Turtle Inc. - Informe

Trabajo Práctico Obligatorio - (72.41) Base de Datos II

Alumno	Legajo	Mail
Alejo Flores Lucey	62622	afloreslucey@itba.edu.ar
Andres Carro Wetzel	61655	acarro@itba.edu.ar
Camila Di Toro	62576	cditoro@itba.edu.ar
Nehuén Gabriel Llanos	62511	nllanos@itba.edu.ar

Introducción
Desarrollo
- Implementación queries y vistas
- Desarrollo de la API
  - Problemas encontrados
- Proceso de Migración

Conclusión

Introducción

Con el objetivo de interiorizarse en el manejo de las bases de datos relacionales y no relacionales (NoSQL), se ha desarrollado un sistema formado por una base de datos relacional PostgreSQL que crea determinadas tablas a la que se le pueden realizar consultas de diferentes tipos. Además de ello, se provee la funcionalidad de poder migrar todas tablas a MongoDB (base de datos no relacional) y realizar queries a la misma. A lo largo de este informe se describirá la implementación de queries/vistas, el desarrollo de la API y cómo funciona el proceso de migración.

Desarrollo

Implementación queries y vistas

Como parte del trabajo práctico se desarrollaron diez queries y dos vistas, tanto en mongo como en PostgresSQL.

En el caso de PostgresSQL, se contaba con la ventaja de contar con esquemas previamente definidos, lo que simplificó el proceso. Sin embargo, en las bases de datos de documentos, la flexibilidad de los esquemas es una característica esencial y no existe una estructura única y normalizada que sea adecuada para todos los casos.

Con el foco en las consultas y vistas que se requeria implementar, tomamos decisiones estratégicas para organizar las collections en MongoDB:

- Los teléfonos de los clientes fueron embebidos dentro de la collection de clientes clientes correspondientes. Los documentos de clientes tienen un campo “teléfono” que contiene un array con los teléfonos correspondientes. Esto facilitó la resolución de las queries que implican la obtención de teléfonos y clientes en simultáneo.
- Los detalles de las facturas fueron embebidos dentro de las facturas. Esto facilitó realizar queries como la 8, donde se solicitan las facturas que contengan los productos de un proveedor particular.
- Los productos son una collection independiente.

Es importante destacar que en los dos casos en los que se optó por de-normalizar los esquemas, se trataba de relaciones de tipo 1:N bajo la semántica "contiene". Se consideró que esta de-normalización era razonable, ya que no se esperaba que los clientes tuvieran una gran cantidad de teléfonos ni que las facturas contuvieran un número excesivamente grande de productos por lo que los arrays mutables no presentan preocupaciones significativas y simplifica considerablemente las operaciones.

Desarrollo de la API

Para el desarrollo de la API se eligió el framework Django, un paquete de Python, de alto nivel, que potencia un desarrollo rápido y diseño limpio y pragmático de servidores HTTP.

Por un lado, se procedió con la creación de los modelos para las tablas de la base de datos preexistente. Para eso, se utilizó una herramienta de Django que inspecciona la base de datos y propone los modelos de Python correspondientes. Dichos modelos deben ser revisados y adaptados para mantener la semántica del sistema.

Por otro lado, se crearon las vistas de Python, que se mapean uno a uno con urls para poder hacer las operaciones ABM (Alta, Baja y Moficación).

Problemas encontrados

Django tiene limitaciones en cuanto a la gestión de PK (Primary Keys) compuestas. Para solucionar este inconveniente, se modificó el esquema original agregando campo `id` en las tablas `teléfono` y `detalle_factura`, con tipo de dato `SERIAL` y utilizando el mismo como PK (Primary Key). Asimismo, se cambió el tipo de dato de las PK (Primary Keys) de las otras tablas para que sean de tipo `SERIAL`.

Proceso de Migración

Al inicio, se encuentran cargadas las tablas únicamente en PostgresSQL. Es posible realizar una migración de los datos a MongoDB haciendo uso del endpoint `/migration`.

Un `POST` sobre el endpoint `/migration` genera que todos los datos en PostgresSQL se agreguen a MongoDB. Luego, pueden probarse las queries sobre los datos ingresados en Mongo, sin necesidad de reiniciar el sistema.

Para realizar dicha migración, se utiliza Django para obtener todas las tuplas de las diferentes tablas, y se utiliza la dependencia PyMongo para conectarse a MongoDB desde Python.

Cabe recalcar que la herramienta de Django, al obtener las tuplas, no sube todo a memoria y luego lo procesa en un ciclo común y corriente, sino que instancia un cursor y recorre las tuplas de manera secuencial. De esta manera, se logra que la migración no sea un proceso demandante para el sistema.

Conclusión

En este informe, se abordó el desarrollo de un sistema formado por dos bases de datos, una relacional y una no relacional para poder realizar queries a las mismas y realizar un proceso de migración entre ellas. Durante el proceso se tomaron decisiones de diseño para la migración, las queries que se ofrecen y también para el desarrollo de la API.

A pesar de los logros, se identificaron áreas de mejora importantes, como la implementación de testeos para corroborar que todo funcione correctamente. Además se sugirió la implementación de autenticación y autorización de usuarios y la posibilidad de proveer una interfaz gráfica para las operaciones de ABM.

Dada la ambigüedad de la consigna y de las subsiguientes consultas a la cátedra sobre la misma, el grupo decidió por sus propios medios el alcance final del proyecto. Esto desembocó en el uso de frameworks y aplicativos de alto nivel que no fueron vistos en la materia; de todas formas, fue una experiencia enriquecedora.

En resumen, este proyecto fue de gran ayuda para la interiorización de los conceptos relacionados y proporciona una base sólida para futuras expansiones y mejoras.