universidade de aveiro

BASE DE DADOS

Plataforma de Gestão de um *Private Torrent Tracker*

Trabalho Prático - P4G7

MESTRADO INTEGRADO EM ENGENHARIA COMPUTACIONAL

Álvaro Freixo, NMEC 93116

João Maria Machado, NMEC 89132

ÍNDICE

INTRODUÇÃO

No presente trabalho pretende-se desenvolver um protótipo demonstrativo com inspiração numa plataforma de distribuição de *torrents* – um *torrent tracker* - meramente para fins educacionais.

O desenvolvimento do tema citado, possibilitou relacionar dados provenientes de realidades distintas, tal como num contexto real de conceção de uma base de dados. Desta forma, aa aplicação de conceitos lecionados na unidade curricular é essencial. Utilização de *querys* complexas, implementação cuidadosa de *stored procedures* e a manipulação de dados foi uma constante .

A base de dados concebida é baseada num private *torrent tracker* real. Este trata-se de uma plataforma privada com uma comunidade coesa e fechada com regras próprias. Algumas das regras são referidas no documento de requisitos previamente entregue no primeiro momento de avaliação. Assim, o presente relatório subentende a leitura prévia do documento de requisitos. No entanto, as regras podem ser sumariadas no seguinte princípio: "...garantir a sobrevivência do conteúdo, a satisfação e a sustentabilidade da comunidade".

O presente relatório encontra-se dividido em 3 partes. Uma primeira parte na qual é explicada o modelo de dados dando destaques a algumas tabelas centrais ao projeto, uma segunda parte focada na introdução das tabelas e tuplos na base de dados e finalmente, uma terceira parte focada nos testes feitos à integridade do modelo e a sua validade.

Para além da análise de requisitos a leitura deste relatório pressupõem que o leitor tem acesso ao diagrama de entidade relação, ao esquema relacional, ao script que contém o DDL (*data definition language*), ao script que contém o DML (*data manipulation language*), ao *script* que contém a inserção dos tuplos para teste da BD e, finalmente, o acesso ao GUI desenvolvido pelo grupo que, por sua vez, implementa de maneira prática o trabalho desenvolvido.

MODELO DE DADOS

Análise de Requisitos

Antes de proceder à criação da base de dados propriamente dita foi necessário discutir os requisitos prévios à alocação dos dados. Para tal, procedeu-se à elaboração de uma lista de requisitos. Esta lista resulta de uma cuidada análise, discutida em grupo. Ao ter por base um site real, foi possivel a enumeração e diferenciação dos tipos de dados necessários à criação de um diagrama de entidade-relação. Um dos aspetos a sublinhar é a criação de uma estrutura de dados capaz de suportar vários tipos de utilizadores com funções diferentes na comunidade, nomeadamente: "Utilizador Normal" capaz de descarregar conteúdo, utilizador conhecido como "*Uploader*" responsável pela inserção de novos conteúdos na plataforma,utilizador do tipo "*Staff*" capaz de moderar as interações na plataforma e recompensar utilizadores pelo bom comportamento e, finalmente, utilizador do tipo *Admin* com plenos poderes sobre a plataforma.

Outro aspeto que importa relevar é a criação de uma estrutura interna de organização de conteúdo. Para melhorar a organização da plataforma e oferecer ao utilizador uma experiência de pesquisa mais amigável os conteúdos são divididos em 3 categorias: Programas, Jogos, Conteúdo cuja informação é detalhada num site de crítica de cinema, que, por sua vez, se subdivide em séries e filmes..

Como forma de permitir a interação entre membros da comunidade e conteúdo implementou-se um sistema de comentários de modo que todos os utilizadores possam tecer comentários relativos a qualquer conteúdo disponibilizado na plataforma.

Finalmente, para incentivar o crescimento da comunidade, implementou-se um sistema de prémios segundo o qual, utilizadores podem ser recompensados pela boa conduta de interação com conteúdos ou até mesmo por recrutarem outros utilizadores.

