Bases de Dados (MiEI) – 2019/2020 – 2º Semestre

Relatório do projeto
2ª fase

Loja de compra e venda de ténis

Realizado por:

Grupo 63

Duarte Moreira 55021

Cláudia Santos 57049

José Murta 55226

[MiEI]

Turno Prático 6

Índice

Loja de compra e venda de ténis

Uma loja de revenda de ténis pretende implementar uma base de dados para gerir o seu stock, as compras e vendas dos produtos, e a informação sobre os seus empregados e clientes.

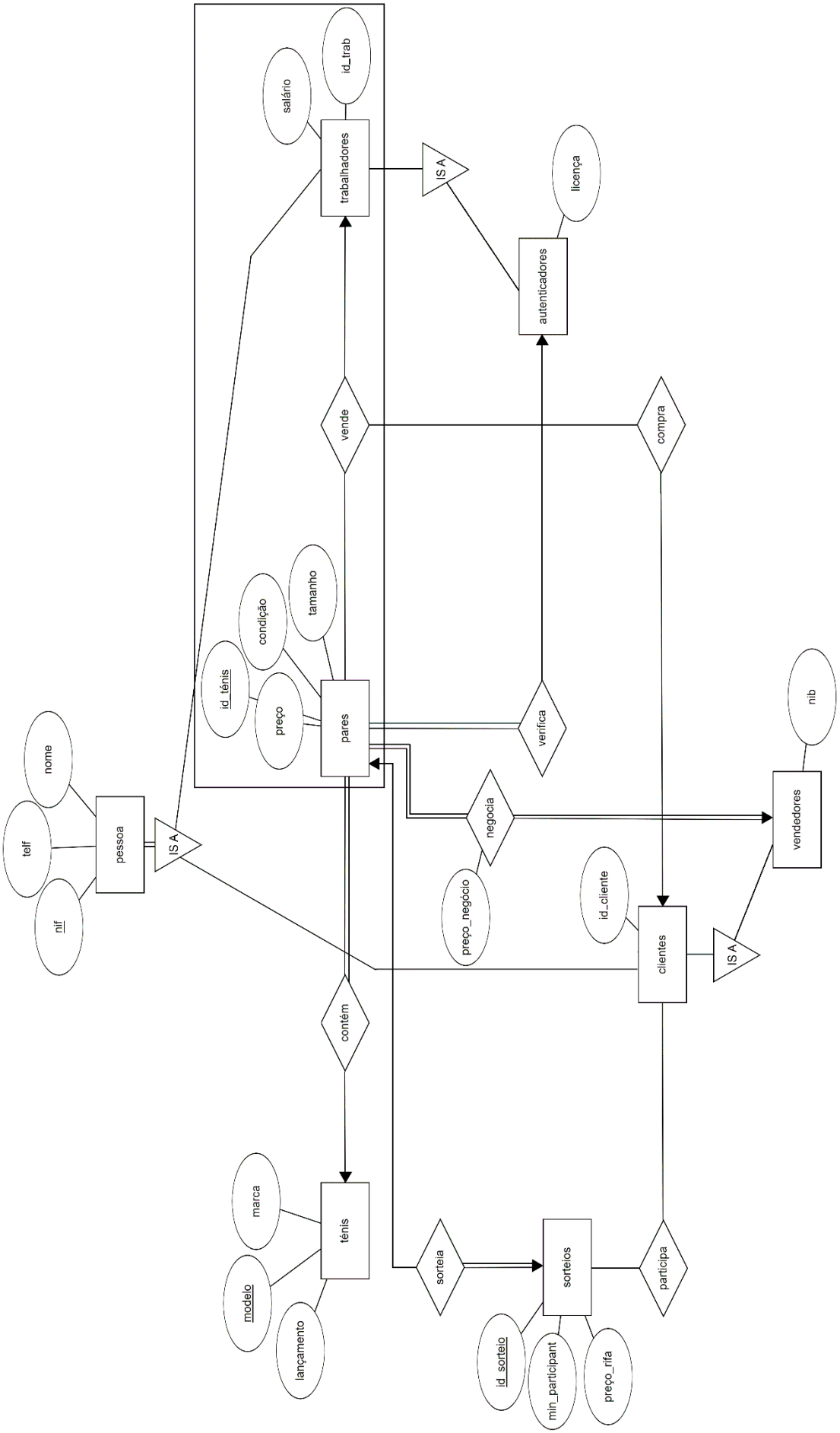
- A loja tem um stock de ténis, cada par com o seu preço, tamanho, modelo, um identificador, marca, ano de lançamento e condição (novo ou usado). Os ténis para a loja são comprados a outros vendedores, com a intenção de serem vendidos a um preço superior ao de compra, para haver alguma margem de lucro.

- Antes de serem comprados para a loja, todos os produtos passam por uma estação de autenticação realizada por profissionais na área. A loja tem dois tipos de trabalhadores, os referidos anteriormente e os outros empregados, todos identificados pelo seu ID, salário, número de telefone e NIF.

- É necessário também guardar informação sobre os clientes, que podem ou não ser vendedores, com atributos similares aos dos empregados, exceto o salário. É importante notar também que cada cliente é um potencial comprador, mas apenas alguns são também vendedores.

