Paquetería Sensible y de Alto riesgo

Base de Datos - C2466-ST0246-1672

Nombre Completo

Juan Andrés Young Hoyos
Samuel Madrid Ossa

Profesora: Catalina Hernandez Acevedo

Medellín, 11 de noviembre de 2024

Índice

1 | Modelamiento Base de datos MongoDb

1.1 | Modelo entidad relación

Modelo entidad relación con retroalimentación dada en la entrega número 2.

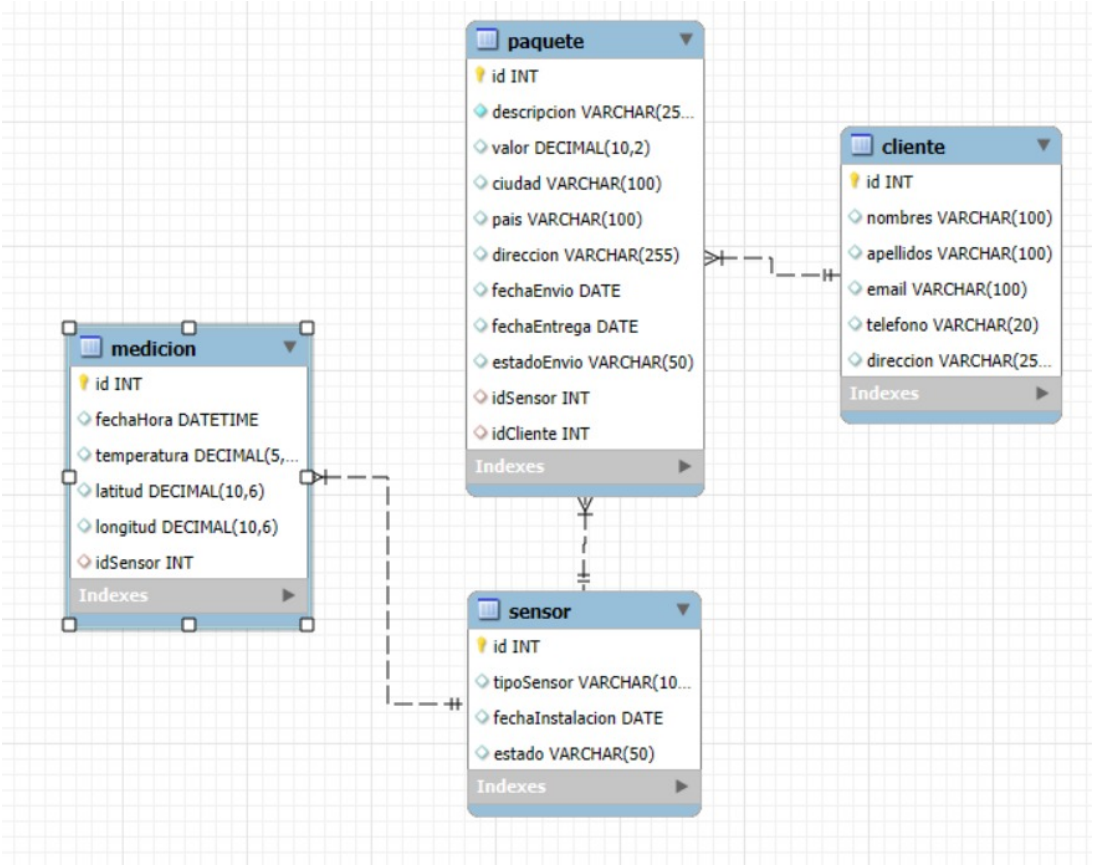

Figura 1.1: Modelos Entidad Relación.

