Diagrama

Descripción generada automáticamenteForma

Descripción generada automáticamente con confianza baja

Una base orientada a objetos es una colección de datos modelados en objetos de los lenguajes de programación orientados a objeto, que tienen la capacidad de encapsular los atributos y las operaciones definidas sobre ellos y mediante las cuales interactúan.

Los objetos se agrupan en clases, que son las mismas utilizadas en los lenguajes de programación orientados a objetos y que se estructuran en una jerarquía de superclases y subclases.

Se definen por datos y comportamientos, llamados atributos y métodos, como un todo. Un atributo es una propiedad o característica de la clase y describe un rango de valores que podrán contener los objetos o instancias de esa clase.

Un método es una acción que la clase puede realizar. Un objeto de la clase "Persona", por ejemplo, tendrá como atributos el DNI, el nombre, la edad, la empresa donde trabaja, etc. El valor de los atributos es propio de cada instancia, pero entre algunos objetos se comparte el comportamiento.

La orden para ejecutar un comportamiento se efectiviza mediante un mensaje; si el comportamiento y los datos son aceptados por el protocolo del objeto, se ejecutará; si no, es rechazado. Aquí tenemos la responsabilidad y ejecución del control por parte del objeto. Para nuestro objeto "Persona", se han definido comportamientos tales como actualizar la edad, cambiar la empresa, etc. Si cambia la empresa, se ejecutará el método "cambiarEmpresa" con el nombre de la empresa.

Al inicio del año 2000, las bases de datos relacionales eran las más utilizadas en la mayoría de los ámbitos de aplicación. La estructura de datos básicos que ofrece la tabla relacional es apropiada para muchas aplicaciones habituales en el ámbito empresarial. Sin embargo, existen casos en los que presenta serios inconvenientes prácticos, generalmente si se requiere gestionar datos muy complejos o no convencionales, por ejemplo, imágenes o documentos extensos. Para ellos, la estructura relacional resulta ineficiente. Algunos ejemplos de gran importancia práctica son las bases de datos multimedia, las bases de datos científicas y los sistemas de apoyo al diseño industrial, como los sistemas CAD/CAM.

Tenemos así una primera generación que fue la de bases de **datos jerárquicas**, luego una **segunda generación de bases de datos relacionales**, y como consecuencia de lo que dijimos, aparece una **tercera generación de bases de datos orientadas a objetos**.

Las bases de datos orientadas a **objetos soportan el paradigma orientado a objetos, almacenando datos y métodos,** y no solo los datos como ocurría en las relacionales. Esta característica evita el acceso a los datos mediante el control que ejercen los métodos almacenados en el objeto. Permiten el desarrollo de aplicaciones utilizando el mismo modelo conceptual de análisis, diseño y programación orientado a objetos, sin necesidad de mapearlo a un modelo conceptual diferente, reduciendo el problema de la conversión

Diagrama

Descripción generada automáticamenteDiagrama

Descripción generada automáticamenteDiagrama

Descripción generada automáticamente

Hay tres conceptos que son fundamentales en el marco teórico de las bases de datos orientadas a objetos: encapsulamiento, herencia y polimorfismo.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

El sistema de gestión de base de datos orientada a objetos es un software específico que sirve de interfaz entre la base de objetos, el usuario y las aplicaciones que la utilizan, con la particularidad de almacenar objetos, permitiendo su persistencia, concurrencia y recuperación. El hecho de que las bases de datos orientadas a objetos y los lenguajes de programación orientados a objetos compartan el mismo modelo de datos permite desarrollar aplicaciones sobre la base de datos orientada a objetos de manera casi transparente. Sin embargo, es necesario indicar desde el lenguaje de programación orientado a objetos qué objetos se guardarán en la base de datos, es decir, cuáles serán persistentes y cuáles no.

La propiedad de **persistencia** define qué objetos deberán permanecer en la base de datos una vez que el programa haya terminado y cuáles no. Los objetos transitorios desaparecen una vez que el programa que los creó termina de ejecutarse, mientras que los objetos persistentes permanecen almacenados en la base de datos.

