13019 - Diseño de bases de datos *Curso 2003-2004*
Fecha: *29 de enero de 2004*

HOJA INFORMATIVA

- No se permiten apuntes, libros, prácticas, etc...
- No se considerarán las respuestas escritas en lápiz.
- Cada pregunta debe ser contestada en su hoja correspondiente. No se corregirán las respuestas que no cumplan esta norma.
- Si no tiene suficiente con el espacio asignado para responder la pregunta, puede continuar en una hoja adicional. No olvide poner el nombre, los apellidos y el número de pregunta.
- No se corregirán las hojas sin nombre y apellidos.

Algunas funciones ODMG/C++:

Tipos y funciones de fecha y hora:

Item	Descripción
`d_Interval`	Lapso de tiempo
`d_Date`	Fecha (año, mes, día)
`d_Time`	Hora concreta del día
`d_Timestamp`	Fecha y hora

```
d_Date period1Start, period1End;
d_Date period2Start, period2End;
d_Interval lapso;
d_Boolean flag;

lapso = period1End - period1Start;
flag = period2Start.is_valid_date;
if (flag) {
  ...
}

flag = overlaps(period1Start, period1End, period2Start, period2End);
if (flag) {
  ...
}
```

Acceso a base de datos:

```
d_Database db;
const char * const db_name = "nombre";
...
db.open(db_name);
...
db.close();
```

Transacciones:

```
d_Transaction tx;
tx.begin();
...
tx.commit();
```

Creación y borrado de instancias persistentes:

```
d_Database db;          // Apunta a una base de datos abierta.
T *pA, *pB;             // Punteros a instancias de la clase T.
d_Ref<T> refC(pA);  // Puntero "inteligente." a una instancia de T.

// Creación de objetos persistentes
pA = new(&db, "T") T();
pB = new(pA, "T") T();

// Borrado de objetos persistentes
delete pB;
refC.delete_object();
```

Colecciones:

Colección	Ordenada	Permite duplicados
`d_Set<T>`	No	No
`d_Dictionary<K,V>`	No	No
`d_Bag<T>`	No	Si
`d_List<T>`	Si	Si
`d_Varray<T>`	Si	Si

Operaciones sobre colecciones:

```
d_Collection<d_Ref<T>> lista;
d_Ref<T> elemento;

lista.insert_element(elemento);   // Inserta un elemento en la colección.
                                  // La operación depende del tipo de la
                                  // colección.
lista.remove_element(elemento);   // Elimina un elemento de la colección.
lista.retrieve_first_element();   // Devuelve el primer elemento de la lista.
lista.retrieve_last_element();    // Devuelve el último elemento de la lista.
lista[i];                         // Devuelve el i-simo elemento de la lista.

d_Long cardinalidad = lista.cardinality(); // Número de elementos.

d_Boolean existe = lista.contains_element(elemento);
```

Iteración:

```
d_List<d_Ref<T>> lista; // o cualquier otro objeto d_Collection

d_Iterator<d_Ref<T>> iterador;
d_Ref<T> elemento;

for ( iterador = lista.create_iterator();
      iterador.not_done();
      iterador.advance() ) {

   elemento = iterador.get_element();
   ...
}
```

Extents:

```
d_Database db;    // Base de datos abierta
d_Extent<T> extent_de_T(db);
```

Clases relación:

Clase relación	Clase base
`d_Rel_Ref<T, const char *MT>`	`d_Ref<T>`
`d_Rel_Set<T, const char *MT>`	`d_Set<d_Ref<T>>`
`d_Rel_List<T, const char *MT>`	`d_List<d_Ref<T>>`

13019 - Diseño de bases de datos *Curso 2003-2004*
Fecha: *29 de enero de 2004*

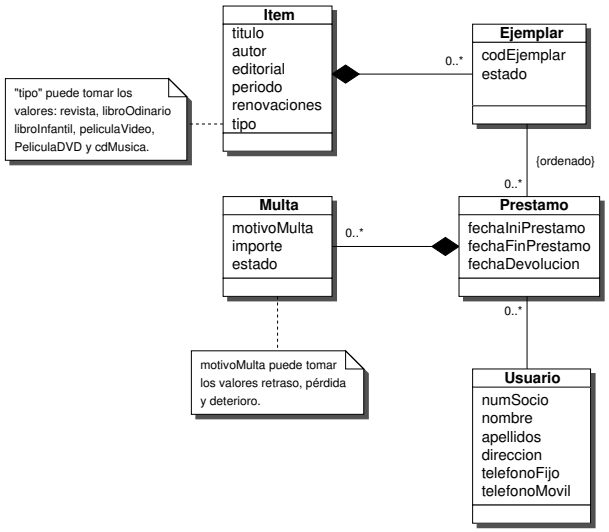
Nombre: _____ Apellidos: _____

1. (*4.00 puntos*) Desarrolle un sistema para hacer el seguimiento de los registros de préstamo de una biblioteca. Los usuarios pueden tomar prestados libros, revistas, películas en DVD o video y discos compactos de música. El sistema debe gestionar cada copia de los items de la librería; por ejemplo, la biblioteca puede contar con cinco copias del libro "Dune", tres copias en DVD de la película "La uvas de la ira", etc... El sistema debe mantener la información de los usuarios (nombre y apellidos, dirección, números de teléfono, etc...). Cada usuario tiene asignado un *carnet de lector* de la biblioteca que es único e intransferible.

