BASES DE DATOS OBJETO-RELACIONALES

Hugo Pelayo Aseko

Bases de datos

4 de mayo de 2023

Contents

[Introducción 3](#_Toc134542758)

[Tipos de datos 6](#_Toc134542759)

[Definición de tipos de objeto 6](#_Toc134542760)

[Herencia 6](#_Toc134542761)

[Tipos de datos colección 7](#_Toc134542762)

[Bibliografía 7](#_Toc134542763)

# Introducción

Los sistemas de gestión de bases de datos objeto-relacional surgieron de investigaciones realizadas en la década de 1990. Estas investigaciones ampliaron los conceptos de las bases de datos relacionales añadiendo conceptos de objetos. Los investigadores intentaron mantener como componente central de la arquitectura un lenguaje de consulta declarativo basado en cálculo de predicados. El proyecto de investigación más destacado, Postgres (UC Berkeley), dio lugar a dos productos que se derivan de esa investigación: Illustra y PostgreSQL.

A mediados de la década de 1990 aparecieron los primeros productos comerciales, como Illustra, Omniscience y UniSQL. También se desarrolló la primera versión de la base de datos Valentina, creada por el desarrollador ucraniano Ruslan Zasukhin, fundador de Paradigma Software, como un SDK de C++. En la siguiente década, PostgreSQL se convirtió en una base de datos comercialmente viable y es la base de varios productos actuales que mantienen sus características de ORDBMS.

Los científicos de la computación comenzaron a referirse a estos productos como sistemas de gestión de bases de datos objeto-relacional o ORDBMS. Muchas de las ideas de los primeros esfuerzos de bases de datos objeto-relacional se han incorporado en gran medida en SQL:1999 a través de tipos estructurados. De hecho, cualquier producto que cumpla con los aspectos orientados a objetos de SQL:1999 podría describirse como un producto de gestión de bases de datos objeto-relacional. Por ejemplo, IBM Db2, Oracle Database y Microsoft SQL Server, afirman admitir esta tecnología.

Las bases de datos objeto-relacionales son bases de datos que siguen el modelo de datos relacional proporcionando características de la programación orientada a objetos.

En un DBMS relacional, los datos se organizan en tablas con filas y columnas, mientras que, en un modelo de datos orientado a objetos, los datos se representan como objetos que contienen propiedades y métodos. Un ORDBMS (del inglés *Object Relational Daatbase Management System*) combina estas dos formas de modelado de datos, permitiendo que los objetos se almacenen y se recuperen a través de una interfaz SQL, que es el lenguaje de consulta estructurado utilizado en los sistemas de gestión de bases de datos relacionales.

Las bases de datos objeto-relacionales permiten una mayor flexibilidad en el modelado de datos que los sistemas de bases de datos relacionales tradicionales. Al utilizar un modelo de datos orientado a objetos, se pueden representar relaciones complejas entre los datos, como objetos que contienen otros objetos, o incluso objetos que son heredados de otros objetos.

Además, los ORDBMS también permiten que se definan tipos de datos personalizados y que se definan funciones y procedimientos almacenados, lo que puede simplificar el proceso de programación de aplicaciones que utilizan la base de datos.

Los ORDBMS también pueden ser escalables y eficientes en términos de rendimiento. Algunos sistemas ORDBMS utilizan técnicas de optimización de consultas, como la indexación y el almacenamiento en caché, para acelerar la recuperación de datos y mejorar el rendimiento de la base de datos.

El modelo objeto-relacional es una combinación de las técnicas orientadas a objetos y las bases de datos relacionales, lo que proporciona una gran cantidad de ventajas. Con este modelo, los desarrolladores pueden acceder a los datos de manera más intuitiva, como si fueran entidades de la vida real, lo que facilita la comprensión y el manejo de los datos.

El uso de tipos de objetos permite organizar y acceder a los datos de una manera más eficiente, y al mismo tiempo se mantiene la alta capacidad de concurrencia y el rendimiento de las bases de datos relacionales. Los tipos de objetos permiten modelar objetos de la vida real, consiguiendo una implementación que se abstrae de los detalles innecesarios de los objetos que se modela, y almacenar los datos orientados a objetos de forma permanente en una base de datos.

La programación orientada a objetos es especialmente útil para la construcción de componentes reutilizables y aplicaciones complejas. En PL/SQL, la programación orientada a objetos se basa en tipos de objetos, lo que permite a los desarrolladores de aplicaciones con lenguajes orientados a objetos acceder directamente a las mismas estructuras de datos creadas en la base de datos de Oracle, ya que esta ofrece soporte nativo para la manipulación de datos utilizando el paradigma de la programación orientada a objetos.

Además, los objetos pueden incluir acciones para realizar ciertas tareas sobre los datos, lo que facilita la obtención de información de manera más sencilla y eficiente. Por ejemplo, un objeto "Pedido" puede incluir un método para calcular el importe total de los artículos comprados, mientras que un objeto "Cliente" puede tener métodos que permitan obtener su historial de compras. Al utilizar estos métodos, las aplicaciones pueden obtener la información necesaria sin tener que realizar complejas consultas a la base de datos.

