Ejercicios Conceptuales

Ejercicio Conceptual 11.1:

¿Qué es una base de datos orientada a objetos?

OODBMS significa sistema de gestión de bases de datos orientadas a objetos. Este sistema de gestión de bases de datos orientado a objetos es un DBMS donde los datos se representan en forma de objetos, tal como se usan en la programación orientada a objetos. Es decir, que cuando se realiza una consulta, estas bases de datos devuelven los objetos en su totalidad. Un objeto devuelto significa que sus atributos y métodos son tan utilizables como lo eran antes de que el objeto fuera alguna vez almacenado en la base de datos.

Ejercicio conceptual 11.2:

¿Cuál es la principal diferencia entre una base de datos relacional y una base de datos orientada a objetos?

Su principal distinción es que las **bases de datos relacional** sólo pueden contener tipos de datos primitivos, como un entero o texto. No son capaces de almacenar tipos de datos más complejos a menos que los datos sean serializados o divididos en componentes primitivos. De esta manera, un beneficio de las **bases de datos orientadas a objetos** es que, cuando se integra con un lenguaje de programación orientado a objetos, existe una consistencia mucho mayor entre la base de datos y el lenguaje de programación.

Sin embargo, a pesar de presentar una ventaja significativa cuando se trabaja con lenguajes de programación orientados a objetos también existen desventajas y, en algunos escenarios, la base de datos relacional sería la más adecuada.

- No hay duda de que las bases de datos relacionales son mucho más simple que las bases de datos orientadas a objetos; solo tienen datos primitivos que se almacenan de manera muy uniforme. Cuando se trata de datos muy simplistas que puede estar contenida en solo una o dos tablas, es más eficiente la base de datos relacional.
- No tienen estándares establecidos. Esto significa que hay menos herramientas de usuario para las bases de datos de objetos, y menos soporte en general para desarrolladores.
- La multitud de APIs de algún idioma específico también hace que sea más difícil portar una base de datos de objetos a una aplicación que está escrita en otro lenguaje de programación, porque probablemente implicaría el uso de una API diferente.

Ejercicio conceptual 11.3:

¿Qué entiende por Complejidad, Herencia, Persistencia y Encapsulación?

- **Complejidad:** las OODBMS tienen la capacidad de representar la compleja estructura interna (del objeto) con una complejidad multinivel.
- **Encapsulación:** una clase de objeto encapsula datos y acciones (métodos) que se pueden realizar en esos datos. De hecho, un objeto puede restringir el acceso directo a los datos subyacentes, que requieren que las modificaciones a los datos sean posibles solo a través de un método de objetos. Por ejemplo, una clase de empleado puede incluir un método para recuperar salario y otro método para modificar el salario. El método de modificación salarial podría incluir restricciones sobre salarios mínimos y máximos, y la clase podría permitir que no se manipule el salario fuera de estos métodos.
- **Herencia:** las clases de objetos pueden heredar las características de una clase padre. Los integrantes de la clase de empleado puede heredar todas las propiedades de una clase de personas (fecha de nacimiento, nombre, etc.) al agregar propiedades y métodos, como salario, fecha de empleado, etc.
- **Persistencia:** se permite crear objetos persistentes (el objeto permanece en la memoria incluso después de la ejecución). Esta característica puede resolver automáticamente el problema de recuperación y concurrencia.

Ejercicio conceptual 11.4:

¿Qué se entiende por SQL embebido?

Utilizando el ejemplo de SQL embebido diseñe un código para leer la lista de "Estudiantes" que cursan "Inglés" en "Idiomas" y asignarles una "Nota" de "10".

Cuando se quieren almacenar datos de aplicaciones que utilizan programas orientados a objetos a RDBMS se deben aplicar operaciones de lógica de negocios desde fuentes externas a la tabla, por ejemplo, mediante el uso de SQL incorporado o procedimientos precodificados. Para construir una aplicación eficaz y eficiente en el modelo relacional, el desarrollador debe tener un conocimiento exhaustivo de las tablas, las relaciones entre ellas y de estos componentes lógicos externos.
Para simplificar estos procedimientos se utiliza **SQL embebido**. Este tipo de consultas proporciona acceso a la base de datos desde un lenguaje de programación de propósito general requerido en los casos siguientes:

- Cuando no todas las consultas se pueden expresar en SQL, por ejemplo, las consultas recursivas no se pueden escribir en SQL.
- Existan acciones no declarativas: por ejemplo, imprimir informes no se puede hacer desde SQL.

El lenguaje de propósito general en el que se incrusta SQL es llamado lenguaje de host. En este lenguaje los comandos de SQL son embebidos y los datos son intercambiados entre el lenguaje de host y DBMS usando cursores. De esta forma, una consulta SQL pasa del lenguaje del host a una DBMS que calcula el conjunto de respuestas y un cursor se puede ver como un puntero en el conjunto de respuestas. Una vez obtenido este conjunto, la DBMS devuelve el cursor al lenguaje de programación, que puede usar el cursor para obtener un registro a la vez de acceso a la respuesta materializada.

