

**Área Departamental de Engenharia de Electrónica e Telecomunicações e de Computadores**

**Trabalho Prático - Fase 1**

|  |  |  |
| --- | --- | --- |
| Autores: | 43597 | Marco Borges |
|  | 45368 | Pedro Pires |
|  | 45827 | Daniel Azevedo |

Relatório de trabalho realizado no âmbito de Sistemas de Informação 2, do curso de Licenciatura em Engenharia Informática e de Computadores Semestre de Verão 2020/2021

Professor: Afonso Remédios

16 – 05 – 2021

<< Esta página foi intencionalmente deixada em branco >>

Resumo

Cada vez mais jogos suportam modo online e multijogador, sendo que para tal necessitam de um sistema de informação, o qual suporta estas funcionalidades. A empresa “Jogos & Joguinhos” pretende criar um sistema de informação com a finalidade de suportar o seu novo jogo online.

A solução apresentada passa por utilizar um sistema de gestão de base de dados relacional de modo a criar o modelo físico para o negócio a partir de um modelo conceptual desenhado em conjunto com os responsáveis da empresa, de modo a perceber as necessidades para este novo sistema.

**Palavras-chave:** Jogo Online, Multijogador, Microsoft SQL Server, Modelo Entidade-Relação, Modelo Relacional, SQL, Transact SQL

Abstract

More and more games support online and multiplayer modes, to achieve this an information system is necessary as a support for these functionalities. The company “Jogos & Joguinhos” pretends to create an information system to support such features of their new online game.

The presented solution uses a data base management system to create the physic model for transaction logic implementation from a conceptual model designed with the company’s responsible to answer all requirements.

**Keywords:** Online Game, Multiplayer, Microsoft SQL Server, ER Model, Relational Model, SQL, Transact SQL

Índice

[Lista de Figuras iv](#_Toc72096173)

[1. Introdução 5](#_Toc72096174)

[2. Modelo de Dados 6](#_Toc72096175)

[a. Modelo Entidade-Relação 7](#_Toc72096176)

[b. Modelo Relacional 8](#_Toc72096177)

[c. Modelo Físico 10](#_Toc72096178)

[3. Referências 15](#_Toc72096179)

# Lista de Figuras

1. Modelo Entidade-Relação ………………………………………………………………………………. 7

# Introdução

Neste trabalho foi-nos proposto desenvolver para a empresa “Jogos & Joguinhos” um sistema de informação, cujo objetivo é suportar o seu novo jogo online. Com a finalidade de implementar o referido sistema, optamos inicialmente por definir os seus modelos de dados abstratos. Estes são responsáveis por definir a estrutura de dados ou informações que deverão ser implementadas na base de dados.

Posteriormente, efetuamos a normalização do modelo até à terceira forma normal. O objetivo desta etapa é avaliar a qualidade do modelo e transformá-lo, se necessário, num modelo equivalente com menos redundância e mais estabilidade.

Na etapa seguinte desenvolvemos as diferentes funcionalidades impostas na descrição do enunciado. Para o efeito recorremos aos procedimentos armazenadas para a elaboração dos mesmos. Estes dispõem de diversas vantagens ao nível de: encapsulamento, reutilização, desempenho e segurança do sistema.

# Modelo de Dados

Sendo o modelo de dados a fundação do projeto é necessariamente elaborado, revisto e remodelado numa fase inicial, de modo a garantir que cumpre as necessidades descritas no enunciado. Primeiramente foi desenvolvido o modelo Entidade-Relação (ER) a partir do qual mais tarde fora derivado o modelo relacional e definidas todas as restrições de integridade necessárias, sendo este último usado para realizar o modelo físico.