1.2 | Carga de trabajo

Tipo	Información	Frecuencia	Criticidad	Latencia	Tamaño
Lectura	Cada entidad tiene un conjunto de operaciones CRUD. Las más importantes son las consultas en tiempo real de los estados de los sensores.	Cada segundo.	La lectura es crítica para las métricas, ya que su consulta es esencial para la empresa.	Mínima de 30 segundos, suficiente para el análisis de eventos relevantes.	Alcanzan su máximo cuando un cliente consulta información que supera el límite del BSON (Max: 16MB).
Escritura	Las creaciones de documentos son orgánicas, excepto para las mediciones.	Cada segundo.	Es crítica porque una mala métrica puede implicar la pérdida de un paquete.	Mínima de 1 segundo por medición para mantener la calidad.	La escritura más relevante es la del paquete, pero en frecuencia, destacan las mediciones.
Lectura por pipeline	Involucra varias entidades en una consulta.	Frecuencia de una vez por conexión.	Facilita la navegación del cliente.	Máximo de 4 segundos para mantener una buena experiencia de usuario.	No supera el máximo de BSON debido al paginado implementado.

Cuadro 1.1: Carga de trabajo en el modelo MongoDB

1.3 | Justificación del uso de documentos embebidos o referenciados

		Embed	Reference
sencillez	¿Mantener juntas las piezas de información conduciría a un modelo de datos y un código más simples?	Yes	
Mantener juntos	¿La cardinalidad de las relaciones (actual y futura) es uno a uno o uno a pocos?	Yes	
Atomicidad Consulta	¿La aplicación consulta los datos juntos?	Yes	
Complejidad actualización	¿Mantener juntas las piezas de información conduciría a un modelo de datos y un código más simples?	Yes	
Archivo	¿Los datos de información deben archivarse al mismo tiempo?	Yes	
Cardinalidad	¿Existe una alta cardinalidad (actual y futura) del lado de "muchos" de la relación?	No	Yes
Duplicidad datos	¿La duplicación de datos sería demasiado complicada de gestionar y no deseada?	No	Yes
Tamaño documento	¿El tamaño combinado de las piezas de información consumiría demasiada memoria o transferiría ancho de banda para la aplicación?	No	Yes
Crecimiento documento	¿El subdocumento embebido crecería sin límites?	No	Yes
Workload	¿Las piezas de información se escriben en diferentes momentos en una carga de trabajo que requiere mucha escritura?		Yes
Individualidad	¿Los subdocumentos pueden existir sin el documento padre?		Yes

Figura 1.2: Tabla comparativa

Creando nuestro modelo a partir de esta tabla, encontramos que las mediciones deberían referenciar al sensor que las toma, y que los paquetes referencien al cliente propietario. La cantidad de variables ambientales medidas en un paquete no es tan alta, por lo que también se podría considerar embeber la entidad paquete con la entidad sensor para simplificar la distribución de información, aunque esto puede resultar en información innecesaria en algunas consultas.

1.4 | Patrón usado

Inicialmente, optamos por el patrón Outlier. Sin embargo, tras una investigación profunda, decidimos cambiar al patrón de Árbol, que se adapta mejor al proyecto debido a su estructura jerárquica, permitiendo una organización y acceso eficiente de los datos.

1.5 | Creación Base de datos MongoDb

Este enlace del proyecto final lleva a los JSON necesarios o los puedes ver en la entrega interactiva.

- Enlace.

1.6 | Consulta MongoDb

Ambas consultas incluyen **aggregate** y **lookup**:

- Consulta 1: Busca todos los paquetes asociados a un tipo específico de sensor.

```python
async def retrieve_packages_by_sensor_type(sensor_type: str):
    """Obtener todos los paquetes asociados con un tipo específico de se
    packages = []
    async for package in package_collection.aggregate([
        {
            # Realiza un lookup para unir los documentos de paquetes con
            # utilizando id_sensor y _id como campos de unión
            "$lookup": {
                "from": "sensors_collection",  # La colección con la que
                # Define una variable sensorId que convierte el id_senso
                "let": {"sensorId": {"$toObjectId": "$id_sensor"}},
                # Define una pipeline para filtrar los sensores basados
                "pipeline": [
                    {"$match": {
                        "$expr": {
                            "$and": [
                                {"$eq": ["$_id", "$$sensorId"]},  # Comp
                                {"$eq": ["$tipo_sensor", sensor_type]}
                            ]
                        }
                    }}
                ],
                "as": "sensor_info"  # Nombra los datos unidos como sens
            }
```

Figura 1.3: Paquetes asociados a un tipo específico de sensor.