Com estes requisitos considerados, procede-se à elaboração de um diagrama de entidade- relação.

Diagrama de Entidade Relação

O diagrama de entidade relação consiste na planificação da organização de informação na base de dados. Após todas as considerações feitas pela análise de requisitos é possível começar a planificar a estrutura dos dados bem como a maneira como estes se relacionam entre si.

Um processo importante, que acompanha a criação deste diagrama corresponde à escolha de chaves primárias para cada entidade incorporada no próprio diagrama. Uma chave primária é uma chave que identifica unicamente um tuplo numa tabela. Esta deve ser atribuída cuidadosamente, pois tem de ser imperativamente imutável, ou seja, não pode consistir numa caraterística que possa ser suscetível de modificação, caso contrário a integridade dos dados pode ser seriamente comprometida.

Um dos casos que mais importa mencionar é o caso do utilizador, no qual foi necessário criar um atributo próprio para atuar como chave primária – ID_User, é de notar que este atributo foi definido como um *integer* (*INT*) que é automaticamente incrementado à medida que os utilizadores vão aderindo à plataforma. Esta estratégia de escolha de chaves primárias foi utilizada em várias tabelas, sendo possível garantir a integridade dos dados na BD.

Uma outra situação que se revelou importante, a quando da criação do diagrama de entidade relação (e consequentemente durante todo o processo de criação da base de dados), foi o uso adequado de relações de hierarquias de classes (*Is-A*). Estas consistem numa classificação de dados que afirma que os dados classificados pertencem, hierarquicamente, a uma (ou várias) entidades posteriormente especificadas. Um exemplo desta implementação no nosso modelo de dados pode ser observado na figura 2.1.

.

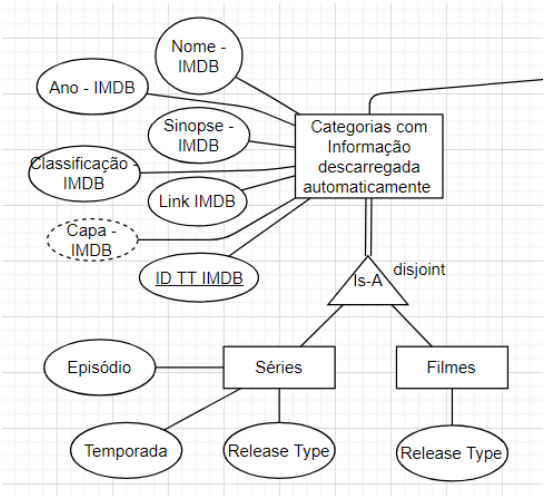

Figura 2.1: Exemplo de uma hierarquia de classes na implementação do diagrama entidade-relação. Neste caso é possível observar que uma categoria cuja informação é descarregada automaticamente é uma série ou filme.

Após considerar as propriedades previamente discutidas, foi possível elaborar um esquema de entidade-relação. Neste estão explicitados todos os dados que possam ser úteis na construção da base de dados de um *private torrent tracker,* bem como a maneira com estes se relacionam entre si (relações), a cardinalidade dos diferentes domínios dos atributos, explicitação das chaves primárias e atributos derivados, e, também, a obrigatoriedade que cada entidade tem (ou não) em participar numa relação.