- Para finalizar, esta loja terá um sistema de sorteios, onde a loja sorteia um par de ténis e cada participante tem que pagar uma pequena entrada para se habilitar a ganhar; para além disso, de forma a que a loja não tenha prejuízo, é necessário apresentar um número mínimo de participantes para tal se realizar e cada participante apenas pode comprar no máximo uma rifa de cada sorteio (por exemplo, se o sorteio se basear num par com um custo de 100€ e o preço de cada rifa for 5€, então o número mínimo de participantes teria de ser, supostamente, por volta de 20).

Modelo ER


salário
id_trab
trabalhadores
IS A
licença
autenticadores
vende
compra
condição
tamanho
id_ténis
preço
pares
verifica
nome
telf
nif
pessoa
IS A
negocia
nib
vendedores
preço_negócio
id_cliente
clientes
IS A
contém
marca
modelo
lançamento
ténis
sorteia
sorteios
participa
id_sorteio
min_participant
preço_rifa

Pontos relevantes sobre o diagrama

- Para um cliente ser vendedor, ele terá que, obrigatoriamente, fazer parte de, no mínimo, uma relação “negocia”, de modo a diferenciá-lo dos clientes comuns, o que leva a que a relação seja total em ambos os lados.

- A agregação que inclui as entidades pares, trabalhadores, e a relação vende foi criada para garantir que fosse impossível que clientes comprassem pares não vendidos pelos trabalhadores, já que, sem este mecanismo, os clientes teriam que ter uma relação de compra direta com os pares, e não poderíamos assegurar que, por exemplo, um cliente iria comprar um par que já tinha sido adquirido. Isto deve-se ao facto de todas as relações mencionadas serem parciais, o que nos levou a considerar a agregação algo necessário.

Esquema Relacional

- **tenis (<u>modelo</u>, marca, lancamento)**
  - ❖ Os ténis são o nosso “ponto de partida” para a loja, sendo que são caracterizados pelo seu modelo (chave primária), marca, tendo em conta que um modelo só pode corresponder a uma marca, e ano de lançamento. Podem e vão existir diversos pares do mesmo ténis, ou seja, que tenham o mesmo modelo, marca, e ano de lançamento. Pode também não existir nenhum par de um ténis registado na loja.

- **pares (<u>id_tenis</u>, tamanho, preco, condicao, modelo, nif.autenticadores, nif.vendedores, preco_negocio)**
  - ➔ **modelo é chave estrangeira de tenis**
  - ➔ **nif.vendedores é chave estrangeira de vendedores**
  - ➔ **nif.autenticadores é chave estrangeira de autenticadores**
  - ❖ Os pares são os elementos individuais, ou seja, designam cada par da loja, que é único, e que será vendido por um trabalhador da mesma, tendo cada um o seu ID, tamanho, preço de venda (“preco”) na loja e condição (novo ou usado), sendo que todos os pares com estas mesmas características são enumerados no atributo “stock”. O conjunto de entidades “pares” estará associado com o conjunto de entidades “tenis”, sendo “modelo” a chave estrangeira associada. Cada par está no máximo associado a um ténis e tem de ter sempre esta associação, sendo que o contrário não se verifica. Cada par existente na loja necessita de ser legítimo, logo é verificado por um único autenticador, que é identificado por “nif.autenticadores”. Todos os pares existentes na loja foram adquiridos através de negócios com vendedores (identificados por “nif.vendedores”) em que é acordado um preço, o “preco_negocio”.

- **pessoa (<u>nif</u>, telf, nome)**
  - ❖ As pessoas são identificadas pelo seu NIF (“nif”), e têm como atributos o nome (“nome”) e o número de telemóvel (“telf”). As pessoas apresentam duas subcategorias, designadas de clientes e trabalhadores, e têm que obrigatoriamente ser pelo menos uma destas.

- **trabalhadores (<u>nif</u>, id_trab, salario)**

  ➔ nif é chave estrangeira de pessoa

  ❖ Os trabalhadores da loja são identificados pelo seu ID (“id_trab”), e têm como atributos o seu salário, o seu NIF (“nif”), e o nome e contacto telefónico (como atributos da entidade “pessoa”). Os trabalhadores apresentam uma subcategoria, designada de autenticadores.

- **autenticadores (<u>nif</u>, licenca)**

  ➔ nif é chave estrangeira de trabalhadores

  ❖ Os autenticadores são uma subcategoria de trabalhadores, que têm como função analisar a legitimidade de cada par de ténis que os clientes vendedores trazem até à loja, e caso tal se verifique, os autenticadores então compram os pares para poderem ser vendidos na loja, a um preço superior ao de compra para a loja conseguir sempre obter lucro. Os autenticadores são caracterizados pelo seu NIF bem como pela sua licença de autenticação (“licenca”).

- **clientes (<u>nif</u>, id_cliente)**

  ➔ nif é chave estrangeira de pessoa

  ❖ Para uma pessoa ser cliente na loja é necessário que a mesma possua uma ficha/cartão de cliente, que guarda as suas informações de modo a que qualquer cliente possa participar nos sorteios da loja. Assim todo o cliente terá registado o seu ID (“id_cliente”), o seu NIF (“nif”), o seu nome e o seu contacto telefónico (que são atributos da entidade “pessoa”). Um par apenas pode ser vendido no máximo a um cliente, mas cada cliente pode comprar diversos pares. No caso dos sorteios, cada cliente pode participar em 0 ou mais sorteios, só podendo ter uma participação por sorteio. Os clientes podem ainda ser vendedores que negociam os seus pares com os autenticadores.

