*Artículo original/ Original article*
*ISSN 2953-6553*

# Análisis comparativo de bases de datos relacionales y NoSQL: criterios de selección para aplicaciones web

# Comparative analysis of relational and NoSQL Databases: Selection Criteria for Web Applications

María Gabriela Aguirre Vicuña [1] 0000-0003-3046-6939, Mateo Josue Echeverria Betancourt[2] 0009-0000-9834-2882
maria.aguirre@ucacue.edu.ec, mateo.echeverria.33@est.ucacue.edu.ec

[1] Universidad Católica de Cuenca, Azuay/Cuenca, Ecuador.
[2] Universidad Católica de Cuenca, Azuay / Cuenca, Ecuador.

DOI 10.36500/atenas.5.011

***Resumen***

**La elección entre un sistema de bases de datos relacional y uno NoSQL constituye un componente esencial en el desarrollo de aplicaciones web modernas. El objetivo de este trabajo fue comparar las características, limitaciones, ventajas y principales casos de uso de ambos modelos, para que oriente la selección más adecuada según las necesidades de cada proyecto. La investigación se desarrolló mediante la revisión documental de la literatura científica y documentación técnica oficial, analizando los criterios de escalabilidad, flexibilidad, seguridad, consistencia, rendimiento y estructura de datos. Los resultados evidencian que las bases de datos NoSQL destacan por su escalabilidad horizontal, flexibilidad estructural y capacidad para gestionar grandes volúmenes de datos distribuidos, mientras que las bases de datos relacionales ofrecen mayores garantías de integridad y consistencia, mediante las propiedades de ACID, siendo apropiadas para aplicaciones transaccionales. Finalmente se concluye que no existe un modelo superior en todos los escenarios y que la selección responde a los requerimientos funcionales y no funcionales de cada aplicación, considerando la tendencia actual hacia arquitecturas de persistencia políglota.**

***Abstract***

**Choosing between a relational database system and a NoSQL system is a crucial aspect of modern web application development. This study aimed to compare the characteristics, limitations, advantages, and primary use cases of both models to guide the selection of the most suitable option based on specific project needs. The research was conducted through a review of scientific literature and official technical documentation, analyzing criteria such as scalability, flexibility, security, consistency, performance, and data structure. The results demonstrate that NoSQL databases excel in horizontal scalability, structural flexibility, and the ability to manage large volumes of distributed data, whereas relational databases offer superior guarantees regarding integrity and consistency leveraging ACID properties making them well-suited for transactional applications. Ultimately, the study concludes that no single model is superior in every scenario; rather, the choice depends on the functional and non-functional requirements of each application, while also considering the current trend toward polyglot persistence architectures.**

***Palabras claves: Base de Datos Relacionales, Base de Datos NoSQL, Aplicaciones Web, Escalabilidad***
***Keywords: Relational Databases, NoSQL databases, Web applications, Scalability***

Recibido: 2026-07-17, Aprobado tras revisión: 2026-09-29

## I. INTRODUCCIÓN

Las bases de datos constituyen un componente fundamental en el desarrollo web, debido a que permiten almacenar, organizar, consultar y administrar la información necesaria para su funcionamiento. El modelo relacional propuesto por Codd (1970), estableció fundamentos teóricos para la representación estructurada de datos porque tiene una base matemática sólida, usa un lenguaje estándar de consultas (SQL) y mantiene la información ordenada y sin errores. Posteriormente, el crecimiento de las aplicaciones web y la diversificación de los tipos y volúmenes de la información impulsaron el desarrollo y la adopción de tecnologías NoSQL, caracterizadas por incorporar distintos modelos de almacenamiento y mecanismos de distribución de datos (Cattell, 2011; Han et al., 2011).

La evolución de los sistemas de gestión de bases de datos ha ampliado las alternativas disponibles para el desarrollo de aplicaciones web. Los gestores relacionales pueden incorporar mecanismos de replicación, particionamiento y escalabilidad distribuida, mientras que determinadas tecnologías NoSQL ofrecen capacidades transaccionales, validación de esquemas y diferentes niveles de consistencia. Estas características evidencian que las diferencias entre ambos enfoques no deben establecerse mediante clasificaciones absolutas, sino considerando las funcionalidades particulares de cada gestor, su arquitectura y configuración. Asimismo, la evaluación del rendimiento requiere analizar las cargas de trabajo y las condiciones de ejecución, debido a que los resultados de estudios comparativos no son necesariamente generalizables a otros escenarios (Taipalus, 2024).

