Universidad Don Bosco

Dirección de Educación a Distancia

Ingeniería en Ciencias de la Computación

Diseño y Programación de Software Multiplataforma (DPS941)

Foro de Discusión I

Presentado por

Ricardo Arturo De Paz Núñez (DN192246)

Kevin Alexander Fernandez Monge (FM150385)

Julio Danilo Flores Fuentes (FF201999)

Christian Alexander Hernández Funes (HF171856)

Kelvin Vladimir García Juárez (GJ111587)

Fecha de Entrega 29 de Octubre de 2023

Bases de datos NoSQL que Firebase ofrece: Cloud Firestore y Realtime Database.

Cloud Firestore

1. ¿Cuáles son las diferencias fundamentales entre bases de datos SQL y NoSQL?

En el siguiente cuadro compararemos algunas de las diferencias que nosotros consideramos “clave” entre las bases de datos SQL y NoSQL en términos de modelo de datos, escalabilidad, consultas, transacciones, esquema de datos y flexibilidad para cambios.

Las bases de datos SQL son ideales para datos estructurados y relaciones complejas, mientras que las bases de datos NoSQL son más adecuadas para datos no estructurados y cambios frecuentes en el esquema de datos.

Aspecto	Bases de Datos SQL	Bases de Datos NoSQL
Modelo de Datos	Tablas estructuradas	Diversos modelos (documento, clave-valor, grafo, etc.)
Escalabilidad	Escalamiento vertical	Escalamiento horizontal
Consultas	Lenguaje SQL, consultas complejas	Consultas más simples y rápidas
Transacciones	Admite transacciones ACID	Flexibilidad en transacciones, a menudo sin ACID
Esquema de Datos	Estructura rígida	Esquema flexible
Flexibilidad para Cambios	Menos flexible para cambios en el esquema	Mayor flexibilidad en cambios en el esquema

Ejemplo:

Base de Datos SQL:

Supongamos que estamos construyendo un sistema de gestión de usuarios para un sitio web utilizando una base de datos SQL. En este caso, podríamos tener una tabla llamada "Usuarios" con una estructura predefinida que contiene información sobre los usuarios, como su *nombre, dirección de correo electrónico y contraseña*. La tabla podría verse de la siguiente manera:

Tabla: Usuarios

ID	Nombre	Correo Electrónico	Contraseña
1	Juan	juan@email.com	********
2	María	maria@email.com	********
3	Pedro	pedro@email.com	********

En una base de datos SQL, se espera que todos los registros de la tabla tengan la misma estructura, y cada campo tiene un tipo de dato específico. Las consultas SQL se utilizan para recuperar, insertar, actualizar y eliminar datos de esta tabla.

Base de Datos NoSQL:

Supongamos que estamos desarrollando una aplicación móvil de redes sociales donde los usuarios pueden crear perfiles con información personalizada. Utilizar una base de datos NoSQL sería más apropiado en este escenario, ya que los perfiles de los usuarios pueden variar considerablemente en contenido.

En una base de datos NoSQL, como Firebase Cloud Firestore, podríamos tener una colección llamada "Usuarios" que contiene documentos individuales para cada usuario. Cada documento puede tener su propia estructura de datos, lo que permite una mayor flexibilidad. Un ejemplo podría ser:

Colección: Usuarios

Documento 1:
```
{
 "Nombre": "Juan",
 "Correo Electrónico": "juan@email.com",
 "Amigos": ["María", "Pedro"],
 "Intereses": ["Deportes", "Música"]
}
```

Documento 2:
```
{
 "Nombre": "María",
 "Correo Electrónico": "maria@email.com",
 "Publicaciones": [
  {"Texto": "¡Hoy es un gran día!", "Fecha": "2023-10-25"},
  {"Texto": "Foto de la playa", "Fecha": "2023-10-24"}
 ]
}
```

En este ejemplo de base de datos NoSQL, cada usuario tiene una estructura de datos diferente según sus preferencias y actividades en la aplicación. No es necesario que todos los documentos sigan el mismo formato rígido. Esto proporciona flexibilidad para adaptarse a cambios en los perfiles de los usuarios con facilidad.

