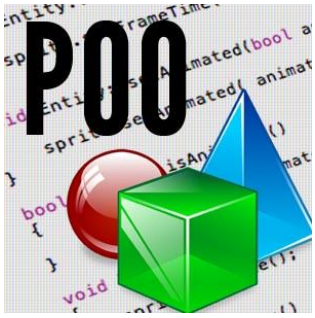
Martínez García Gabriela
Tarea 1
01/Septiembre/2021

Modelos de datos

1 Modelo orientado a objetos



Las bases de datos orientadas a objetos en lugar de incorporar tablas como lo hacen las relacionales, utilizan objetos.

El modelo de datos orientado a objetos es una extensión del paradigma de programación orientado a objetos. Los objetos de una base de datos tienen las mismas características conocidas de los objetos de los lenguajes orientados a objetos (herencia, polimorfismo, encapsulación, otros).
Este modelo admite relaciones uno a varios, uno a uno, y varios a varios.

Los objetos entidad que se utilizan en los programas orientados a objetos son análogos a las entidades que se utilizan en las bases de datos orientadas a objetos puros, pero con una gran diferencia: los objetos del programa desaparecen cuando el programa termina su ejecución, mientras que los objetos de la base de datos permanecen. A esto se le denomina persistencia.

Ventajas

-Modelo de objetos intuitivamente más cercano el mundo real.
-Extensibilidad - herencia.
-Valores complejos.
-Eliminación de la impedancia incorrecta.
-Lenguaje de consulta más expresivo
-El estrechamiento acoplamiento entre datos y aplicaciones permite que el esquema capture más el significado de las aplicaciones.
-Soporte para transacciones largas.
-Mejor soporte para aplicaciones como ingeniería del software o diseño asistido por computadora (CAD)
-Podría decirse que tienen mejor rendimiento, aunque los benchmarks se han aplicado principalmente en áreas como el soporte de ingeniería, a las que los sistemas de gestión base de datos orientados a objetos están mejor adaptado

Desventajas

-La falta de un fundamento teórico, por lo que el significado exacto de modelo de datos orientado a objetos no está bien definido.
-Con un modelo de datos orientado a objetos es más difícil conseguir personal experimentado.
-Falta de estándares.
-La competencia de los sistemas de gestión de base de datos relacionales y objeto-relacionales.
-La encapsulación está comprometida para optimizar las consultas.
-Un modelo de datos orientado a objetos es inherentemente más complejo que el modelo de datos relacional; el sistema de gestión de base de datos orientado objetos proporciona más complejidad que el sistema de gestión de un modelo de datos relacional. La complejidad lleva a mayores costos de implementación y mantenimiento.
-Los sistemas de gestión de bases de datos orientados a objetos generalmente proporcionan control de acceso de grano grueso. Se necesita un mecanismo de seguridad más fino para la mayoría de las aplicaciones comerciales

Al crear un sistema de BD Orientado a Objetos se deben tener en cuenta características están divididas en tres grupos:

- **Mandatorias:** Son aquellas obligatorias
- **Opcionales:** No son obligatorias y se ponen para hacer que el sistema sea mejor
- **Abiertas:** Son aquellas en las que el diseñador puede poner de su parte y que están relacionadas con la programación.

El modelo de datos orientado a objetos está basado en los siguientes componentes:

- La estructura básica es un objeto: equivale a una entidad individual del modelo Entidad-Relación.
- Una clase es la definición del objeto. Esta definición explica los métodos y atributos que tiene. Las clases se organizan en una jerarquía de clase que se parece a un árbol invertido donde cada clase tiene solo un padre. Por ejemplo, la clase cliente y la clase proveedor comparten una clase: persona.
- Un método representa una acción del mundo real. Por ejemplo: localizar el nombre de un cliente, cambiar el teléfono de un cliente o imprimir su dirección. Son equivalentes a los procedimientos en un lenguaje de programación. Definen el comportamiento de un objeto.
- La herencia es la capacidad de un objeto de heredar los atributos y los métodos de los objetos que están sobre él en una jerarquía de clase. Por ejemplo, las clases cliente y proveedor, como subclases de la clase persona heredarán los atributos de la clase persona.