El problema es que, al existir estos dos tipos de bases de datos, el desarrollador se enfrenta con frecuencia a la misma interrogante: cuál le conviene más para su proyecto. Esta decisión no es menor, porque influye en el rendimiento, la consistencia de los datos, los costos y lo fácil o difícil que será mantener la aplicación. De aquí nace la pregunta que guía esta investigación: ¿qué tipo de base de datos, relacional o NoSQL, ofrece mayores ventajas para una aplicación web moderna que maneja grandes volúmenes de información y un crecimiento constante de usuarios? El objetivo del trabajo es analizar comparativamente las características técnicas, capacidades y limitaciones de las bases de datos relacionales y NoSQL, a partir de la literatura científica y documentación técnica, con el propósito de establecer criterios de decisión que orienten la selección de tecnologías de almacenamiento para páginas web.

En este contexto, la selección de una tecnología de almacenamiento representa una decisión técnica que influye en el rendimiento, la integridad de los datos, la escalabilidad, la seguridad y el mantenimiento de las aplicaciones web. La diversidad de soluciones disponibles plantea la necesidad de establecer criterios que permitan relacionar los requisitos funcionales y no funcionales de un proyecto con las capacidades de los sistemas de gestión de bases de datos. Por consiguiente, se formula la siguiente pregunta de investigación: ¿qué criterios técnicos deben considerarse para orientar la selección entre tecnologías de bases de datos relacionales y NoSQL en el desarrollo de aplicaciones web?

## II. MÉTODO

La investigación se desarrolló mediante un enfoque cualitativo, de tipo documental, sustentado en una revisión bibliográfica narrativa de literatura científica y documentación técnica relacionada con los sistemas de gestión de bases de datos relacionales y NoSQL. Este enfoque permitió analizar las características, capacidades y limitaciones de ambos modelos, con el propósito de establecer criterios técnicos que orienten su selección en el desarrollo de aplicaciones web.

De acuerdo con Snyder (2019), las revisiones de literatura permiten integrar conocimientos procedentes de diferentes investigaciones, siempre que exista claridad sobre el propósito de la revisión y el procedimiento utilizado. Por su parte, Grant y Booth (2009) señalan que las revisiones bibliográficas pueden adoptar distintos niveles de sistematización, según sus objetivos y la forma en que se identifican, seleccionan y analizan las fuentes.

A partir de estas consideraciones, el estudio se orientó hacia un análisis comparativo de las propiedades técnicas de ambos modelos de bases de datos, tomando como referencia aspectos relacionados con la estructura de la información, escalabilidad, rendimiento, consistencia, flexibilidad y seguridad.

## III. RESULTADOS Y DISCUSIÓN

### Bases de datos relacionales

Una base de datos relacional es aquella que guarda la información en tablas (también llamadas relaciones), formadas por filas y columnas, donde cada fila es un registro y cada columna representa un atributo. Este modelo fue planteado por Codd (1970) y se basa en la teoría de conjuntos y en la lógica. Para trabajar con los datos utiliza principalmente el lenguaje SQL, y las tablas se relacionan entre sí por

medio de claves primarias y foráneas. La normalización permite organizar la información y reducir redundancias, mientras que las restricciones contribuyen a mantener su integridad. Algo muy característico de estos sistemas es el soporte de transacciones con propiedades ACID (atomicidad, consistencia, aislamiento y durabilidad), que permiten controlar la ejecución de operaciones y la recuperación ante fallos, de acuerdo con las garantías de cada gestor (Codd, 1970; Stonebraker, 2010).

