ORM (sqlalchemy) II dalis - ryšiai

Many To One

(daug tėvų turi po vieną vaiką. Atvirkščiai - kiekvienas vaikas gali turėti daug tėvų)

```
from sqlalchemy import Column, Integer, String, ForeignKey, create_engine
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy.orm import relationship, sessionmaker

engine = create_engine('sqlite:///many2one_test.db')
Base = declarative_base()


class Tevas(Base):
    __tablename__ = "tevas"
    id = Column(Integer, primary_key=True)
    vardas = Column("Vardas", String)
    pavarde = Column("Pavardė", String)
    vaikas_id = Column(Integer, ForeignKey('vaikas.id'))
    vaikas = relationship("Vaikas")


class Vaikas(Base):
    __tablename__ = "vaikas"
    id = Column(Integer, primary_key=True)
    vardas = Column("Vardas", String)
    pavarde = Column("Pavardė", String)
    mokymo_istaiga = Column("Mokymo įskaita", String)

Base.metadata.create_all(engine)
```

Kaip sukurti sesiją su sukurta DB

(engine importuoti iš sqlalchemy ORM klasės, jei ji kitame faile)

```
Session = sessionmaker(bind=engine)
session = Session()
```

Kaip įrašyti tėvą ir jo vaiką (Crud)

```
vaikas = Vaikas(vardas="Vaikas", pavarde="Tevaika", mokymo_istaiga = "Čiurlionio
gimnazija")
tevas = Tevas(vardas="Tevas", pavarde="Tevaika", vaikas=vaikas)
session.add(tevas)
session.commit()
```

Kaip pakeisti tėvo ar vaiko duomenis (cRUd)

Priskirti naują vaiką:

```
vaikas = Vaikas(vardas="Naujas vaikas", pavarde="Tevaika")
tevas = session.query(Tevas).get(1)
tevas.vaikas = vaikas
session.commit()
```

Pakeisti tėvo vaiko duomenis:

```
tevas = session.query(Tevas).get(1)
tevas.vaikas.pavarde = "Naujapavardaitis"
session.commit()
```

Kaip ištrinti tėvą (cruD)
(vaikas lieka, nes nenustatytas "cascade" trynimas)

```
tevas = session.query(Tevas).get(1)
session.delete(tevas)
session.commit()
```

One To Many
(kiekvienas tėvas gali turėti po daug vaikų. Atvirkščiai - kiekvienas vaikas gali turėti tik vieną tėvą)

```
from sqlalchemy import Column, Integer, String, Float, ForeignKey, create_engine
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy.orm import relationship, sessionmaker

engine = create_engine('sqlite:///one2many_test.db')
Base = declarative_base()


class Tevas(Base):
    __tablename__ = "tevas"
    id = Column(Integer, primary_key=True)
    vardas = Column("Vardas", String)
    pavarde = Column("Pavardė", String)
    vaikai = relationship("Vaikas")


class Vaikas(Base):
    __tablename__ = "vaikas"
    id = Column(Integer, primary_key=True)
    vardas = Column("Vardas", String)
    pavarde = Column("Pavardė", String)
    mokymo_istaiga = Column("Mokymo įskaita", String)
    tevas_id = Column(Integer, ForeignKey("tevas.id"))
    tevas = relationship("Tevas")


Base.metadata.create_all(engine)
```

Kaip sukurti sesiją su sukurta DB
(engine importuoti iš sqlalchemy ORM klasės, jei ji kitame faile)

```
Session = sessionmaker(bind=engine)
session = Session()
```

Kaip įrašyti tėvą ir jo vaikus (Crud)

```
vaikas = Vaikas(vardas="Vaikas", pavarde="Vaikaitis")
vaikas2 = Vaikas(vardas="Vaikas 2", pavarde="Vaikaitis 2")
tevas = Tevas(vardas="Tevas", pavarde="Vaikaitis")
tevas.vaikai.append(vaikas)
```

```
tevas.vaikai.append(vaikas2)
session.add(tevas)
session.commit()
```