Algunas bases de datos orientadas a objetos han sido diseñadas para trabajar bien con lenguajes de programación orientados a objetos tales como Delphi, Ruby, Python, Perl, Java, Visual Basic.NET, etc.

2 Modelo objeto relacional

El término base de datos objeto-relacional (BDOR) se usa para describir una base de datos que ha evolucionado desde el modelo relacional hasta una base de datos híbrida, que contiene ambas tecnologías: relacional y de objetos.

El modelo de base de datos objeto-relacional integra los conceptos de la tradicional base de datos relacional y los conceptos de paradigma de objetos que se utiliza en la programación orientada a objetos (POO).

El objetivo de este concepto es poder aplicar la tecnología madura de bases de datos relacionales sobre la organización de los datos complejos es decir datos de texto e imagen, mapas, datos en el rango de audio etc. Las bases de datos Objeto-relacional son compatibles con estos objetos de datos y las operaciones de mayor complejidad.

En una base de datos objeto relacional los dominios de dicha base de datos ya no son sólo atómicos por esta razón no cumplen la 1FN debido a que las tuplas también pueden ser una relación, que llevará a la creación de una relación de relaciones es así que no se puede aplicar el concepto de normalización.
De este modo, se genera la posibilidad de guardar objetos más complejos en una sola tabla con referencias a otras relaciones, con lo que se acerca más al paradigma de programación orientada a objetos.

Características de los datos complejos:

- **Colecciones:** También conocidos como conjuntos, este tipo de datos clasifican los arrays y los conjuntos en que los elementos pueden aparecer varias veces.
- **Tipos estructurados:** Los tipos estructurados permiten representación directa de los atributos compuestos en los diagramas entidad-relación.
- **Objetos de gran tamaño:** Desde ya hace varios años que se necesita almacenar datos con atributos muy grandes (Varios Mbytes), como libros, canciones, etc. E incluso aún más grandes; como mapas de alta resolución, video, etc. que puede llegar fácilmente a los Gbytes.

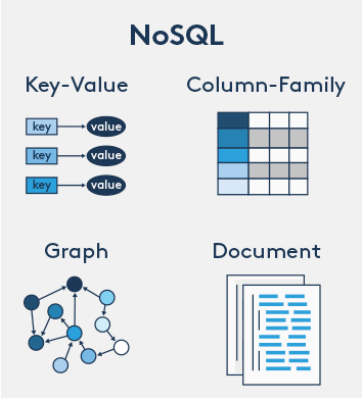
Las BDOR las podemos ver como un híbrido de las BDR y las BDOO que intenta aunar los beneficios de ambos modelos, aunque por descontado, ello suponga renunciar a algunas características de ambos. Los objetivos que persiguen estas bases de datos son: mejorar la representación de los datos mediante la orientación a objetos y simplificar el acceso a datos, manteniendo el sistema relacional.

En una BDOR se siguen almacenando tablas en filas y columnas, aunque la estructura de las filas no está restringida a contener escalares o valores atómicos, sino que las columnas pueden almacenar tipos estructurados (tipos compuestos como vectores, conjuntos, etc.) y las tablas pueden ser definidas en función de otras, que es lo que se denomina herencia directa.
Internamente tanto las tablas como las columnas son tratados como objetos, esto es, se realiza un mapeo objeto-relacional de manera transparente.
Como consecuencia de esto, aparecen nuevas características, entre las que podemos destacar las siguientes:

