5 SQL et programmation objet

5.1 Introduction

Il est possible de considerer une table comme un objet et donc nous pouvons y lier attributs et méthodes qui permettent de manipuler cette table.

Modèle de classe (python)

```
class User(object) :
    def __init__(self, name, age, password='toto'):
        self.name = name
        self.age = age
        self.password = password
```

Modèle de table (SQL)

```
CREATE TABLE User (
    user_pk INTEGER NOT NULL PRIMARY KEY,
    name VARCHAR(20),
    age INTEGER,
    password VARCHAR(10) DEFAULT 'toto')
```

Ce concept s'appelle : Mapping objet-relationnel (ou, en anglais object-relational mapping ou ORM).

C' est un type de programme informatique qui se place en interface entre un programme applicatif et une base de données relationnelle pour simuler une base de données orientée objet. Ce programme définit des correspondances entre les schémas de la base de données et les classes du programme applicatif.

Principal avantage : Il n'est plus nécessaire de reconnaître le SQL

Inconvénients :

- Pour des requêtes complexes, la traduction en SQL peut ne pas être optimale et il y a des baisses de performance.
- On ajoute une couche supplémentaire entre la base et l'utilisateur ou d'éventuels trous de sécurité peuvent s'engouffrer.

5.2 Un ORM python : SQLAlchemy

A faire vous même 1.

- Installez sqlalchemy

```
>>> !pip install sqlalchemy
```

- Connectez-vous ou créez une base de données

```
from sqlalchemy import create_engine
```

```
engine = create_engine('sqlite:///ma_base.db')
```

- Créez votre 1ère table

```
from sqlalchemy import Column, Integer, Text, MetaData, Table

metadata = MetaData()
messages = Table(
    'messages', metadata,
    Column('id', Integer, primary_key=True),
    Column('message', Text),
)

messages.create(bind=engine)
```

- Insérez des données

```
insert_message = messages.insert().values(message='Hello, World!')
engine.execute(insert_message)
```

- Cherchez de la donnée

```
from sqlalchemy import select
stmt = select([messages.c.message])
message, = engine.execute(stmt).fetchone()
print(message)
```

A faire vous même 2.

- Connectez-vous ou créez une base de données

```
from sqlalchemy import create_engine

engine = create_engine('sqlite:///ma_base2.db')
```

- Créez un objet Base

```
from sqlalchemy.ext.declarative import declarative_base
Base = declarative_base()
```

- Créez une nouvelle classe d'objets qui va hériter de Base

```
from sqlalchemy import Column, Integer, String

class Message(Base):
    __tablename__ = 'messages'

    id = Column(Integer, primary_key=True)
    message = Column(String)
```

- Enregistrez la table correspondante dans la base

```
Base.metadata.create_all(engine)
```

- Créez de la donnée et enregistrez-là dans la table

```
from sqlalchemy.orm import sessionmaker

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

message = Message(message="Hello World!")
message.message # 'Hello World!

session.add(message)
session.commit()
```

- Interrogez la base

```
query = session.query(Message)
instance = query.first()
print (instance.message) # Hello World!
```

Tutoriel complet en anglais : https://riptutorial.com/sqlalchemy

Documentation plus détaillée : https://docs.sqlalchemy.org/en/14/core/tutorial.html