El **polimorfismo** es la propiedad de una operación mediante la cual puede ser aplicada a distintos objetos, siendo el sistema quien define qué acción exactamente debe ejecutarse. Un ejemplo clásico de polimorfismo es el siguiente: tenemos tres clases diferentes, Círculo, Rectángulo y Cuadrado, que heredan de la superclase Forma. La clase Forma tiene el método "calcularÁrea", que se implementa de distinta forma en cada una de las subclases. En la subclase Círculo, será el algoritmo "pi por radio al cuadrado"; en la subclase Rectángulo, será "base por altura"; y en la subclase Cuadrado, será "lado al cuadrado".

La **herencia** es el mecanismo que permite la reutilización y extensibilidad del software. Una clase implementa el tipo de objeto, y una subclase hereda las propiedades de su clase padre, agregando nuevos atributos y métodos. Este concepto lleva a una organización jerárquica que facilita el mantenimiento: se modifican los niveles superiores y los inferiores heredan la modificación.

Con respecto al **encapsulamiento**, podemos decir que en las metodologías de desarrollo clásicas, datos y procesos se tratan por separado. Hay un modelo de datos donde figuran las estructuras y las relaciones, y un modelo de procesos donde se representa el intercambio entre entidades, almacenamientos y procesos. El hecho de que un objeto sea una estructura indivisible de atributos que lo describen y métodos de comportamiento permite, por un lado, el tratamiento de ambos aspectos desde el inicio del modelado del sistema y no en etapas, como en las metodologías tradicionales, y por otro lado, permite reducir el potencial de errores, ya que la responsabilidad de control y ejecución reside en el mismo objeto. Cada objeto oculta su funcionalidad mediante la propiedad de encapsulamiento, estableciendo atributos y métodos públicos y privados para permitir la comunicación entre ellos. Presenta una interfaz que recibe o envía mensajes a modo de peticiones para realizar operaciones con las aplicaciones de usuario..

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

El **manifiesto de las bases de datos orientadas a objetos** fue definido en 1992 y establece las reglas que deben cumplir los sistemas de gestión de bases de datos para ser considerados orientados a objetos. Estas reglas, conocidas como el **manifiesto de Atkinson**, incluyen las siguientes características clave:

1. **Soporte de objetos complejos**: Debe ser posible manipular objetos complejos, que están formados por la unión de objetos simples, aplicando constructores y sin necesidad de descomponerlos en objetos más básicos.
2. **Identidad del objeto**: Cada objeto debe tener un identificador único, llamado **OID (Object Identifier)**, que es independiente de los valores de sus atributos. Este identificador es inmutable y generado por el sistema, y no es visible para el usuario externo.
3. **Encapsulamiento**: Los programadores tienen acceso solo a la especificación de la interfaz de los métodos; los datos y la implementación de estos métodos están ocultos dentro de los objetos.
4. **Tipos o clases**: El esquema de una base de datos orientada a objetos contiene un conjunto de clases o tipos que definen la estructura de los datos y los métodos que pueden aplicarse a cada uno de ellos.
5. **Jerarquía de tipos o clases**: Las clases deben poder heredar atributos y métodos mediante especialización o generalización, formando una jerarquía. Los subtipos heredan todas las funciones de sus supertipos.
6. **Sobrecarga y polimorfismo**: Diferentes clases pueden implementar distintos métodos para una misma operación, y el sistema determina cuál ejecutar en función del contexto.
7. **Lenguaje de Manipulación de Datos (Data Manipulation Language - DML)**: Debe ser un lenguaje de programación de propósito general que permita expresar cualquier algoritmo y que sea completo para la manipulación de datos.
8. **Extensibilidad de tipos de datos**: Debe permitir la creación de nuevos tipos de datos a partir de los existentes, sin hacer distinción entre los tipos definidos por el usuario y los del sistema.
9. **Persistencia de datos**: Los datos deben mantenerse después de que la aplicación que los creó haya finalizado, sin que el usuario deba realizar una copia explícita.
10. **Gestión eficiente del almacenamiento**: El sistema debe ofrecer funciones de gestión eficiente, como el uso de índices, almacenamiento en memoria intermedia, selección de rutas de acceso y optimización de consultas.
11. **Soporte para la concurrencia**: El sistema debe tener mecanismos para controlar el acceso simultáneo de múltiples usuarios a los datos.
12. **Recuperación ante fallas**: El sistema debe ofrecer mecanismos para mantener la integridad de los datos y la continuidad del servicio en caso de fallas.
13. **Lenguaje de consulta específico para objetos**: Debe proporcionar un lenguaje de consulta que permita interactuar de forma adecuada con los objetos almacenados en la base de datos.

