FIAP GRADUAÇÃO

Oracle com asociações

- Introdução
- Problema
- Exercícios

Introdução

- trabalhamos com uma única entidade no tópico passado
- criamos as funções para inserir, atualizar, apagar e consultar essa entidade
- agora vamos trabalhar em situações que devemos atualizar duas tabelas associadas
- vamos apresentar a solução usando o exemplo entre cliente, venda e itens de venda.

Problema

Imagine uma funcionalidade do sistema onde temos que registrar uma venda. Note que, uma venda sempre esta associada a um cliente, ou seja, na tabela do banco o id do cliente precisa ser gravada como uma fk na tabela venda. Como regra de negócio, vamos imaginar que se o cliente não existe devemos inserir ele no banco de dados primeiro e depois inserir a venda e os itens de venda. Caso o cliente já exista, apenas cadastramos a venda e seus respectivos itens.

```
1       def insere_venda(cliente: dict, venda: dict):
```

Observe que dentro do dicionário venda temos uma lista de dicionários representando os itens de venda.

Banco de dados do Problema

Considere a seguinte tabela para o cliente:

```
1  create table t_cliente(
2      id number generated by default as identity,
3      nome varchar2(100),
4      documento varchar2(30),
5      email varchar2(50),
6      primary key(id)
7  );
```

E a tabela venda:

```
1  create table t_venda(
2      id number generated by default as identity,
3      valor number(8, 2),
4      data date,
5      id_cliente number,
6      primary key(id),
7      foreign key(id_cliente) references t_cliente(id)
8  );
```

Banco de dados do Problema 2

E a tabela itemvenda:

```
1  create table t_itemvenda(
2      id number generated by default as identity,
3      produto varchar2(100),
4      quantidade number,
5      valor number(8, 2),
6      id_venda number,
7      primary key(id),
8      foreign key(id_venda) references venda(id)
9  );
```

Considerações

- veja como criamos as chaves primárias das tabelas
- o Oracle criará automaticamente os ids dos registros da tabela
- contudo, vamos precisar desses ids para inserirmos os registros de venda e os de itemvenda
- ou seja, o modo como fazemos o insert no banco de dados também terá que ser modificado
- fique atento também a como lidaremos com a coluna date do banco de dados

Funcionalidade em Python

```
1  import banco
2
3  def cadastra_venda(cliente: dict, venda: dict):
4      cli = banco.recupera_cliente_documento(cliente['
           documento'])
5      if not cli:
6          banco.insere_cliente(cliente)
7          venda['id_cliente'] = cliente['id']
8      else:
9          venda['id_cliente'] = cli[0]
10     itens_venda = venda['itens']
11     venda.pop('itens')
12     banco.insere_venda(venda)
13     idvenda = venda['id']
14     for iv in itens_venda:
15         iv['id_venda'] = idvenda
16         banco.insere_itemvenda(iv)
```

Veja que esta função representa uma funcionalidade da aplicação contendo as regras de negócio levantadas na análise de requisitos. Vamos chamar este módulo de negocio.py

Módulo banco — funções de acesso ao banco

Vamos implementar a função que recupera um cliente pelo documento

```
1  import oracledb
2
3  def recupera_cliente_documento(doc_valor: str):
4      sql = '''select id, nome, email, documento from
           t_cliente where documento = :documento'''
5      with get_conexao() as conn:
6          with conn.cursor() as cur:
7              cur.execute(sql, {"documento": doc_valor})
8              return cur.fetchone()
```

Estou considerando que existe uma função para pegar a conexão e implementei a versão mais curta para recuperar um registro do banco de dados.

Módulo banco — funções de acesso ao banco

Nas funções que inserem a venda e o cliente, precisamos recuperar o id gerado pelo banco Oracle e colocar esse id nos respectivos dicionários, veja a implementação:

```
1  def insere_cliente(cliente: dict):
2      ins = '''insert into t_cliente(nome, email, documento)
           values(:nome, :email, :documento) returning id into
           :id'''
3      with get_conexao() as conn:
4          with conn.cursor() as cur:
5              new_id = cur.var(oracledb.NUMBER)
6              #colocando a variavel no dicionario
7              cliente['id'] = new_id
8              cur.execute(ins, cliente)
9              #colocando o VALOR da variavel no dicionario
10             cliente['id'] = new_id.getvalue()[0]
11         conn.commit()
```

Quando forem implementar o `insere_venda` e o `insere_itemvenda` percebam que a estrutura será muito parecida com o `insere_cliente`. Basicamente mudará o comando insert.

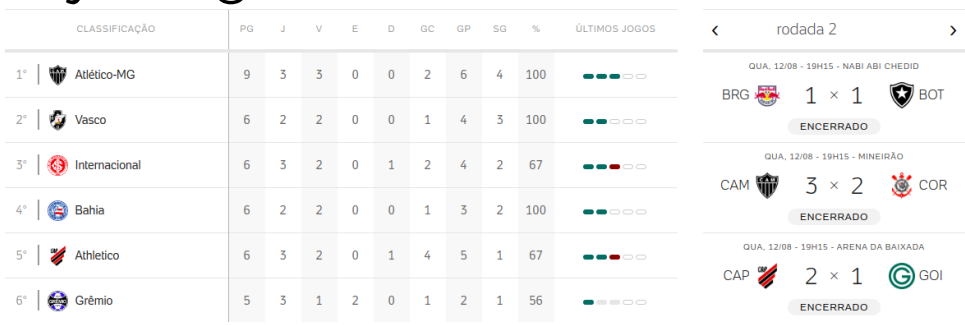
Módulo banco — consulta

Mostrarei como fazer uma consulta usando o join do SQL, vamos supor que queremos mostrar todas as vendas e itens venda do banco de dados.

```
1
2  def consulta_venda(cli_id: int):
3      sql = '''select v.id, v.data, v.valor, iv.produto, iv.
           quantidade, iv.valor from t_venda v join t_itemvenda
            iv on v.id=iv.id_venda where v.id_cliente = :
           id_cliente'''
4      with get_conexao() as conn:
5          with conn.cursor() as cur:
6              cur.execute(ins, {"id_cliente": cli_id})
7              return cur.fetchall()
```

Exercícios

1. Implemente as funções não implementadas de `cadastra_venda` no banco.py.
2. Crie em negocio.py uma função para chamar a função `consulta_venda` do banco.py. Faça testes para verificar o funcionamento desta função e da `cadastra_venda`
3. Veja a figura abaixo:



CLASSIFICAÇÃO	PG	J	V	E	D	GC	GP	SG	%	ÚLTIMOS JOGOS
1° Atlético-MG	9	3	3	0	0	2	6	4	100	
2° Vasco	6	2	2	0	0	1	4	3	100	
3° Internacional	6	3	2	0	1	2	4	2	67	
4° Bahia	6	2	2	0	0	1	3	2	100	
5° Athletico	6	3	2	0	1	4	5	1	67	
6° Grêmio	5	3	1	2	0	1	2	1	56	

Defina as tabelas de banco de dados para Time e Partida, implemente a função `cadastra_partida` que cadastra uma partida no banco de dados. Caso os times não existam ainda no banco, você deverá criá-los dentro desta função. Não esqueça de atualizar as informações de times.

Exercícios

4. Crie uma outra função que permite mostrar a classificação dos times como a imagem acima está exibindo.
5. Faça um programa em Python que cadastra algumas partidas e mostra os resultados dos times após o cadastro dessas partidas. Como na tabela de classificação da figura do eslaide anterior.