2. ¿Cuáles son las diferencias específicas entre Cloud Firestore y Realtime Database?

Las diferencias especificas entre Cloud Firestore y Realtime Database en términos de modelo de datos, consultas, escalabilidad, seguridad, disponibilidad y soporte para consultas sin conexión. La elección entre ambas bases de datos dependerá de las necesidades específicas de tu proyecto y de cuáles de estas características se adaptan mejor a tus requisitos.

Aspecto	Realtime Database	Cloud Firestore
Modelo de Datos	Árbol JSON simple	Colecciones de documentos similares a JSON
Consultas	Consultas con ordenamiento y filtrado limitados	Consultas indexadas con ordenamiento y filtrado compuestos
Escalabilidad	Escalabilidad limitada (hasta 200,000 conexiones simultáneas)	Escalabilidad automática (hasta 1 millón de conexiones simultáneas)
Seguridad	Lenguaje de reglas en cascada	Reglas sin formato de cascada y compatibilidad con Identity and Access Management (IAM)
Disponibilidad	Disponibilidad zonal en una región	Regional y multirregional con ajuste de escala automático
Consultas sin conexión	Soporte para clientes de Apple y Android	Soporte para clientes de Apple, Android y la Web

3. Basándose en su investigación, ¿cuál de estas bases de datos consideran que sería la mejor opción para implementar en una aplicación desarrollada en React Native?

La elección entre ambas bases de datos dependerá de las necesidades específicas de nuestra aplicación y de cuáles de estas características son más importantes para nuestro caso de uso.

Aspecto	Cloud Firestore	Realtime Database
Modelo de Datos	Colecciones de documentos similares a JSON	Árbol JSON simple
Consultas	Consultas indexadas con ordenamiento y filtrado compuestos	Consultas con ordenamiento y filtrado limitados
Escalabilidad	Escalabilidad automática (hasta 1 millón de conexiones simultáneas)	Escalabilidad limitada (hasta 200,000 conexiones simultáneas)
Seguridad	Reglas sin formato de cascada y compatibilidad con Identity and Access Management (IAM)	Lenguaje de reglas en cascada
Disponibilidad	Regional y multirregional con ajuste de escala automático	Disponibilidad zonal en una región
Consultas sin conexión	Soporte para clientes de Apple, Android y la Web	Soporte para clientes de Apple y Android

Para una aplicación desarrollada en React Native, la elección entre Cloud Firestore y Realtime Database depende de nuestras necesidades específicas, por ejemplo:

- Si necesitamos consultas avanzadas, alta escalabilidad, un modelo de datos flexible y reglas de seguridad más detalladas, Cloud Firestore es una buena opción.
- Si son más simples, no requerimos de consultas complejas y deseamos una solución eficiente para datos en tiempo real, Realtime Database podría ser suficiente.
- Además, consideramos los costos asociados, ya que Cloud Firestore se cobra principalmente por operaciones, mientras que Realtime Database se cobra por ancho de banda y almacenamiento.

En pocas palabras, la elección dependerá de cuál de estas características se adapta mejor a los requisitos específicos de nuestra aplicación en React Native.

Parte 2: Ejercicio Práctico - Diseño de Bases de Datos:

1. En esta fase, deben elaborar dos estructuras de base de datos: una utilizando SQL y otra utilizando NoSQL (Firebase Firestore o Realtime Database). Ambas bases de datos deben permitir almacenar las notas de los alumnos becarios de UDB VIRTUAL.
|

Base de Datos SQL:

Tenemos los datos:

Tabla Notas:

- ID (Clave primaria)
- ID_Alumno
- Nombre_Alumno
- Apellido_Alumno
- DUI_Alumno
- ID_Materia
- Nombre_Materia
- Nota
- Fecha

Aplicando Normalización:

Paso 1: Identificar las Entidades y Atributos

- Identifica las entidades involucradas en tus datos.
  Alumnos, Materias y Notas.
- Identifica los atributos o campos que pertenecen a cada entidad. Por ejemplo, los atributos de Alumnos pueden ser Nombre, Apellido y DUI y los atributos de Notas son: ID_Alumno, ID_Materia, Nota y Fecha.