EJERCICIO

```
#include <stdio.h>
#include <string.h>

// Declaración de la estructura del estudiante
struct Estudiante {
    char ssn[10];
    char nombre[50];
    char curso[20];
    int nota;
};

int main() {
    // Variables
    char curso[] = "Inglés";
    char departamento[] = "Idiomas";
    struct Estudiante E;

    // Consulta para obtener el número del departamento
    #EXEC SQL SELECT dnum INTO :dnum
    FROM Departamento
    WHERE dnombre = :departamento;

    // Declarar cursor para seleccionar estudiantes del curso especificado
    #EXEC SQL DECLARE Est CURSOR FOR
    SELECT * FROM Estudiante
    WHERE curso = :curso AND dno = :dnum
    FOR UPDATE;

    // Abrir cursor
    #EXEC SQL OPEN Est;

    // Fetch de estudiantes y asignación de nota
    #EXEC SQL FETCH Est INTO :E.ssn, :E.nombre, :E.curso, :E.nota;
    while (SQLCODE == 0) {
        // Asignar nota de 10
        E.nota = 10;

        // Actualizar nota del estudiante en la base de datos
        #EXEC SQL UPDATE Estudiante
        SET nota = :E.nota
        WHERE CURRENT OF Est;

        // Fetch siguiente estudiante
        #EXEC SQL FETCH Est INTO :E.ssn, :E.nombre, :E.curso, :E.nota;
    }
    // Cerrar cursor
    #EXEC SQL CLOSE Est;

    return 0;
}
```

Ejercicio conceptual 11.5:

¿Qué administra un OMG?

El Object Management Group (OMG) fue fundado con el propósito de desarrollar estándares para un modelo de objeto común que sería portátil e interoperable. Un foco del grupo ha estado en desarrollo de estándares para el modelo de objetos y los servicios conectados con él, y otro en estándares de modelado.
Estos estándares permiten que los clientes de ODBMS escriban aplicaciones portátiles y los mismos incluyen:

- Modelo de objetos
- Idiomas de especificación de objeto: entre los que se incluyen
    - Lenguaje de definición de objetos (ODL) para la definición de esquema
    - Formato de intercambio de objetos (OIF) para intercambiar objetos entre bases de datos
    - Lenguaje de consulta de objetos (OQL): Lenguaje declarativo para consultar y actualizar objetos de base de datos
    - Enlaces de lenguaje (C ++, Java, Smalltalk): que vincula el lenguaje del programa con el lenguaje de la base de datos e incluye
    - Lenguaje de manipulación de objetos Mecanismos para invocar OQL desde el lenguaje Procedimientos para operar en bases de datos y transacciones

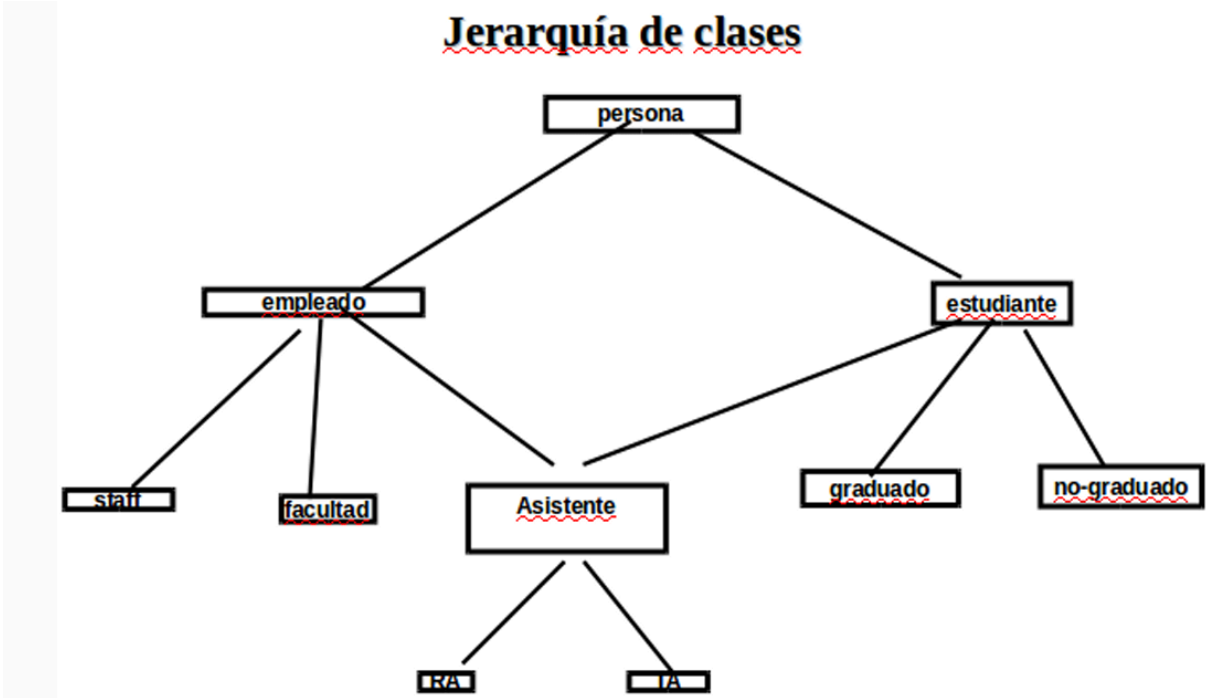
Ejercicio conceptual 11.6:

¿Qué entiende por Clase, Subclase, Jerarquía y Jerarquía de contención?

