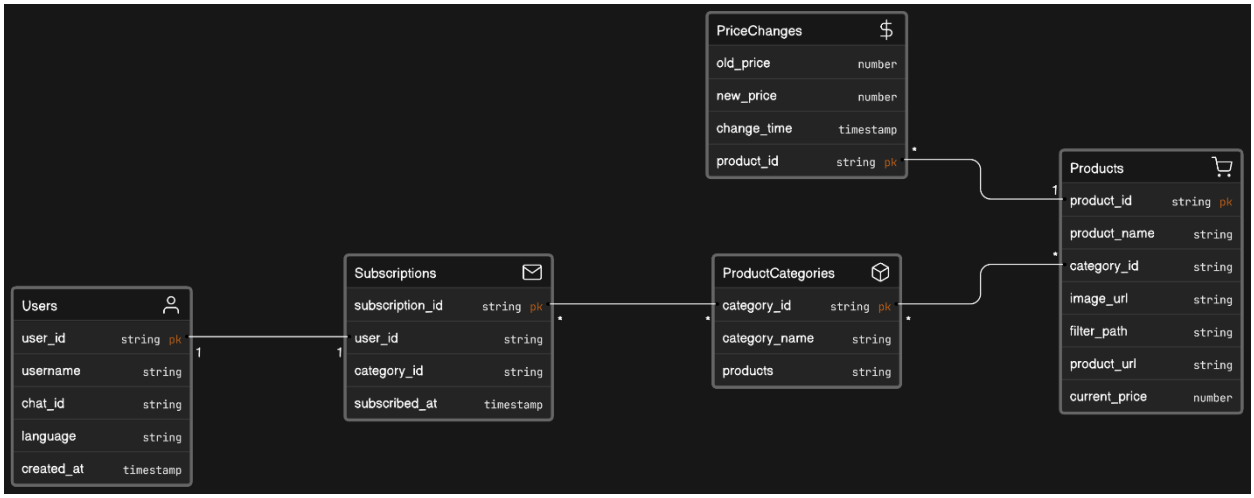
Лабораторная работа 9
Гунькин Денис П3А

Тема: Python и БД. ORM
Цель: освоение концепции ORM
Задача. Для своей схемы базы данных:
- инициализировать проект с использованием SQLAlchemy.
- определить модели для представления различных сущностей
- произвести добавление, редактирование и удаление данных
- выполнить запросы на выборку данных из базы данных
Отчет должен содержать
1 Текст задания
2 Схема базы данных
з. Листинг файла .py
4. Скриншоты с результатами выполнения запросов

1. Схема связей таблиц БД:



