UNIVERSIDADE DE ÉVORA

Escola de Ciências e Tecnologia
Mestrado em Engenharia Informática
Disciplina: Tópicos Avançados de Bases de Dados
Docente: Prof. Teresa Gonçalves

Programação Server-Sided

Trabalho Prático

Trabalho Elaborado Por:
João Aiveca, Nº 10712
Ricardo Dias, Nº11246
Marlene Oliveira, Nº11327

Ano Letivo 2013/2014

Índice

Introdução

Um histórico de alterações numa base de dados pode ser extremamente útil em cenários de *disaster recovery*. Este tipo de mecanismo permite obter o estado da base de dados ou de apenas alguns elementos da mesma com uma simples consulta do histórico, sendo então possível usar a informação obtida para recuperar um estado anterior da base de dados.

A construção do histórico implica o uso de programação do lado do servidor, o que permitirá criar primitivas que irão não só criar a estrutura de tabelas que irá armazenar as informações do histórico, bem como os *triggers* que irão gerir as operações de escrita nestas tabelas. O registo constante de informação quando ocorrem operações como INSERT, DELETE e UPDATE é efetuado automaticamente, não sendo necessária intervenção do utilizador.

Para este trabalho foi criada uma base de dados para teste, na qual foram aplicadas todas as primitivas criadas.

O Sistema

Descrição do Sistema

O sistema criado permite consultar a informação presente numa base de dados e criar automaticamente novas tabelas, nas quais serão armazenados os históricos das tabelas originais. Este histórico permite ao utilizador consultar várias informações sobre as operações que ocorreram na base de dados (por exemplo, ver o histórico completo de uma determinada tabela, consultar histórico por operação, etc).

O sistema permite ainda consultar, alterar, remover e inserir dados na base de dados, sendo estas operações automaticamente registadas no histórico. Para que estas operações sejam mais fáceis de executar para o utilizador, criou-se uma *webapp*. Uma captura de ecrã da página inicial pode ser consultada na figura 1.

Figura 1 - Página inicial da *webapp* do trabalho.

Neste sistema é usada uma arquitetura cliente servidor. O cliente (*webapp*) comunica com o servidor de modo a obter a informação que se encontra na base de dados e a fazer operações sobre a mesma. Uma representação gráfica do nosso sistema pode ser consultada na figura 2.

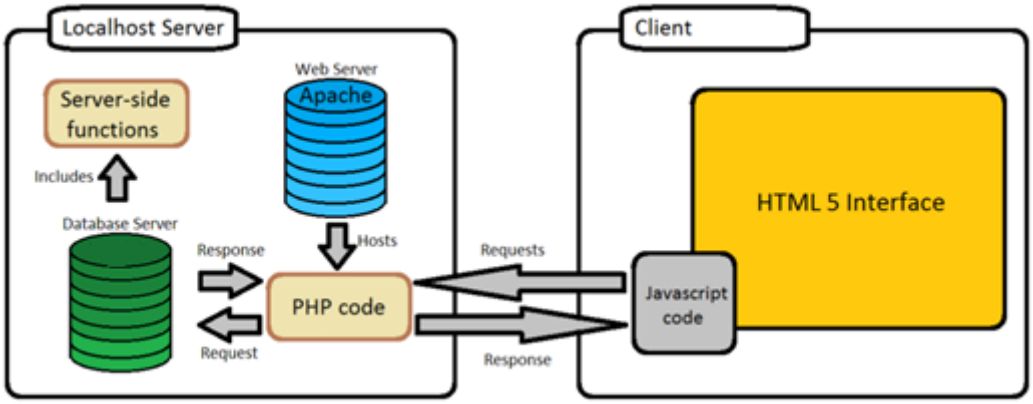

Figura 2 - Representação gráfica do sistema.

Os componentes do sistema funcionam do seguinte modo:

- *Web Server* – Hospedado na máquina de teste, esta estrutura terá em execução dois servidores. O servidor Apache, que hospeda o código PHP, serve de intermédio entre o cliente e a base de dados. O *database server*, por sua vez, hospeda a base de dados e as funções do lado do servidor que o cliente pretende utilizar. Não há interação direta entre o servidor *PostgreSQL* e o cliente, sendo enta feita apenas através de *bindings* PHP;
- *Client* - Utilizável em qualquer sistema com um *browser* disponível com capacidade de ler HTML5. O cliente trata-se de uma página escrita em HTML5, para comunicar com o servidor PHP, de modo a obter a informação das tabelas da base de dados assim como a informação do histórico de operações.

Funcionalidades do Sistema

O cliente representado por um *webapp* em HTML e PHP permite a manipulação da base de dados.

Este sistema permite efetuar pesquisas, inserções, atualizações e eliminações de dados sobre as tabelas que constituem a base de dados de teste (Equipa, Jogadores, Jogos e Golos). Estas operações estão definidas de modo a respeitar a integridade dos dados. É também possível consultar estatísticas sobre os dados inseridos nas tabelas disponíveis.

A *webwapp* permite ainda a consulta das operações realizadas sobre uma tabela assim como a sua quantidade, sendo também possível obter as operações num intervalo de tempo definido.

A Base de Dados do Sistema

Devido à impossibilidade de passar argumentos como parâmetro das funções dos *triggers* que controlam a geração do histórico [1], foi criada uma base de dados sobre a qual são aplicados os *triggers* criados, que permitem escrever nas tabelas do histórico. Esta base de dados serve apenas para testar o histórico. O modelo entidade associação desta base de dados pode ser consultado na figura 3.