- **vendedores (<u>nif</u>, nib)**

  ➔ nif é chave estrangeira de clientes

  ❖ Os vendedores são uma sub-categoria dos clientes que negociam os seus pares com os autenticadores para poderem ser vendidos na loja. Apresentam assim, na sua ficha de cliente, para além dos atributos de clientes, também o seu Número de Identificação Bancária (“nib”) para as transações monetárias se realizarem através de uma transferência bancária.

- **sorteios (<u>id_sorteio</u>, preco_rifa, min_participant, estado, id_tenis)**
  ➔ id_tenis é chave estrangeira de pares

  ❖ Os sorteios da loja são realizados com um sistema de rifas, em que o prémio de cada sorteio é um par da loja caracterizado pelo seu respectivo ID (“id_tenis”), sendo que o sorteio terá um número mínimo de participantes (“min_participant”) e cada participante terá de pagar um valor para poder participar no sorteio ficando assim com uma probabilidade de ganhar. À medida que mais clientes se vão juntando ao sorteio, a sua possibilidade de cada participante ganhar vai diminuindo. Os sorteios são identificados pelo seu ID (“id_sorteio”). O atributo “estado” verifica se o sorteio está a decorrer ou já se encontra finalizado.

- **vende (<u>id_tenis</u>, nif)** -> um para muitos, parcial em ambos
  ➔ id_tenis é chave estrangeira de pares
  ➔ nif é chaves estrangeira de trabalhadores

  ❖ Cada par existente na loja pode ou não ser vendido, sendo que a venda é realizada por um único trabalhador, no entanto cada trabalhador pode ou não vender um número indeterminado de pares.

- **compra (<u>id_tenis</u>, nif.trabalhadores, nif.clientes)** -> um para muitos, parcial em ambos
  ➔ id_tenis e nif.trabalhadores são chaves estrangeiras de vende
  ➔ nif.clientes é chave estrangeira de clientes

  ❖ Por cada compra realizada por um cliente, existe uma venda de um par correspondente, tendo em conta que cada cliente pode realizar um número indeterminado de compras. No entanto, cada venda realizada está associada a um único cliente, uma vez que cada par (relativo à venda) é único.

- **participa (<u>nif</u>, <u>id_sorteio</u>)** -> muitos para muitos, parcial para ambos
  ➔ nif é chave estrangeira de clientes
  ➔ id_sorteio é chave estrangeira de sorteios

  ❖ Cada cliente tem a possibilidade de participar nos vários sorteios existentes na loja (podendo ter apenas uma participação por sorteio) e, como indicado anteriormente, cada sorteio pode ter um número indeterminado de participantes.

§ contem (modelo, id_tenis) -> um para muitos, total do lado dos muitos

- modelo é chave estrangeira de tenis
- id_tenis é chave estrangeira de pares

§ verifica (id_tenis, nif) -> um para muitos, total do lado de muitos

- id_tenis é chave estrangeira de pares
- nif é chave estrangeira de autenticadores

§ sorteia (id_tenis, id_sorteio) -> um para um, total para o lado dos sorteios

- id_tenis é chave estrangeira de pares
- id_sorteio é chave estrangeira de sorteios

§ negocia (nif.vendedores, nif.autenticadores, id_tenis, preco_negocio) -> um para muitos, total em ambos

- nif.vendedores é chave estrangeira de vendedores
- nif.autenticadores e id_tenis são chaves estrangeiras de verifica

Dependências funcionais

Tenis
modelo -> marca, lancamento

Pares
id_tenis -> tamanho, preco, condicao, modelo, stock, id_trab, id_cliente, preco_negocio
tamanho, condicao, modelo -> preco

Pares na BCNF:
2ª dependência viola **BCNF** pois a dependência não é trivial, e "tamanho, condicao, modelo" não é superchave de Pares
P1(tamanho, condicao, modelo, preco)
P2(id_tenis, tamanho, condicao, modelo, stock, id_trab, id_cliente, preco_negocio)

Este esquema na BCNF preserva as dependências, e garante que não há redundância, logo não há necessidade de fazer mais alterações.

Pessoa
nif -> telf, nome

Trabalhadores
nif -> id_trab, salario
id_trab -> salario

Trabalhadores na BCNF:
2ª dependência viola **BCNF** pois a dependência não é trivial, e "id_trab" não é superchave de Trabalhadores
T1(id_trab, salario)
T2(nif, id_trab)

Este esquema na BCNF preserva as dependências, e garante que não há redundância, logo não há necessidade de fazer mais alterações.

Autenticadores
nif -> licenca

Clientes
nif -> id_cliente

Vendedores
nif -> nib

Sorteios
id_sorteio -> preco_rifa, min_participant, id_tenis, estado
id_tenis, preco_rifa -> min_participant

Sorteios na BCNF:
2ª dependência viola **BCNF** pois a dependência não é trivial, e “id_tenis, preco_rifa” não é superchave de Sorteios
S1(id_tenis, preco_rifa, min_participant, estado)
S2(id_sorteio, preco_rifa, id_tenis)

Este esquema na BCNF preserva as dependências, e garante que não há redundância, logo não há necessidade de fazer mais alterações.

