Universidad Nacional Autónoma de México

Facultad de Ingeniería

TAREA I

Modelo Orientado a Objetos y Modelos NoSQL

Asignatura: Bases de Datos

Profesor: Ing. Fernando Arreola

Estudiante: Ericka Aguilar Lara

No. de cuenta: 321227122

Grupo: 1

Universidad Nacional Autónoma de México

Ingeniería en Computación

7 de febrero de 2026

1. Modelo Orientado a Objetos

1.1. Descripción

El Modelo Orientado a Objetos es la base de las Bases de Datos Orientadas a Objetos (OODBMS). Este paradigma se caracteriza por integrar tanto los datos como sus relaciones en una única estructura denominada objeto [1], lo que permite una representación más natural de la información.

La orientación a objetos funciona como representación del mundo real mediante la resolución de problemas tangibles o digitales con ayuda de objetos. Este enfoque ve a los sistemas como entidades dinámicas compuestas por componentes que interactúan entre sí [2].

Un aspecto esencial de este modelo es la persistencia, característica que garantiza la recuperación futura de los datos. En los sistemas de gestión de base de datos orientada a objetos (OODBMS por sus siglas en inglés), la persistencia implica almacenar los valores de atributos de los objetos con un nivel de transparencia que permite a los desarrolladores trabajar directamente con el lenguaje de programación orientado a objetos, sin necesidad de implementar mecanismos adicionales de almacenamiento [2].

1.2. Ventajas

El modelo orientado a objetos destaca por su capacidad para representar el mundo real, facilitando la modelización de entidades y relaciones de forma natural. Entre sus principales ventajas se encuentran: la extensibilidad mediante herencia, el manejo eficiente de valores complejos, y elimina la desconexión entre los datos y la aplicación, lo que permite que trabajen mejor juntos [1].

Adicionalmente, ofrece lenguajes de consulta más expresivos y soporte para transacciones de larga duración, siendo especialmente adecuado para dominios especializados. En contextos como la ingeniería de software o el diseño asistido por computadora (CAD) [1].

Según IONOS las bases de datos orientadas a objetos permiten almacenar y consultar conjuntos de datos complejos de forma rápida y sencilla. Además, asignan automáticamente códigos de identificación a cada objeto y funcionan de manera óptima con lenguajes de programación orientados a objetos, facilitando la integración entre la capa de aplicación y la de persistencia [3].

1.3. Desventajas

No tiene un fundamento teórico sólido, lo que dificulta una definición precisa y universal del modelo de datos orientado a objetos. Además, no hay tantas personas especializadas en este paradigma, sumado a la falta de estándares consolidados en el mercado. La fuerte competencia de las bases de datos relacionales y objeto-relacionales también limita su adopción.

A nivel técnico, la encapsulación suele verse comprometida para optimizar el rendimiento de las consultas, mientras que la gestión de la concurrencia mediante bloqueos en jerarquías de herencia es compleja y afecta al rendimiento. El modelo es más complejo que el relacional, lo que se traduce en mayores costos de implementación y mantenimiento. Otras desventajas

incluyen la ausencia de vistas y mecanismos de control de acceso de grano grueso, insuficientes para muchas aplicaciones comerciales que requieren seguridad más detallada [1].

1.4. Casos de Uso

Las aplicaciones principales del modelo orientado a objetos incluyen el diseño asistido por ordenador (CAD) y entornos de desarrollo de software integrados (IDEs) [1].

2. Modelos NoSQL

NoSQL, también conocido como "no solo SQL" o "no SQL", es un enfoque para el diseño de bases de datos que permite el almacenamiento y la consulta de datos fuera de las estructuras tradicionales que se encuentran en las bases de datos relacionales [5].

2.1. Modelo Clave-Valor

2.1.1. Descripción

El modelo clave-valor es considerado la forma más simple de base de datos NoSQL. Se organiza como un diccionario de pares clave-valor sin esquema fijo, donde cada elemento tiene una clave única que sirve como identificador y un valor asociado que puede ser cualquier tipo de dato, desde simple texto hasta objetos complejos [5].

Este modelo es particionable (proceso de almacenar una base de datos grande en varias máquinas. Una sola máquina, o servidor de base de datos, puede almacenar y procesar únicamente una cantidad limitada de datos. La partición de bases de datos supera esta limitación al dividir los datos en trozos más pequeños, denominados particiones, y almacenarlos en varios servidores de bases de datos [6]) y permite un escalado horizontal (se plantea el escalado horizontal cuando no se pueden obtener suficientes recursos para las cargas de trabajo, incluso cuando funcionan con el máximo nivel de rendimiento. Con el escalado horizontal, los datos se dividen en varias bases de datos, o particiones, en varios servidores, y cada partición se puede escalar o reducir verticalmente de forma independiente [5]).