Estas reglas aseguran que un sistema gestor de bases de datos orientado a objetos sea capaz de manejar de manera eficiente la estructura y comportamiento de los datos, facilitando la integración con lenguajes de programación orientados a objetos y proporcionando características como la persistencia, la encapsulación y la reutilización a través de la herencia y el polimorfismo.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

El manifiesto Atkinson propone también cinco opciones de implementación no obligatorias:

1. **Herencia múltiple**: un objeto tiene características de padres diferentes.
2. **Herencia selectiva**: hereda solo algunas de las características del padre.
3. **Verificación e inferencia de tipos**: le quita al programador la tarea de especificar los tipos de datos.
4. **Distribución de los datos** en diferentes servidores.
5. **Transacciones complejas** entre usuarios interactivos cooperantes o secuencias de subtransacciones.
6. **Versionado de objetos**: algunos sistemas de orientados a objeto cuentan con capacidades para manejar múltiples versiones del mismo objeto; esta característica es útil para consultar una versión anterior hasta que la nueva se haya aprobado y verificado, por ejemplo..

Texto

Descripción generada automáticamente

Una de las motivaciones para el desarrollo de las bases de datos orientadas a objeto fue el deseo de representar objetos complejos, que pueden ser de dos tipos: **estructurados y no estructurados**.

Un objeto complejo estructurado está constituido por componentes y se define aplicando recurrentemente en diversos niveles los constructores de tipos disponibles, como mínimo listas y arreglos.

Uno no estructurado casi siempre es un tipo de datos que requiere una gran cantidad de espacio de almacenamiento, como por ejemplo un tipo de datos que representa una imagen. Se conocen como **objetos binarios extensos o *binary large object***. Estos objetos son no estructurados en el sentido de que el sistema gestor de la base de datos no sabe qué estructura tienen; solo la aplicación que los usa sabe cómo interpretarlos.

**Se consideran complejos** porque necesitan un área de almacenamiento sustancial y no forman parte de los tipos de datos estándar que suelen ofrecer los sistemas gestores de base de datos. Como el tamaño de los objetos es considerable, el sistema gestor podría obtener una porción del objeto y proporcionarla al programa antes de recuperar todo el objeto. También podría usar técnicas de buffers y de almacenamiento en memoria caché para obtener por anticipado partes del objeto.

Como un sistema gestor de base de datos orientado a objetos permite a los usuarios crear nuevos tipos y como un tipo incluye tanto la estructura como las operaciones, podemos considerar que un sistema gestor de **base de datos tiene un sistema de tipo extensible.** Se pueden crear bibliotecas de nuevos tipos definiendo estructura y operaciones. Otras aplicaciones pueden utilizar estos tipos e incluso modificarlos creando subtipos de los tipos provistos en la biblioteca.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamenteDiagrama

Descripción generada automáticamente

**ODMG (Object Data Management Group**) es el estándar de facto para los modelos de objetos. Este grupo fue creado en 1991, con el propósito de definir estándares para los sistemas gestores de bases de datos, ya que las bases de datos orientadas a objetos no contaban con un estándar, a diferencia del análisis y diseño orientado a objetos que se realiza con UML (Unified Modeling Language).