Vende
id_tenis -> nif

Compra
id_tenis -> nif.trabalhadores, nif.clientes

Participa
nif-> id_sorteio

Após analisar o esquema da base de dados usando a teoria das dependências funcionais, apesar de observarmos algumas alterações possíveis para que o esquema ficasse na forma normal Boyce-Codd, **decidimos mantê-lo inalterado**, uma vez que estas alterações eram mínimas, e o nosso uso de IDs como chaves primárias em quase todas as tabelas faz com que não tenhamos redundância já na primeira criação do esquema. Assim, alterá-lo para a BCNF apenas torná-lo-ia desnecessariamente mais complexo.

O mesmo se aplica à tentativa de conversão para a 3ª Forma Normal, visto que as tabelas obtidas através da conversão para a BCNF foram exatamente iguais às obtidas na conversão para a 3FN, logo, achámos desnecessário apresentar estas últimas.

Alterações relativas à 1ª fase

1. Inserção de uma **entidade de generalização "pessoas"**, que irá representar de forma global todos os clientes e trabalhadores representados na base de dados. Sem esta entidade, teríamos redundância nas tabelas que continham atributos iguais, tais como "nome", "NIF" e "telefone"; e, considerando que podemos ter trabalhadores que sejam clientes e vice-versa, poderia mesmo ocorrer repetição de tuplos.

2. Retirada do **atributo stock**, uma vez que, para o que se pretende, é mais correto obtê-lo realizando uma view que nos apresente uma tabela de pares disponíveis. Exemplo: Dado um determinado modelo de ténis, devolve-nos uma tabela que contenha todos os pares disponíveis correspondentes.

3. Adicionado **novo atributo** à tabela sorteios: **estado** - Representa se um sorteio está a decorrer ou se já terminou. Este atributo permite que os sorteios sejam terminados, e não apenas criados. Desta forma, a leitura da tabela passa a ser mais intuitiva, já que os sorteios deixam de ser algo dispensável na base de dados.

Triggers e Views

View “stock”

```
CREATE OR REPLACE VIEW STOCK AS
  (SELECT PARES.ID_TENIS, TAMANHO, CONDICAO, PRECO
        FROM PARES LEFT OUTER JOIN COMPRA ON (PARES.ID_TENIS = COMPRA.ID_TENIS)
        WHERE NIF_TRABALHA IS NULL)
  MINUS
  (SELECT ID_TENIS, TAMANHO, CONDICAO, PRECO
        FROM PARES INNER JOIN SORTEIOS USING (ID_TENIS)
  WHERE ESTADO = 'TERMINADO');
```

A vista “stock” é implementada visto que é algo bastante importante em qualquer sistema de Base de Dados de uma loja, e o seu objetivo é identificar todos os pares disponíveis na loja, incluindo os seus atributos mais relevantes, ou seja, os pares da loja que ainda não foram vendidos nem sorteados. Para além disso, esta vista será vantajosa quando, por exemplo, for necessário identificar quais os pares de um determinado modelo disponíveis.

View “pares_vend_sort”

```
CREATE OR REPLACE VIEW PARES_VEND_SORT AS
        (SELECT ID_TENIS, MODELO, PRECO
         FROM PARES INNER JOIN COMPRA USING (ID_TENIS))
                UNION
        (SELECT ID_TENIS, MODELO, PRECO
        FROM PARES INNER JOIN SORTEIOS USING (ID_TENIS)
        WHERE ESTADO = 'TERMINADO');
```

A vista “pares_vend_sort” é utilizada sempre que for importante saber quais os pares de ténis que já existiram em stock na loja, bem como as suas características, mas que já foram vendidos ou sorteados anteriormente. Esta vista é utilizada numa função que avalia o stock de um determinado modelo de ténis, e também serve para simplificar as consultas de um funcionário, pois, por exemplo, quando um cliente procura por um determinado par, o funcionário poderá indicar que de momento o par não está disponível, mas que anteriormente já esteve.

View “lucro”

```
CREATE OR REPLACE VIEW LUCRO AS
   WITH DINHEIRO_VENDAS AS
          (SELECT SUM(PRECO) AS DV
           FROM PARES INNER JOIN COMPRA USING (ID_TENIS)),
          DINHEIRO_GASTO AS
          (SELECT -1*SUM(PRECO_NEGOCIO) AS PN
           FROM PARES),
          DINHEIRO_SORTEIOS AS
          (SELECT SUM(PRECO_RIFA) AS PR
           FROM SORTEIOS INNER JOIN PARTICIPA USING (ID_SORTEIO)
           WHERE ESTADO = 'TERMINADO')
          SELECT DV + PN + PR AS LUCROTOTAL
   FROM DINHEIRO_VENDAS, DINHEIRO_GASTO, DINHEIRO_SORTEIOS;
```

A vista “lucro” é bastante relevante no que toca a qualquer loja ou qualquer empresa, uma vez que, ao gerir uma loja, deve-se sempre ter em conta o lucro obtido. O lucro é calculado através da soma do valor obtido em vendas e em sorteios terminados, subtraindo o valor gasto ao comprar os pares para a loja.