- Consulta 2: Busca todos los paquetes asociados a un usuario por su email.

```python
async def retrieve_packages_by_user_email(user_email: str):
    """Obtener todos los paquetes asociados a un correo electrónico específico del
    packages = []
    async for package in package_collection.aggregate([
        {
            # Realiza un lookup para unir los documentos de paquetes con la colecc
            # utilizando id_cliente en paquetes y _id en usuarios como campos de u
            "$lookup": {
                "from": "users_collection",  # La colección con la que se unirá
                # Define una variable clientId que toma el valor de id_cliente en
                "let": {"clientId": "$id_cliente"},
                # Define una pipeline para filtrar los usuarios basados en la conv
                "pipeline": [
                    {"$match": {
                        "$expr": {
                            "$and": [
                                # Compara _id en la colección de usuarios con clie
                                {"$eq": ["$_id", {"$toObjectId": "$$clientId"}]},
                                # Compara el email en usuarios con el user_email p
                                {"$eq": ["$email", user_email]}
                            ]
                        }
                    }}
                ],
                "as": "user_info"  # Nombra los datos unidos como user_info en el
            }
```

Figura 1.4: Paquetes asociados a un usuario por su email.

2 | Conclusiones

1. **¿Qué es lo más relevante que se debe tener en cuenta en la elección de la base de datos de un proyecto?**
   Depende de la necesidad lo que se considere lo más relevante, pero sí hay una lista clave de cosas que forman parte de la elección de la base de datos para un proyecto y estas son:
   - Modelo de los datos: estructurales y no estructurales.
   - Escalabilidad: horizontal o vertical, es decir, si se amplía la infraestructura o se mejora la ya existente, respectivamente.
   - Consistencia o disponibilidad.
   - Soporte de integración.
   - Costo.
2. **¿Qué aspecto es el que da más dificultad durante el proceso de diseño y la implementación de la base de datos?**
   El aspecto que suele generar más dificultad durante el diseño e implementación de una base de datos es encontrar el modelo de datos óptimo que garantice tanto el rendimiento como la escalabilidad, especialmente para aplicaciones con necesidades complejas de almacenamiento y acceso a datos, como en el caso de IoT.
3. **¿Por qué son importantes las bases de datos o por qué no lo son?**
   Las bases de datos son muy importantes en la actualidad porque, en primer lugar, hacen posible el manejo de grandes volúmenes de datos, y no serían útiles para aplicaciones simples o temporales, datos de un solo uso o cuando hay alternativas aún más simples como archivos de texto.
4. **¿Qué aspecto o tema generó más dificultad a la hora de realizar el proyecto?**
   Crear las relaciones, debido a que había que pensar en la lógica de negocio y en la escalabilidad simultáneamente debido a la poca planificación que involucra las bases de datos no relacionales.
5. **¿Qué aprendizaje generó la realización del proyecto?**
   Nos dio un vistazo de muchos aspectos que no se tienen en cuenta cuando solo se indaga en las implementaciones de las tecnologías y nos dio una primera probada de lo sofisticado y hermoso que puede llegar a ser la creación de una buena base de datos y lo provechoso que es.
6. **¿Qué lecciones aprendidas dejó la realización del proyecto? ¿Qué no deben hacer, qué se debe hacer, qué se debe mejorar en todo el proceso de diseño e implementación de base de datos?**
   Aprendimos que para lograr un buen diseño primero debemos planificar, y al momento de planificar hay que pensar en la implementación sin dejar de lado la lógica de negocio. También hay que entender bien cómo se forman las relaciones entre entidades, estando en todo momento con la certeza de la utilidad de las mismas. Jamás hay que comenzar codificando porque no se llegará a nada, y tampoco hay que pasar por alto la posibilidad de un crecimiento de la base de datos a futuro, ya que esto podría generar una deuda técnica.
7. **Expectativas, temas y metodología de clase**
   Todas las expectativas del curso se cumplieron, y los temas que vimos fueron relevantes y bien seleccionados. No hubo temas que se sintieran innecesarios, y el contenido estuvo bien balanceado; agregar más podría restarles profundidad a los temas actuales. La metodología de la clase fue buena y ayudó a entender bien los contenidos.