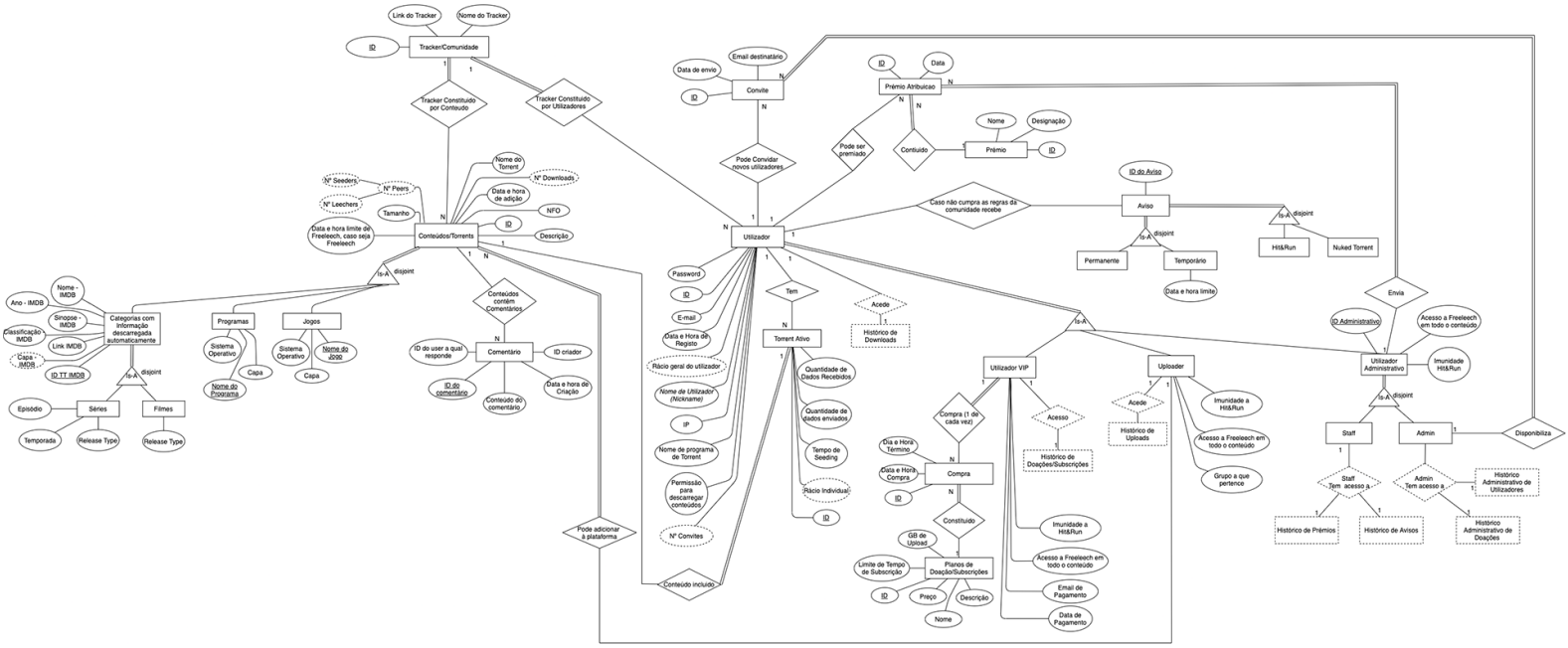

Figura 2.2: Imagem do diagrama entidade relação elaborado (Para uma melhor resolução recomenda-se a visualização da imagem anexada).

Esquema Relacional

Após a criação do diagrama de entidade\relação pode agora proceder-se à criação do esquema relacional. Contrariamente ao diagrama de entidade relação, o esquema relacional consiste numa forma de visualizar os dados sob a forma de tabelas, como maneira de diminuir o nível de abstração no processo de implementação e permitir a aproximação à linguagem SQL. Ou seja, a passagem do DER (diagrama entidade\relação) para o ER (esquema relacional) implica uma aproximação entre aquilo que foi previamente discutido a nível de requisito e a implementação que será feita no DDL (*data definition language*).

No processo de transição do DER para o ER, aquilo que era previamente definido como atributo (ou até mesmo relação) será agora sob a forma de uma tabela na qual são explicitados os seus atributos (também evidenciando a chave primária).

