Sistemas de Bases de Datos 2

**Artículo Bases de Datos Objetos Relacional**

Miguel Zamora

Mauricio Morales

Tania Gutiérrez

**Estructura**

 Titulo

 Resumen

 Palabras Clave

 Introducción (Tema y el ¿Cómo?)

 Problema (que se toca en la introducción)

 Teoría

 Metodología

 Caso

 Conclusión

 Recomendaciones

 Trabajos Futuros

 Referencias Bibliográficas

**Titulo**

Como las Bases de Datos Objetos Relacional pueden apoyar al Sistema Argos en la Escuela de Ingeniería Informática

**Objetivo General**

Desarrollar una Base de Datos Orientada a Objetos para reemplazar módulos del Sistema Argos de las Escuelas de la Universidad Católica Andrés Bello sede Guayana.

**Objetivos Específicos**

● Identificar los módulos más relevantes del Sistema Argos.

● Modelar una nueva Base de Datos Orientada a Objetos a partir de los datos extraídos de los módulos seleccionados.

● Implementar el diseño previamente elaborado.

● Realizar recomendaciones para trabajos futuros.

**Resumen**

Con el pasar de los años se han estudiado varias características del lenguaje de definición de datos extendido, así como del lenguaje de consultas y, en especial, dar soporte a los atributos valorados como conjuntos, a la herencia y a las referencias a las tuplas. Estas extensiones intentan conservar los fundamentos relacionales—en particular, el acceso declarativo a los datos—a la vez que se extiende la potencia de modelado. Pero esto es algo que el modelo E-R no podía cumplir por no estar a un nivel semántico suficientemente rico para representar los datos y relaciones complejos necesarios para aplicaciones avanzadas.

Los lenguajes de programación orientados a objetos presentaron una forma alternativa de los programas de diseño, en los que las estructuras de datos y sus operaciones tienen importancia primaria. La programación orientada a objetos se reconoce ampliamente como un método para producir código reutilizable muy confiable. Con el auge de la tecnología, la promesa de los primeros lenguajes orientados a objetos, y la popularidad de los lenguajes orientados a objetos C++ y Java, también influyeron en el modelado de las bases de datos, evaluando la posibilidad de que las técnicas actuales adoptaran esta tecnología, naciendo Los Sistemas de Bases de Datos Orientados a Objetos.

Este sistema ofrece un camino de migración cómodo para los usuarios de las bases de datos relacionales que desean usar las características orientadas a objetos. Algunas características que este sistema trajo fueron los tipos de conjuntos incluyen las relaciones anidadas, los conjuntos, los multiconjuntos y los arreglos que estos permitían que los atributos de las tablas sean conjuntos. Proporciona herencia con subtipos y subtablas, así como referencias a objetos (tuplas).

La norma SQL:1999 extiende el lenguaje de definición de datos y de consultas con los nuevos tipos de datos y la orientación a objetos. Se han estudiado varias características del lenguaje de definición de datos extendido, así como del lenguaje de consultas y, en especial, da soporte a los atributos valorados como conjuntos, a la herencia y a las referencias a las tuplas. Estas extensiones intentan conservar los fundamentos relacionales—en particular, el acceso declarativo a los datos—a la vez que se extiende la potencia de modelado.

Las extensiones persistentes de C++ y Java integran la persistencia de forma elegante y ortogonalmente a sus elementos de programación previos, por lo que resulta fácil de usar.

**Palabras Clave**

* Relaciones anidadas.
* Modelo relacional anidado.
* Conjuntos: Tipos de Conjuntos
* Arreglos (Arrays).
* Tipos complejos:

Tipos de objetos grandes.

Multiconjuntos.

Tipos estructurados.

* Métodos.
* Funciones constructoras.
* Herencia:

Simple.

Múltiple.

* Herencia de tipos.
* Tipo más concreto.
* Herencia de tablas.
* Tipos de referencia.
* Anidamiento y des anidamiento.
* Funciones y procedimientos en SQL.