Paso 2: Crear Tablas Separadas

- Separamos las entidades.

Tabla Alumnos:

- ID (Clave primaria)
- Nombre
- Apellido
- DUI

Tabla Materias:

- ID (Clave primaria)
- Nombre de la materia

Tabla Notas:

- ID (Clave primaria)
- ID_Alumno

- ID_Materia
- Nota
- Fecha

Paso 3: Establecer Relaciones

Definimos las relaciones entre las tablas utilizando las claves primarias y externas. Tabla Notas, las claves ID_Alumno y ID_Materia se utilizan como claves externas para relacionar las notas con los alumnos y las materias.

Paso 4: Eliminar Redundancia de Datos

- Cada entidad y sus atributos se almacenan en tablas separadas. Esto elimina la redundancia de datos, ya que la información del alumno y la materia se almacena solo una vez en las tablas Alumnos y Materias.

Con esta estructura normalizada podemos, realizar consultas eficientes y mantener la integridad de los datos sin duplicación innecesarias.
Por lo que la estructura queda de la siguiente manera:

Alumnos:
  - ID (Clave primaria)
  - Nombre
  - Apellido
  - DUI

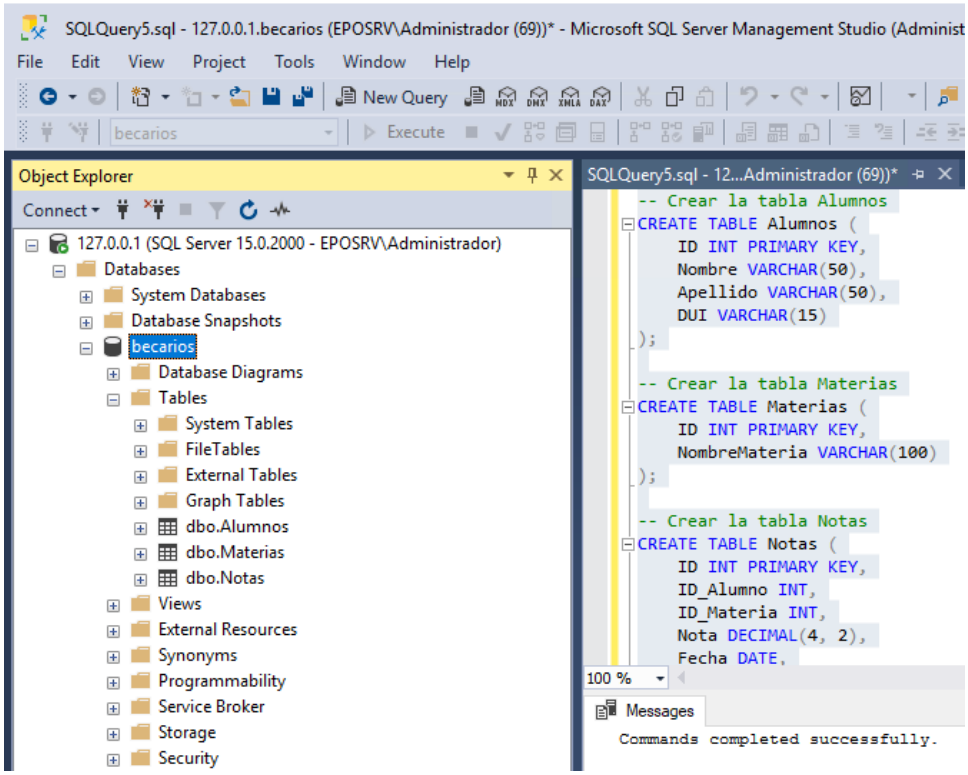
Materias:
  - ID (Clave primaria)
  - Nombre de la materia

