**Лабораторна робота №2**

**Тема:** Розгортання mongo docker – контейнерів.

**Мета:** Навчитися розгортати mongo docker – контейнери.

**Короткі теоретичні відомості**

Перша публічна версія MongoDB була випущена в 2009 році, а тепер це висхідна зірка в світі NoSQL. система задумувалась як масштабована база даних - назва Mongo походить від слова «humongous», отриманого об'єднанням «huge» (гігантскій) і «monstrous» (жахливий), а в якості основних проектних цілей були поставлені висока продуктивність і простота доступу до даних. Це документно - орієнтована база даних, яка дозволяє не тільк зберігати, але й писати складні запити для вибірки даних. Схема бази даних не нав'язується (в цьому MongoDB схожа на Riak, але відрізняється від Postgres), тому один документ може містити поля або типи, відсутні у всіх інших документах. Але не думайте, що гнучкість MongoDB перетворює її в іграшку. Цю базу даних використовують такі гігантські сайти, як Foursquare і bit.ly, а в Європейському центрі ядерних досліджень (ЦЕРН) вона застосовується для зберігання данних, що надходять з великого адронного коллайдера.

Mongo поєднує в собі потужні засоби запитів, характерною для реляційних баз даних, і розподілену архітектуру, властиву таким сховищ, як Riak або HBase. Засновник проекту Дуайт Меррімен (Dwight Merriman) говорить, що MongoDB - це та база даних, з якою він хотів би працювати в компанії DoubleClick, де обіймав посаду технічного директора і відповідав за побудову великомасштабного сховища даних, яке проте могло б відповідати на довільні запити.

Mongo - сховище JSON-документів (хоча, строго кажучи, данні зберігаються в двійковому варіанті JSON, який називається BSON). Документ Mongo можна уподібнити рядку реляційної таблиці без схеми, в якій допускається довільна глибина вкладеності значень.

Mongo - відмінний вибір для зростаючого класу веб-проектів, в яких необхідно працювати з великими масивами даних, але бюджет занадто малий для придбання дорогого обладнання. Завдяки відсутності структурованої схеми, Mongo може рости і змінюватися разом з моделлю даних. Якщо ви працюєте в недавно утвореної компанії, яка плекає грандіозні плани або вже накопичила стільки даних, що виникла потреба в горизонтальним масштабування, то придивіться до MongoDB.

**Хід роботи**

**Етап 1. Встанновлення mongo в docker – контейнер.** Завантажте зображення mongo за допомогою команди docker pull mongo(рис. 1):

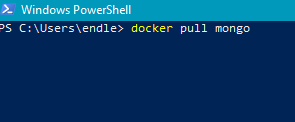


Рисунок 1 – Виконання команди docker для завантаження зображення mongo

Запустіть контейнер за допомогою команди docker run -p 27017:27017 --name server-mongo -d mongo(рис. 2)

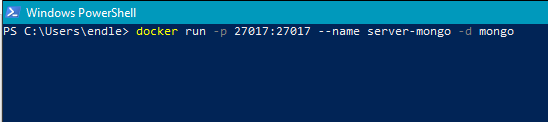


Рисунок 2 – Запуск контейнера

Для підключення до сервера нам знадобиться локальна установка MongoDB. Перейдіть на офіційний сайт MongoDB[www.mongodb.com] завантажте і скачкайте MongoDB.

Перейдіть в теку MongoDB і запустіть звідти PowerShell(рис. 3)

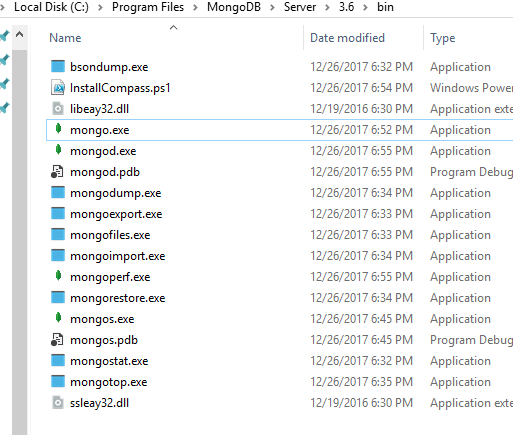


Рисунок 4 – Запуск Power Shell із директорії mongo

Виконайте команду ./mongo(рис. 5). Клієнт mongo автоматично знайте контейнер mongo, який ми запустили пізніше.