**Introducción**

Las aplicaciones tradicionales de las bases de datos consisten en tareas de procesamiento de datos, como la gestión bancaria y de nóminas, con tipos de datos relativamente sencillos, que se adaptan bien al modelo relacional. A medida que los sistemas de bases de datos se fueron aplicando a un rango más amplio de aplicaciones, como el diseño asistido por computadora y los sistemas de información geográfica, las limitaciones impuestas por el modelo relacional se convirtieron en un obstáculo. La solución fue la introducción de bases de datos basadas en objetos, que permiten trabajar con tipos de datos complejos.

El primer obstáculo al que se enfrentan los programadores que usan el modelo relacional de datos es el limitado sistema de tipos soportado por el modelo relacional. Las aplicaciones con el pasar de los años van creciendo su complejidad y se necesitan tipos de datos del mismo nivel de complejidad, como las estructuras de registros anidados, los atributos multivaluados y la herencia, que los lenguajes de programación tradicionales soportan.

El modelo de datos relacional orientado a objetos extiende el modelo de datos relacional ofreciendo un sistema de tipos más rico que incluye tipos de datos complejos y orientación a objetos. Hay que extender de manera acorde los lenguajes de consultas relacionales, en especial SQL, para que puedan trabajar con este sistema de tipos más complejo. Estas extensiones intentan conservar los fundamentos relacionales mientras extienden la potencia de modelado. Los sistemas de bases de datos basados en el modelo objeto-relación, ofrecen un medio de migración cómodo para los usuarios de las bases de datos relacionales que deseen usar características orientadas a objetos.

El segundo obstáculo es la dificultad de acceso a los datos de la base de datos desde los programas escritos en lenguajes de programación como C++ o Java. La mera extensión del sistema de tipos soportado por las bases de datos no resulta suficiente para resolver completamente este problema. Las diferencias entre el sistema de tipos de las bases de datos y el de los lenguajes de programación hace más complicados el almacenamiento y la recuperación de los datos, y se debe minimizar. Es deseable, para muchas aplicaciones, contar con estructuras o extensiones del lenguaje de programación de forma nativa que permitan el acceso directo a los datos de la base de datos, sin tener que pasar por un lenguaje intermedio como SQL.

**Planteamiento del Problema**

En la Universidad Católica Andrés Bello, es muy importante la buena gestión de la información de sus estudiantes, profesores y todo personal administrativo, por lo que la velocidad de acceso y actualización a esta es fundamental para que el personal pueda realizar sus tareas con eficacia y productividad.

Uno de los sistemas que son ampliamente utilizados por el personal de la Universidad es el Sistema Argos, el cual es el encargado de realizar las diferentes consultas a la gran base de datos que se encuentra en el servidor de la UCAB, y el cual provee gran parte de la información que utiliza el personal para realizar sus actividades en el día.

Uno de los grandes problemas de este sistema, es la gran cantidad de tiempo que le lleva a hacer las consultas, ya sea por el gran volumen de datos o por la manera en que están relacionadas las diferentes tablas que requiere una consulta específica; por lo que la mayoría del tiempo, el personal pierde una cantidad considerable de tiempo a la espera de la respuesta del sistema.

**Marco Teórico**

Primero hay que aclarar el principio que hizo posible lo que vendrá a continuación y es la SQL:1999 (también conocida como SQL3), que fue la cuarta revisión del estándar SQL. Introdujo un gran número de nuevas características, las cuales fueron las que abrieron paso a la evolución Orientado a Objetos de las Bases de Datos.

Para hablar de un enfoque orientado a objetos se tienen que aclarar ciertos aspectos, como el núcleo de este enfoque que son los objetos, estos son una entidad provista de un conjunto de propiedades o atributos (datos) y de comportamiento o funcionalidad (métodos). Es la unidad individual que en tiempo de ejecución realiza las tareas de un programa. Estos objetos interactúan unos con otros, en contraposición a la visión tradicional en la cual un programa es una colección de subrutinas. Cuando los objetos similares trabajan en conjunto se hace lo que se llama una Clase; esta consta de métodos y de datos que resumen las características comunes de dicho conjunto. En un lenguaje de programación orientada a objetos se pueden definir muchos objetos a partir de una clase.