Algunos de los gestores más conocidos que usan este modelo son PostgreSQL, MySQL y Oracle. La documentación oficial define a PostgreSQL como un sistema objeto-relacional de código abierto, que incorpora mecanismos de integridad, consultas SQL y diferentes opciones de administración de datos (PostgreSQL Global Development Group, s. f.). Entre las ventajas de estos sistemas están la integridad de la información, el uso de un lenguaje de consulta ampliamente adoptado y la posibilidad de resolver consultas complejas que combinan varias tablas mediante uniones (joins). Sin embargo, cuando la estructura de los datos cambia con frecuencia, pueden requerirse modificaciones y migraciones del esquema. Aunque tradicionalmente se asocian con la escalabilidad vertical, también existen alternativas de replicación, particionamiento y distribución, por lo que sus posibilidades de crecimiento dependen del gestor y de la arquitectura utilizada (Cattell, 2011; PostgreSQL Global Development Group, s. f.).

**Bases de datos NoSQL**

El término NoSQL (que significa Not only SQL) se usa para un grupo variado de sistemas que no siguen necesariamente el modelo relacional y que emplean diferentes mecanismos de almacenamiento y consulta (Han et al., 2011). Su desarrollo estuvo relacionado con las necesidades de aplicaciones web que manejan grandes cantidades de datos y requieren distribuir la información entre varios servidores. Según Sadalage y Fowler (2012), una característica de varios de estos sistemas es la flexibilidad de su esquema, lo que facilita almacenar datos semiestructurados y modificar su estructura. Sin embargo, esta flexibilidad no significa que todos carezcan de validaciones o restricciones. En la literatura se identifican cuatro tipos principales (Cattell, 2011; Han et al., 2011), que se resumen en la Tabla 1; por ejemplo, MongoDB guarda la información en documentos de formato BSON, relacionado con la representación JSON (MongoDB, Inc., s. f.).

**Tabla 1**

*Tipos de Bases de Datos NoSQL*

| Tipo | Descripción | Ejemplos |
|---|---|---|
| Clave-valor | Guardan pares de clave y valor; son muy rápidas y de modelo sencillo. | Redis, Amazon DynamoDB |
| Documental | Almacenan documentos flexibles (JSON/BSON) con estructura variable. | MongoDB, CouchDB |
| Orientada a columnas | Organizan los datos en familias de columnas; buenas para grandes analíticas. | Apache Cassandra, HBase |
| Grafos | Representan entidades y sus relaciones; ideales para datos muy conectados. | Neo4j, Amazon Neptune |

*Nota. Adaptado de Cattell (2011) y Han et al. (2011).*

Las bases de datos NoSQL presentan distintos modelos de consistencia y capacidades transaccionales, por lo que no es adecuado atribuirles de manera generalizada un comportamiento basado exclusivamente en el modelo BASE. Algunas implementaciones priorizan la disponibilidad y admiten consistencia eventual, mientras que otras incorporan garantías transaccionales. Por ejemplo, MongoDB admite transacciones multidocumento bajo las condiciones y restricciones establecidas por el sistema (MongoDB, Inc., s. f.; Davoudian et al., 2018). En consecuencia, las diferencias entre bases de datos relacionales y NoSQL deben analizarse considerando las características específicas de cada gestor y su configuración.

**Modelos de consistencia: ACID, BASE y el teorema CAP**

Para entender mejor esta diferencia es útil mencionar el teorema CAP, formulado a partir de la propuesta de Brewer y desarrollado formalmente por Gilbert y Lynch (2002). Este plantea que, ante una partición de red, un sistema distribuido no puede garantizar simultáneamente consistencia y disponibilidad en los términos definidos por el teorema. Esto no significa que siempre deba elegirse entre tres propiedades ni que todos los gestores relacionales se comporten de una manera y todos los NoSQL de otra. Además, ACID describe propiedades de las transacciones, mientras que BASE se relaciona con enfoques de disponibilidad y consistencia eventual; por tanto, no son equivalentes al teorema CAP (Gilbert y Lynch, 2002; Davoudian et al., 2018).

**Análisis comparativo**

Tomando en cuenta los seis criterios que se mencionaron en la metodología, en la Tabla 2 se resumen las principales diferencias técnicas entre las bases de datos relacionales y las NoSQL. La comparación muestra que las capacidades de estructura, escalabilidad, consistencia, rendimiento, flexibilidad y seguridad no dependen únicamente de la clasificación SQL o NoSQL, sino también de las características del gestor, la configuración y las necesidades de la aplicación (Cattell, 2011; Davoudian et al., 2018).

