



Universidad Don Bosco, El Salvador

Nombres: Carlos Alfredo Artiga Mármol, AM221132

David Guillermo Campos Hernández, CH220048

Materia: DPS

Tema: Investigación para foro de discusión número 1

Repositorio: <https://github.com/The-301/Foro1-DPS.git>

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

- **Modelo de datos:**

-SQL: Las bases de datos SQL utilizan un modelo de datos tabular y relacional, donde los datos se organizan en tablas con filas y columnas. Cada fila representa un registro y las tablas están vinculadas mediante relaciones.

-NoSQL: Las bases de datos NoSQL utilizan un modelo de datos no tabular y no relacional. Pueden adoptar varios modelos, como documentos, columnas, grafos o clave-valor, y no requieren un esquema fijo.

- **Esquema:**

-SQL: Las bases de datos SQL tienen un esquema rígido y predefinido, lo que significa que la estructura de la base de datos debe definirse antes de insertar datos.

-NoSQL: Las bases de datos NoSQL suelen ser esquemas flexibles y permiten agregar campos sin requerir una estructura fija. Esto es especialmente útil en aplicaciones con cambios frecuentes en los requisitos de datos.

- **Escalabilidad:**

-SQL: Las bases de datos SQL suelen escalar verticalmente, lo que significa que se agregan más recursos a un único servidor para aumentar su capacidad. Esto puede ser costoso y tiene un límite.

-NoSQL: Las bases de datos NoSQL se diseñan para escalar horizontalmente, lo que implica agregar más servidores a la infraestructura. Esto proporciona una escalabilidad más sencilla y rentable.

- **Transacciones:**

-SQL: Las bases de datos SQL son ideales para aplicaciones que requieren transacciones ACID (Atomicidad, Consistencia, Aislamiento y Durabilidad). Garantizan la integridad de los datos, pero pueden ser menos eficientes para aplicaciones de alto rendimiento.

- **Consultas:**

-SQL: Las bases de datos SQL admiten consultas complejas utilizando SQL, lo que las hace adecuadas para aplicaciones con requisitos de consulta sofisticados.

-NoSQL: Las bases de datos NoSQL son más adecuadas para aplicaciones con necesidades de consulta más simples o donde se prioriza la velocidad de lectura/escritura sobre la complejidad de la consulta.

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

- **Modelo de datos:**

- Cloud Firestore: Utiliza un modelo de datos de documentos y colecciones. Los datos se organizan en documentos JSON anidados dentro de colecciones.

- Realtime Database: Utiliza un modelo de datos JSON en tiempo real basado en un árbol de datos. Los datos se almacenan como pares clave-valor.

- **Escalabilidad:**

- Cloud Firestore: Proporciona una escalabilidad más sencilla y eficiente, ya que permite consultas y escalabilidad horizontal de forma nativa.

- Realtime Database: Ofrece escalabilidad en tiempo real, pero puede requerir más esfuerzo de optimización para aplicaciones altamente escalables.

- **Consultas:**

- Cloud Firestore: Admite consultas más ricas y estructuradas, incluyendo consultas compuestas y filtrado avanzado.

- Realtime Database: Ofrece consultas más simples y menos flexibilidad en comparación con Cloud Firestore.

- **Conexión en tiempo real:**

-Cloud Firestore: Ofrece una sincronización en tiempo real de los datos y una función de escucha que permite a las aplicaciones reaccionar a cambios en los datos en tiempo real.

-Realtime Database: Es conocida por su sincronización en tiempo real, lo que la hace ideal para aplicaciones que requieren actualizaciones en tiempo real.

3. ¿Cuál de estas bases de datos consideran que sería la mejor opción para implementar en una aplicación desarrollada en React Native?

La elección entre Cloud Firestore y Realtime Database para una aplicación React Native dependerá de los requisitos específicos de la aplicación. Por ejemplo, si la aplicación necesita una escalabilidad sencilla, una estructura de datos más rica y una flexibilidad en las consultas, Cloud Firestore es una buena opción. Pero no me confío aún, ya que, por otro lado, si la aplicación se centra en la sincronización en tiempo real y los datos se organizan de manera simple, Realtime Database podría ser la más adecuada.

Ya que en general, Cloud Firestore se considera la opción más versátil y escalable, y es la elección preferida para muchas aplicaciones. Entonces podemos concluir que como mejor opción es Cloud Firestore, y como opinión personal ya he trabajado anteriormente con ella, así que me resulta mejor.