Entre las características más útiles de los objetos tenemos la herencia, es una propiedad que permite que los objetos sean creados a partir de otros ya existentes, obteniendo características (métodos y atributos) similares a los ya existentes. Este mecanismo nos permite crear clases derivadas a partir de clase base.

Las aplicaciones de bases de datos tradicionales consisten en tareas de procesamiento de datos, para que una base de datos funcione estos datos tienen que relacionarse. En las [bases de datos](http://www.alegsa.com.ar/Dic/base%20de%20datos.php) relacionales, una relación o vínculo entre dos o más [entidades](http://www.alegsa.com.ar/Dic/entidad.php) describe alguna interacción entre las mismas. En [SQL](http://www.alegsa.com.ar/Dic/sql.php) las relaciones pueden implementarse como [tablas](http://www.alegsa.com.ar/Dic/tabla.php) (lo mismo que las [entidades](http://www.alegsa.com.ar/Dic/entidad.php)), aunque no todas las relaciones terminan en tablas, algunas relaciones son más bien conceptuales.

En los últimos años, ha crecido la demanda de formas de abordar tipos de datos más complejos. Considérense, por ejemplo, las direcciones. Mientras que una dirección completa se puede considerar como un elemento de datos del tipo cadena de caracteres, esa forma de verlo esconde detalles como la calle, la población, la provincia, y el código postal, que pueden ser interesantes para las consultas. Por otra parte, si una dirección se representa dividiéndola en sus componentes (calle, población, provincia y código postal) la escritura de las consultas sería más complicada, pues tendrían que mencionar cada campo. Una alternativa mejor es permitir tipos de datos estructurados, que admiten el tipo dirección con las subpartes *calle, población, provincia y código\_postal*.

Con sistemas de tipos complejos se pueden representar directamente conceptos del modelo E-R, como los atributos compuestos, los atributos multivaluados, la generalización y la especialización, sin necesidad de una compleja traducción al modelo relacional. La posibilidad de usar tipos de datos complejos como los conjuntos y los arreglos puede resultar útil en muchas aplicaciones, pero se debe usar con cuidado.

Los tipos de Datos complejos que maneja las Bases de Datos Objetos Relacional, son los tipo de Datos Estructurados, Multiconjuntos y Datos de Gran Tamaño.

Los tipos estructurados permiten representar directamente los atributos compuestos de los diagramas E-R. En SQL estos tipos se denominan tipos definidos por el usuario. Ahora se pueden usar esos tipos para crear atributos compuestos en las relaciones, con sólo declarar que un atributo es de uno de estos tipos.

**create type** Nombre **as**(

nombredepila **varchar**(20),

apellidos **varchar**(20)

)**final**

**create type** Dirección **as**(

calle **varchar**(20)

ciudad **varchar**(20)

códigopostal **varchar**(9)

)**not final**

Se puede tener acceso a los componentes de los atributos compuestos usando la notación “punto”; por ejemplo, nombre.nombredepila devuelve el componente nombre de pila del atributo nombre. El acceso al atributo nombre devolvería un valor del tipo estructurado Nombre.

SQL soporta dos tipos de conjuntos: arrays y multiconjuntos; los tipos array se añadieron en SQL:1999, mientras que los tipos multiconjunto se agregaron en SQL:2003. Recuérdese que un multiconjunto es un conjunto no ordenado, en el que cada elemento puede aparecer varias veces. Los multiconjuntos son como los conjuntos, salvo que los conjuntos permiten que cada elemento aparezca, como mucho, una vez.

Supóngase que se desea registrar información sobre libros, incluido un conjunto de palabras clave para cada libro. Supóngase también que se deseara almacenar el nombre de los autores de un libro en forma de array; a diferencia de los elementos de los multiconjuntos, los elementos de los arreglos están ordenados, de modo que se puede distinguir el primer autor del segundo, etc.

