



UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO



Facultad de ingeniería

Bases de datos

PROFESOR: FERNANDO ARREOLA FRANCO

ALUMNA: ZÁRATE DÍAZ SOFÍA VIRIDIANA

TAREA I

GRUPO:01

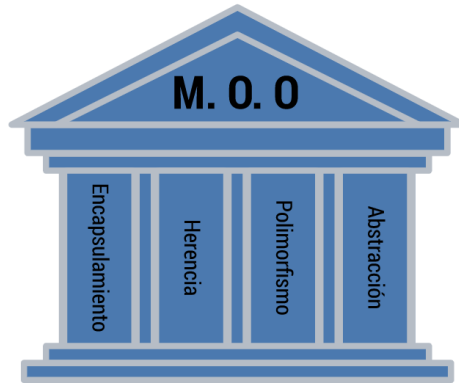
SEMESTRE:2022-2

FECHA:01-09-21

Modelo de datos

○ Modelo orientado a objetos

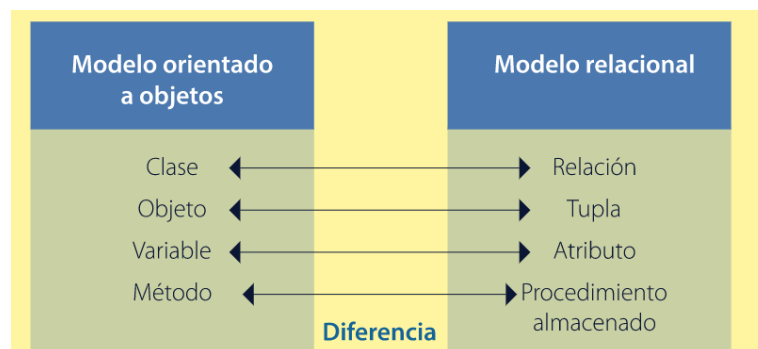
Los componentes se almacenan como objetos y no como datos, tal y como hace una base relacional, cuya representación son las tablas.



Pilares del modelo orientado a objetos

Los manejadores de bases de datos orientados a objetos deben tomar en cuenta las siguientes operaciones:

- Ser capaces de definir sus propios tipos de datos.
- El tamaño de los datos puede ser muy grande.
- La duración de las transacciones puede ser muy larga.
- Recuperar rápidamente objetos complejos.
- Lenguajes de consulta de objetos, un ejemplo es OQL (Object Query Language).
- Mecanismos de seguridad basados en la noción de objeto.
- Funciones para definir reglas deductivas.



Persistencia

Los datos se almacenan a pesar del término del programa de aplicación.

En el caso de 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 un objeto con la transparencia necesaria para que el desarrollador de aplicaciones no tenga que implementar ningún mecanismo distinto al mismo lenguaje de programación orientado a objetos.

Modelo objeto/relacional

Una *base de datos relacional* es un tipo de base de datos que almacena y proporciona acceso a puntos de datos relacionados entre sí. Las bases de datos relacionales se basan en el modelo relacional, una forma intuitiva y directa de representar datos en tablas. En una base de datos relacional, cada fila de la tabla es un registro con un ID único llamado *clave*. Las columnas de la tabla contienen atributos de los datos, y cada registro generalmente tiene un valor para cada atributo, lo que facilita el establecimiento de las relaciones entre los puntos de datos.

Este tipo de modelo se utiliza generalmente en informes de lista donde se necesitan datos mezclados (numéricos, de fecha, de serie) para transmitir información a los usuarios. Se puede realizar agregación en este tipo de informe, pero no es un elemento clave del informe.

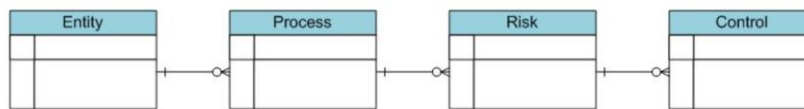
La siguiente lista describe las características clave del modelo de datos relacional:

- Modelado como un diagrama Entidad-Relación.
- Datos altamente normalizados.
- Normalmente dividido por objetos relacionados con otros objetos. Todos los atributos para un objeto incluido el textual, así como el numérico, pertenecen al objeto.

Ejemplo

El modelo relacional es el mejor para mantener la consistencia de los datos en todas las aplicaciones y copias de la base de datos (denominadas instancias). Por ejemplo, cuando un cliente deposita dinero en un cajero automático y, luego, mira el saldo de la cuenta en un teléfono móvil, el cliente espera ver que ese depósito se refleje inmediatamente en un saldo de cuenta actualizado. Las bases de datos relacionales se destacan en este tipo de consistencia de datos, lo que garantiza que múltiples instancias de una base de datos tengan los mismos datos todo el tiempo.

