Pontifícia Universidade Católica do Paraná

Henrique Poledna

Mapeamento Objeto Relacional com o ORM SQLAlchemy

Curitiba

[2024]

Pontifícia Universidade Católica do Paraná

Henrique Poledna

Mapeamento Objeto Relacional com o ORM SQLAlchemy

TDE apresentado ao Curso de Cibersegurança da Pontifícia Universidade Católica do Paraná.

Orientador: Rodrigo da Silva do Nascimento

Curitiba

[2024]

1 Introdução

Neste trabalho foi desenvolvido um banco de dados utilizando SQLAlchemy/SQLite. O conceito básico do sistema é que um cliente informa qual é a sua comida favorita, escolhe um dia do mês e recebe esse item todos os meses na data selecionada.

2 Descrição textual das entidades e atributos

2.1 Entidade: Clientes

Tabela: users

Descrição: Esta tabela armazena informações sobre os clientes.

Atributos:

- **id** (`Integer`, `Primary Key`): Identificador único do cliente. Serve como chave primária para a tabela.
- **nome** (`String(50)`): Nome completo do cliente. O valor máximo permitido é 50 caracteres.
- **comida_favorita** (`String(60)`): Comida favorita do cliente. O valor máximo permitido é 60 caracteres.
- **idade** (`Integer`): Idade do cliente.
- **dia_todo_mes_recebe** (`Integer`): Dia do mês em que o cliente recebe a entrega. Pode representar o dia específico do mês (de 1 a 31).
- **endereco_completo** (`String(50)`): Endereço completo do cliente. O valor máximo permitido é 50 caracteres.

2.2 Entidade: Estoque

Tabela: alimento_que_recebe

Descrição: Esta tabela contém informações sobre o estoque de alimentos disponíveis na empresa.

Atributos:

- **id** (`Integer`, `Primary Key`): Identificador único do item de estoque. Serve como chave primária para a tabela.
- **comida_estoque** (`String(60)`): Nome do alimento que está em estoque. O valor máximo permitido é 60 caracteres.
- **quantidade** (`Integer`): Quantidade disponível do alimento em estoque.

2.3 Entidade: Entrega

Tabela: batatinha_em_casa

Descrição: Esta tabela armazena informações sobre as entregas realizadas.

Atributos:

- **id** (`Integer`, `Primary Key`): Identificador único da entrega. Serve como chave primária para a tabela.
- **cliente_id** (`Integer`, `Foreign Key`): Identificador do cliente que receberá a entrega. Esta é uma chave estrangeira que referencia a coluna `id` da tabela `users` (Clientes).
- **data_entrega_escolhida** (`Integer`): Data escolhida para a entrega. Pode ser representada como um número inteiro que indica o dia do mês em que a entrega deve ocorrer.
- **endereco_entrega** (`String(100)`): Endereço para onde a entrega deve ser feita. O valor máximo permitido é 100 caracteres.

2.4 Entidade: Fornecedor

Tabela: fornecedores

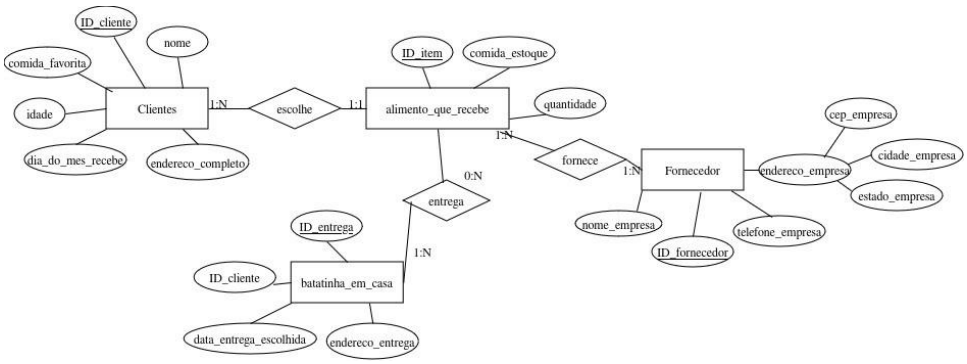
Descrição: Esta tabela contém informações sobre os fornecedores da empresa.