**create type** Editor **as**(

nombre **varchar**(20),

sucursal **varchar**(20)

)

**create** **type** Libro **as**(

título **varchar**(20),

array\_autores **varchar**(20) **array** [10],

fecha\_publicación **date**,

editor **Editor**,

conjunto\_palabras\_clave **varchar**(20) **multiset**

)**create** **table** libros **of** Libro

Dentro de las bases de datos objeto relacional se debe tener en cuenta que los objetos poseen una identidad que los diferencian. Esto se implementa a través de un Identificador Único (Object Identifier), generado por el sistema. El valor de este identificador no es visible para el usuario externo, el sistema lo emplea internamente para identificar cada objeto de manera única, así como también, crear y manejar referencias entre objetos. El concepto de identidad hace que sea necesario distinguir:

• Igualdad de identidad: dos objetos son iguales si tienen el mismo OID. En este caso se suele decir que son el mismo objeto. Normalmente se representa por el símbolo “=”.

• Igualdad de valor: dos objetos son iguales si los valores de sus atributos son iguales. Normalmente se representa por el símbolo “==”.

Para manejar estas bases de datos hace falta un lenguaje especial debido a que los lenguajes de las bases de datos se diferencian de los lenguajes de programación tradicionales en que trabajan directamente con datos que son persistentes; es decir, los datos siguen existiendo una vez que el programa que los creó haya concluido. Las relaciones con sus tuplas de las bases de datos son ejemplos de datos persistentes. Por el contrario, los únicos datos persistentes con los que los lenguajes de programación tradicionales trabajan directamente son los archivos. Sin embargo, los lenguajes de programación persistentes son lenguajes de programación extendidos con estructuras para el tratamiento de los datos persistentes.

La manera tradicional de realizar las interfaces de las bases de datos con los lenguajes de programación es incorporar SQL dentro del lenguaje de programación, como puede existir el caso de los lenguajes de programación orientados a objetos, como C++ y Java, para hacerlos lenguajes de programación persistentes. Las características de estos lenguajes permiten que los programadores trabajen con los datos directamente desde el lenguaje de programación, sin tener que recurrir a lenguajes de tratamiento de datos como SQL. Por tanto, ofrecen una integración más estrecha de los lenguajes de programación con las bases de datos que, por ejemplo, SQL incorporado.

**Marco Metodológico**

**Área de Estudio:**

El trabajo de investigación se enfocará en el estudio de la base de datos del sistema argos de la escuela de Ingeniería Informática de la UCAB extensión Guayana, principalmente enfocándonos en su gestión de estudiantes, profesores, personal administrativo, asignaturas, carreras, semestres, prelaciones, horarios y notas.

**Tipo de estudio:**

La investigación se realizará con el propósito de reconocer la distribución actual del sistema para así proponer una base de datos orientada a objetos que permita manejar más eficiente todos aquellos datos que puedan ser representados de manera estructurada o en forma de conjunto o multiconjunto.

**Método de recolección de datos:**

Para la recolección de datos se utilizó la técnica de observación para reconocer la distribución de la base de datos actual y la investigación para determinar los posibles cambios que se propondrán

**Procedimientos del uso información:**

Se necesitó de la aprobación del director de escuela para acceder a la distribución de la base de datos actual de la cual se obtuvieron tablas que podrían representarse mediante un sistema objeto relacional como lo son: Estudiantes, Profesores, Personal administrativo, Notas, Asignaturas, Prelaciones y Horarios. Una vez que se obtuvieron estas tablas mediante los tipos de datos estructurados, multiconjuntos y la herencia, se representaron en un enfoque objeto relacional.

**Plan de Tabulación:**

Finalmente, con ambas distribuciones obtenidas se realizaron pruebas que nos permita conocer la eficiencia de cada uno de ellos mediante distintas consultas que nos permitan comparar sus resultados y así determinar cuál se debería aplicar para la gestión de los datos

**Caso Practico**

En el Caso Práctico que nos trae aquí se agarra algunas tablas que implementa la estructura de Base de Datos Relacionales que tiene el sistema Argos usado en la Escuela de Ingeniería Informática con el objetivo de agarrar esa estructura estándar y actualizarla a una estructura Objeto-Relacional.

