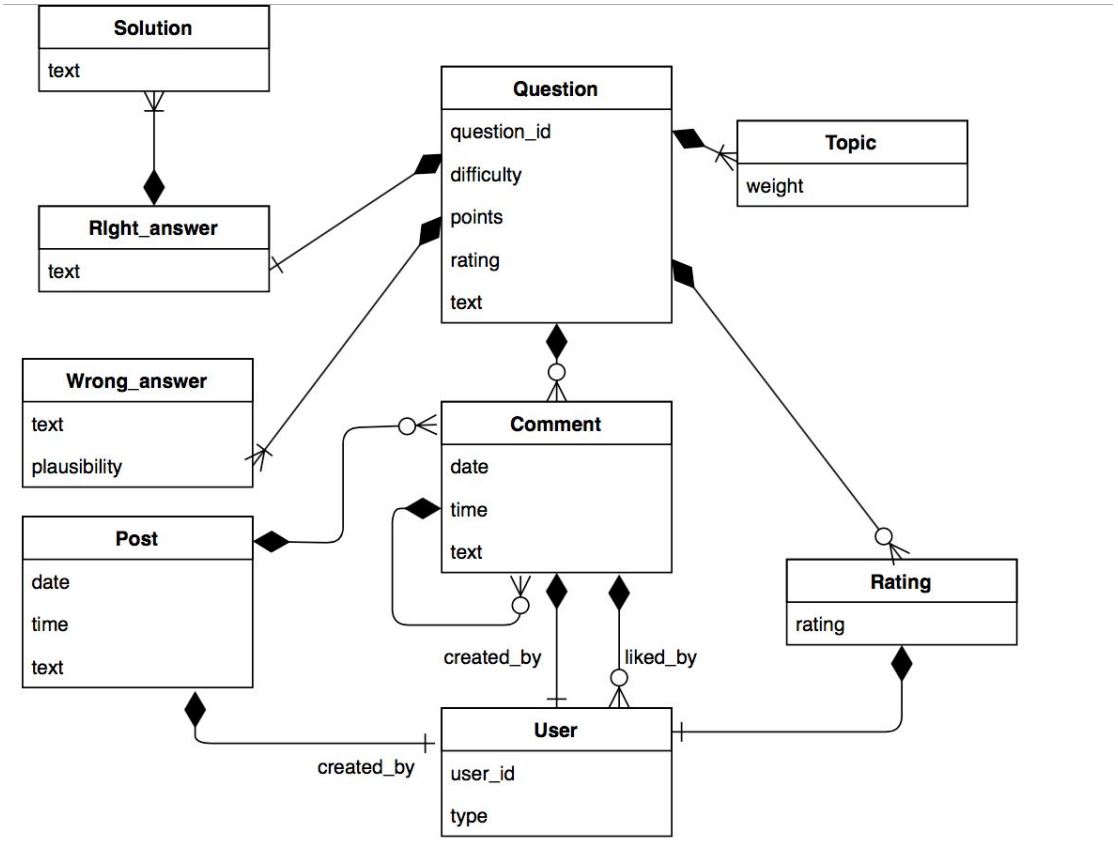

Figura 2: Diagrama do modelo NoSQL

5. Correções e outras mudanças

A seguir serão listadas as mudanças que fizemos no banco, seja para atender a uma correção apontada pela monitora, ou para melhorar algum aspecto do banco.

- Os esquemas relacionais, físicos e do noSQL foram passados para o inglês. Isso foi feito porque o Rails tem convenções de nomes para os atributos e arquivos que ficam bastante estranhas quando aplicadas a um código em português.
- Conforme indicado pela correção, algumas entidades receberam novos atributos para caracterizá-las melhor.
- Foi adicionada a entidade nivel_ponto (levels_points) para que a quantidade de pontos necessárias para alcançar os níveis em um determinado 'estudável' (learnable) fosse representada no modelo.
- Os avisos dos professores são considerados posts comuns e ficarão no BD noSQL. Um post de um professor poderá ser destacado simplesmente pelo fato de ser de um professor.
- Para que admins também conseguissem cadastrar questões, criamos a entidade 'cadastrador' (registrant).
- Para diminuir a confusão entre 'tema' e 'tópico', alteramos os nomes dessas entidades para 'tópico' (topic) e 'subtópico' (subtopic), respectivamente.
- Ao passar para o modelo físico, criamos um atributo a mais em 'Sessão de Estudo' (Study_Session) para funcionar como chave primária. Isso foi feito porque o Rails não lida bem com chaves compostas e, em especial, chaves compostas usadas como chaves estrangeiras.
- Como admin é uma palavra reservada no SQL, o nome da entidade 'admin' foi alterado para 'coordenador' (coordenator).

6. Plataforma Escolhida

Decidimos implementar nosso código em Ruby on Rails junto ao Postgresql e ao MongoDB. Para fazer a interação entre o Rails e o MongoDB foi usada a gema Mongoid, que facilita bastante o acesso aos dados. Além disso, criamos um container no Docker para que a execução do código em diferentes máquinas fosse simples.

7. Rodando o Código

Antes de rodar o código, é necessário ter instalado:

- Ruby 2.5.1
- Rails 5.2
- Docker Compose

Para rodar nossa aplicação, execute os comandos abaixo na raiz do projeto:

```
$ docker-compose up
$ docker-compose run web rake db:setup
```

Para rodar os exemplos de acesso aos bancos, acesse localhost:3000 no seu navegador depois de subir o Docker. Esses exemplos podem ser encontrados em *brainpin/controllers/db_example_controllers.rb*. É relativamente difícil de encontrar a saída do exemplo na console do Rails devido a grande quantidade de coisas que são impressas referentes a configuração.

8. Testando inserção no banco relacional

É possível testar a inserção e busca de outras tabelas no banco que não foram contempladas no exemplo. Para cada tabela, foi criada uma view que pode ser acessada no browser com a url “localhost:3000/NOMEDATABELA”, em que NOMEDATABELA deve ser substituído pela tabela em questão. Atualmente, isto é válido somente para tabelas do modelo relacional.