Atributos:

- **id** (`Integer`, `Primary Key`): Identificador único do fornecedor. Serve como chave primária para a tabela.
- **nome_empresa** (`String(100)`): Nome da empresa fornecedora. O valor máximo permitido é 100 caracteres.
- **telefone_empresa** (`String(20)`): Número de telefone da empresa fornecedora. O valor máximo permitido é 20 caracteres.
- **endereco_empresa** (`String(100)`): Endereço da empresa fornecedora. O valor máximo permitido é 100 caracteres.
- **cidade_empresa** (`String(50)`): Cidade onde a empresa fornecedora está localizada. O valor máximo permitido é 50 caracteres.
- **estado_empresa** (`String(50)`): Estado onde a empresa fornecedora está localizada. O valor máximo permitido é 50 caracteres.
- **cep_empresa** (`String(10)`): Código postal (CEP) da empresa fornecedora. O valor máximo permitido é 10 caracteres.

3 Diagrama

Imagem



4 Implementação em Python

O link para o acesso direto ao código é este: <https://github.com/poledna0/sqlalchemy_TDE/blob/main/main.py>.

A seguir, são apresentadas as partes em que são declaradas as entidades e seus respetivos atributos:

```
class Clientes(Base_banc):
    __tablename__ = 'users'
    id = Column(Integer, primary_key=True)
    nome = Column(String(50))
    comida_favorita = Column(String(60))
    idade = Column(Integer)
    dia_todo_mes_recebe = Column(Integer)
    endereco_completo = Column(String(50))

class Estoque(Base_banc):
    __tablename__ = 'alimento_que_recebe'
    id = Column(Integer, primary_key=True)
    comida_estoque = Column(String(60))
    quantidade = Column(Integer)

class Entrega(Base_banc):
    __tablename__ = 'batatinha_em_casa'
    id = Column(Integer, primary_key=True)
    cliente_id = Column(Integer, ForeignKey('users.id'))
    data_entrega_escolhida = Column(Integer)
    endereco_entrega = Column(String(100))
    cliente = relationship("Clientes")
```

```
class Fornecedor(Base_banc):
    __tablename__ = 'fornecedores'
    id = Column(Integer, primary_key=True)
    nome_empresa = Column(String(100))
    telefone_empresa = Column(String(20))
    endereco_empresa = Column(String(100))
    cidade_empresa = Column(String(50))
    estado_empresa = Column(String(50))
    cep_empresa = Column(String(10))
```

Essa foi a escolha dos atributos e das entidades para o sistema, durante minha pesquisa vi muitas pessoas recomendando o uso dessa função para cada entidade, assim melhorando a leitura no Terminal das informações,

```
  def __repr__(self):
      return (f"<Fornecedor(id={self.id}, nome_empresa={self.nome_empresa}, 
telefone_empresa={self.telefone_empresa}, "
              f"endereco_empresa={self.endereco_empresa}, 
cidade_empresa={self.cidade_empresa}, "
              f"estado_empresa={self.estado_empresa}, 
cep_empresa={self.cep_empresa})>")
```

Esse é um dos exemplos para a adição de informaçao a cada tabela:

```
estoques = [
    Estoque(comida_estoque='batata', quantidade=123),
    Estoque(comida_estoque='bis', quantidade=23),
    Estoque(comida_estoque='tomate', quantidade=10),
    Estoque(comida_estoque='temaki', quantidade=42),
    Estoque(comida_estoque='feijão', quantidade=765),
    Estoque(comida_estoque='farinha de trigo', quantidade=29),
    Estoque(comida_estoque='cerveja', quantidade=92)
]
```

Esse é um exemplo aprendido para consulta de informação, nesse exemplo eu chamo a mari e mostro as informações preenchidas na tabela dos clientes:

```
consulta_usuario = sessao.query(Clientes).filter_by(nome='Mari').first()
print(consulta_usuario)

for instance in sessao.query(Clientes).order_by(Clientes.id):
    print(instance.nome, instance.comida_favorita, instance.idade)
```

Agora um exemplo de atualização de informação, o usado nesse exmeplo é o usuario pedro:

```
usuario_atualiza = sessao.query(Clientes).filter_by(nome='Pedro').first()
usuario_atualiza.comida_favorita = "sushi"
usuario_atualiza.idade = 26
sessao.commit()
```

agora por último deletar um usuário presente no banco:

```
pessoa_delet = sessao.query(Clientes).filter_by(nome='Henrique').first()
sessao.delete(pessoa_delet)
sessao.commit()
```