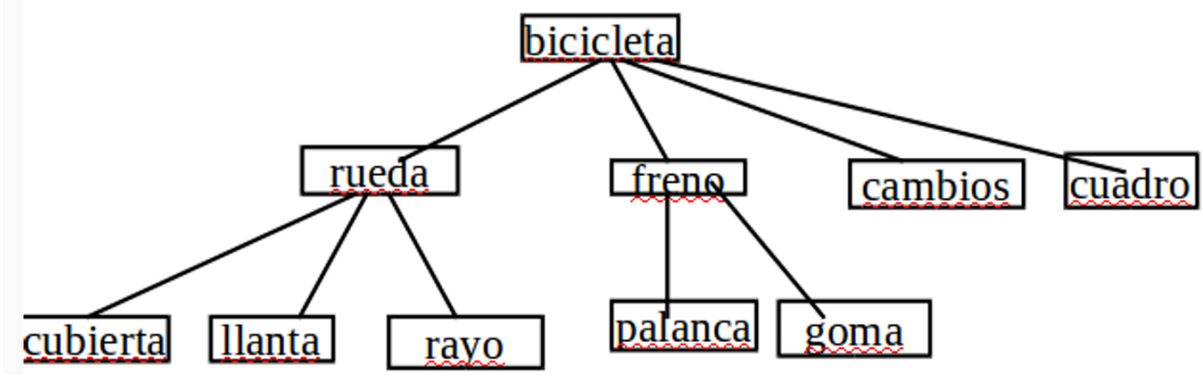
Dentro del modelo de datos (el cual ya se mencionó anteriormente) se pueden encontrar diferentes categorías para caracterizar a los objetos.

- **Clases:** Objetos similares con el mismo conjunto de propiedades y que describen conceptos similares del mundo real se recopilan en una clase. Cada clase se puede definir mediante una declaración de ODL.También, para cada clase de ODL, se puede declarar una extensión. Esta extensión es el conjunto actual de objetos que pertenecen a la clase.
- **Subclases y herencia:** Una clase también puede declararse como una subclase de otra clase. En este caso, la subclase definida hereda todas las propiedades de la superclase: atributos, relaciones y métodos. Cuando se define una subclase, se da la propiedad de sustituibilidad, por la cual cualquier método de superclase se puede invocar sobre los objetos de cualquier subclase. Esto implica una reutilización del código.
- **Jerarquía:** Dentro de las clases definidas se establece una especie de jerárquia, en la cual existe una clase principal y otras derivadas y/o relacionadas. En la siguiente

figura se muestra un ejemplo en el cual existe una clase Persona y dos subclases derivada de esta, empleado y estudiante, que a su vez tienen otras subclases. Es importante notar qua la subclase asistente vincula las subclases empleado y estudiante

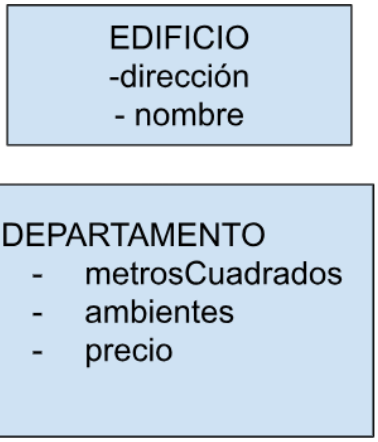


- **Jerarquía de contención:** En la siguiente figura se muestra un ejemplo de la jerarquía de contención que puede darse en los sistemas orientados a objetos. Cada enlace en la jerarquía de contención debe leerse como "es parte de", ya que, por ejemplo en la figura, la clase bicicleta contiene todas las subclases (partes de una bicicleta) que salen de la misma.



Ejercicio conceptual 11.7:

Piense una ejemplo de clase que contenga, por lo menos una subclase y varios atributos. Para esto utilice el formato declaratorio de ODL. Utilice la clase definida para diseñar un diagrama de jerarquía de contención.




```
// Definición de la clase "Edificio"
class Edificio {
    attribute String direccion;
    attribute String nombre;
    relationship Set<Departamento> tieneDepartamentos inverse Departamento.esParteDe;
};

// Definición de la subclase "Departamento" que hereda de "Edificio"
class Departamento : Edificio {
    attribute double metrosCuadrados;
    attribute int ambientes;
    attribute double precio;
    relationship Edificio esParteDe inverse Edificio.tieneDepartamentos;
};
```

Ejercicio conceptual 11.8:

Siguiendo el ejemplo de SQL3, diseñe una búsqueda de "Estudiantes" que colaboran en un proyecto de investigación llamado "Bases".