**Tabla 2**

*Comparación técnica entre bases de datos relacionales y NoSQL*

| Criterio | Bases de datos relacionales (SQL) | Bases de datos NoSQL |
|---|---|---|
| Estructura de datos | Tablas con filas y columnas; esquema fijo. Organizan la información mediante tablas relacionadas. Generalmente utilizan esquemas definidos, aunque algunos gestores admiten datos semiestructurados. | Documentos, clave-valor, columnas o grafos; esquema flexible. Comprenden modelos documentales, clave-valor, de columnas y de grafos. La flexibilidad del esquema depende del gestor. |
| Escalabilidad | Admiten escalabilidad vertical y alternativas horizontales, como réplicas de lectura, particionamiento y arquitecturas SQL distribuidas. | Muchos gestores incorporan mecanismos de distribución horizontal, como fragmentación y replicación; su implementación depende del sistema. |
| Consistencia | Ofrecen mecanismos transaccionales ACID, con garantías y niveles de aislamiento definidos por cada gestor. | Las garantías varían según el sistema. Algunos ofrecen consistencia eventual y otros admiten transacciones ACID multidocumento, como MongoDB. |
| Rendimiento | Puede resultar favorable para determinadas consultas relacionales, operaciones JOIN y cargas transaccionales. Depende del diseño, los índices y la configuración. | Puede resultar favorable para determinados patrones de acceso, modelos documentales y cargas distribuidas. Depende de la implementación y del escenario evaluado. |
| Flexibilidad | Permiten modificar esquemas y utilizar tipos de datos flexibles, | Algunos modelos permiten estructuras variables entre |

| | | |
|---|---|---|
| | aunque los cambios pueden requerir migraciones y ajustes de integridad. | registros; también pueden aplicar validaciones y restricciones de esquema. |
| Seguridad | Pueden incorporar autenticación, autorización, cifrado y controles de acceso. Requieren medidas contra la inyección SQL. | También pueden incorporar autenticación, autorización, cifrado y controles de acceso. Requieren medidas contra la inyección NoSQL y otras vulnerabilidades. |

*Nota. Elaboración propia a partir de Cattell (2011), Davoudian et al. (2018), Sadalage y Fowler (2012), Taipalus (2024), MongoDB, Inc. (s. f.) y Google Cloud (s. f.). Las características presentadas describen capacidades generales y no comportamientos universales; su implementación depende del gestor, la arquitectura, la configuración y la carga de trabajo.*

En cuanto al rendimiento, los estudios que se han hecho no dan un resultado único. Makris et al. (2021), al comparar MongoDB y PostgreSQL en el escenario evaluado por los autores, encontraron diferencias en los tiempos de respuesta según las operaciones realizadas. Este resultado muestra que una tecnología puede responder mejor que otra en determinadas consultas, sin que ello permita afirmar que será más rápida en cualquier aplicación. La evaluación requiere considerar los datos utilizados, los índices, las cargas de trabajo, la configuración y las condiciones en las que se ejecutan las pruebas (Makris et al., 2021; Taipalus, 2024).

Taipalus (2024), mediante una revisión sistemática de estudios comparativos de gestores de bases de datos, identificó limitaciones en la representatividad de algunas cargas de trabajo y en la información disponible para reproducir los experimentos. Este hallazgo permite discutir los resultados de Makris et al. (2021) con mayor cuidado: sus conclusiones son útiles para el escenario estudiado, pero no deben extenderse automáticamente a todas las aplicaciones web. Por ello, antes de seleccionar un gestor conviene realizar pruebas que representen las operaciones reales del proyecto.

Otra diferencia importante es la flexibilidad. En las bases relacionales suelen definirse esquemas y restricciones que ayudan a organizar los datos, pero ciertos cambios pueden requerir ajustes y migraciones. En cambio, algunos sistemas NoSQL permiten modificar con mayor facilidad la estructura de los documentos. Esta ventaja depende del modelo utilizado, porque también existen gestores relacionales con soporte para datos semiestructurados y bases NoSQL que aplican validaciones de esquema (Sadalage y Fowler, 2012; PostgreSQL Global Development Group, s. f.; MongoDB, Inc., s. f.). Por tanto, la flexibilidad debe evaluarse junto con las necesidades de integridad, mantenimiento y evolución de la aplicación.

