Работа с базами данных (авторизация пользователей)

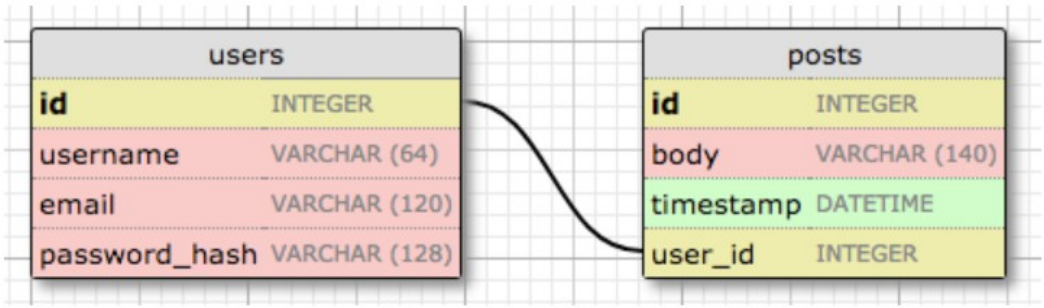
1. Изучить книгу М.Гринберга Мега-учебник Flask ч.4, ч.8

https://habr.com/ru/post/346344/

https://habr.com/ru/post/347450/

Рассмотрим как устанавливаются связи между таблицами

2. Связь «один-ко-многим»



В структру классов моделей данных необходимо внести изменения:

```
from datetime import datetime
from app import db

class User(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(64), index=True, unique=True)
    email = db.Column(db.String(120), index=True, unique=True)
    password_hash = db.Column(db.String(128))
    posts = db.relationship('Post', backref='author', lazy='dynamic')

    def __repr__(self):
        return '<User {}>'.format(self.username)

class Post(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    body = db.Column(db.String(140))
    timestamp = db.Column(db.DateTime, index=True, default=datetime.utcnow)
    user_id = db.Column(db.Integer, db.ForeignKey('user.id'))

    def __repr__(self):
        return '<Post {}>'.format(self.body)
```

« Поле `user_id` – внешний ключ для `user.id`, что означает, что оно ссылается на

значение id из таблицы users. В этой ссылке user — это имя таблицы базы данных, которую Flask-SQLAlchemy автоматически устанавливает как имя класса модели, преобразованного в нижний регистр. Класс User имеет новое поле сообщений, которое инициализируется db.relationship. Это не фактическое поле базы данных, а высокоуровневое представление о взаимоотношениях между users и posts, и по этой причине оно не находится в диаграмме базы данных. Для отношения «один ко многим» поле db.relationship обычно определяется на стороне «один» и используется как удобный способ получить доступ к «многим». Так, например, если у меня есть пользователь, хранящийся в u, выражение u.posts будет запускать запрос базы данных, который возвращает все записи, написанные этим пользователем. Первый аргумент db.relationship указывает класс, который представляет сторону отношения «много». Аргумент backref определяет имя поля, которое будет добавлено к объектам класса «много», который указывает на объект «один». Это добавит выражение post.author, которое вернет автора сообщения. Аргумент lazy определяет, как будет выполняться запрос базы данных для связи.»

Для работы с новостями конкретного пользователя, например для их добавления, необходимо добавить id пользователя в соответствующий объект, например, так:

```
u = User.query.get(1)
p = Post(body='my first post!', author=u)
db.session.add(p)
db.session.commit()
```

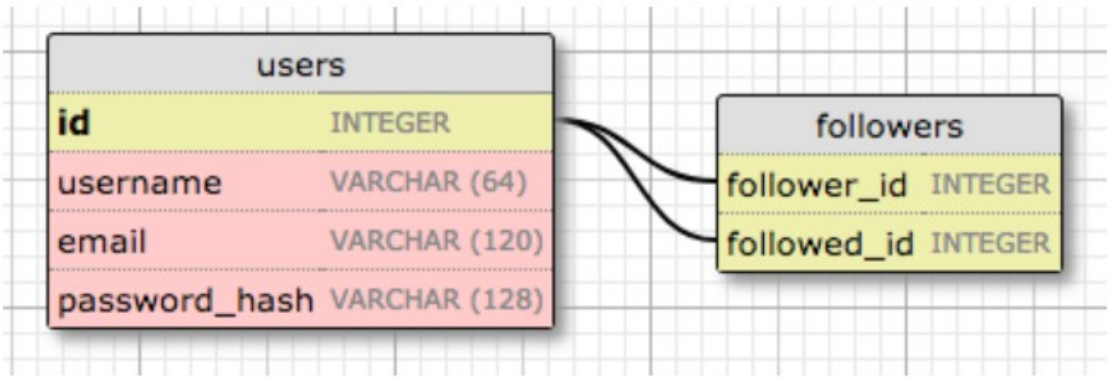
3. Связь «многие-ко-многим»

Рассмотрим отношение пользователь и его подписчики, т. е. и первым и вторым объектами отношений является таблица пользователей, такое отношение называется самореферентным. Но и при нем связь «многие-ко-многим» осуществляется через добавление дополнительной таблицы:

```
followers = db.Table('followers',
db.Column('follower_id', db.Integer, db.ForeignKey('user.id')),
db.Column('followed_id', db.Integer, db.ForeignKey('user.id'))
)
```

«Таблица followers — это таблица ассоциаций отношений или таблицей

относительных связей. Внешние ключи (foreign keys) в этой таблице указывают на записи в пользовательской таблице, поскольку они связывают пользователей с пользователями. Каждая запись в этой таблице представляет собой одну связь между пользователем-подписчиком follower user и подписанным пользователем followed user.»



К модели User lдобавим поле, поддерживающее связь:

```
followed = db.relationship( 'User', secondary=followers,
  primaryjoin=(followers.c.follower_id == id),
  secondaryjoin=(followers.c.followed_id == id),
  backref=db.backref('followers', lazy='dynamic'), lazy='dynamic')
```

Теперь для добавления подписчика необходимо просто добавить один объект класса User к другому:

```
user1.followed.append(user2)
```

4. В качестве задания к лабораторной работе необходимо создать БД, содержащую не менее трех таблиц, реализующие связи «один-ко-многим» и «многие-ко-многим» и создать для них обработчики операций добавления/удаления/изменения данных