ODMG define un modelo para la semántica de los objetos en una base de datos. **La versión 3.0** propone los siguientes componentes principales de la arquitectura de un sistema gestor de base de datos:

* **Lenguaje de Definición de Objetos (ODL, Object Definition Language)**: La definición de una base de datos está contenida en un esquema creado mediante este lenguaje, que es equivalente **al DDL (Data Definition Language**) de los sistemas gestores de bases de datos tradicionales.
* **Lenguaje de Consulta de Objetos (OQL, Object Query Language)**: Con una sintaxis similar a SQL, pero con características adicionales como la identidad de objetos, objetos complejos, operaciones, herencia, polimorfismo y relaciones.
* **Conexión con lenguajes como C++, Smalltalk y Java**: ODMG permite que un sistema gestor de bases de datos se conecte con estos lenguajes de programación. Esta lista de lenguajes no es limitante, pero define al menos la interoperabilidad con los mencionados.

Existen dos tipos de semántica para la referencia entre un objeto complejo y sus componentes en cada nivel. **El primer tipo es conocido como *agregación***, y es una relación conocida como "es parte de", que denota una jerarquía del todo a la parte.

El segundo tipo se lo conoce como ***generalización/especialización*** y denota una relación del tipo "es un".

En el ejemplo, vemos la relación de agregación entre el objeto "perro" y sus componentes (ojo, oreja, cola), que son parte del objeto "perro". En el objeto "mariposa", con los suyos (ala, antena), son parte de "mariposa".

En el ejemplo también está conceptualizada una relación de generalización/especialización entre la clase "animal" y sus componentes "perro" y "mariposa", ya que ambos "son un" tipo de "animal".

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

eh los componentes básicos de una base de datos orientada a objetos son los objetos.

eh un objeto es una representación abstracta de una entidad del mundo real que tiene una identidad o identificador único dentro de la base de datos y propiedades que incluyen sus atributos y un comportamiento que le proporciona la capacidad de interaccionar con otros objetos incorporadas en sí mismo.

eh los objetos que comparten propiedades y comportamientos forman clases según el estándar odmg.

eh los elementos que componen un objeto

1-son identidad, que es la propiedad que permite diferenciar a cada objeto de los restantes. Se representa por medio de un object ID (object identifier), que es el único para ese objeto, es asignado por el sistema en el momento en que el objeto es creado y no puede ser alterado bajo ninguna circunstancia.

eh No es visible para el usuario externo, sino que el sistema lo utiliza internamente para identificar al objeto y para referenciarlo desde otros objetos.

2-constructores de tipos y objetos. Los lenguajes de definición de datos orientados a objetos permiten definir los tipos de datos y los tipos de objetos necesarios para construir los objetos de una aplicación concreta.

3-eh referencia entre objetos. Los atributos cuyo dominio sea un tipo de objetos implementa como referencia usando el object identifier (el ID) y representa relaciones entre los objetos.

4-estado de un objeto es el valor que tiene asignado dicho objeto en un momento determinado. Esto significa que cuando las propiedades de un objeto cambian, su valor debe ser actualizado en la base de datos, haciendo pasar el objeto a un nuevo estado.

5-comportamiento de los objetos determina las tareas que puede realizar a partir de mensajes recibidos de otros objetos. Se representa por medio de operaciones que se ejecutan entre ellos.

toda operación que deba realizarse sobre un objeto tiene que estar previamente declarada. Las operaciones se usan para crear y eliminar objetos, para cambiar los valores de sus atributos, para conocer su valor o para realizar operaciones complejas.

6-y finalmente, clasificación e instanciación de objetos. La base de datos orientada a objetos clasifican los objetos de acuerdo a sus similitudes. Una clase es un conjunto de objetos que comparten unas propiedades, tipos de objeto y un comportamiento operaciones. La instanciación es el proceso inverso al de clasificación y consiste en generar los objetos de una clase.