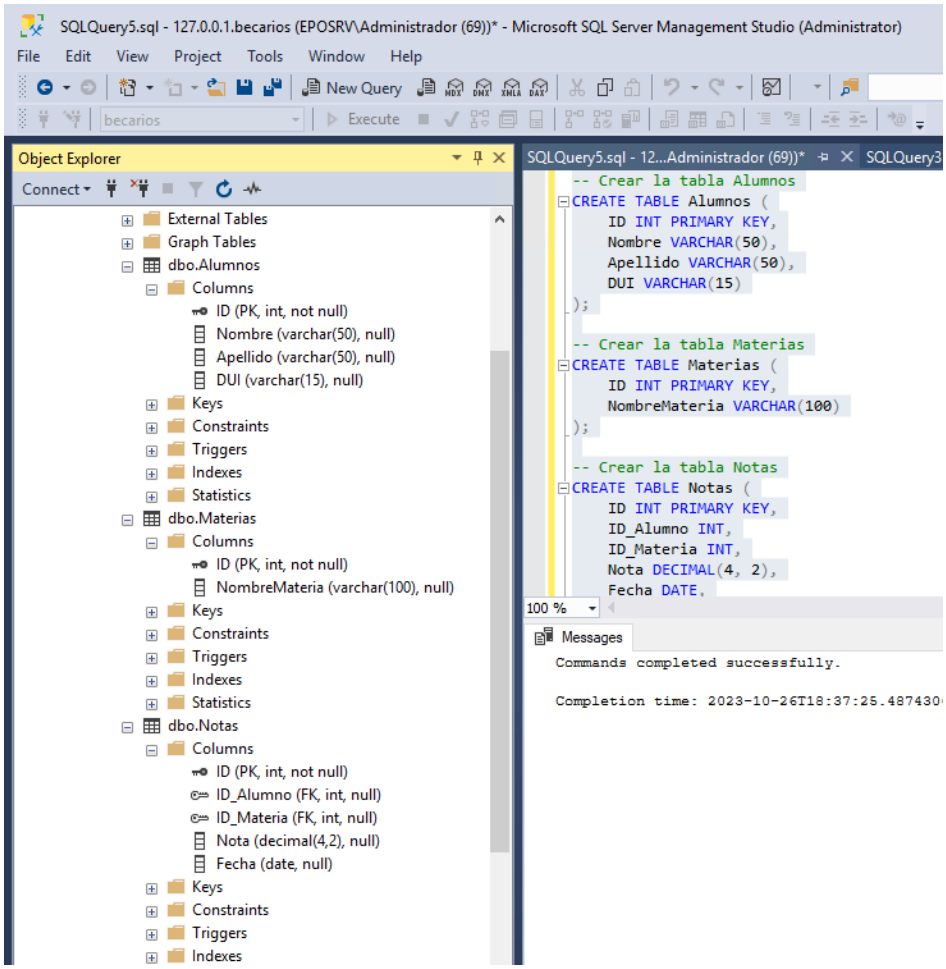
Notas:
  - ID (Clave primaria)
  - ID_Alumno (Clave externa que hace referencia a la tabla Alumnos)
  - ID_Materia (Clave externa que hace referencia a la tabla Materias)
  - Nota
  - Fecha (Fecha en la que se registró la nota)