   Cada categoría de *item* de la biblioteca cuenta con un periodo de préstamo estándar y un número máximo de renovaciones. Por ejemplo, los libros infantiles tienen un periodo de préstamo de un mes, mientras que los libros para adultos sólo se prestan durante dos semanas. El sistema debe mantener un registro de la fecha prevista de devolución y cualquier multa asociada al préstamo.

   El sistema debe también gestionar, además de las multas asociadas al retraso en la devolución, las multas debidas a los desperfectos y las pérdidas de las copias, junto con el importe correspondiente a dicho evento.

   En la figura siguiente se muestra un diagrama UML con una descripción aproximada del sistema:



   (a) (*0.25. puntos*) Añada en el diagrama de objetos la jerarquía de objetos representada implícitamente por la clase Item.

   (b) (*1.00 puntos*) Escriba, de acuerdo con el estándar ODMG/C++, la declaración de los atributos y relaciones de las clases descritas.

   (c) (*0.25 puntos*) Implemente el método void Item::ejemplares(void) que escriba en la salida estándar el número de ejemplares totales del Item, el código de cada ejemplar y su estado.

   (d) (*0.25 puntos*) Implemente el método void Ejemplar::prestamos(void) que muestre en la salida estándar la lista ordenada de todos los préstamos del ejemplar (fechas de préstamo y devolución y número de socio de la persona que lo tomo prestado).

(e) (*0.25 puntos*) Implemente el método void Usuario::prestamos(void) que escriba la lista de todos los Ejemplares que ha tomado prestados un usuario (fechas de préstamo y devolución y datos del Item).

(f) (*0.50 puntos*) Implemente el método void Usuario::multas(void) que muestre todas las multas asociadas al usuario (ejemplar, motivo, importe y estado) y el importe total asociado, el importe total recaudado y el importe total pendiente de cobrar.

(g) (*0.75 puntos*) Escriba un programa que "anule" los carnets de socio de todos los usuarios con un importe "excesivo" de multas sin pagar, en donde "excesivo" es un importe especificado por el bibliotecario.

(h) (*0.75 puntos*) Escriba un programa que "busque" las copias en préstamo con un retraso en la devolución "excesivo", en donde "excesivo" es un número de días de retraso especificado por el bibliotecario.

13019 - Diseño de bases de datos *Curso 2003-2004*
Fecha: *29 de enero de 2004*

Nombre: _____ Apellidos: _____

2. (*4.00 puntos*) Partiendo del sistema descrito en la pregunta anterior y del digrama UML allí representado, conteste a las siguientes cuestiones:

   (a) (*0.50 puntos*) Modifique el digrama de objetos para incluir expresamente la identidad de los objetos del modelo.

   (b) (*2.00 puntos*)Proyecte el diagrama de objetos del apartado (a) en el modelo relacional (u objeto–relacional).

   (c) (*0.25 puntos*) Escriba una consulta SQL que muestre el número total de multas impuestas a un usuario.

   (d) (*0.25 puntos*) Escriba una consulta SQL que muestre el importe total de todas las multas impuestas a un usuario.

   (e) (*0.25 puntos*) Escriba una consulta SQL que muestre el importe total de todas las multas pagadas por un usuario.

   (f) (*0.25 puntos*) Escriba una consulta SQL que muestre la lista de todas las multas impagadas de un usuario.

   (g) (*0.50 puntos*) Escriba una consulta SQL que muestre las copias prestadas con un retraso en la devolución "excesivo", en donde "excesivo" es un número de días de retraso especificado por el bibliotecario.

   Nota: dada una fechaFinPrestamo, el número de días enteros de retraso se puede calcular en PL/SQL de cualquiera de las siguientes formas:

   - SELECT FLOOR(SYSDATE-fechaFinPrestamo) [INTO var] FROM DUAL;
   - SELECT TO_CHAR(SYSDATE,'DDD')-TO_CHAR(fechaFinPrestamo,'DDD') [INTO var] FROM DUAL;

13019 - Diseño de bases de datos *Curso 2003-2004*
Fecha: *29 de enero de 2004*

Nombre: _____ Apellidos: _____

3. (*1.00 puntos*) Criterios de diseño del OQL.

13019 - Diseño de bases de datos *Curso 2003-2004*
Fecha: *29 de enero de 2004*

Nombre: _____ Apellidos: _____

4. (*1.00 puntos*) Desarrolle los siguientes puntos:
   (a) (*0.50 puntos*) Cómo aborda el modelo objeto-relacional de *Oracle* la necesidad de columnas multievaluadas?: Vectores (**VARRAY**) y tablas anidadas.
   (b) (*0.50 puntos*) Criterios para decidir entre vectores y tablas anidadas para implementar una columna multievaluada.