La seguridad constituye un criterio fundamental en la selección de sistemas de gestión de bases de datos, independientemente del modelo de almacenamiento utilizado. Su evaluación comprende mecanismos de autenticación, autorización, cifrado, auditoría y protección frente a vulnerabilidades. Estas capacidades deben analizarse de manera independiente de las garantías de integridad y consistencia transaccional, puesto que las propiedades ACID no sustituyen los controles de seguridad (PostgreSQL Global Development Group, s. f.; MongoDB, Inc., s. f.).

En los sistemas relacionales, como PostgreSQL, la administración de roles y privilegios permite establecer restricciones de acceso a los objetos de la base de datos. Asimismo, pueden implementarse mecanismos de autenticación y conexiones cifradas. En sistemas NoSQL, como MongoDB, también se dispone de autenticación, control de acceso basado en roles y mecanismos de cifrado (PostgreSQL Global Development Group, s. f.; MongoDB, Inc., s. f.). Por consiguiente, no resulta adecuado afirmar que un modelo es intrínsecamente más seguro que otro únicamente por su estructura de almacenamiento.

Otro aspecto relevante corresponde a las vulnerabilidades de inyección. En las bases de datos relacionales, la inyección SQL puede producirse cuando una aplicación construye consultas mediante la incorporación insegura de entradas proporcionadas por los usuarios. De manera similar, determinadas implementaciones NoSQL pueden presentar vulnerabilidades cuando las entradas no confiables modifican operadores o estructuras de consulta. La prevención requiere utilizar mecanismos seguros de construcción de consultas, validar las entradas y aplicar el principio de mínimo privilegio. Se trata, por tanto, de un riesgo asociado también a la implementación de la aplicación y no solamente al modelo de datos.

En consecuencia, la selección de un gestor debe considerar las funcionalidades de seguridad disponibles, su configuración y los requerimientos de protección de la aplicación. La evaluación debe contemplar tanto las características del producto como las prácticas de implementación y administración.

**Casos de uso**

En la literatura se reconocen situaciones en las que cada modelo puede resultar conveniente, como se muestra en la Tabla 3. Las bases de datos relacionales suelen ser una alternativa apropiada para aplicaciones que necesitan transacciones e integridad estricta, como los sistemas bancarios y financieros, los de gestión empresarial o la parte transaccional de una tienda en línea (Stonebraker, 2010; Sadalage y Fowler, 2012). En estos casos, una operación incompleta, por ejemplo, una transferencia que descuenta dinero de una cuenta, pero no lo deposita en otra, sería un problema grave. No obstante, el criterio

decisivo es que el gestor seleccionado ofrezca las garantías transaccionales requeridas, ya que determinadas tecnologías NoSQL también pueden incorporarlas (MongoDB, Inc., s. f.).

Las bases NoSQL, en cambio, pueden acomodarse bien a aplicaciones que manejan grandes cantidades de datos heterogéneos, que cambian con frecuencia o que necesitan distribución de la carga. Algunos ejemplos son las redes sociales, las plataformas de Big Data y de Internet de las Cosas, los sistemas de mensajería y la gestión de contenidos (Cattell, 2011; Han et al., 2011). Por ejemplo, una base documental como MongoDB puede guardar perfiles de usuario con estructuras distintas, mientras que una base orientada a columnas, como Cassandra, puede utilizarse para determinadas cargas distribuidas. Sin embargo, estos escenarios no excluyen a los gestores relacionales: la elección depende de las consultas, las relaciones entre datos, las garantías requeridas y la capacidad de operación del sistema.