Figura 1. Diagrama Entidad-Relación



Modelos No SQL

Las Bases de Datos NoSQL (*“Not Only SQL”*) pertenecen al modelo no relacional. Las **principales características y ventajas** de este tipo son:

- ✓ SQL no es el lenguaje de consulta/modificación de datos principal, aunque sí lo soportan, de ahí el nombre No Sólo SQL.
- ✓ Los datos **no tienen que almacenarse en tablas**.
- ✓ Son más eficientes en el **procesamiento de los datos** que las BBDD relacionales, por eso son la elección para aplicaciones que hacen un uso intensivo de estos (*“streaming”*, etc.).
- ✓ Utilizan lo que se conoce como **consistencia eventual** que consiste en que los **cambios realizados** en los datos serán **replicados** a todos los nodos del sistema, lo cual aumenta el rendimiento de estos sistemas en contraposición a las propiedades **ACID de las BBDD** relacionales (*“Atomicity, Consistency, Isolation and Durability”* – Atomicidad, Consistencia/Integridad, Aislamiento y Durabilidad).

Desventajas y ventajas

- Los **Sistemas de Gestión de Bases de Datos NoSQL** no contemplan por definición la atomicidad de las instrucciones, es decir, cuando una operación sobre los datos consta de varios pasos, no se tienen que ejecutar todos, cosa que sí sucede en los modelos relacionales (transacciones completas). Hay algunas BBDD NoSQL que contemplan **la atomicidad**.

- Los gestores NoSQL no contemplan obligatoriamente la consistencia o integridad de la BBDD, esto quiere decir que no se comprueba que la operación a ejecutar sobre los datos se pueda completar desde un estado de la **Base de Datos** válido a otro válido (por ejemplo, no violación de ninguna restricción de tipos de datos o reglas).

Utilizar el **mecanismo de consistencia eventual** se puede dar el caso de que la misma consulta a diferentes **máquinas del sistema** produzca resultados diferentes porque las modificaciones de la BBDD aún no han sido replicadas a todos los nodos. Algunas BBDD de este tipo contemplan la propiedad de consistencia.




- Estas BBDD utilizan sus **propios lenguajes de consulta de datos** y APIs, por lo que no tienen una gran interoperabilidad (por ejemplo, dificultad de migraciones de una BBDD a otra, integración con aplicaciones, consultas heredadas en SQL, etc.).
- No hay estandarización para este tipo de BBDD, algo que sí es un punto fuerte de las relacionales.
- Las Bases de Datos NoSQL **funcionan ampliamente en máquinas Linux**, pero no existe en general soporte a otros Sistemas Operativos.
- Las interfaces de gestión de estas BBDD **no son intuitivas** ni sencillas y en algunos casos carecen de ellas gestionándose directamente desde consola de comandos.

¿Cuándo es importante usarlas?

1. Cuando se necesita una BBDD para una aplicación que hace una **consulta/lectura** intensiva de grandes cantidades de datos.
2. Cuando no hay la necesidad de que los **datos sean consistentes**.
3. Si los datos a almacenar no tienen una **estructura fija**.

Una misma aplicación puede usar una BBDD relacional y una BBDD NoSQL y guardar cosas diferentes en cada una de ellas.

Algunos **ejemplos** de uso de este tipo de BBDD:

-  Amazon.
-  Facebook.
-  Google.

Gestores de BBDD NoSQL:

- ❖ Cassandra: <http://cassandra.apache.org/>
- ❖ Redis: <https://redis.io/>
- ❖ MongoDB: <https://www.mongodb.com/es>

Bibliografía;

[1] Modelo Orientado a Objetos [Online].Available:

[Modelo Orientado a Objetos \(unam.mx\)](#)

[2] Bases de datos NoSQL.Qué son y tipos que nos podemos encontrar [Online].Available: [bbdd-nosql-wp-acens.pdf](#)

[3] (2018, Agosto 3) Modelo de datos NoSQL[Online]. Available: [Modelos de datos NoSQL – EAMinds](#)

[4] El modelo de datos relacional [Online]. Available: [El modelo de datos relacional - Documentación de IBM](#)

[5] ¿Qué es una base de datos relacional? [Online]. Availabe: [¿Qué es una base de datos relacional? | Oracle México](#)