2.1.2. Ventajas

Los sistemas de bases de datos clave-valor ofrecen ventajas para aplicaciones que demandan rendimiento y crecimiento. Presentan una alta escalabilidad horizontal que supera a otros modelos NoSQL, permitiendo distribuir datos y consultas de manera eficiente a través de múltiples servidores [4, 5]. Están especialmente diseñados para alto rendimiento y baja latencia (la latencia se refiere al retraso que ocurre entre el momento en que un usuario realiza una acción en una red o aplicación web y el momento en que obtiene una respuesta [8]), garantizando tiempos de respuesta rápidos y consistentes incluso bajo cargas intensas [4]. Su simplicidad estructural facilita enormemente la implementación y el mantenimiento, mientras que su flexibilidad permite almacenar cualquier tipo de dato como valor, sin restricciones de esquema.

2.1.3. Desventajas

Este modelo también tiene limitaciones funcionales importantes. Ofrece una consulta limitada, ya que no es eficiente para extraer múltiples registros a la vez o ejecutar consultas complejas con filtros sofisticados [5]. Carece de capacidad para gestionar relaciones entre datos, lo que limita su uso en dominios con datos interconectados. Además, las consultas por valor son muy restringidas, pues el acceso principal es a través de la clave única, no mediante el contenido almacenado en el valor.

2.1.4. Casos de Uso

Son ampliamente utilizados para almacenamiento en caché y gestión de sesiones de usuario, como mantener el estado de un carrito de compras en línea [4, 5]. En el sector del entretenimiento, se usan en aplicaciones de juegos, manejando en tiempo real tablas de clasificación y datos de sesión de jugadores [4]. En tecnología publicitaria (AdTech) e Internet de las Cosas (IoT), donde se procesan millones de eventos y lecturas de sensores que requieren operaciones ultra rápidas [4].

2.2. Modelo Documental

2.2.1. Descripción

Las bases de datos documentales almacenan datos en formato de documentos semiestructurados, típicamente en JSON, XML o BSON. Cada documento es una unidad autocontenida con todos los datos relacionados, manteniendo una estructura jerárquica que se asemeja a los objetos en el código de las aplicaciones [4, 5]. Este modelo no requiere que los esquemas coincidan entre documentos, ofreciendo gran flexibilidad.

2.2.2. Ventajas

La principal ventaja es su esquema flexible, que permite un desarrollo iterativo rápido y adaptación a cambios sin migraciones costosas [4]. Presentan un modelo intuitivo que se alinea bien con estructuras de programación orientada a objetos, facilitando la integración entre la aplicación y la persistencia [4]. Los datos autocontenidos reducen la necesidad de operaciones de unión (joins) complejas [4, 5]. Además, permite la evolución natural de la estructura según cambian las necesidades de la aplicación [4].

2.2.3. Desventajas

Pueden presentar problemas con transacciones complejas que afecten múltiples documentos, lo que puede llevar a la corrupción de los datos [5]. Existe riesgo de redundancia al almacenar datos relacionados en un mismo documento, lo que puede llevar a duplicación. En configuraciones distribuidas, a menudo ofrecen solo consistencia eventual, lo que puede no ser adecuado para todas las aplicaciones.

2.2.4. Casos de Uso

Son muy versátiles y se usan en Sistemas de Gestión de Contenidos (CMS) como WordPress, donde cada artículo es un documento flexible [5]. Son ideales para perfiles de usuario y catálogos de productos en tiendas online, ya que cada ítem puede tener su propia estructura de datos sin un esquema fijo [4]. Su compatibilidad nativa con JSON las hace perfectas para aplicaciones web modernas y APIs REST, evitando conversiones de datos [4]. Además, su formato semiestructurado es útil para almacenar registros y logs de aplicaciones, centralizando la información para su análisis.

2.3. Modelo de Grafos

2.3.1. Descripción

El modelo de grafos está diseñado para gestionar datos altamente interconectados. Representa la información como nodos (entidades), bordes/aristas (relaciones) y propiedades (atributos) [4, 5]. Los nodos pueden representar cualquier objeto, persona o lugar, mientras que los bordes definen las relaciones entre ellos, pudiendo tener dirección, tipo y propiedades propias. No existen límites prácticos para la cantidad y tipo de relaciones que un nodo puede tener [4].

2.3.2. Ventajas

Ofrece consultas de relaciones extremadamente eficientes, optimizado para navegar y analizar conexiones complejas entre datos [4, 5]. Proporciona flexibilidad máxima para modelar relaciones complejas, capturando matices en las interconexiones entre datos. Presenta un modelado intuitivo de dominios con relaciones ricas como redes sociales o sistemas de recomendaciones. Incluye potentes algoritmos de grafos disponibles para análisis avanzado de conexiones.

2.3.3. Desventajas

Es menos adecuado para datos tabulares o sin relaciones complejas, donde su complejidad no aporta ventajas. Presenta complejidad de implementación mayor que otros modelos NoSQL, requiriendo conocimientos especializados. Tiene escalabilidad especializada que puede ser más compleja de gestionar que en otros modelos. Ofrece una curva de aprendizaje más pronunciada para desarrolladores acostumbrados a modelos relacionales.