Листинг файла main.py

```
from sqlalchemy import create_engine, Column, Integer, String, ForeignKey, Float,
DateTime
from sqlalchemy.orm import declarative_base, relationship, Session
from datetime import datetime

Base = declarative_base()

class User(Base):
```

```
    __tablename__ = 'users'

    user_id = Column(Integer, primary_key=True, autoincrement=True)
    username = Column(String(255))
    chat_id = Column(Integer)
    language = Column(String(255))
    created_at = Column(DateTime)

    subscriptions = relationship("Subscription", back_populates="user")


class ProductCategory(Base):
    __tablename__ = 'product_categories'

    category_id = Column(Integer, primary_key=True, autoincrement=True)
    category_name = Column(String(255))

    products = relationship("Product", back_populates="category")
    subscriptions = relationship("Subscription", back_populates="category")


class Subscription(Base):
    __tablename__ = 'subscriptions'

    subscription_id = Column(Integer, primary_key=True, autoincrement=True)
    user_id = Column(Integer, ForeignKey('users.user_id'))
    category_id = Column(Integer, ForeignKey('product_categories.category_id'))
    subscribed_at = Column(DateTime)

    user = relationship("User", back_populates="subscriptions")
    category = relationship("ProductCategory", back_populates="subscriptions")


class PriceChange(Base):
    __tablename__ = 'price_changes'

    product_id = Column(Integer, ForeignKey('products.product_id'),
primary_key=True)
    old_price = Column(Float)
    new_price = Column(Float)
    change_time = Column(DateTime)

    product = relationship("Product", back_populates="price_changes")


class Product(Base):
    __tablename__ = 'products'

    product_id = Column(Integer, primary_key=True, autoincrement=True)
    product_name = Column(String(255))
    category_id = Column(Integer, ForeignKey('product_categories.category_id'))
```

```
    image_url = Column(String(255))
    filter_path = Column(String(255))
    product_url = Column(String(255))
    current_price = Column(Float)

    category = relationship("ProductCategory", back_populates="products")
    price_changes = relationship("PriceChange", back_populates="product")


engine = create_engine('postgresql://admin:admin@localhost:5432/telegram')
Base.metadata.create_all(engine)

session = Session(engine)

new_user = User(username='JohnDoe', chat_id=12345, language='English',
created_at=datetime.now())
session.add(new_user)
session.commit()

users = session.query(User).all()
users = session.query(User).all()
print("Users:")
for user in users:
    print(f"User ID: {user.user_id}, Username: {user.username}, Chat ID:
{user.chat_id}, Language: {user.language}, Created At: {user.created_at}")

# Вывод данных из таблицы 'product_categories'
categories = session.query(ProductCategory).all()
print("\nProduct Categories:")
for category in categories:
    print(f"Category ID: {category.category_id}, Category Name:
{category.category_name}")

# Вывод данных из таблицы 'subscriptions'
subscriptions = session.query(Subscription).all()
print("\nSubscriptions:")
for subscription in subscriptions:
    print(f"Subscription ID: {subscription.subscription_id}, User ID:
{subscription.user_id}, Category ID: {subscription.category_id}, Subscribed At:
{subscription.subscribed_at}")
users = session.query(User).filter_by(language='English').all()
for user in users:
    print(f"Username: {user.username}, Language: {user.language}")

user_to_update = session.query(User).filter_by(username='JohnDoe').first()
if user_to_update:
    user_to_update.language = 'French'
    session.commit()

user_to_delete = session.query(User).filter_by(username='JohnDoe').first()
if user_to_delete:
```

```
    session.delete(user_to_delete)
    session.commit()
```

Результат в консоли:

Users:

User ID: 1, Username: user1, Chat ID: 1001, Language: English, Created At: 2023-01-01 10:00:00

User ID: 2, Username: user2, Chat ID: 1002, Language: Russian, Created At: 2023-01-02 12:30:00

User ID: 3, Username: user3, Chat ID: 1003, Language: Spanish, Created At: 2023-01-03 15:45:00

User ID: 4, Username: user4, Chat ID: 1004, Language: French, Created At: 2023-01-08 16:30:00

User ID: 5, Username: user5, Chat ID: 1005, Language: German, Created At: 2023-01-09 08:00:00

User ID: 6, Username: user6, Chat ID: 1006, Language: Italian, Created At: 2023-01-10 10:30:00

User ID: 7, Username: user7, Chat ID: 1007, Language: Spanish, Created At: 2023-01-20 11:00:00

User ID: 8, Username: user8, Chat ID: 1008, Language: French, Created At: 2023-01-21 13:30:00

User ID: 9, Username: user9, Chat ID: 1009, Language: English, Created At: 2023-01-27 11:00:00

User ID: 10, Username: user10, Chat ID: 1010, Language: German, Created At: 2023-01-28 13:30:00

User ID: 13, Username: JohnDoe, Chat ID: 12345, Language: English, Created At: 2023-11-20 20:10:40.051354

Product Categories:

Subscriptions:

Subscription ID: 1, User ID: 1, Category ID: 1, Subscribed At: 2023-01-05 09:30:00

Subscription ID: 2, User ID: 2, Category ID: 2, Subscribed At: 2023-01-06 11:45:00

Subscription ID: 4, User ID: 4, Category ID: 4, Subscribed At: 2023-01-11 12:00:00

Subscription ID: 5, User ID: 5, Category ID: 5, Subscribed At: 2023-01-12 14:45:00

Username: user1, Language: English

Username: user9, Language: English

Username: JohnDoe, Language: English