A

Tipo abstracto de datos 'Biblioteca'

Autor:
Temesio Vizoso, Silvana

Revista:
Información, Cultura y Sociedad

2010, vol.23, 117-135

Artículo

NOTAS DE INTERÉS PROFESIONAL

TIPO ABSTRACTO DE DATOS «BIBLIOTECA»

[LIBRARY ABSTRACT DATA TYPE]

SILVANA TEMESIO VIZOSO

Resumen: Análisis de la descripción bibliográfica desde un punto de vista conceptual a partir de la especificación formal: tipo abstracto de datos «*biblioteca*». Se detallan otros abordajes usando el modelo entidad-relación y el lenguaje XML. Se concluye la relevancia del enfoque abstracto y su aplicabilidad genérica en un sujeto de cambio permanente.

Palabras clave: TAD; Formatos bibliográficos.

Abstract: Analysis of bibliographic description from a conceptual point of view from the formal specification: «library» abstract data type. Other approaches are detailed using the entity-relationship model and XML language. We conclude the relevance of abstract approach and its general applicability in a subject of constant change.

Keywords: ADT; Bibliographic formats.

Escuela Universitaria de Bibliotecología y Ciencias Afines, EUBCA, Universidad de la República. Emilio Frugoni 1427, 11.200 Montevideo, Uruguay. Correo electrónico: stemesio@adinet.com.uy
Artículo recibido: 07-06-2010. Aceptado: 01-11-2010.
INFORMACIÓN, CULTURA Y SOCIEDAD. No. 23 (2010) p. 117-135

Introducción

Las bibliotecas, el modo de acceder a ellas, los usuarios y sus requerimientos, las revistas y hasta los libros están cambiando. ¿Cómo plantearse algunos temas técnicos desde un punto de vista más conceptual que procedimental? Esta pretende ser una aproximación en ese sentido.

Realizar una abstracción a partir de una realidad compleja que logre resaltar elementos significativos manejables a los efectos de estudio, es siempre un proceso subjetivo. Aún así, la manipulación de esta abstracción puede hacerse en un marco positivo.

En este caso se utiliza un sistema formal –el tipo abstracto de datos– para analizar el tema de los formatos bibliográficos desde un punto de vista operativo. El formato bibliográfico pasa a mirarse entonces como articulador de operaciones específicas como la recuperación de la información en formas variadas. La riqueza descriptiva del objeto es funcional a estas operaciones y no un fin en si mismo. Dependiendo de las operaciones que resulten significativas se modelará la descripción del objeto. En este sentido las operaciones que se detallan son simplistas y rudimentarias porque contribuyen a la presentación del enfoque y no al fondo de las mismas, que justamente está sujeto en estos tiempos a cambios y replanteos.

El enfoque formal tiene la bondad de aislar las distintas facetas para estudiarlas con independencia. El formato bibliográfico visto como infraestructura de acción en forma abstracta, se separa de implementaciones específicas: un software particular con un formato particular. El formato va más allá de una aplicación práctica concreta, tiene un alcance teórico que puede analizarse independientemente de que una aplicación específica lo implemente en forma incorrecta o parcial. Las bondades del formato en cuanto a los métodos o acciones que permite, es independiente del éxito de los emprendimientos que lo utilicen. Más aún, el elemento descriptivo está afín a las operaciones posibles, que especifique el sistema formal en forma abstracta, pero no es el eje del enfoque. Es el proceso de abstracción que define al objeto en su esencia.

Y es justamente esa esencia la que está sujeta a una revisión importante y una aproximación conceptual puede ayudar a enfocar el tema. El propósito de estas reflexiones es aportar al análisis de las descripciones bibliográficas –los formatos en este caso– a través de un modelo de datos y un enfoque abstracto de las operaciones que son objetivo del formato. Al mirar el objeto como sujeto de una acción susceptible de realizarse sobre él, el foco cambia, y detalles que no parecían relevantes pueden empezar a ser significativos y viceversa. La especificación abstracta conceptualiza e incluso valida. Podrán entonces cambiar los libros o las bibliotecas, pero la esencia de la necesidad de acceder a la información y al conocimiento, sigue vigente, y este enfoque permite validar las herramientas con prescindencia de la materialidad de los objetos a tratar.