El objetivo principal de una base de datos objeto-relacional es implantar en las bases de datos relacionales las técnicas de modelado orientado a objetos que se utilizan en lenguajes de programación como Java, C++ o C#, por ejemplo, la abstracción de entidades sobre clases, la implementación de la herencia entre tipos, entre otras características. Una alternativa popular para integrar el modelo de datos objeto-relacional en las bases de datos relacionales consiste en utilizar un sistema de bases de datos relacionales estándar con algún tipo de software de mapeo objeto-relacional (ORM). A diferencia de los productos tradicionales de RDBMS o SQL-DBMS que se enfocaban en la gestión eficiente de datos extraídos de un conjunto limitado de tipos de datos (definidos por los estándares del lenguaje relevante), un ORDBMS permite a los desarrolladores de software integrar sus propios tipos y los métodos que se aplican a ellos en el DBMS.

El ORDBMS se integra con un lenguaje de programación orientado a objetos. Las propiedades características de ORDBMS son:

1. datos complejos
2. herencia de tipos
3. comportamiento de objetos.

La creación de datos complejos en la mayoría de los ORDBMS de SQL se basa en una definición de esquema preliminar a través del tipo definido por el usuario (UDT). La jerarquía dentro de los datos complejos estructurados ofrece una propiedad adicional, la herencia de tipos, es decir, un tipo estructurado puede tener subtipos que reutilicen todos sus atributos y contengan atributos adicionales específicos del subtipo. Otra ventaja, el comportamiento de objeto, está relacionada con el acceso a los objetos del programa. Dichos objetos del programa deben ser almacenables y transportables para el procesamiento de la base de datos, por lo tanto, generalmente se les llama objetos persistentes. Dentro de una base de datos, todas las relaciones con un objeto de programa persistente son relaciones con su identificador de objeto. Todos estos puntos se pueden abordar en un sistema relacional adecuado, aunque el estándar SQL y sus implementaciones imponen restricciones arbitrarias y una complejidad adicional.

En la programación orientada a objetos (OOP), el comportamiento del objeto se describe a través de los métodos, que son operaciones que se pueden realizar sobre el objeto. Los métodos designados con un nombre se distinguen por el tipo de sus parámetros y el tipo de objetos a los que están asociados, conjunto de elementos que se conoce como firma del método. Los lenguajes de OOP llaman a esto el principio de polimorfismo, que se define brevemente como *one interface, many implementations*, en castellano "una interfaz, muchas implementaciones" [1]. Otros principios de OOP, como la herencia y la encapsulación, están relacionados tanto con los métodos como con los atributos. La herencia de métodos está incluida en la herencia de tipos.

Una base de datos objeto relacional es aquella que incorpora aquellos aspectos más destacados del paradigma de la programación orientada a objetos, como ejemplo tenemos la encapsulación, propiedad que permite ocultar la información que contiene un objeto impidiendo la libre manipulación de los atributos de un objeto, permitiendo la consulta y modificación de estos atributos únicamente a través de métodos públicos que ofrece la clase que describe el objeto. Tenemos luego la herencia, que es una propiedad a través de la cual ciertos objetos pueden heredar los comportamientos y atributos de otra clase comúnmente llamada clase madre. Tenemos también el polimorfismo que es una propiedad que nos permite tener varios métodos con el mismo nombre, pero definiendo diferentes funcionalidades según convenga.

En una base de datos objeto relacional los datos no son atómicos ya que podemos tener estructuras que engloban otros tipos, por ende, no se puede garantizar que estén en 1FN, por tanto, el concepto de normalización no se aplica. Los objetos de una base de datos constan generalmente de atributos y métodos, ambos recogidos en estructuras que pueden llegar a ser de gran tamaño y que a la vez describen las diferentes relaciones que hay entre objetos de nuestra base de datos objeto-relacional.

# Tipos de datos

Un tipo de dato hace referencia a una estructura definida por el usuario que puede contener uno o más tipos de datos. Contiene atributos y generalmente funciones y procedimientos con los cuales podemos operar sobre los atributos. Las variables simples o built-in permiten guardar únicamente un tipo de dato, e.g. INT, VARCHAR, DATETIME, entre otros. Sin embargo, las estructuras nos permiten englobar varios de los tipos mencionados anteriormente e incluso otras estructuras, de tal manera que se pueden anidar.

Normalmente pensamos en los objetos como un grupo de atributos y métodos con los cuales podemos operar sobre los atributos. Por ejemplo, si tenemos un objeto Persona, a este le podemos asignar los atributos documento nacional de identidad, edad, fecha de nacimiento, estudios, progenitores, entre otros. Partiendo de estos atributos podemos generar métodos que nos permitan trabajar sobre estos datos, así como consultarlos o modificarlos. Podemos en estas situaciones hacer ciertas optimizaciones como, por ejemplo, no guardar la edad y tener un método que nos puede calcular la edad de cualquier objeto persona en base a la fecha de nacimiento.

# Definición de tipos de objeto

# Herencia

# Tipos de datos colección

# Bibliografía

|  |  |
| --- | --- |
| [1] | Wikipedia, «Object-relational database,» 31 enero 2023. [En línea]. Available: https://en.wikipedia.org/wiki/Object%E2%80%93relational\_database. [Último acceso: 7 mayo 2023]. |
| [2] | Wikipedia, «Bases de datos objeto-relacional,» 11 junio 2021. [En línea]. Available: https://es.wikipedia.org/wiki/Base\_de\_datos\_objeto-relacional. [Último acceso: 7 mayo 2023]. |
| [3] | «BBD Objeto relacional - Bases De Datos Avanzada,» [En línea]. Available: https://sites.google.com/a/espe.edu.ec/bases-de-datos-ii/introduccion/bdd-objeto-relacional. [Último acceso: 2023 mayo 2023]. |