Kaip nuskaityti tėvo vaikus (cRud)

```
tevas = session.query(Tevas).get(1)
for vaikas in tevas.vaikai:
    print(vaikas.vardas, vaikas.pavarde)
```

Kaip redaguoti tėvo vaikus (crUd)

redaguojamas pirmo (0) vaiko vardas:

```
tevas = session.query(Tevas).get(1)
tevas.vaikai[0].vardas = "Vaikas 1"
session.commit()
```

Kaip gauti vaiko tėvą

```
vaikas = session.query(Vaikas).get(1)
print(vaikas.tevas.vardas)
```

Kaip ištrinti tėvo vaiką

(vaikas lieka, nes nenustatytas "cascade" trynimas, tik be tėvo ID)

```
tevas = session.query(Tevas).get(1)
vaikas1 = tevas.vaikai[0]
tevas.vaikai.remove(vaikas1)
session.commit()
```

Many To Many

(kiekvienas tėvas gali turėti po daug vaikų. Kiekvienas vaikas gali turėti po daug tėvų)

```
from sqlalchemy import Column, Integer, String, Float, ForeignKey, create_engine,
Table
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy.orm import relationship, sessionmaker

engine = create_engine('sqlite:///many2many_test.db')
Base = declarative_base()

association_table = Table('association', Base.metadata,
    Column('tevas_id', Integer, ForeignKey('tevas.id')),
    Column('vaikas_id', Integer, ForeignKey('vaikas.id'))
)

class Tevas(Base):
    __tablename__ = 'tevas'
    id = Column(Integer, primary_key=True)
    vardas = Column("Vardas", String)
    pavarde = Column("Pavardė", String)
    vaikai = relationship("Vaikas", secondary=association_table,
back_populates="tevai")

class Vaikas(Base):
    __tablename__ = 'vaikas'
    id = Column(Integer, primary_key=True)
```

```
    vardas = Column("Vardas", String)
    pavarde = Column("Pavardė", String)
    tevai = relationship("Tevas", secondary=association_table,
back_populates="vaikai")

Base.metadata.create_all(engine)
```

Kaip sukurti sesiją su sukurta DB

(engine importuoti iš sqlalchemy ORM klasės, jei ji kitame faile)

```
Session = sessionmaker(bind=engine)
session = Session()
```

Kaip pridėti daug tėvų su daug vaikų

```
tevas1 = Tevas(vardas="Tėvas", pavarde="Tėvaika")
tevas2 = Tevas(vardas="Motina", pavarde="Tevienė")
vaikas1 = Vaikas(vardas="Vaikas", pavarde="Tėvaika")
vaikas2 = Vaikas(vardas="Vaikė", pavarde="Tėvaikytė")

tevas1.vaikai.append(vaikas1)
tevas2.vaikai.append(vaikas1)
tevas2.vaikai.append(vaikas2)

session.add(tevas1)
session.add(tevas2)
session.commit()
```

Kaip peržiūrėti susijusius įrašus

Kaip gauti visus tėvo vaikus:

```
tevas = session.query(Tevas).get(2)
for vaikas in tevas.vaikai:
    print(vaikas.vardas, vaikas.pavarde)
```

Kaip gauti visus vaiko tėvus:

```
vaikas = session.query(Vaikas).get(1)
for tevas in vaikas.tevai:
    print(tevas.vardas, tevas.pavarde)
```

Kaip pakeisti tėvo vaiko įrašą:

```
tevas = session.query(Tevas).get(2)
tevas.vaikai[0].vardas = "Vaikas 1"
session.commit()
```

Kaip ištrinti tėvo vaiko įrašą:

```
tevas = session.query(Tevas).get(2)
vaikas1 = tevas.vaikai[0]
tevas.vaikai.remove(vaikas1)
session.commit()
```