Nesta etapa ocorre também um processo extremamente importante no âmbito da questão da integridade dos dados, a definição de chaves estrangeiras. Uma chave estrangeira (Foreign key) consiste numa chave que é importada de outra tabela. A omissão, ou definição indevida de chaves estrangeiras pode facilmente levar a conflitos de dados. A figura 2.2 mostra um exemplo da definição de chaves estrangeiras.

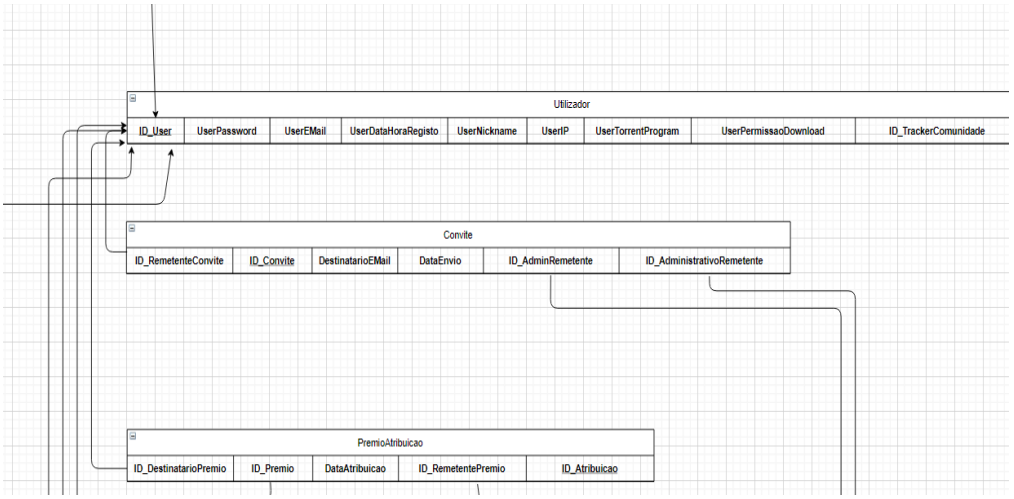

Figura 2.3: Exemplo de uma tabela cuja chave (*ID_User*) é chave estrangeira de 8 tabelas diferentes, o que é expectável, visto que várias entidades são moldadas pelo utilizador.

Uma última etapa a considerar antes de prosseguir para a DDL é a normalização de dados. Este processo tem apenas um propósito: a eliminação de redundâncias. Para tal efeito identificam-se potências dependências

funcionais nas tabelas do ER, isto é, maneiras como dados numa tabela se relacionam entre si. Seguidamente classificam-se as dependências funcionais. Através da eliminação ordenada e sucessiva dos diferentes tipos de dependências funcionais podemos atingir a forma normal de Boyce-Codd (BCNF). Este algoritmo foi aplicado\verificado em todas as tabelas do nosso ER, desse modo evitando potenciais redundâncias.

DATA DEFINITION LANGUAGE (DDL)

Criação de Tabelas

Nesta secção do relatório será explorado o processo de implementação do modelo de dados em SQL. Serão abordados alguns dos princípios que, durante a realização do trabalho, se mostraram determinantes.

Na secção anterior foi elaborado um esquema relacional, que nesta etapa será o principal guia para a criação de tabelas no SQL. O trecho de código 3.1 exemplifica a criação da tabela de utilizador.

```
CREATE TABLE TorrentTracker.Utilizador(
        ID_User                                 TorrentTracker.ID                   IDENTITY(1,1),
        UserPassword                    VARCHAR(MAX)                    NOT NULL,
        UserEMail                               VARCHAR(MAX)                    NOT NULL,
        UserDataRegisto                 SMALLDATETIME                   NOT NULL,
        UserNickName                    VARCHAR(256) NOT NULL,
        UserIP                                  VARCHAR(MAX),
        UserTorrentProgram              VARCHAR(256),
        UserPermissaoDownload   BINARY(1)                               NOT NULL,
        ID_TrackerComunidade    TorrentTracker.ID,

        PRIMARY KEY (ID_User)
)
GO
```

Figura 3.1: Exemplo de criação de uma tabela em SQL.

