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

1. 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.

2. 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.

a. Modelo Entidade-Relação

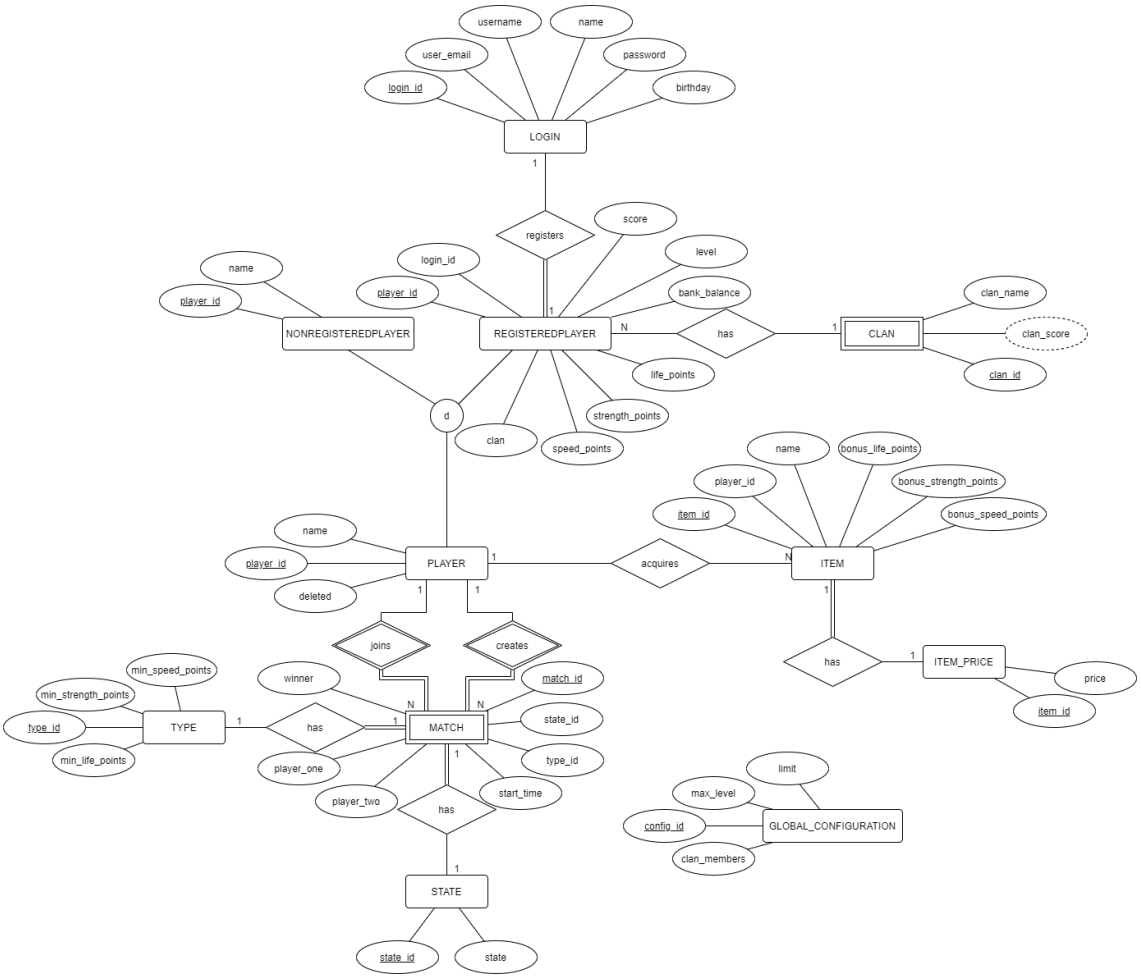

Figura 1: Modelo Entidade-Relação

b. 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

3. 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*.

a. 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.

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.

i. 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.

ii. B

iii. C

iv. D

v. E

vi. F

vii. G

viii. 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.

ix. I

x. J

xi. 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.

xii. L

xiii. 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.

c. Teste das Funcionalidades

4. 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.

5. Referências

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