NoSQL. Mongo DB

MongoDB є однією з найпопулярніших систем управління базами даних типу NoSQL. Вона призначена для зберігання та роботи з великими об'ємами даних у вигляді документів у форматі BSON (Binary JSON).

Основні особливості MongoDB:

1. **Документо-орієнтована модель даних**: Дані зберігаються у вигляді документів, які є BSON-об'єктами (Binary JSON). Кожен документ може мати різну структуру, що дозволяє зберігати дані в довільному форматі.
2. **Гнучкість схеми (Schemaless)**: У MongoDB немає строгої схеми даних. Це означає, що поля можуть бути додані або видалені без необхідності перепроектування бази даних.
3. **Розширюваність та реплікація**: MongoDB надає можливість горизонтального масштабування (розподіл даних на кілька серверів) та має механізми для реплікації даних (створення копій для забезпечення вищого рівня надійності).
4. **Мови запитів та агрегації**: MongoDB підтримує мову запитів, подібну до SQL, а також можливість використання агрегаційних функцій для обробки та аналізу даних.
5. **Шардування (Sharding)**: MongoDB надає можливість розподілу даних на різні сервери (шарди) для підвищення продуктивності та обробки великого обсягу даних.
6. **Велика спільнота та документація**: MongoDB має велику та активну спільноту, а також детальну документацію, що спрощує вивчення та використання цієї бази даних.
7. **Підтримка для різних мов програмування**: MongoDB можна використовувати з багатьма мовами програмування, включаючи Python, JavaScript, Java, C#, і інші.

MongoDB часто використовується в сучасних веб-додатках, аналітиці, системах керування контентом та інших областях, де необхідна гнучкість та масштабованість в роботі з даними.

MongoDB можна використовувати з Python, використовуючи офіційний драйвер `pymongo`. Ось декілька кроків, які потрібно виконати для початку роботи з MongoDB у Python:

1. **Встановлення pymongo**:
   Встановіть пакет `pymongo` за допомогою pip:

   ```
   pip install pymongo
   ```

2. **Підключення до бази даних**:
   Спочатку потрібно встановити з'єднання з MongoDB сервером. Використовуйте URL сервера та порт, якщо вони не за замовчуванням.

   ```
   from pymongo import MongoClient

   client = MongoClient('mongodb://localhost:27017/')  # Підключення до локального серве
   ра MongoDB
   ```

3. **Вибір бази даних**:
   Виберіть базу даних, з якою ви будете працювати. Якщо база даних не існує, вона буде автоматично створена.

   ```
   db = client['mydatabase']  # Вибір бази даних з назвою 'mydatabase'
   ```

4. **Створення та робота з колекціями**:
   MongoDB використовує колекції для зберігання даних. Колекція є аналогічною таблиці у реляційних базах даних. Вона може створюватися автоматично при додаванні першого документу.

   ```
   collection = db['mycollection']  # Створення/вибір колекції 'mycollection'
   ```

5. **Робота з документами**:
   Додайте, оновіть, видаліть та вибирайте документи з колекції. Ось декілька прикладів:

```
# Вставка документа
document = {"name": "John Doe", "age": 30, "city": "New York"}
collection.insert_one(document)

# Оновлення документа
collection.update_one({"name": "John Doe"}, {"$set": {"age": 31}})

# Видалення документа
collection.delete_one({"name": "John Doe"})

# Вибірка документів
result = collection.find({"city": "New York"})
for document in result:
    print(document)
```

6. **Запити та фільтрація**:
   MongoDB дозволяє виконувати різноманітні запити та фільтрацію даних. Наприклад:

```
# Знайти всіх користувачів, які мають вік більше 25
result = collection.find({"age": {"$gt": 25}})

# Знайти користувача з певним ім'ям
result = collection.find_one({"name": "John Doe"})
```

7. **Використання агрегаційних функцій**:
   MongoDB підтримує агрегаційні функції для обробки та аналізу даних.

```
# Приклад агрегаційного запиту
pipeline = [
    {"$group": {"_id": "$city", "count": {"$sum": 1}}}
]
result = collection.aggregate(pipeline)
```

8. **Закриття з'єднання**:
   Все підключення до бази даних повинно бути закрите, коли воно не потрібно більше.

```
client.close()
```

Це загальний огляд того, як працювати з MongoDB у Python за допомогою `pymongo`. За потреби ви можете переглянути детальну документацію `pymongo` для отримання більш докладної інформації та прикладів використання.