2.3.4. Casos de Uso

Es ideal para redes sociales y análisis de comunidades donde las relaciones entre usuarios son fundamentales [4, 5]. Se utiliza en motores de recomendación que analizan patrones de comportamiento y conexiones entre productos o contenidos [4, 5]. Es efectivo para detección de fraude al identificar patrones sospechosos en relaciones transaccionales [4, 5]. Se aplica en gráficos de conocimiento y ontologías para representar conocimiento complejo [4]. También se usa en seguridad en la nube para descubrir relaciones de riesgo entre recursos [4].

2.4. Modelo Columnar (Columnas Anchas)

2.4.1. Descripción

Las bases de datos columnares, también conocidas como de almacén de columnas anchas, almacenan datos en columnas en lugar de filas. Esta organización permite a los usuarios acceder específicamente a las columnas necesarias sin cargar datos irrelevantes, optimizando tanto el almacenamiento como la consulta [5]. Apache Cassandra y Apache HBase son ejemplos representativos de este modelo [5].

2.4.2. Ventajas

Ofrece compresión eficiente al almacenar datos del mismo tipo juntos, reduciendo significativamente los requisitos de almacenamiento [4]. Permite consultas agregadas rápidas sobre columnas específicas, ideal para análisis estadísticos y sumarizaciones [4]. Presenta escalabilidad horizontal robusta en clústeres distribuidos, manejando grandes volúmenes de datos. Es ideal para análisis donde se consultan subconjuntos específicos de columnas repetidamente.

2.4.3. Desventajas

Es ineficiente para operaciones de fila completa (OLTP tradicional) que requieren acceso a todos los atributos de una entidad [4]. Presenta complejidad de gestión mayor que otros sistemas, especialmente para equipos nuevos [5]. Puede tener latencia en actualizaciones que afectan múltiples filas, ya que modificar datos en muchas filas requiere acceder a múltiples columnas. Ofrece transacciones complejas limitadas, no siendo óptimo para aplicaciones con alta concurrencia de escritura.

2.4.4. Casos de Uso

Es excelente para análisis de big data y data warehousing donde se realizan consultas analíticas sobre grandes conjuntos de datos [4, 5]. Se utiliza en sistemas de gestión de registros y series temporales para análisis histórico. Es apropiado para análisis en tiempo real de datos masivos provenientes de diversas fuentes [5]. Se aplica en aplicaciones de IoT que generan grandes volúmenes de datos estructurados que requieren análisis agregado.

Referencias

[1] Powerdata, "Conceptos básicos sobre modelo de datos orientado a objetos," *Powerdata: El valor de la gestión de datos*, 28 de feb. 2017. [En línea]. Disponible: https://blog.powerdata.es/el-valor-de-la-gestion-de-datos/conceptos-basicos-sobre-modelo-de-datos-orientado-a-objetos. [Accedido: 4 de feb. 2026].

[2] CUAED-UNAM, "Modelo Orientado a Objetos," *Unidad de Apoyo para el Aprendizaje (UAPA)*, 2024. [En línea]. Disponible: https://repositorio-uapa.cuaed.unam.mx/repositorio/moodle/pluginfile.php/2731/mod_resource/content/1/UAPA-Modelo-Orientado-Objetos/index.html. [Accedido: 4 de feb. 2026].

[3] Equipo editorial de IONOS, "Base de datos orientada a objetos: el secreto mejor guardado de los modelos de bases de datos," *IONOS Digital Guide*, 18 ene. 2023. [En línea]. Disponible: https://www.ionos.mx/digitalguide/hosting/cuestiones-tecnicas/base-de-datos-orientada-a-objetos/. [Accedido: 4 de feb. 2026].

[4] Amazon Web Services, "¿Qué es NoSQL?," *AWS*, 2024. [En línea]. Disponible: https://aws.amazon.com/es/nosql/. [Accedido: 4 de feb. 2026].

[5] IBM, "¿Qué es una base de datos NoSQL?," *IBM Think*, 12 dic. 2022. [En línea]. Disponible: https://www.ibm.com/mx-es/think/topics/nosql-databases. [Accedido: 4 de feb. 2026].

[6] Amazon Web Services, "¿Qué es la partición de bases de datos?," *AWS*, 2026. [En línea]. Disponible: https://aws.amazon.com/es/what-is/database-sharding/. [Accedido: 7 de feb. 2026].

[7] Microsoft Azure, "Escalado horizontal frente a escalado vertical," *Microsoft Azure*, [En línea]. Disponible: https://azure.microsoft.com/es-mx/resources/cloud-computing-dictionary/scaling-out-vs-scaling-up#scaling-out-horizontally. [Accedido: 7 de feb. 2026].

[8] Fortinet, "¿Qué es la latencia?," *Fortinet Cyberglossary*, [En línea]. Disponible: https://www.fortinet.com/lat/resources/cyberglossary/latency. [Accedido: 7 de feb. 2026].