Script SQL:

```
- -- Crear la base de datos "becarios" si no existe
CREATE DATABASE becarios;

-- Usar la base de datos "becarios"
USE becarios;

-- Crear la tabla Alumnos
CREATE TABLE Alumnos (
   ID INT PRIMARY KEY,
   Nombre VARCHAR(50),
   Apellido VARCHAR(50),
   DUI VARCHAR(15)
);

-- Crear la tabla Materias
```

```
CREATE TABLE Materias (
    ID INT PRIMARY KEY,
    NombreMateria VARCHAR(100)
);

-- Crear la tabla Notas
CREATE TABLE Notas (
    ID INT PRIMARY KEY,
    ID_Alumno INT,
    ID_Materia INT,
    Nota DECIMAL(4, 2), -- Cambia el tipo de dato según tus necesidades
    Fecha DATE,
    FOREIGN KEY (ID_Alumno) REFERENCES Alumnos(ID),
    FOREIGN KEY (ID_Materia) REFERENCES Materias(ID)
);
```





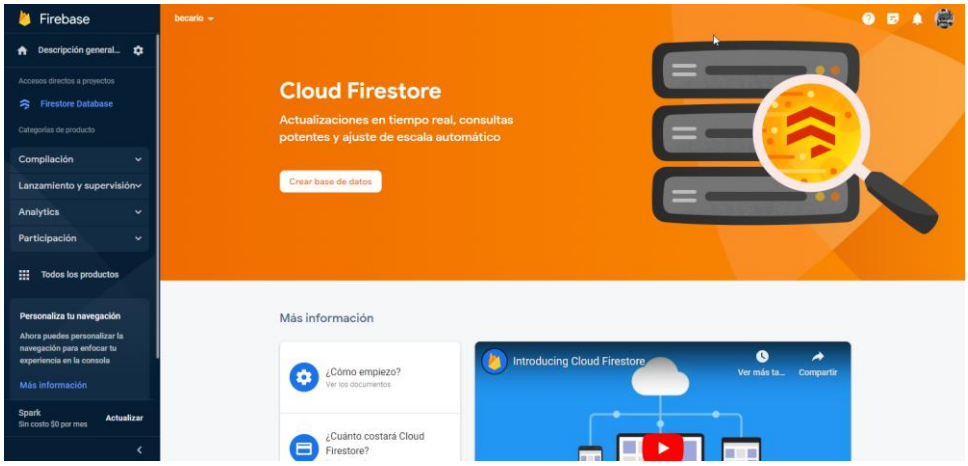
Base de Datos NoSQL (Firebase Firestore):

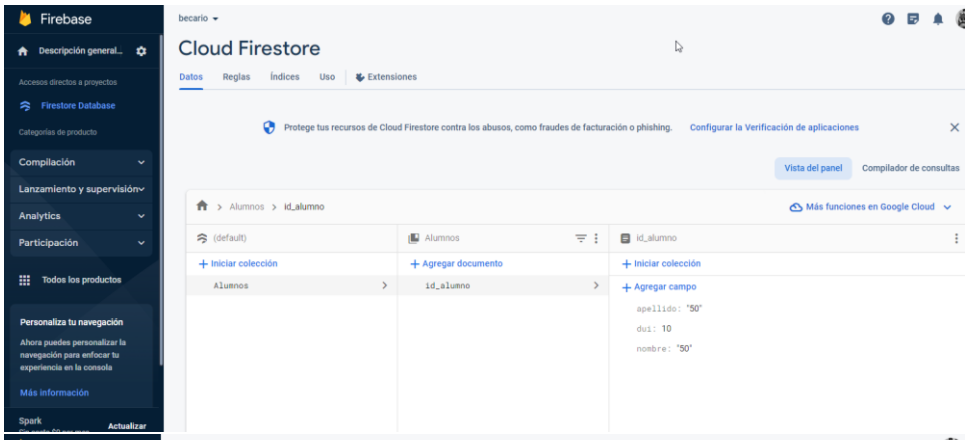
En el caso de la base NoSQL como Firestore, están diseñadas para proporcionar una mayor flexibilidad en la estructura de datos, lo que significa que no siguen el enfoque estricto de normalización. Firestore no tiene tablas, sino colecciones y documentos.