Trigger “verifica_preco”

```
CREATE OR REPLACE TRIGGER VERIFICA_PRECO
AFTER INSERT OR UPDATE OF PRECO OR UPDATE OF PRECO_NEGOCIO ON PARES
   DECLARE EXISTE NUMBER;
   BEGIN
      SELECT COUNT(*) INTO EXISTE
      FROM PARES WHERE PRECO < PRECO_NEGOCIO;
      IF EXISTE > 0
         THEN
                  RAISE_APPLICATION_ERROR (-20100, 'O PRECO DO PAR DEVE SER SUPERIOR AO
PRECO_NEGOCIO!');
      END IF;
   END;
/
```

Independentemente da loja, o objetivo é sempre vender produtos a um preço superior ao preço a que os mesmo foram adquiridos, e por isso é necessário garantir que um par de ténis será colocado à venda por um preço superior ao comprado, de modo a que a loja tenha lucro e seja possível sustentar a mesma.

Trigger “min_participant”

```
CREATE OR REPLACE TRIGGER MIN_PARTICIPANT
          BEFORE INSERT ON SORTEIOS
   FOR EACH ROW
   DECLARE PRECONEGOCIADO NUMBER;
   BEGIN
     SELECT PRECO_NEGOCIO INTO PRECONEGOCIADO FROM PARES WHERE ID_TENIS =
:NEW.ID_TENIS;
      :NEW.MIN_PARTICIPANT := PRECONEGOCIADO/:NEW.PRECO_RIFA;
      :NEW.ESTADO := 'A DECORRER';
   END;
/
```

Este trigger servirá para calcular o número mínimo de participantes necessários para um sorteio, de modo a que a loja obtenha lucro sorteando esse par, sendo que será calculado a partir da divisão entre o valor que foi gasto ao obter esse par, e o preço da rifas para o sorteio indicado. Para além disso, este trigger irá alterar o estado do sorteio para “a decorrer”, indicando que o sorteio está disponível para a participação de clientes.

Trigger “end_sorteio”

```
CREATE OR REPLACE TRIGGER END_SORTEIO
   BEFORE UPDATE OF ESTADO ON SORTEIOS
   FOR EACH ROW
   DECLARE NAOATINGIDO NUMBER;

   BEGIN
      SELECT COUNT(*) INTO NAOATINGIDO
      FROM PARTICIPA
      WHERE ID_SORTEIO = :NEW.ID_SORTEIO;

      IF NAOATINGIDO < :NEW.MIN_PARTICIPANT AND :NEW.ESTADO = 'TERMINADO'
         THEN
           RAISE_APPLICATION_ERROR (-20100, 'O SORTEIO NAO PODE SER TERMINADO, UMA VEZ
QUE O NUMERO MINIMO DE PARTICIPANTES NAO FOI ATINGIDO.');
      END IF;
   END;
/
```

Este trigger irá garantir que um sorteio só será declarado como “terminado”, isto é, não aceitando mais participantes, quando o número de participantes for superior ao mínimo de participantes calculado anteriormente.

Trigger “new_par_sort”

```
CREATE OR REPLACE TRIGGER NEW_PAR_SORT
BEFORE INSERT ON SORTEIOS
FOR EACH ROW
DECLARE EXISTE NUMBER;
BEGIN
   SELECT COUNT (*) INTO EXISTE FROM COMPRA WHERE ID_TENIS = :NEW.ID_TENIS;
   IF EXISTE > 0
      THEN
         RAISE_APPLICATION_ERROR (-20100, 'NAO SE PODE ADICIONAR UM PAR JA SORTEADO OU
VENDIDO A ESTA TABELA.');
   END IF;
END;
/
```

É necessário um trigger que verifique se um par que pretendemos sortear não foi vendido anteriormente, pois caso contrário, poderíamos sortear pares que já não existiam na loja.

Trigger “new_par_compra”

```
CREATE OR REPLACE TRIGGER NEW_PAR_COMPRA
BEFORE INSERT ON COMPRA
FOR EACH ROW
DECLARE EXISTE NUMBER;
BEGIN
   SELECT COUNT (*) INTO EXISTE FROM SORTEIOS WHERE ID_TENIS = :NEW.ID_TENIS;
   IF EXISTE > 0
      THEN
         RAISE_APPLICATION_ERROR (-20100, 'NAO SE PODE ADICIONAR UM PAR JA SORTEADO OU
VENDIDO A ESTA TABELA.');
   ELSE
      INSERT INTO VENDE VALUES (:NEW.ID_TENIS, :NEW.NIF_TRABALHA);
   END IF;
END;
/
```

Bem como para o caso dos sorteios, também é necessário verificar se um par que pretendemos vender não foi já sorteado.

Trigger “worker_sell_himself”

```
CREATE OR REPLACE TRIGGER WORKER_SELL_HIMSELF
BEFORE INSERT ON COMPRA
FOR EACH ROW
BEGIN
  IF :NEW.NIF_TRABALHA = :NEW.NIF_CLIENTE
    THEN
      RAISE_APPLICATION_ERROR (-20100, 'O TRABALHADOR NAO PODE VENDER UM PAR A SI MESMO.');
    END IF;
END;
/
```

Visto que um trabalhador da loja pode também ser cliente, este trigger irá garantir que um trabalhador não poderá vender um par de ténis a si mesmo. Assim, antes de se inserir uma compra na tabela respectiva, será verificado se o NIF do trabalhador que está a vender o par é diferente do NIF do cliente que o está a comprar.