**Tabla 3**

*Casos de uso recomendados según el modelo de base de datos*

| Contexto | Requisitos Técnicos | Alternativas que deben evaluarse |
|---|---|---|
| Sistemas financieros | Transacciones, consistencia, auditoría e integridad. | Gestores SQL o NoSQL con garantías transaccionales adecuadas. |
| Comercio electrónico | Pagos, inventarios, catálogos y consultas. | SQL, NoSQL o arquitectura híbrida según el componente. |
| Redes sociales | Datos heterogéneos, relaciones y crecimiento. | Gestores relacionales, documentales o de grafos. |
| Internet de las Cosas | Alta frecuencia de escritura y almacenamiento distribuido. | Gestores especializados, SQL o NoSQL según la carga. |
| Gestión de contenidos | Estructuras variables y consultas de recuperación. | Bases documentales o relacionales con soporte para datos semiestructurados. |

*Nota. Elaboración propia. Los escenarios presentados constituyen ejemplos de aplicación y las alternativas tecnológicas se plantean en función de sus requisitos técnicos. La selección de un sistema de gestión de bases de datos depende de las necesidades de consistencia, escalabilidad, rendimiento, flexibilidad y seguridad de cada aplicación.*

Cabe señalar que los dos enfoques no son excluyentes entre sí. Sadalage y Fowler (2012) hablan de la persistencia políglota (polyglot persistence), que consiste en que una misma aplicación use distintas tecnologías de almacenamiento según lo que necesite. Por ejemplo, una tienda en línea podría usar una

base relacional para los pagos y una base documental para determinadas funciones del catálogo. Esta combinación puede resultar útil cuando existen requisitos diferentes, aunque también aumenta la complejidad de integración, mantenimiento y consistencia entre sistemas.

A partir de los criterios técnicos examinados en la revisión bibliográfica, se propone una matriz de apoyo a la selección de tecnologías de bases de datos para aplicaciones web. Este instrumento relaciona los requisitos de un proyecto con las capacidades que deben evaluarse en los gestores relacionales y NoSQL, considerando que ambos enfoques pueden ofrecer funcionalidades similares mediante diferentes arquitecturas y configuraciones. La matriz no pretende determinar una tecnología universalmente superior, sino orientar un proceso de evaluación fundamentado en las necesidades específicas de cada aplicación. Constituye una síntesis elaborada a partir de la revisión documental y no el resultado de una validación experimental.

**Tabla 4**

*Matriz de criterios para la selección de tecnologías de bases de datos*

| Criterio | Condición del proyecto | Orientación para la selección |
|---|---|---|
| Estructura de datos | Datos estructurados con relaciones y restricciones definidas. | Evaluar gestores relacionales. |
| Flexibilidad | Documentos heterogéneos o estructuras que cambian frecuentemente. | Evaluar gestores documentales o relacionales con soporte para datos semiestructurados. |
| Consistencia | Operaciones que requieren transacciones atómicas sobre varios registros. | Priorizar gestores con garantías ACID adecuadas, independientemente del modelo. |
| Escalabilidad | Crecimiento sostenido de usuarios y necesidad de distribuir cargas. | Evaluar mecanismos de escalabilidad horizontal disponibles en SQL y NoSQL. |
| Rendimiento | Consultas complejas, lecturas intensivas o escrituras de alta frecuencia. | Comparar gestores mediante pruebas representativas de la carga prevista. |
| Seguridad | Información sensible, permisos diferenciados y requisitos de auditoría. | Verificar autenticación, autorización, cifrado, auditoría y prevención de inyecciones. |

*Nota. Elaboración propia. La matriz constituye un instrumento orientativo de análisis técnico; no establece una superioridad universal entre modelos. La selección debe fundamentarse en los requisitos funcionales y no funcionales de cada aplicación.*

La aplicación de la matriz requiere identificar los requisitos prioritarios del proyecto y contrastarlos con las características verificables de los gestores candidatos. Cuando una aplicación presenta necesidades diferenciadas de almacenamiento, puede considerarse una arquitectura de persistencia políglota, siempre que su implementación resulte técnica y operativamente justificable (Sadalage y Fowler, 2012). Asimismo, los criterios relacionados con el rendimiento deben validarse mediante pruebas representativas, debido a que los resultados de estudios comparativos dependen de las condiciones experimentales y las cargas de trabajo evaluadas (Taipalus, 2024). Como limitación del presente trabajo, las orientaciones propuestas proceden de una revisión documental y no de pruebas propias de rendimiento o de una evaluación empírica de la matriz; por ello, deben utilizarse como punto de partida para decisiones que requieren verificación en cada proyecto.