Com vista a facilitar o processo de elaboração dos diferentes modelos foram primeiramente definidas as entidades presentes no enunciado do projeto, com vista a estabelecer um ponto de partida:

1. **PLAYER** guarda a informação sobre o estado da conta do jogador e associa o seu player\_id ao respetivo username;
2. **LOGIN** armazena toda a informação pessoal do jogador;
3. **GLOBAL\_CONFIGURATION** é a configuração global do jogo que afeta todos os jogadores;
4. **CLAN** informação sobre os clãs existentes;
5. **REGISTEREDPLAYER** jogador registado para o qual é armazenado informação quanto à sua conta de jogo, tais como, nível, balanço bancário, pontos de vida/força/velocidade, o seu clã e pontuação;
6. **NONREGISTEREDPLAYER** um armazenamento mais simples da informação do jogador registado, em que apenas é atribuído um nome que é apresentado no jogo a um player\_id;
7. **STATE** os diversos estados possíveis para uma partida;
8. **TYPE** os diversos tipos possíveis de partida, por exemplo, contabilizada ou amigável, bem como os requisitos para participar na mesma;
9. **MATCH** armazena informação sobre quem participa na match, o estado e tipo da mesma, bem como a hora de começo e o vencedor;
10. **ITEM** guarda os atributos, o estado (ativo/inativo) e o nome do item bem como o player\_id do jogador que possui o item;
11. **ITEM\_PRICE** associa um preço a cada item.

# Modelo Entidade-Relação

Figura 1: Modelo Entidade-Relação

# Modelo Relacional

**PLAYER** (player\_id, name, deleted)

PK: player\_id

RI1: deleted é um boolean que identifica se o jogador eliminou a conta

RI2: name não pode ser null

**LOGIN** (login\_id, user\_email, username, name, password, birthday)

PK: login\_id

AK: user\_email

RI1: username deve ser único

RI2: user\_email não pode ser null e têm que ser único

RI3: username é obrigatório e têm que ser único

**GLOBAL\_CONFIGURATION** (config\_id, max\_level, clan\_members, limit)

PK: config\_id

R1: Nenhum dos atributos pode ser null.

**REGISTEREDPLAYER** (player\_id, login\_id, score, level, bank\_balance, life\_points, strength\_points, speed\_points, clan)

PK: player\_id

AK: login\_id

FK: {player\_id} para PLAYER(player\_id), {clan} para CLAN(clan\_id), {login\_id} para LOGIN(login\_id)

RI1: lvl, life\_points, strength\_points e speed\_points estão compreendidos no intervalo [1, lvlMax];

RI2: bank\_balance e score >= 0.

**NONREGISTEREDPLAYER** (player\_id, name)

PK: player\_id

FK: {player\_id} para PLAYER (player\_id)

**STATE** (state\_id, state)

PK: state\_id

**TYPE** (type\_id, min\_life\_points, min\_strength\_points, min\_speed\_points)

PK: type\_id

R1: min\_life\_points, min\_strength\_points, and min\_speed\_points must be >= 1

**MATCH** (match\_id, state\_id, type\_id, player\_one, player\_two, start\_time, winner)

PK: match\_id

FK: {player\_one} para PLAYER (player\_id), {player\_two} para PLAYER (player\_id), {state\_id} para STATE (state\_id), {type\_id} para TYPE (type\_id)

RI1: start\_time <= System.currentTime()

**CLAN** (clan\_id, , clan\_name, clan\_score)

PK: clan\_id

AK: clan\_name

RI2: clan\_score é a média das pontuações de todos os jogadores de um clã

**ITEM** (item\_id, player\_id, name, bonus\_life\_points, bonus\_strenght\_points, bonus\_speed\_points)

PK: item\_id

FK: {player\_id} para Player (player\_id)

AK: name

RI1: bonus\_life\_points, bonus\_strenght\_points e bonus\_speed\_points >= 0

**ITEM\_PRICE** (item\_id, price)

PK: item\_id

FK: {item\_id} para ITEM (item\_id)

RI1: price >= 0

# Funcionalidades em Transact-SQL

Nesta parte deste documento são apresentadas as funcionalidades requisitadas pelo enunciado na primeira fase do projeto em *Transact-SQL*. Estas funcionalidades destinam-se a demonstrar como pode ser usado o modelo relacional criado na primeira parte deste relatório utilizando controlo transacional, de modo a otimizar o desempenho dos acessos.