- **Tipos definidos por el usuario**. Se pueden crear nuevos tipos de datos definidos por el usuario, y que son compuestos o estructurados, esto es, será posible tener en una columna un atributo multivaluado (un tipo compuesto).
- **Tipos Objeto**. Posibilidad de creación de objetos como nuevo tipo de dato que permiten relaciones anidadas.
- **Reusabilidad**. Posibilidad de guardar esos tipos en el gestor de la BDOR, para reutilizarlos en tantas tablas como sea necesario
- **Creación de funciones**. Posibilidad de definir funciones y almacenarlas en el gestor. Las funciones pueden modelar el comportamiento de un tipo objeto, en este caso se llaman métodos.
- **Tablas anidadas**. Se pueden definir columnas como arrays o vectores multidimensionales, tanto de tipos básicos como de tipos estructurados, esto es, se pueden anidar tablas
- **Herencia** con subtipos y subtablas.

3 Modelos NoSQL

Hablar de bases de datos NoSQL es hablar de estructuras que nos permiten almacenar información en aquellas situaciones en las que las bases de datos relacionales generan ciertos problemas debido principalmente a problemas de escalabilidad y rendimiento de las bases de datos relacionales donde se dan cita miles de usuarios concurrentes y con millones de consultas diarias.
Las bases de datos NoSQL son sistemas de almacenamiento de información que no cumplen con el esquema entidad–relación. Tampoco utilizan una estructura de datos en forma de tabla donde se van almacenando los datos, sino que para el almacenamiento hacen uso de otros formatos como clave–valor, mapeo de columnas o grafos



Esta forma de almacenar la información ofrece ciertas ventajas sobre los modelos relacionales. Entre las ventajas más significativas podemos destacar:

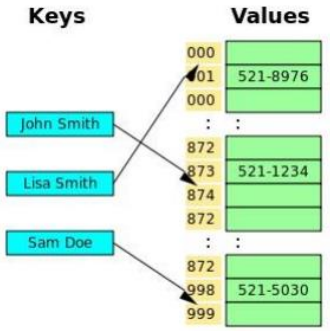
- Se ejecutan en máquinas con pocos recursos: Estos sistemas, a diferencia de los sistemas basados en SQL, no requieren de apenas computación, por lo que se pueden montar en máquinas de un coste más reducido.
- Escalabilidad horizontal: Para mejorar el rendimiento de estos sistemas simplemente se consigue añadiendo más nodos, con la única operación de indicar al sistema cuáles son los nodos que están disponibles.
- Pueden manejar gran cantidad de datos: Esto es debido a que utiliza una estructura distribuida, en muchos casos mediante tablas Hash.
- No genera cuellos de botella: El principal problema de los sistemas SQL es que necesitan transcribir cada sentencia para poder ser ejecutada, y cada sentencia compleja requiere además de un nivel de ejecución aún más complejo, lo que constituye un punto de entrada en común, que ante muchas peticiones puede ralentizar el sistema.

Algunas de las diferencias más destacables que nos podemos encontrar entre los sistemas NoSQL y los sistemas SQL están:

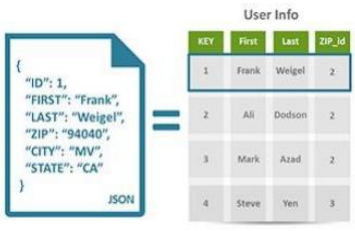
- No utilizan SQL como lenguaje de consultas. La mayoría de las bases de datos NoSQL evitan utilizar este tipo de lenguaje o lo utilizan como un lenguaje de apoyo. Por poner algunos ejemplos, Cassandra utiliza el lenguaje CQL, MongoDB utiliza JSON o BigTable hace uso de GQL.
- No utilizan estructuras fijas como tablas para el almacenamiento de los datos. Permiten hacer uso de otros tipos de modelos de almacenamiento de información como sistemas de clave–valor, objetos o grafos.
- No suelen permitir operaciones JOIN. Al disponer de un volumen de datos tan extremadamente grande suele resultar deseable evitar los JOIN. Esto se debe a que, cuando la operación no es la búsqueda de una clave, la sobrecarga puede llegar a ser muy costosa. Las soluciones más directas consisten en desnormalizar los datos, o bien realizar el JOIN mediante software, en la capa de aplicación.
- Arquitectura distribuida. Las bases de datos relacionales suelen estar centralizadas en una única máquina o bien en una estructura máster–esclavo, sin embargo, en los casos NoSQL la información puede estar compartida en varias máquinas mediante mecanismos de tablas Hash distribuidas.