Trigger “edit_part_sort”

```
CREATE OR REPLACE TRIGGER EDIT_PART_SORT
BEFORE INSERT OR DELETE OR UPDATE ON PARTICIPA
FOR EACH ROW
DECLARE STATUS VARCHAR2(10);
BEGIN
  SELECT ESTADO INTO STATUS FROM SORTEIOS WHERE ID_SORTEIO = :NEW.ID_SORTEIO OR ID_SORTEIO = :OLD.ID_SORTEIO;
  IF STATUS = 'TERMINADO'
    THEN
      RAISE_APPLICATION_ERROR (-20100, 'NAO E POSSIVEL REALIZAR AS MODIFICACOES PRETENDIDAS UMA VEZ QUE ESTE SORTEIO FOI DECLARADO COMO TERMINADO.');
  END IF;
END;
/
```

Após um sorteio ser encerrado, não deverá ser possível adicionar, atualizar ou remover os participantes na tabela “participa”, visto que, se um sorteio está encerrado. quaisquer alterações neste serão completamente desnecessárias.

Trigger “edit_estado”

```
CREATE OR REPLACE TRIGGER EDIT_ESTADO
BEFORE UPDATE OF ESTADO ON SORTEIOS
FOR EACH ROW
BEGIN
   IF :OLD.ESTADO = 'TERMINADO' AND :NEW.ESTADO = 'A DECORRER'
      THEN
         RAISE_APPLICATION_ERROR (-20100, 'NAO E POSSIVEL REABRIR UM SORTEIO QUE FOI
DECLARADO COMO TERMINADO.');
   END IF;
END;

/
```

Após um sorteio ser terminado, não deverá ser possível alterar o seu estado para a “a decorrer”, visto que recomeçar um sorteio depois do mesmo terminar não faria sentido.

Trigger “checkYear”

```
CREATE OR REPLACE TRIGGER CHECKYEAR
BEFORE INSERT ON TENIS
FOR EACH ROW
DECLARE COMPARATION NUMBER;
BEGIN
   SELECT TO_CHAR(SYSDATE, 'YYYY') - :NEW.LANCAMENTO INTO COMPARATION FROM DUAL;
   IF (COMPARATION < 0)
      THEN
         RAISE_APPLICATION_ERROR (-20100, 'NAO E POSSIVEL INSERIR UM TENIS COM UM ANO DE
LANCAMENTO SUPERIOR AO ANO ATUAL.');
   END IF;
END;
/
```

Para além das verificações que realizamos na criação da tabela ténis, também é necessário avaliar se o ano de lançamento de um novo tuplo que queremos adicionar é ou não válido quando comparado com o ano presente (menor ou igual a este).

Limitações na implementação da base de dados

A nossa base de dados não permite determinar um vencedor de um sorteio. Seria necessário um mecanismo que o fizesse aleatoriamente, o que iria aumentar a complexidade da base de dados para algo que nem está nos nossos objetivos.

Poderíamos ter utilizado a tabela “pares” para armazenamento apenas dos pares de ténis disponíveis, e seria mais fácil de os consultar. No entanto, escolhemos armazenar todos os pares, disponíveis, sorteados, e vendidos, pois esta tabela referência quase todas as outras, e se permitissemos a eliminação de tuplos, tornaria a base de dados menos sólida. Preferimos utilizar vistas um pouco mais complexas para aceder aos pares disponíveis, de forma mais segura, na nossa opinião.

Optámos também por não utilizar uma data de lançamento dos ténis com dia e mês, utilizando apenas o ano, uma vez que, para além de aumentar a complexidade de realização, também é algo que pode ser dependente da loja onde são lançados, o que levaria a alguma discrepância entre as datas.

Consultas interessantes

- **Qual o lucro atual da loja, considerando os pares adquiridos, vendidos e sorteados?**

```
WITH DINHEIRO_VENDAS AS
     (SELECT SUM(PRECO) AS DV
      FROM PARES INNER JOIN COMPRA USING (ID_TENIS)),
  DINHEIRO_GASTO AS
     (SELECT -1*SUM(PRECO_NEGOCIO) AS PN
      FROM PARES),
   DINHEIRO_SORTEIOS AS
     (SELECT SUM(PRECO_RIFA) AS PR
      FROM SORTEIOS INNER JOIN PARTICIPA USING (ID_SORTEIO)
      WHERE ESTADO = 'TERMINADO')
         SELECT DV + PN + PR AS LUCROTOTAL
  FROM DINHEIRO_VENDAS, DINHEIRO_GASTO, DINHEIRO_SORTEIOS;
```

(Utilizámos esta consulta como uma view, dado que saber o lucro da loja é muito importante para a sua gestão)

- **Quais os trabalhadores da loja que também são clientes desta mesma?**

```
SELECT NOME
FROM PESSOA INNER JOIN TRABALHADORES ON (PESSOA.NIF = TRABALHADORES.NIF)
                INNER JOIN CLIENTES ON (PESSOA.NIF = CLIENTES.NIF);
```

- **Quais os pares disponíveis na loja?**

```
(SELECT PARES.ID_TENIS, TAMANHO, CONDICAO, PRECO
          FROM PARES LEFT OUTER JOIN COMPRA ON (PARES.ID_TENIS =
         COMPRA.ID_TENIS)
          WHERE NIF_TRABALHA IS NULL)
                  MINUS
  (SELECT ID_TENIS, TAMANHO, CONDICAO, PRECO
          FROM PARES INNER JOIN SORTEIOS USING (ID_TENIS)
         WHERE ESTADO = 'TERMINADO');
```