**Imagen de la pantalla de un celular con texto

Descripción generada automáticamente con confianza mediaImagen de la pantalla de un celular con texto

Descripción generada automáticamente con confianza media**

Al igual que en las bases de datos relacionales, el modelo lógico de datos de las bases de datos orientadas a objeto es independiente de la localización física de los datos.

los datos también se almacenan en archivos emplazados en diferentes dispositivos físicos que el administrador de la base de datos tiene que gestionar.

eh muchos de los conceptos de diseño físico de las bases de datos relacionales se pueden aplicar a las bases de datos orientadas a objetos.

sin embargo, hay ciertas cuestiones que deben ser estudiadas en particular.

por ejemplo, los índices de acceso a los datos.

eh los principales tipos de estructuras de indexación que podemos encontrar en las bases de datos son los índices de atributos, que se usan para indexar los objetos de una clase según alguno de sus atributos.

de esta manera, se pueden recuperar rápidamente los objetos de una clase que cumplan cierta propiedad sobre el atributo indexado.

por ejemplo, indexar los objetos de una clase empleados por su atributo DNI.

y los índices de objetos, que se usan para indexar los objetos de una clase según su identificador o ID.

este índice contiene una entrada con el identificador de cada objeto de la clase y un puntero a su dirección de memoria.

eh esta alternativa resulta muy útil para navegar por los objetos, ya que para pasar de un objeto de una clase a otra que referencia desde alguno de sus atributos se utilice el índice.

eh todos los accesos, creación, modificación y borrado de objetos persistentes se debe realizar dentro de una transacción, que son unidades lógicas de trabajo que llevan a la base de datos de un estado consistente a otro estado consistente.

el modelo ASUME una secuencia lineal de transacciones que se ejecutan de modo controlado.

la concurrencia se maneja en base a bloqueos de lectura y escritura.

el modelo especifica operaciones para iniciar, terminar, el commit y abortar transacciones.

eh utilizando el bloqueo de dos fases, así como la operación de checkpoint. Esta última operación hace permanentes los

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

ventajas:

eh proporcionan un modelo de datos mucho más rico y extensible y así complementan, aunque no sustituyen, a las bases de datos relacionales.

esto se debe a que un objeto permite encapsular tanto un estado como un comportamiento.

un objeto puede almacenar todas las relaciones que tenga con otros objetos y los objetos pueden agruparse para formar objetos complejos mediante herencia.

eh logran una equivalencia al de los lenguajes de programación orientados a objetos, como C++ o Java.

esto tiene la gran ventaja de que, al compartir el modelo de datos, se pueden integrar las bases de datos orientadas a objeto con el software usado para desarrollar aplicaciones de manera directa y casi transparente.

en las bases de datos relacionales es necesario usar dos lenguajes: uno de programación para la aplicación y SQL para el acceso a la base de datos.

eh disponen de nuevas características para el modelado de datos, por ejemplo, eh almacenar grandes objetos binarios, eh como imágenes, sonido o video.

eh las herramientas de gestión de documentos, índices y operadores específicos para buscar texto, eh inclusive como soporte de XML y otras extensiones para soportar datos geográficos.

eh además permiten la reusabilidad de clases, lo que repercute en una mayor facilidad de mantenimiento y un menor tiempo de desarrollo.

Imagen de la pantalla de un celular de un mensaje en letras blancas

Descripción generada automáticamente con confianza baja

inconvenientes que aún presentan los sistemas gestores de base de datos:

por un lado, la carencia de un modelo de datos universal.

no hay ningún modelo de datos que esté universalmente aceptado para los sistemas gestores de bases de datos y los modelos carecen en su mayoría de una base teórica.

eh por otra parte, los sistemas gestores de base de datos relacionales tienen una experiencia de uso considerable.

SQL es un estándar aprobado y, además, el modelo relacional tiene una sólida base teórica.

los productos relacionales disponen de muchas herramientas de soporte que sirven tanto para desarrolladores como para usuarios finales, lo que no es el caso de las bases de datos orientadas a objeto.