Numa primeira seção será apresentado o modelo físico proposto para a base de dados, isto é, o *script* em *TSQL* que cria a base de dados, com base no modelo em 2.b.

Na segunda parte será justificada a utilização e implementação de funções, procedimentos, gatilhos e vistas de modo a satisfazer as restantes necessidades apresentadas no enunciado do trabalho, sendo que na terceira e última parte deste capítulo será feita uma breve introdução ao código de teste de todas as funcionalidades implementadas em *TSQL*.

# Modelo Físico

Para a criação do modelo físico do esquema relacional proposto é necessário o *script* **create\_tables.sql** que tem como objetivo criar as tabelas correspondentes na base de dados. Associado a estas tabelas existem ainda funções e gatilhos que têm como objetivo garantir o correto funcionamento de acordo com as regras de negócio e restrições apresentadas em 2.b.

# Outras Funcionalidades

De modo aos utilizadores poderem interagir com a base de dados corretamente foi criado um conjunto adicional de funcionalidades, requisitadas pelo enunciado, de modo a garantir o correto uso das funcionalidades e do controlo transacional.

# A

Por tratar-se do *script* de criação da base de dados o seu nível de isolamento pode ser baixo, tendo sido utilizado *READ UNCOMMITTED*. Neste *script*, tal como é dito no ponto 3.a foram criadas todas as tabelas de acordo com o modelo relacional e restrições presentes no ponto 2.b.

# B

# C

# D

# E

# F

# G

# H

Nesta funcionalidade é pedido pelo enunciado que se crie um procedimento armazenado (ficheiro p\_CreatePlayerWithOptions.sql) que crie um novo jogador, com as opções de passar um item e/ou um clã a associar ao jogador.

Se for passado um item\_id, e for um id válido, o item é associado ao jogador. O mesmo se passa com clan\_id.

# I

# J

# K

De modo a satisfazer este requisito foi criada a função f\_getPlayerGamesFromYear presente no ficheiro com o mesmo nome, que retorna uma tabela com toda a informação das matches encontradas onde o jogador especificado participou no ano determinado.

# L

# M

Esta transação foi definida com o isolamento SERIALIZABLE porque não podemos correr o risco de apanhar *phantom* *tuples* no momento de verificação de quantos jogos o vencedor fez no total de modo a calcular o seu nível.

Primeiro é feito uma verificação no *trigger*, para caso do vencedor passado como parâmetro não se encontrar no match a atualizar. Nessa situação é lançada uma mensagem de erro para o utilizador a informar do mesmo.

Depois verificamos se o match ainda está a decorrer, para o efeito de não estar a contabilizar jogos por iniciar, ou já terminados/contabilizados.

Para atualizar os atributos dos jogadores envolvidos na partida recorremos à uma função que calcula o nível com base no seguinte algoritmo (1 + Nº Vitórias - Nº Derrotas % limit).

De forma a simplificar o mecanismo foi assumido que num encontro entre dois jogadores, apenas é incrementado os atributos e nível do vencedor.

Para calcular as vitórias contabilizamos todas as partidas em que o jogador venceu. No caso das derrotas procuramos todas as partidas em que este esteve envolvido, e subtraímos às vitórias.

# Teste das Funcionalidades

# Conclusões

Neste documento foi apresentada uma solução possível de um sistema de gestão de base de dados relacional para suportar o novo jogo online da empresa “Jogos & Joguinhos”. Para realizar este sistema foram explorados diversos tópicos, nomeadamente, controlo transacional, funções, procedimentos armazenados e gatilhos.

Apesar deste sistema ser funcional consideramos que ainda existe trabalho a ser desenvolvido nesta vertente, como por exemplo, diminuição de complexidade temporal de cálculo da pontuação de um clã em dado momento de O(n) para O(1), em que n é o número de membros do clã, com recurso a *caches*.

Outro aspeto a melhorar será completar este documento com a justificação das alíneas em falta.

# Referências

[1] Fundamentals of Database System 7th ed., R. Elmasri e Shamkant B. Navathe, Pearson Education, 2016, ISBN: 0-13-397077-9