Para hacer esto primero hay que implementar los tipos de datos complejos, se usaran los tipos de dato colecciones/multiconjuntos y los tipo estructurados por ser lo que mejor se adaptan para gestionar, organizar y administrar mejor las diferentes listas de estudiantes, profesores y personal administrativo presentes en la escuela.

Esto se demuestra cuando entendemos que los tipos de dato de colecciones son datos no ordenados en que cada elemento puede aparecer varias veces y se puede usar para los atributos multivaluados que están presentes en algún dato de los sujetos. Podemos evitar la normalización debido a que ya no cumple la 1FN.

En el caso del tipo de dato estructurado se puede usar para poder representar los atributos compuestos de manera más simplificada disminuyendo la cantidad de datos.

Basados en estas características nuestro caso práctico estará enfocado principalmente en la gestión de estudiantes, profesores y personal administrativo donde se hará uso del tipo de dato Colección y Estructurado para demostrar que la eficacia y mejoras que puede traer la actualización de las Base de Datos Objetos-Relacional.

Ventajas que trae la implementación:

* Su flexibilidad
* Soporte para el manejo de tipos de datos complejos
* Manipula datos complejos de forma rápida y ágil

Desventajas a considerar:

* La falta de estándares en la industria orientada a objetos
* La implantación de nueva tecnología requiere que los usuarios iniciales acepten cierto riesgo

**Conclusión**

Las Bases de Datos Objeto-Relacional tienen la capacidad de cubrir las necesidades de datos de aplicaciones donde la tecnología relacional comienza a tener problemas de desempeño, escalabilidad, flexibilidad y/o complejidad de mantenimiento.

La importancia del Objeto-Relacional radica en el manejo de objetos complejos y a la persistencia de ellos. Por otro lado, los estándares de persistencia han repercutido de manera importante en el desarrollo y auge de los Objeto-Relacional, marcando importante influencia.

Para finalizar, es importante destacar que aún existe inmadurez en el mercado para la adopción de esta tecnología Orientada a Objetos, por lo que debe analizarse con detalle la presencia del proveedor para adoptar su producto en una línea de producción sustantiva.

**Recomendaciones**

Hay diferencias considerables entre los diversos productos de bases de datos en cuanto al soporte de las características relacionales orientadas a objetos. Probablemente Oracle tenga el soporte más amplio entre los principales fabricantes de bases de datos. El sistema de bases de datos de Informix ofrece so porte para muchas características relacionales orientadas a objetos.

Tanto Oracle como Informix ofrecían características relacionales orientadas a objetos antes de la finalización de la norma SQL:1999 y presentan algunas características que no forman parte de SQL:1999.

Los diagramas UML son un método útil para representar clases y relaciones, y coinciden bien con el modelo de datos orientados a objetos. A partir de un diagrama UML es relativamente fácil traducir el diseño en definiciones de clase que correspondan de manera directa a los ítems en el diagrama, incluidas las relaciones entre clases.

Las relaciones entre clases, llamadas asociaciones, pueden ser unidireccionales o bidireccionales, que se representan mediante la definición de un inverso para la relación. Pueden especificar una relación “uno” con un miembro de una clase, o una “muchos”, que se representa al especificar Set antes del nombre de la clase relacionada.

La agregación es un tipo especial de relación que conecta un todo a sus partes. La generalización se indica mediante líneas que conectan las subclases a la superclase, con un triángulo sobre la línea en la superclase.

**Trabajos Futuros**

Para el futuro se tiene planteado completar el proyecto de adaptar el Sistema Argos a una estructura Orientada a Objetos, llevando su Base de Datos Relacional a Objetos Relacional mejorando mucho el rendimiento de esta.

Realizado esto ya no se quedaría solo en la Escuela de Ingeniería sino que se extendería a todas las escuelas de la Universidad Católica Andrés Bello Extensión Guayana.

**Referencias Bibliográficas**

* Silberschatz, A., Korth, H. y Sudarshan, S. (2006). Fundamentos de Bases de Datos. McGrawHill, España.
* Ricardo, C. (2009). Bases de Datos. McGrawHill, España