## IV. CONCLUSIONES

Se concluye que no existe un modelo de base de datos que sea mejor que el otro en todos los casos, sino que cada uno responde mejor a ciertas necesidades. Respondiendo a la pregunta de investigación, lo que se encontró indica que, para una aplicación web que maneja grandes volúmenes de información y que tiene un crecimiento constante de usuarios, las bases de datos NoSQL ofrecen ventajas importantes en cuanto a escalabilidad horizontal, flexibilidad del esquema y disponibilidad, por lo que es una alternativa adecuada para escenarios con datos masivos y que cambian constantemente (Cattell, 2011; Han et al., 2011).

Sin embargo, esto no quiere decir que las bases relacionales vayan a desaparecer, porque siguen siendo la mejor alternativa cuando se necesita integridad y consistencia estricta, gracias a las propiedades ACID (Codd, 1970; Stonebraker, 2010). Además, los estudios revisados muestran que en muchos casos las bases relacionales, como PostgreSQL, mantienen un rendimiento muy bueno e incluso mejor que algunas soluciones NoSQL (Makris et al., 2021), así que la decisión no debería tomarse por moda, sino analizando bien lo que el proyecto realmente necesita.

Para finalizar, la elección entre una base relacional y una NoSQL depende de varios factores: el tipo y la cantidad de datos, qué tan estricta debe ser la consistencia, cuánta escalabilidad se necesita y el contexto de la aplicación. La tendencia actual hacia la persistencia políglota (Sadalage y Fowler, 2012) demuestra que lo más conveniente no siempre es quedarse con un solo modelo, sino combinar lo mejor de cada tecnología para crear aplicaciones más sólidas y escalables.

## REFERENCIAS BIBLIOGRÁFICAS


Cattell, R. (2011). Scalable SQL and NoSQL data stores. *ACM SIGMOD Record, 39*(4), 12–27. https://doi.org/10.1145/1978915.1978919

Codd, E. F. (1970). A relational model of data for large shared data banks. *Communications of the ACM, 13*(6), 377–387. https://doi.org/10.1145/362384.362685

Davoudian, A., Chen, L., & Liu, M. (2018). A survey on NoSQL stores. *ACM Computing Surveys, 51*(2), 1–43. https://doi.org/10.1145/3158661

Gilbert, S., & Lynch, N. (2002). Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. *ACM SIGACT News, 33*(2), 51–59. https://doi.org/10.1145/564585.564601

Google Cloud. (s. f.). *Optimizing schema design for Spanner*. https://docs.cloud.google.com/spanner/docs/whitepapers/optimizing-schema-design

Grant, M. J., & Booth, A. (2009). A typology of reviews: An analysis of 14 review types and associated methodologies. *Health Information & Libraries Journal, 26*(2), 91–108. https://doi.org/10.1111/j.1471-1842.2009.00848.x

Han, J., Haihong, E., Le, G., & Du, J. (2011). Survey on NoSQL database. In *2011 6th International Conference on Pervasive Computing and Applications* (pp. 363–366). IEEE. https://doi.org/10.1109/ICPCA.2011.6106531

Makris, A., Tserpes, K., Spiliopoulos, G., Zissis, D., & Anagnostopoulos, D. (2021). MongoDB vs PostgreSQL: A comparative study on performance aspects. *GeoInformatica, 25*(2), 243–268. https://doi.org/10.1007/s10707-020-00407-w

MongoDB, Inc. (s. f.). *MongoDB manual*. Recuperado el 18 de junio de 2026, de https://www.mongodb.com/docs/manual/

PostgreSQL Global Development Group. (s. f.). *PostgreSQL documentation*. Recuperado el 18 de junio de 2026, de https://www.postgresql.org/docs/

Sadalage, P. J., & Fowler, M. (2012). *NoSQL distilled: A brief guide to the emerging world of polyglot persistence*. Addison-Wesley.

Snyder, H. (2019). Literature review as a research methodology: An overview and guidelines. *Journal of Business Research, 104*, 333–339. https://doi.org/10.1016/j.jbusres.2019.07.039

Stonebraker, M. (2010). SQL databases v. NoSQL databases. *Communications of the ACM, 53*(4), 10–11. https://doi.org/10.1145/1721654.1721659

Taipalus, T. (2024). Database management system performance comparisons: A systematic literature review. *Journal of Systems and Software, 208*, Article 111872. https://doi.org/10.1016/j.jss.2023.111872