Dependiendo de la forma en la que almacenen la información, nos podemos encontrar varios tipos distintos de bases de datos NoSQL:

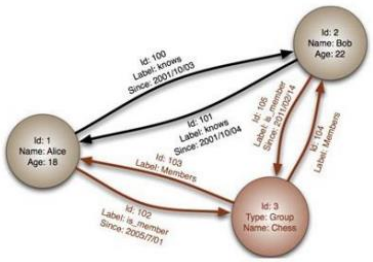
- Base de datos clave-valor:
  Son el modelo de base de datos NoSQL más popular, además de ser la más sencilla en cuanto a funcionalidad. En este tipo de sistema, cada elemento está identificado por una llave única, lo que permite la recuperación de la información de forma muy rápida, información que habitualmente está almacenada como un objeto binario (BLOB). Se caracterizan por ser muy eficientes tanto para las lecturas como para las escrituras. Algunos ejemplos de este tipo son Cassandra, BigTable o HBase.



- Base de datos documentales:
  Este tipo almacena la información como un documento, generalmente utilizando para ello una estructura simple como JSON o XML y donde se utiliza una clave única para cada registro. Este tipo de implementación permite, además de realizar búsquedas por clave–valor, realizar consultas más avanzadas sobre el contenido del documento. Son las bases de datos NoSQL más versátiles. Se pueden utilizar en gran cantidad de proyectos, incluyendo muchos que tradicionalmente funcionarían sobre bases de datos relacionales. Algunos ejemplos de este tipo son MongoDB o CouchDB.



- Base de datos en grafos:
  En este tipo de bases de datos, la información se representa como nodos de un grafo y sus relaciones con las aristas del mismo, de manera que se puede hacer uso de la teoría de grafos para recorrerla. Para sacar el máximo rendimiento a este tipo de bases de datos, su estructura debe estar totalmente normalizada, de forma que cada tabla tenga una sola columna y cada relación dos. Este tipo de bases de datos ofrece una navegación más eficiente entre relaciones que en un modelo relacional. Algunos ejemplos de este tipo son Neo4j, InfoGrid o Virtuoso.



- Bases de datos orientada a objetos: En este tipo, la información se representa mediante objetos, de la misma forma que son representados en los lenguajes de programación orientada a objetos (POO) como ocurre en JAVA, C# o Visual Basic .NET. Algunos ejemplos de este tipo de bases de datos son Zope, Gemstone o Db4o.

Referencias

- 2.1 El modelo de datos orientados a objetos - Unidad 2: Sistemas BDOO. (2021). Recuperado de: https://sites.google.com/site/unidad2sistemasbdoo/2-1-el-modelo-de-datos-orientados-a-objetos
- 3.3.5.- Modelo orientado a objetos | GBD01.- Sistemas de almacenamiento de la información. (2021). Recuperado de: https://ikastaroak.birt.eus/edu/argitalpen/backupa/20200331/1920k/es/ASIR/GBD/GBD01/es_ASIR_GBD01_Contenidos/website_335_modelo_orientado_a_objetos.html
- 4.- Características de las bases de datos objeto-relacionales. | AD05.- Bases de datos objeto-relacionales y orientadas a objetos. (2021). Recuperado de: https://ikastaroak.birt.eus/edu/argitalpen/backupa/20200331/1920k/es/DAM/AD/AD05/es_DAM_AD05_Contenidos/website_4__caractersticas_de_las_bases_de_datos_objetorelacionales.html
- Bases de datos NoSQL. Qué son y tipos que nos podemos encontrar (2021). Recuperado de: https://www.acens.com/wp-content/images/2014/02/bbdd-nosql-wp-acens.pdf