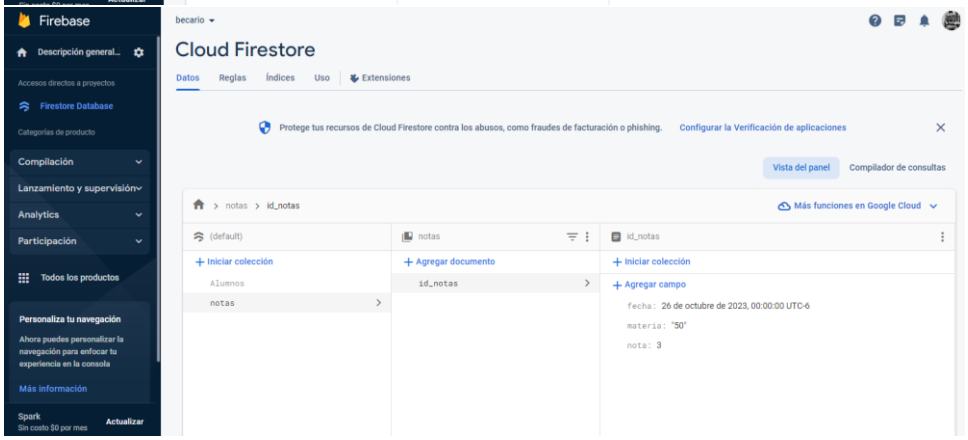
Colección "Alumnos":

- Documento por cada alumno con un ID único.
    - Nombre
    - Apellido
    - DUI
    - **Subcolección "Notas":**
        - Documento por cada nota con un ID único.
            - Materia
            - Nota
            - Fecha

Creación de base de datos:







Con esta estructura cada documento de alumno contendrá sus datos personales y una subcolección de notas en la que se puede almacenar las notas de diferentes materias con las fechas correspondientes.

Comparación entre la base de datos relacional y la base de datos NoSQL :

Base de Datos Relacional

- Las tablas se utilizan para almacenar conjuntos de datos relacionados con una estructura predefinida y rígida.
- Cada fila en la tabla representa una entidad específica, como un alumno o una materia, con atributos específicos.
- Se utilizan claves primarias y claves externas para establecer relaciones entre diferentes tablas y mantener la integridad referencial.
- Los datos se organizan en filas y columnas, y cada columna tiene un tipo de datos específico y restricciones definidas.

Colecciones en la Base de Datos NoSQL:

- Las colecciones almacenan conjuntos de documentos no estructurados, donde cada documento es una entidad independiente.
- Cada documento puede contener diferentes campos y estructuras, lo que proporciona flexibilidad en la organización de los datos.
- No hay una estructura de esquema fijo y las colecciones pueden contener documentos con diferentes estructuras de datos.
- Los datos se almacenan en formato JSON o BSON, lo que permite un almacenamiento más flexible y dinámico.

Similitudes entre Tablas y Colecciones:

- Ambos almacenan conjuntos de datos relacionados, ya sean entidades en el caso de las tablas o documentos en el caso de las colecciones.
- Tanto las tablas como las colecciones permiten la recuperación eficiente de datos relacionados con consultas específicas.
- Ambos pueden utilizar identificadores únicos para cada fila o documento, lo que facilita el acceso y la gestión de datos.
- Tanto las tablas como las colecciones pueden contener relaciones entre diferentes entidades, aunque la forma en que se gestionan estas relaciones difiere entre los modelos de base de datos relacional y NoSQL.

En conclusión, aunque las tablas y las colecciones tienen diferencias fundamentales en términos de estructura y flexibilidad, ambas tienen como objetivo almacenar conjuntos de datos relacionados y permitir la recuperación eficiente de información. Las tablas proporcionan una estructura predefinida y relaciones más estrictas, mientras que las colecciones permiten una mayor flexibilidad en la organización y almacenamiento de datos.

Referencias.

https://firebase.google.com/docs/firestore/rtdb-vs-firestore?hl=es-419