Os *scripts* que permitem criar as tabelas da base de dados do trabalho podem ser consultados no ficheiro *Step1_Create_Tables.sql*. Os scripts de criação dos *triggers* que fazem o registo das operações no histórico e das suas respetivas funções podem ser consultados no ficheiro *Step2_Create_Functions_Triggers.sql*. Os scripts correspondentes às inserções de dados na base de dados podem ser consultados no ficheiro *Step3_Inserts.sql*.

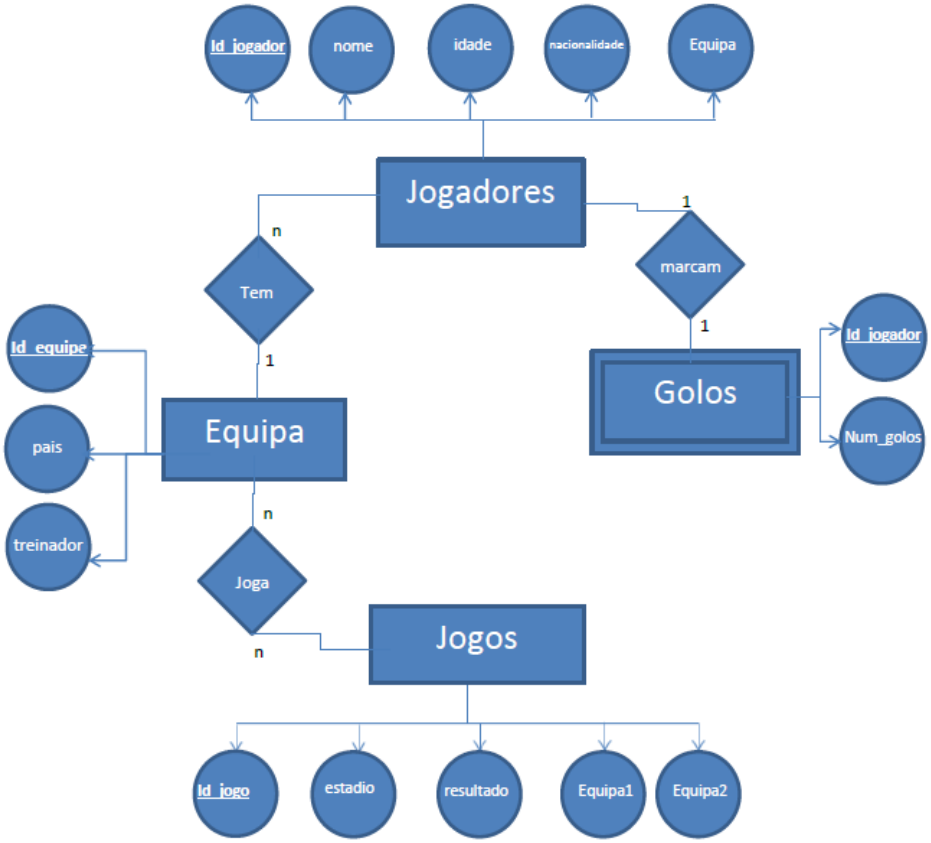

Figura 3 – Modelo Entidade-Associação da base de dados do trabalho.

Tabela No Modelo Relacional	Chaves Estrangeiras
jogadores(id_jogador,nome,idade,nacionalidade,equipa)	**Atributo *'equipa'*** (referência para o campo '*id_equipa*' da tabela 'equipa').
equipa(*id_equipa*, pais, treinador)	Não tem.
jogos(*id_jogo*, estado,resultado,equipa1,equipa2)	**Atributos 'equipa1' e 'equipa2'** (ambos os atributos são uma referência para o campo '*id_equipa*' da tabela '*equipa*').
golos(id_jogador,num_golos)	**Atributo 'id_jogador'** (referência para o campo '*id_jogador*' da tabela '*jogadores*').

Tabela 1 – Modelo relacional da base de dados do trabalho.

As tabelas de histórico são criadas com recurso a um *stored procedure*, que verifica quais as tabelas existentes na base de dados onde o *script* do procedimento é executado e cria as tabelas do histórico daquela base de dados. O nome atribuído às tabelas de histórico segue a convenção seguinte: *<nome_tabela_original>_hist*. Por exemplo, histórico da tabela 'jogador' é guardado numa outra tabela que tem o nome '*jogador_hist*'. Além dos atributos que aparecem na tabela normal, a

tabela que guarda o histórico tem alguns atributos extra. Os atributos extra destas tabelas são os seguintes:

- '*operacao*': correspondente à operação efetuada (INSERT, DELETE, UPDATE);
- '*utilizador*': correspondente ao nome do utilizador que fez a operação na base de dados;
- '*data_hora_op*': um *timestamp* que corresponde à data e à hora em que ocorreu a operação na base de dados.

O código SQL do *stored procedure* encontra-se no ficheiro *Step2_Create_Functions_Triggers.sql*.

Além do que foi referido, o *stored procedure* que cria as tabelas de histórico foi alterado de modo a que este crie automaticamente os *triggers* destas tabelas (funciona agora de um modo dinâmico).

Conclusão

O sistema criado permite aplicar conhecimentos adquiridos nas aulas, nomeadamente a criação e uso de funções se *triggers*. Além disto foi ainda necessário compreender como funcionam os *stored procedures* e aplicar os mesmos, o que também correu bem.

O sistema tem embutido um mecanismo que permite criar o histórico da base de dados que o suporta, mantendo um registo de todas as operações que ocorrem sobre os dados da mesma. Através deste registo é possível saber diversas informações, sendo um exemplo disto o número total de um determinado tipo de operação.

Referências

[1] "*Postgresql trigger function with parameters*". 2011. Acedido em 04-06-2014 a partir de: http://stackoverflow.com/questions/7726237/postgresql-trigger-function-with-parameters

[2] *Database Systems*. P. Atzeni, S. Ceri, S. Paraboschi and R.Torlone. McGraw-Hill. 1999.