Question 1 : Quelles commandes avez-vous utilisées pour effectuer les opérations UPDATE et DELETE dans MySQL ? Avez-vous uniquement utilisé Python ou également du SQL ? Veuillez inclure le code pour illustrer votre réponse.

Réponse :

Pour effectuer les opérations UPDATE et DELETE dans MySQL, j'ai utilisé une combinaison de Python et de SQL. Plus précisément, j'ai utilisé Python avec la bibliothèque `mysql.connector` pour exécuter des requêtes SQL paramétrées.

Code pour l'opération UPDATE :

```
def update(self, user):
    """ Update given user in MySQL """
    self.cursor.execute(
        "UPDATE users SET name = %s, email = %s WHERE id = %s",
        (user.name, user.email, user.id)
    )
    self.conn.commit()
```

Code pour l'opération DELETE :

```
def delete(self, user_id):
    """ Delete user from MySQL with given user ID """
    self.cursor.execute("DELETE FROM users WHERE id = %s", (user_id,))
    self.conn.commit()
```

Question 2 : Quelles commandes avez-vous utilisées pour effectuer les opérations dans MongoDB ? Avez-vous uniquement utilisé Python ou également du SQL ? Veuillez inclure le code pour illustrer votre réponse.

Réponse :

Pour effectuer les opérations dans MongoDB, j'ai utilisé uniquement Python avec la bibliothèque `pymongo`. MongoDB étant une base de données NoSQL, il n'utilise pas SQL mais plutôt des méthodes Python qui correspondent aux opérations CRUD.

Code pour la connexion MongoDB :

```
from pymongo import MongoClient
from bson import ObjectId

# Connexion à MongoDB
self.client = MongoClient('mongodb://user:pass@mongo:27017/')
self.db = self.client['mydb']
self.collection = self.db['users']
```

Code pour l'opération CREATE (INSERT) :

```
def insert(self, user):
    """ Insert given user into MongoDB """
    user_doc = {
        'name': user.name,
        'email': user.email
    }
    result = self.collection.insert_one(user_doc)
    return str(result.inserted_id)
```

Code pour l'opération READ (SELECT) :

```
def select_all(self):
    """ Select all users from MongoDB """
    users = []
    for doc in self.collection.find():
        user_id = str(doc.get('_id', doc.get('id', None)))
        name = doc.get('name', '')
        email = doc.get('email', '')
        users.append(User(user_id, name, email))
    return users
```

Code pour l'opération UPDATE :

```
def update(self, user):
    """ Update given user in MongoDB """
    from bson import ObjectId
    try:
        if len(user.id) == 24:
            query = {'_id': ObjectId(user.id)}
        else:
            query = {'_id': user.id}
    except:
        query = {'_id': user.id}

    update_doc = {
        '$set': {
            'name': user.name,
            'email': user.email
        }
```

```
    }
    self.collection.update_one(query, update_doc)
```

Code pour l'opération DELETE :

```
def delete(self, user_id):
    """ Delete user from MongoDB with given user ID """
    from bson import ObjectId
    try:
        if len(str(user_id)) == 24:
            query = {'_id': ObjectId(user_id)}
        else:
            query = {'_id': user_id}
    except:
        query = {'_id': user_id}

    self.collection.delete_one(query)
```

- Question 3 : Comment avez-vous implémenté votre product_view.py ? Est-ce qu'il importe directement la ProductDAO ? Veuillez inclure le code pour illustrer votre réponse.

Réponse :

Non, mettre la ProductDAO directement dans la vue ne suivrait pas les concepts d'un MVC. Il faut insérer directement le contrôleur de produit dans la vue de produit pour bien implémenter MVC.

```
from models.product import Product
from controllers.product_controller import ProductController

class ProductView:
    @staticmethod
    def show_options():
        controller = ProductController()
        # ...
        if choice == '1':
            products = controller.list_products()
            ProductView.show_products(products)
        elif choice == '2':
            name, brand, price = ProductView.get_inputs()
            product = Product(None, name, brand, price)
            controller.create_product(product)
```

Question 4 : Si nous devions créer une application permettant d'associer des achats d'articles aux utilisateurs (Users → Products), comment structurerions-nous les données dans MySQL par rapport à MongoDB ?

Réponse :

En MySQL, étant une base de données relationnelle, nous pouvons définir des relations à l'aide de clés primaires et de clés étrangères afin de lier, par exemple, des articles et des produits. En MongoDB, étant une base de données non relationnelle, une duplication des données peut avoir lieu, puisqu'il est nécessaire d'insérer les données imbriquées directement dans un document, comme dans l'exemple d'un utilisateur (user).