(Utilizámos esta consulta como uma view, porque, tal como anteriormente, é muito importante para um funcionário saber o stock atual da loja)

- **Quais as marcas de ténis disponíveis em sorteios?**

```
SELECT DISTINCT MARCA
FROM SORTEIOS INNER JOIN PARES USING (ID_TENIS)
              INNER JOIN TENIS USING (MODELO);
```

- **Quais os trabalhadores que já participaram em sorteios?**

```
WITH CLIENTESTRAB AS (
SELECT PESSOA.NIF, NOME
FROM PESSOA INNER JOIN TRABALHADORES ON (PESSOA.NIF = TRABALHADORES.NIF)
            INNER JOIN CLIENTES ON (PESSOA.NIF = CLIENTES.NIF))
SELECT NOME
FROM PARTICIPA INNER JOIN CLIENTESTRAB USING (NIF);
```

Descrição da interface APEX

A nossa interface APEX contempla 7 páginas diferentes: Inicio, Relatório de ténis, Relatório de pares, Relatório de compras, Relatório de sorteios, Relatório de marcas e Relatório de consultas. A aplicação foi criada utilizando o **servidor local**.

Início:

A página Início é a página de entrada da aplicação onde existem ligações para as páginas subsequentes.

Relatório de ténis:

Nesta página é possível consultar todos os detalhes disponíveis dos ténis que são comercializados pela loja.

Relatório de pares:

A página Relatório de pares fornece todos os detalhes sobre os pares de ténis por vender, vendidos, em sorteio ou já sorteados.

Relatório de compras:

Nesta página é possível observar a listagem de todos os pares vendidos pela loja, e as respectivas pessoas responsáveis por cada uma das transações.

Relatório de sorteios:

Na página Relatório de sorteios é possível constatar todos os sorteios passados ou a decorrer na loja.

Relatório de marcas:

Esta página apresenta uma listagem de todas as marcas de todos os ténis comercializados pela loja.

Relatório de consultas:

Esta página apresenta o resultado das consultas relevantes referidas anteriormente.

Explicação da implementação de funcionalidades

1. **Existência de uma página de entrada onde existam ligações para as páginas subsequentes:**

   Página Início - através da utilização de botões com links para todas as páginas da aplicação.

2. **Listagem de dados da Base de Dados onde códigos referentes a chaves externas sejam substituídos pelo valor de outros atributos de fácil compreensão:**

   Página Relatório de pares - os NIFs (chave estrangeira de autenticadores e de vendedores) são substituídos pelo atributo “nome”.

   Página Relatório de compras - os NIFs (chave estrangeira de trabalhadores e de clientes) são substituídos pelo atributo “nome” e os ID_TENIS (chave estrangeira de pares) são substituídos pelo atributo “modelo”.

   Página Relatório de sorteios - ID_TENIS (chave estrangeira de pares) são substituídos pelo atributo “modelo”.

3. **Listagem de dados da Base de Dados onde sejam apresentados valores derivados:**

   Página Relatório de ténis - a nova coluna “stock” apresenta a disponibilidade dos pares disponíveis por cada modelo (“disponível” ou “esgotado”), calculada através da função “evaluateStock”.

   Página Relatório de sorteios - a nova coluna “participantes” apresenta o número de participantes atuais de cada sorteio, calculado através da função “numParticipantes”.

4. **Possibilidade de inserir, remover e atualizar tuplos da Base de Dados:**

   Página Relatório de sorteios - através da utilização de um report-form, que permite a alteração e remoção dos valores de cada tuplo, para além do botão “Create” na página, que permite a criação de novos tuplos.

5. **Possibilidade de preencher valores de atributos correspondentes a relações sem se ter conhecimento de códigos:**

   Página Relatório de sorteios - na criação ou alteração de um tuplo, utiliza-se uma select list baseada numa LOV com os modelos de ténis disponíveis em loja, para se poder selecionar o par a ser sorteado sem necessitar de conhecer o seu ID_TENIS (chave estrangeira de pares).

6. **Existência de links de navegação (breadcrumbs) nas várias páginas:**

   Em todas as páginas, existe uma barra de navegação no lado esquerdo, com breadcrumbs para possível acesso a todas as páginas da aplicação, em qualquer momento.

7. **Existência de dois reports interligados, onde um apresente detalhes do outro (drill-down):**

   Páginas Relatório de ténis & Relatório de marcas - existe uma select list baseada numa LOV com os nomes de todas as marcas existentes, provenientes do Relatório de Marcas. Ao selecionar uma marca, são apresentados detalhes sobre todos os ténis dessa mesma. Por outro lado, no Relatório de Marcas, é possível selecionar uma marca, que irá abrir o Relatório de Ténis, e apresentar, tal como anteriormente, todos os detalhes sobre todos os ténis da marca selecionada.

8. **Existência de um detalhe condicional:**

   Página Relatório de ténis - na criação de uma região “Valor da marca”, que, ao selecionar uma marca da select list, devolve o valor médio dos preços de cada par desta marca, obtidos através de uma query com a tabela “ténis”.
   (O valor da marca é importante para se ter noção do seu prestígio no mercado).

9. **Existência de um form master-detail, onde seja possível inserir, remover e alterar dados do detail:**

   Como Master, temos o Relatório de Sorteios, onde é possível criar, alterar e remover tuplos, e, ao editar um destes, é apresentado o Detail (Relatório de Participantes) que, no caso, é a lista de participantes do sorteio selecionado, em que é possível fazer alterações relativas a estes.