Desde já, uma das boas práticas que foi implementada foi o uso cuidadoso da declaração de variáveis a "*NOT NULL*". Quando uma variável é declarada no SQL esta pode, preemptivamente, ser verificada como nunca ser nula. Ou seja, é requerido pelo modelo de dados que esta variável tenha um valor "existente". Este cuidado revela-se particularmente importante a quando da inserção de novos dados na base de dados pois determinados dados podem não existir e o modelo de dados pode continuar perfeitamente válido

(aliás, em determinados casos é mesmo necessário que determinados atributos não existam).

Um aspeto que importa referir é o uso de variáveis declaradas com *"IDENTITY(1,1)"*. No decorrer da declaração de tabelas várias tabelas tem como atributo chave um ID, este ID é único e imutável (chave primária), logo é conveniente que o modelo de dados a quando da sua criação o interprete como tal. Daí a utilidade de declarar este ID's com o atributo *"IDENTITY(1,1)"*. Ao criar as variáveis deste modo, garantimos que à medida que são introduzidos dados no modelo estes vêm sobre a forma de um inteiro que vai incrementando.

Uma vantagem nativa ao SQL que também se provou útil é o facto de uma variável poder ser inicializada com o atributo "*SMALLDATETIME*" que atribui um valor de uma data seguida de uma hora a uma variável. Este tipo de variáveis é particularmente útil quando, como por exemplo, tratamos de variáveis relacionadas com planos de subscrição que precisam de ser cuidadosamente calculadas para garantir que o utilizador tem acesso justo ao plano pelo qual pagou.

Após todas as considerações anteriores com a inicialização de variáveis foi possível fazer um pequeno teste à robustez do modelo de dados inserindo alguns tuplos gerados aleatoriamente com recurso ao site Generate Data. Geramos entre 100 (no máximo) a 5 (no mínimo) tuplos para cada tabela de modo aleatório.

Stored Procedures e Querys

Após termos implementado o modelo de dados planeado na secção anterior e termos populado a base de dados construída, analisamos cuidadosamente que tipo de acessos à base de dados iremos necessitar iremos necessitar para a tornar funcional. Grande parte destes acessos podem ser facilitados com o uso de stored procedures. Stored procedure trata-se de funções cujos *outputs* ficam guardados em memória na base de dados para referências\utilizações futuras.

Um dos *stored procedures* mais importantes na elaboração deste trabalho trata-se do *stored procedure* do *login*. Este procedimento é tão importante à nossa base de dados pois é este que garante que a plataforma de *torrent tracking* é realmente fechada. Na figura 3.2 podemos ver a implementação do stored procedure do *login* de um utilizador.

```
-- Login Page
CREATE PROCEDURE TorrentTracker.LoginProcedure @utilizador VARCHAR(256), @password VARCHAR(MAX)
  AS
  BEGIN
    SELECT * FROM TorrentTracker.Utilizador
    WHERE UserPassword=@password and UserNickName=@utilizador
  END
GO
```

Figura 3.2 : *Stored procedure* implementado para o login de utilizador quando este deseja aceder à plataforma. Deve notar-se que para cada par de *inputs* utilizador e *password* a tabela de retorno deve ter no máximo uma linha e no mínimo zero. Caso contrário teríamos uma falha flagrante de segurança.

Um outro aspecto, ainda a respeito do procedimento de login que importa mencionar foi o procedimento de segurança adotado. Isto é, para garantir que, se ocorrer algum problema de fuga de dados o utilizador não é comprometido. Para tal efeito implementou-se um sistema de encriptação de *password* desse modo, mesmo que a informação da base de dados caía nas mão de alguém mal intencionado, os utilizadores continuam a ser os únicos com acesso à sua conta.