MATERA

Exercícios para Analista de Desenvolvimento de Sistemas (Foco em PL/SQL)

Olá aqui você vai encontrar alguns exercícios simples com foco em PL/SQL.
Caso possua dúvidas, não hesite em perguntar.
Boa sorte!

Exercício 1 - Você recebeu uma procedure que realiza a inserção de apostas no banco de dados. O objetivo deste teste é analisar o código existente, identificar possíveis problemas de performance e propor melhorias.

—---------- ínicio código fonte —----------

criando tabela apostas_temp:

```
CREATE TABLE apostas_temp (
    aposta_id INT NOT NULL,
    usuario_id INT NOT NULL,
    valor DECIMAL(10, 2) NOT NULL,
    data_aposta DATE NOT NULL,
    PRIMARY KEY (aposta_id)
);
```

criando tabela apostas_final:

```
CREATE TABLE apostas_final (
    aposta_id INT NOT NULL,
    usuario_id INT NOT NULL,
    valor DECIMAL(10, 2) NOT NULL,
    data_aposta DATE NOT NULL,
    PRIMARY KEY (aposta_id)
);
```

criando procedure processa_apostas

```
CREATE OR REPLACE PROCEDURE processa_apostas IS
    --
    CURSOR c_apostas IS
        SELECT aposta_id, usuario_id, valor, data_aposta
        FROM apostas_temp;
    --
    v_error_message VARCHAR2(4000);
    --
BEGIN
    --
    DBMS_OUTPUT.PUT_LINE('Início do processamento das apostas.');
    --
    FOR r_aposta IN c_apostas LOOP
        --
        IF r_aposta.valor > 0 AND r_aposta.data_aposta IS NOT NULL THEN
            --
            BEGIN
```

```
                --
                INSERT INTO apostas_final (aposta_id, usuario_id, valor,
data_aposta)
                VALUES (r_aposta.aposta_id, r_aposta.usuario_id, r_aposta.valor,
r_aposta.data_aposta);
                --
                COMMIT;
                --
            EXCEPTION
                WHEN OTHERS THEN
                    --
                    v_error_message := SQLERRM;
                    DBMS_OUTPUT.PUT_LINE('Erro ao processar aposta ID ' ||
r_aposta.aposta_id || ': ' || v_error_message);
            END;
            --
        END IF;
        --
    END LOOP;
    --
    COMMIT;
    --
    DBMS_OUTPUT.PUT_LINE('Processamento concluído com sucesso.');
    --
EXCEPTION
    WHEN OTHERS THEN
        --
        ROLLBACK;
        DBMS_OUTPUT.PUT_LINE('Erro durante o processamento: ' || SQLERRM);
END processa_apostas;
/
```

—---------- fim código fonte —----------

Exercício 2 - Você foi designado para criar um fluxo de processo para um sistema de apostas. Sua tarefa é desenvolver uma procedure em PL/SQL que:

2.1. Receba as seguintes informações de entrada:
- Nome do apostador (nome)
- Idade do apostador (idade)
- E-mail do apostador (email)
- Valor da aposta (valor_aposta)

2.2. Realize as seguintes validações e ações:
- Verifique se o email do apostador já existe na tabela de apostadores e caso não exista, siga com os próximos passos de validação e insert.
- Caso o email não exista na base, verifique se a idade do apostador é maior de 18 anos. Se não for, a procedure deve retornar uma mensagem de erro apropriada e não prosseguir com o processo.
- Caso o email do apostador não exista na base e o mesmo seja maior de idade, insira suas informações em uma tabela de apostadores e gere um ID sequencial para ele. Para isso, você deve usar uma sequência existente no banco de dados para garantir a unicidade do ID.

- Após o cadastro do apostador, faça um insert na tabela **apostas_temp** criando um id de aposta sequencial para o campo **aposta_id**, id gerado para o apostador para o campo **usuario_id**, **valor** da aposta para o campo valor e **data_aposta** com data e hora do momento do insert.
- Por último, faça a chamada da procedure existente **processa_aposta.sql** para que as apostas sejam processadas.

Exercício 3 - Dada a procedure `atualizar_valores_aposta`, que calcula o valor de uma aposta com base em duas variáveis principais: o tempo de uma partida e os pontos feitos pelo time. Sua tarefa é identificar e corrigir qualquer bug na lógica de cálculo da procedure.

—---------- ínicio código fonte —----------

criando tabela times

```
CREATE TABLE times (
    id NUMBER PRIMARY KEY,
    nome VARCHAR2(100),
    pontos NUMBER
);
```

criando tabela partidas

```
CREATE TABLE partidas (
    id NUMBER PRIMARY KEY,
    nome VARCHAR2(100),
    data_hora TIMESTAMP,
    id_time1 NUMBER,
    id_time2 NUMBER,
    FOREIGN KEY (id_time1) REFERENCES times(id),
    FOREIGN KEY (id_time2) REFERENCES times(id)
);
```

criando tabela apostas

```
CREATE TABLE apostas (
    id NUMBER PRIMARY KEY,
    id_partida NUMBER,
    id_time NUMBER,
    valor_aposta NUMBER,
    FOREIGN KEY (id_partida) REFERENCES partidas(id),
    FOREIGN KEY (id_time) REFERENCES times(id)
);
```

criando procedure atualizar_valores_aposta

```
CREATE OR REPLACE PROCEDURE atualizar_valores_aposta IS
    v_id_partida partidas.id%TYPE;
    v_id_time apostas.id_time%TYPE;
    v_data_hora TIMESTAMP;
    v_valor_aposta apostas.valor_aposta%TYPE;
```

```
    v_pontos NUMBER;
    v_tempo_restante INTERVAL DAY TO SECOND;
    v_fator_tempo NUMBER := 1.5;
    v_fator_pontos NUMBER := 2.0;
    --
    CURSOR cur_apostas IS
        SELECT a.id_partida, a.id_time, a.valor_aposta, p.data_hora
        FROM apostas a
        JOIN partidas p ON a.id_partida = p.id;
    --
BEGIN
    --
    FOR rec IN cur_apostas LOOP
        v_id_partida := rec.id_partida;
        v_id_time := rec.id_time;
        v_data_hora := rec.data_hora;
        v_valor_aposta := rec.valor_aposta;
        --
        SELECT pontos INTO v_pontos
        FROM times
        WHERE id = v_id_time;
        --
        v_tempo_restante := v_data_hora - SYSTIMESTAMP;
        --
        IF v_tempo_restante < INTERVAL '1' HOUR THEN
            v_valor_aposta := v_fator_tempo * (1 + v_pontos / 100);
            UPDATE apostas
            SET valor_aposta = v_valor_aposta
            WHERE id_partida = v_id_partida AND id_time = v_id_time;
        END IF;
    END LOOP;
    --
    COMMIT;
    --
EXCEPTION
    WHEN OTHERS THEN
        ROLLBACK;
        RAISE;
END;
/
```

—---------- fim código fonte —----------