Implementações adicionais:

1. Existência de uma select list baseada numa LOV com os modelos de todos os pares existentes, em que, ao selecionar uma das opções, são apresentados todos os detalhes dos pares desse modelo.

2. Na alteração de tuplos de sorteios, é possível alterar o estado de um sorteio de “a decorrer” para “terminado” (caso o sorteio esteja nas condições adequadas), através do uso da funcionalidade “radio group”, de modo a facilitar a experiência do utilizador, que não necessita de saber a string a inserir para alterar o estado.

Manual de utilizador

1. **Como utilizar a página de entrada da aplicação?**

    - Clicar na página “Inicio”
    - Na região “Navegação”, escolher qual das páginas pretende aceder

2. **Onde e como observar a substituição de atributos relacionados a chave externas, que foram alterados para melhor compreensão do utilizador?**

    Alteração dos atributos “NIF” para “nome”:

    - Aceder à página “Relatório de pares”
    - Nas colunas “Autenticado por” e “Comprado a”, em vez de se observar o NIF dos autenticadores e vendedores, observam-se os seus respetivos nomes
    - Aceder à página “Relatório de compras”
    - Nas colunas “Vendido por” e “Comprado por”, em vez de se observar o NIF dos trabalhadores e clientes, observam-se os seus respetivos nomes

    Alteração dos atributos “ID_TENIS” para “modelo”:

    - Aceder à página “Relatório de compras”
    - Na coluna “Modelo”, em vez de se observar o ID de cada um dos pares, observam-se os seus respetivos modelos
    - Aceder à página “Relatório de sorteios”
    - Na coluna “Modelo”, em vez de se observar o ID de cada um dos pares, observam-se os seus respetivos modelos

3. **Onde e como observar valores derivados?**

    - Aceder à página “Relatório de ténis”
    - Observar a coluna nomeada “Stock”
    - Aceder à página “Relatório de sorteios”
    - Observar a coluna nomeada “Participantes”

4. **Onde e como inserir, remover e atualizar tuplos na aplicação?**

    - Aceder à página “Relatório de sorteios”
    - Para criar um novo sorteio, clicar no botão “Create” e inserir o preço desejado para cada rifa, selecionar o modelo que pretende sortear, e finalmente clicar no botão “Create” de novo
    - Para remover um sorteio existente, clicar no ícone do lápis no tuplo que pretende remover e, na região “Sorteios”, clicar no botão “Delete”
    - Para editar um sorteio, na página “Relatório de sorteios”, clicar no ícone do lápis no tuplo que pretende editar e, na região “Sorteios” é possível modificar o preço da rifa, alterar o estado do sorteio, e mudar o modelo que está a sortear. Finalmente, clicar no botão “Save”

5. **Onde e como preencher valores de atributos correspondentes a chaves externas, sem se ter conhecimento de códigos internos?**

    - Aceder à página “Relatório de sorteios”;
    - Clicar no botão “Create” e observar a lista de modelos de ténis que permite selecionar o modelo, sem conhecer o seu ID
    - Clicar no ícone do lápis no tuplo que se pretende modificar e, na região “Sorteios”, observar a lista de modelos de ténis que permite selecionar o modelo, sem conhecer o seu ID

6. **Onde e como verificar a existência de links de navegação?**

    - Em qualquer página da aplicação, existe uma barra de navegação azul do lado esquerdo do ecrã
    - Selecionar o nome da página que pretende aceder

7. **Onde e como verificar a existência de dois reports (1 e 2) interligados a partir de um drill-down?**

    - Aceder à página “Relatório de ténis”
    - Na região “Marcas”, escolher a marca de ténis na lista que pretende observar
    - Em baixo, irá aparecer um report (1) que contém todos os modelos de ténis dessa marca existentes ou que já existiram na loja
    - Aceder à página “Relatório de marcas” e clicar numa das marcas apresentadas no report (2). Será redirecionado para a página “Relatório de ténis” que irá apresentar o report (1) de todos os modelos correspondentes à marca escolhida

8. **Onde e como observar um detalhe condicional na aplicação?**

    - Aceder à página “Relatório de ténis”
    - Selecionar uma marca na lista da região “Marcas”, e observar na região “Valor da marca” o valor correspondente ao detalhe condicional

9. **Onde e como observar a existência de um form master-detail?**

    - Aceder à página “Relatório de sorteios”
    - Relativamente ao Master, este corresponde à tabela de sorteios apresentada. Ao clicar no ícone do lápis de um tuplo, na região “Sorteios” é possível realizar inserções, alterações ou remoções nos tuplos de sorteios
    - Relativamente ao Detail, clicar no ícone do lápis de um tuplo que pretenda modificar e poderá observar a região “Participantes”, onde é possível observar uma tabela correspondente a todos os participantes do sorteio selecionado, sendo possível realizar alterações nos tuplos de participantes

Implementações adicionais:

1. Aceder à página “Relatório de pares”
   Na região “Modelos”, selecionar na lista um modelo de ténis, e observar a tabela de todos os pares desse modelo, com os detalhes correspondentes

2. Aceder à página “Relatório de sorteios”
   Clicar no ícone do lápis de um tuplo que pretenda editar, e alterar o estado do sorteio como desejar, através dos botões de seleção “a decorrer” ou “terminado”