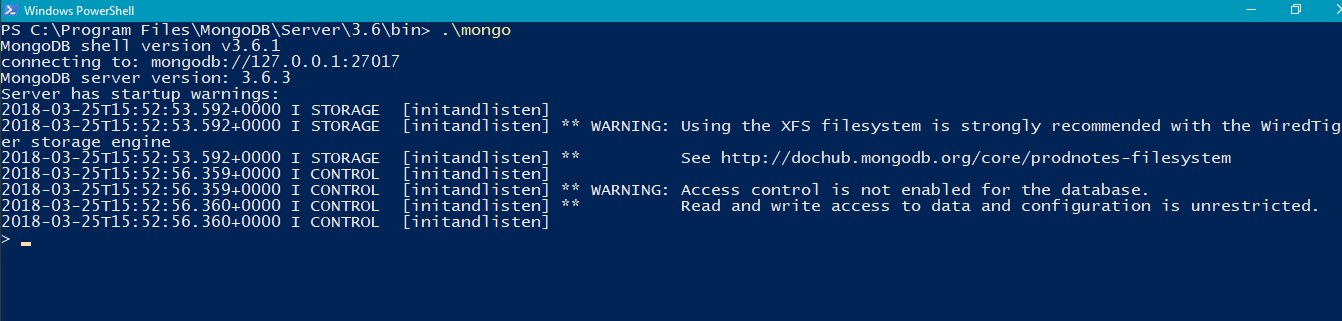


Рисунок 5 – Запуск клієнта mongo

**Етап 2. Операції CRUD і вкладеність.** Для створення бази даних і вказання її, в якості контексту виконайте команду use towns(рис. 6)

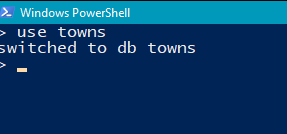


Рисунок 6 – Створення бази даних

Для створення колекції (аналог сегмента в термінології Riak) в Mongo досить просто додати в неї перший запис. Оскільки в Mongo немає схем, то заздалегідь нічого визначати не треба. Більш того, навіть сама база даних towns фізично не існує, поки ми не додамо в неї перший документ. Наступний код створює колекцію towns і вставляє в неї дані(рис. 7):

db.towns.insert({

name: “New York”,

population: 22200000,

last\_census: ISODate(“2009-07-31”),

famous\_for: [ “statue of liberty”, “food” ],

mayor : {

name : “Michael Bloomberg”,

party : “I”

}

})

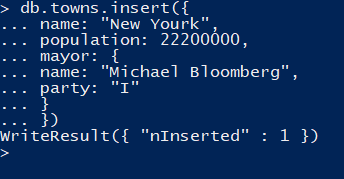


Рисунок 7 – Додавання першого запису

Вище ми згадували, що документи зберігаються в форматі JSON (точніше, BSON), тому в такому форматі ми їх і додаємо. Фігурни дужки {...} позначають об'єкт (аналог асоціативного масиву або хеш-таблиці), що містить ключі і значення, а квадратні дужки [...] - лінійний масив. Значення можуть бути вкладеними, причому глибина вкладеності не обмежена. команда show collections(рис. 8) дозволить переконатися, що колекція дійсно існують.



Рисунок 8 – Список колекцій в базі даних

Переглянутися вміст Колекції дозволяє команда find ()(рис. 9).

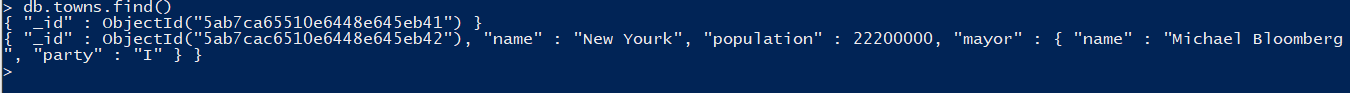


Рисунок 9 – Вміст колекції towns

На відміну від реляційних СУБД, Mongo не підтримує з’єднання на стороні сервера. Один виклик JavaScript-функції витягує документ і всі вкладені в нього дані.

Можливо, ви звернули увагу на доданий системою поле \_id типу ObjectId. Це близький аналог ключового слова SERIAL. Об'єкт ObjectId завжди займає 12 Байтів і складається з тимчасової мітки, ідентифікатора клієнтської машини, ідентифікатора клієнтського процесу та 3-байтового інкрементованого лічильника.

Раніше ми викликали функцію find () без параметрів, щоб отримати всі документи. Для отримання конкретного документа досить задати властивість \_id. Оскільки \_id має тип ObjectId, то для формування запиту необхідно перетворити рядок до цього типу, обернувши його функцією ObjectId (str)(рис 10).

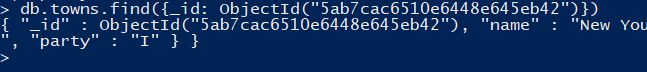


Рисунок 10 – Знаходження конкретного документа по \_id

Функція find() приймає і ще один параметр: об'єкт, який дозволяє вказати, які поля витягати з бази. Якщо вас цікавить тільки назва міста (і його \_id), передайте об'єкт, в якому властивість name приймає значення 1 (або true)(рис. 11).

