Класс base общий для всех моделей создается отдельно в фале database

class Base(DeclarativeBase):  
 pass

from sqlalchemy.orm import Mapped, mapped\_column  
from database import Base

class WorkersOrm(Base):  
 \_\_tablename\_\_ = 'workers'  
  
 id: Mapped[int] = mapped\_column(primary\_key=True)  
 username: Mapped[str]

Создание таблиц и добавление новой информации

from database import engine, session\_factory, Base  
from models import WorkersOrm  
  
  
def create\_tables():  
 engine.echo = False  
 Base.metadata.drop\_all(engine)  
 Base.metadata.create\_all(engine)  
 engine.echo = True  
  
def insert\_data():  
 worker\_bobr = WorkersOrm(username="Bobr")  
 worker\_volk = WorkersOrm(username="Volk")  
 with session\_factory() as session:  
 session.add\_all([worker\_bobr, worker\_volk])  
 session.commit()

class ResumeOrm(Base):  
 \_\_tablename\_\_ = 'resumes'  
  
 id: Mapped[int] = mapped\_column(primary\_key=True)  
 title: Mapped[str]  
 compensation: Mapped[Optional[int]

workload: Mapped[Workload]

Варианты, как можно сделать поле, которое может быть пустым

1) compensation: Mapped[int | None]

2) compensation: Mapped[int] = mapped\_column(nullable=True)

3) from typing import Optional

compensation: Mapped[Optional[int]]

Для создания вариантов используем enum, создаем класс, который наследуется из enum.Enum с вариантами, затем этот класс вставляем в mapped

import enum

class Workload(enum.Enum):  
 parttime = 'parttime'  
 fulltime = 'fulltime'

workload: Mapped[Workload]

Для создания внешнего ключа из sqlalchemy импортируется ForeighnKey, затем в mapped\_column вызывается ForeignKey с названием модели и нужным столбцом, для того, чтобы данные из таблицы удалялись вслед за первичным ключом, то вставляется в ForeignKey атрибут ondelete=’CASCADE’

from sqlalchemy import ForeignKey

worker\_id: Mapped[int] = mapped\_column(ForeignKey('workers.id', ondelete=True, ondelete='CASCADE'))

Для того, чтобы создать поле, в которое будет автоматом подгружаться время создания объекта нужно передать аргументу mapped\_column(server\_default=) два варианта что передавать

from sqlalchemy import text, func

1) created\_at: Mapped[datetime.datetime] = mapped\_column(server\_default=func.now())

func.now(передаёт местное время)

2) created\_at: Mapped[datetime.datetime] = mapped\_column(server\_default=text(“TIMEZONE(‘utc’, now())”))

Для того, чтобы не дублировать информацию, можно создавать общие типы с помощью Annotated, где первым указывается тип данных, а вторым условие, пример:

intpk = Annotated[int, mapped\_column(primary\_key=True)]

id: Mapped[intpk]

Для того, чтобы при изменение модели обновлять время изменений, нужно в mapped\_column, onupdate передать то, что нужно обновлять

Для получение данных из таблицы:

def select\_workers():  
 with session\_factory() as session:  
 worker\_id = 1  
 res = session.get(WorkersOrm, worker\_id) # для получение одного результата  
 print(res.username)  
 query = select(WorkersOrm)  
 res2 = session.execute(query)  
 print(res2.scalars().all())

session.get – для получение одного результата

query = select(WorkersOrm)  
 res2 = session.execute(query)  
 print(res2.scalars().all()) – для получение нескольких результатов, scalars используется для того, чтобы объекты модели были не кортежами, а объектами

Для добавления данных

def insert\_data():  
 new\_obj = WorkersOrm(username='Volk')  
 new\_obj2 = WorkersOrm(username='Zayac')  
 with session\_factory() as session:  
 session.add\_all([new\_obj, new\_obj2])

session.flash()  
 session.commit()

Если нужно внести изменения в БД, но не разрывать соединение с БД, нужно использовать flash

Для изменения данных в таблице

@staticmethod  
def update\_workers(worker\_id: int = 1, new\_username: str = 'Foba'):  
 with session\_factory() as session:  
 edited\_workers = session.get(WorkersOrm, worker\_id)  
 edited\_workers.username = new\_username  
 session.expire(edited\_workers)

session.expire\_all()  
 session.commit()

Чтобы отменить изменения, перед ними сделать session.expire(название объекта, кто меняется) или session.expire\_all() – меняет всё

Для того, чтобы получить последнюю версию данных в бд, нужно сделать refresh запрос

Session.refresh(название объекта)