Ejercicio Practico

SQL Server

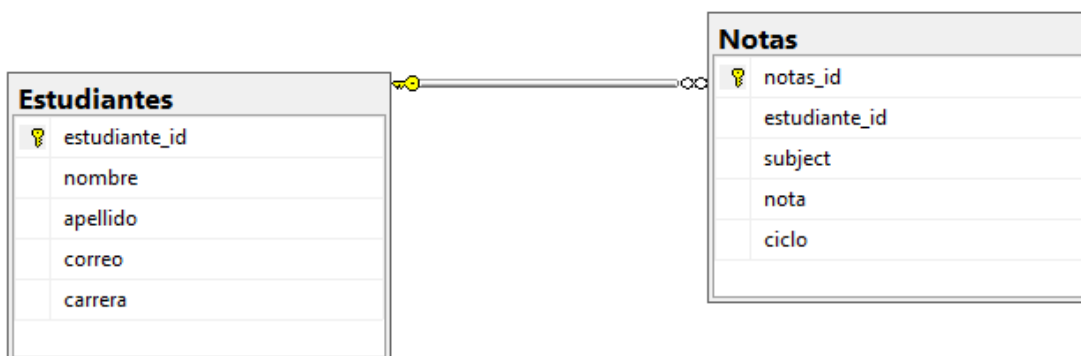
```
--Creamos la DB primero
CREATE DATABASE notas_udb_virtual;
go

--Utilizamos la BD para no meter las tablas en la master
USE notas_udb_virtual;
go

-- Se crea la tabla de estudiantes
CREATE TABLE Estudiantes (
    estudiante_id INT PRIMARY KEY,
    nombre NVARCHAR(50),
    apellido NVARCHAR(50),
    correo NVARCHAR(100),
    carrera NVARCHAR(50)
);

-- Creamos la tabla de notas
CREATE TABLE Notas (
    notas_id INT PRIMARY KEY,
    estudiante_id INT,
    subject NVARCHAR(50),
    nota DECIMAL(4, 2),
    ciclo NVARCHAR(10)
    -- Otros campos relacionados con las notas, si es necesario
);

-- Establecer una relación entre las tablas para relacionar notas con estudiantes
ALTER TABLE Notas
ADD CONSTRAINT FK_Estudiantes_Notas FOREIGN KEY (estudiante_id)
REFERENCES Estudiantes(estudiante_id);
```



FireBase

notas-udb-virtual

Cloud Firestore

?

Vista del panel

Compilador de consultas

🏠

>

estudiantes

>


Pv7K9J1bwiDm..

Más funciones en Google Cloud





<div>(default)</div> <div>+ Iniciar colección</div> <div>estudiantes</div> <div>notas</div>	<div>estudiantes</div> <div>+ Agregar documento</div> <div>Pv7K9J1bwiDmHEGoWVGv</div>	<div>Pv7K9J1bwiDmHEGoWVGv</div> <div>+ Iniciar colección</div> <div>+ Agregar campo</div> <div>apellido: "Campos"</div> <div>carrera: "Tec. Computacion"</div> <div>correo: "dcampos@udbvirtual.com"</div> <div>estudiante_id: 54698</div> <div>nombre: "David"</div>
---	---	---

Ubicación de la base de datos: nam5

[Vista del panel](#)
[Compilador de consultas](#)


[> notas](#)
[> EBZbn0lHgWn3..](#)


[Más funciones en Google Cloud](#)


 (default)	 notas	 EBZbn0lHgWn3wFP7b4D4
+ Iniciar colección	+ Agregar documento	+ Iniciar colección
estudiantes	EBZbn0lHgWn3wFP7b4D4	+ Agregar campo
 notas		<pre>ciclo: 2 id_materia: 34284 materia: "DPS" nota: 7.5</pre>

 Ubicación de la base de datos: nam5



Protege tus recursos de Cloud Firestore contra los abusos, como fraudes de facturación o phishing.

[Configurar la Verificación de aplicaciones](#)



Vista del panel

Compilador de consultas

Ejecutar

Borrar

[Ver la documentación](#)

Alcance de la consulta

Colección

Ruta de acceso

/estudiantes

/estudiantes

/notas

[+ Agregar a la consulta](#)

Resultados de la consulta



Document ID	apellido	carrera	correo	estudiante_id	nombre
Pv7K9J1bwiDmHEGoWVGv	"Campos"	"Tec. Computacion"	"dcampos@udbvirtual.com"	54698	"David"

Items per page: 50 1 - 1 de 1

 Ubicación de la base de datos: nam5

Conclusión

La noSQL en este caso que es FireBase ya que es más intuitiva y sobre todo es más grafica que SQL, como se ve en SQL debo ejecutar la mayoría de cosas en scripts, y la parte visual es más difícil de usar y un poco confusa, además se pueden aprovechar las capacidades de configuración de FireBase que esta proporciona como inserciones más directas y más amigables con los lenguajes modernos de programación, ya que conectar la base de datos de SQL Server cuesta un poco mas y las consultas y demás configuraciones deben de ser precisas y para personas principiantes esto puede ser difícil.