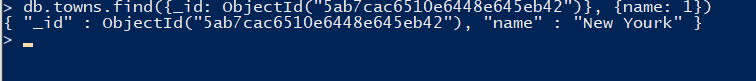


Рисунок 11 – Вказання атрибутів, які потрібно показувати

**Модифікація документів.** Функція update (criteria, operation) приймає два обов’язкових параметра. Перший - критерій відбору - такий же, як для функції find(). Другий - або об'єкт, поля якого замінюють поля відібраних документів, або модифікатор. В даному випадку модифікатор $set записує в поле population значення 2(рис. 12).

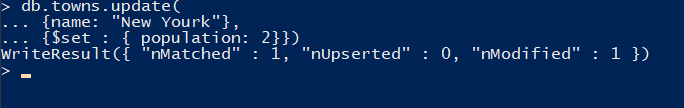
****

Рисунок 12 – Модифікація документу

Може виникнути питання, навіщо взагалі потрібна операція $set. Mongo не працює в термінах атрибутів; на внутрішньому рівні є лише неявне уявлення про атрибути для цілей оптимізації. Однак в інтерфейсі Mongo ніякі атрибути не згадуються, є тільки документи. Навряд чи ви коли-небудь захочете виконати дещо в такому роді (зверніть увагу на відсутність оператора $set):

db.towns.update(

{ \_id : ObjectId(“4d0ada87bb30773266f39fe5”) },

{ state : “OR” }

);

В цьому випадку весь відповідний документ був би замінений пере-даними вами документом ({state: "OR"}). Раз ви не вказали команду $set, Mongo вважає, що ви просто хочете цілком замінити документ. Будьте обережні.

Видалити документи з колекції нескладно. Досить замість функції find () скористатися функцією remove () - і всі документи, які задовольняють критерію, будуть видалені. Важливо відзначити, що видаляється документ цілком, а не тільки елемент, що збігаються або під-документами(рис. 13).

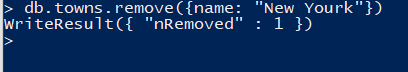


Рисунок 13 – Виділення документа

**Варіанти для самостійних завдань**

1. Дано JSON – схему:

{    
   **"name"**:"Some name",  
   **"age"**:22,  
   **"phones"**:[    
      {    
         **"type"**:"Personal",  
         **"number"**:"33-22-55"  
      }  
   ]  
}

Провести CRUD – операції.

1. Дано JSON – схему:

{    
   **"model"**:"Some car model",  
   **"number"**:"Some car number",  
   **"drivers"**:[    
      {    
         **"name"**:"Some driver name",  
         **"age"**:"Some driver age"  
      }  
   ]  
}

Провести CRUD – операції.

1. Дано JSON – схему:

{    
   **"title"**:"Some book title",  
   **"authors"**:[    
      {    
         **"name"**:"Some author name",  
         **"age"**:"Some author age"  
      }  
   ]  
}

Провести CRUD – операції.

1. Дано JSON – схему:

{    
   **"title"**:"Some title",  
   **"price"**:"some price",  
   **"reviews"**:[    
      {    
         **"name"**:"Some user name",  
         **"review"**:"Some review text"  
      }  
   ]  
}

Провести CRUD – операції.

1. Дано JSON – схему:

{    
   **"title"**:"Some title",  
   **"price"**:"some price",  
   **"reviews"**:[    
      {    
         **"name"**:"Some user name",  
         **"review"**:"Some review text"  
      }  
   ]  
}

Провести CRUD – операції.

1. Дано JSON – схему:

{    
   **"name"**:"Some Student Name",  
   **"studentCard"**:"Some Student Card",  
   **"subjects"**:[    
      {    
         **"title"**:"Some subject title"  
      }  
   ]  
}

Провести CRUD – операції.

1. Дано JSON – схему:

{    
   **"houseNumber"**:"Some house number",  
   **"tenants"**:[    
      {    
         **"name"**:"Some tenant name",  
         **"age"**:"Some tenant age"  
      }  
   ]  
}

1. Дано JSON – схему:

{    
   **"site-url"**:"Some site url",  
   **"description"**:"Some site description",  
   **"admins"**:[    
      {    
         **"login"**:"Some admin login"  
      }  
   ]  
}

Провести CRUD – операції.

1. Дано JSON – схему:

{    
   **"title"**:"Some song title",  
   **"authrs"**:[    
      {    
         **"name"**:"Some author name"  
      }  
   ]  
}

Провести CRUD – операції.

1. Дано JSON – схему:

{    
   **"number"**:"Some room number",  
   **"price"**:"Some room price",  
   **"tenants"**:[    
      {    
         **"name"**:"Some tenant name"  
      }  
   ]  
}

Провести CRUD – операції.

**Контрольні запитання**

1. Що таке MongoDB?
2. Яку модель використовує MongoDB?
3. Як додати запис в БД?
4. Як знайти запис в БД?
5. Як модифікувати запис в БД? Для чого використовується модифікатор $set?
6. Як видалити запис із БД?