Firestore: SERVICIO ADMINISTRADO O EN LA NUBE, DELEGA

=====================================================

Objeto principal en la base de datos Firestore? Coleccion (folder)

Tipo de dato? string



¿Cuál es el caso de uso principal de una base de datos basada en documentos?

ALMACENAR Y PROCESAR MUCHA INFORMACION,

NO Analítica necesariamente

TABLA – COLECCION

TUPLA - DOCUMENTO

Son una implementación de clave valor que varía en la forma semiestructurada en que se trata la información.

Ideal para almacenar datos JSON y XML. Ejemplos: MongoDB, Firestore.

Las top level collection son las colecciones que tienen un documento padre FALSO, LOS HIJOS SON DOCUMENTOS

Utilizamos subcolecciones cuando queremos acceder a sus documentos de manera independiente o son referenciados desde otras colecciones FALSO

Firebase

<https://firebase.google.com/>

*Tipos de datos en Firestore*:

1. **String**: Cualquier tipo de valor alfanumérico
2. **Number**: Soporta enteros y flotantes.
3. **Boolenan**: Los clásicos valores True y False
4. **Map**: Permite agregar un documento dentro de otro.
5. **Array**: Permite agregar un conjunto de datos (soporte multi type) sin nombre e identificador.
6. **Null**: Indica que no se ha definido un valor.
7. **Timestamp**: Permite almacenar fechas (guarda el año, mes, día y hora).
8. **Geopoint**: Guarda una localización geográfica (coordenadas latitud-longitud).
9. **Reference**: Permite referencia un documento (relaciona dos documentos, no importa su colección).

NO MANEJA : VARCHAR

USA NUMBER, NO INT NI FLOAT

CREAS COLECCION: Clientes

DOCUMENTOS

* Cada documento puede tener una lista de campos diferente bajo cliente
* Un document tiene la direccion detallada, en otro document no tiene nada de direccion
* Un documento tiene el telefono y cellular, en el otro solo el cellular
* Un top *level collection* se utilizaria para relaciones de tipo “agregacion”.
* Mientras que una *sub collection* se utilizaria para relaciones tipo “composicion”.

EJEMPLO DE RELACION DE AGREGACION

* Tenemos Estudiantes, Cursos y Notas. Los estudiantes tiene cursos y los cursos tiene estudiantes.
* Si se elimina un curso los estudiantes no deben ser eliminados.
* Lo mismo si se elimina un estudiante los cursos no deben ser elimiandos.

Esto es una relacion de agregacion. Aqui se usaria *top level collection* para estudiantes y cursos.

SUBCOLECCIONES

* Los estudiantes tienen notas y las notas pertenecen a un estudiante.
* Si se elimina un estudiante, tiene sentido eliminar las notas.
* Esto es una relacion de composicion.
* Aqui se usarian las *subcollections*.
* El estudiante tendría una *subcollection* de notas.

Las top level collection son las colecciones que tienen un documento padre FALSO

Un documento pertenece a una coleccion, y esa coleccion a otro documento, pero EL PUNTO INICIAL ES UNA COLECCION PADRE, NO UN DOCUMENTO PADRE

COLECCION – DOCUMENTO – COLECCION - DOCUMENTO

Utilizamos subcolecciones cuando queremos acceder a sus documentos de manera independiente o son referenciados desde otras colecciones FALSO

NO, ES PARA RELACIONAR DOCUMENTOS A UN DOCUMENTO PADRE.

Tabla -> Colección

Tupla -> Documento

Dentro de las Colecciones existen 2 grandes tipos. Las Top level collection o colecciones de nivel superior y las subcollections o subcolecciones. Estas últimas viven únicamente dentro de un documento padre.

**¿Cómo saber cuál escoger?**

Para determinar si tu colección debe ser top level o subcolección no hay una regla escrita en piedra y más bien tiene que ver con el caso de uso en particular y con la experiencia que hayas ganado como desarrollador.

Lo cierto es que no hay una sola forma de estructurar nuestra DB basada en documentos, y por tanto no existe una respuesta correcta, sin embargo a continuación te ofrezco un par de reglas guía que puedes utilizar para transformar tu proyecto que ya trabajaste en bases de datos relacionales en un proyecto no relacional.

**Regla 1. Piensa en la vista de tu aplicación**

La primera pista que te puedo dar es que pienses en un inicio en la manera en que los datos serán extraídos. En el caso de una aplicación, la mejor forma de pensarlo es en términos de las vistas que vas a mostrar a un momento determinado en la aplicación.

Es decir, al armar la estructura en la base de datos que sea un espejo o que al menos contenga todos los datos necesarios para llenar las necesidades que tiene nuestra parte visual en la aplicación.

En el caso de Platziblog por ejemplo si tienes una vista de un blog post individual, generalmente conviene mostrar además de los datos inherentes al post como el contenido, datos adicionales como las etiquetas que tiene o por ejemplo el autor (o autores si es colaborativo), en este caso tal vez convenga guardar estas dos “entidades” (autores y etiquetas) como subcolecciones de cada documento blog post.

**Regla 2. La colección tiene vida propia**

Esta regla se refiere a que la excepción a la regla 1 es cuando tenemos un caso en que la “entidad” que tiene necesidad de vivir y modificarse constantemente de manera independiente a las otras colecciones. Por ejemplo en Platziblog podemos en el ejemplo anterior hacer una excepción a autores porque nos conviene tenerlas como top level collection en el sentido que se añadan, borren, cambien o listen los usuarios sin depender del blog post.

Experimenta aplicando estas dos reglas a un proyecto que ya conozcas en una base de datos relacional y trata de convertirla en un proyecto de Firestore y comentanos los retos a los que te enfrentaste.