Muchos asuntos están siendo revisados y replanteados y abordar estos cambios desde modelos conceptuales seguramente brindará un marco teórico sustentable.

Sistemas Formales

En el contexto de la modelización se puede ver un sistema formal como un conjunto de símbolos que se combinan respetando un conjunto de reglas de modo que solo las frases formadas a partir de estas reglas sintácticas son aceptadas. De todas las frases válidas, un subconjunto de ellas se seleccionan arbitrariamente, calificándolas de axiomas. Aplicando reglas de deducción lógica, puede mostrarse que otras frases del conjunto original de frases válidas, son coherentes con los axiomas. De hecho, el seleccionar los axiomas induce sobre el conjunto de frases válidas una partición. Algunas frases pasan a la categoría de deducibles de los axiomas, los teoremas, y otras serán no deducibles de los axiomas.

Las reglas sintácticas definen el conjunto de predicados afirmables. De algún modo establecen todo lo que se puede decir o escribir, en el marco del sistema formal. De todo lo afirmable debe delimitarse lo verdadero de lo falso. Aquí aparecen los axiomas ARBITRARIAMENTE designados como verdaderos. Ellos, con las reglas de deducción lógicas, hacen emerger como verdaderas a las frases deducibles, a la vez que hunden a las no deducibles.

En general, un sistema formal pretende modelar una situación real. De ésta, por abstracción se recogen los objetos y hechos relevantes.

Cuando se establecen los axiomas, se ha dado un paso importante: queda definido aquello que el sistema formal afirma. El proceso de deducción es independiente de la realidad. Solo depende del sistema formal.

Obviamente el sistema formal y la realidad son distintos, y teoremas del sistema formal pueden no corresponderse con hechos reales constatables. Como los teoremas son generados con independencia de la realidad o de la intuición, es a los supuestos de base que hay que culpar en ese caso. Más importante: la brecha formalismo - realidad es, en definitiva, solamente salvable por la experimentación. No hay modo de demostrar lógicamente su mutua adecuación porque demostrar es aplicar mecánicamente reglas a axiomas para obtener teoremas. En cuanto existen axiomas a partir de los cuales es posible deducir, existe una interpretación del objeto real estudiado. La confrontación del comportamiento real de este objeto con el comportamiento predicho es la única vía para obtener conclusiones sobre la posible invalidez de los axiomas seleccionados. La constatación de esta invalidez y la consecuente reformulación del sistema formal, sucesivamente aplicadas, constituyen la base del método científico tradicional.

Es importante tener en cuenta las conclusiones anteriores en el estudio de especificaciones formales de tipos abstractos de datos (TAD), en cuanto

son, de hecho, sistemas formales particulares. Pretenden modelar el comportamiento de objetos que, pueden a su vez ser creaciones abstractas. En este caso, la contracara del sistema formal no es un objeto manipulable físicamente, sino una idea intuitiva.

Sin embargo, sigue existiendo la posibilidad de confrontar el comportamiento intuitivamente esperado del que se deduzca de la especificación formal. Mas aún, subsiste la posibilidad de demostrar la adecuación de la especificación formal a una idea intuitiva.

Especificación algebraica de TAD

Supongamos que queremos definir un objeto cualquiera, por ejemplo un vehículo automotor.

Lo más normal en estas circunstancias es formarse una imagen visual particular (o eventualmente más de una en sucesión) de este objeto que se desea caracterizar. De hecho, la presencia de esta imagen es el antecedente inmediato de la definición, lo cual puede condicionar su generalidad. Nuestra intención es obtener las definiciones lo más abstractas, es decir, menos dependientes de una realización en particular y por lo tanto lo más generales que sea posible. Si bien podemos definir el objeto vehículo automotor por la vía de recorrer imágenes de realizaciones particulares, preferiremos otra aproximación: evitar el riesgo de comprometernos con casos particulares (coches, motos, camiones, ómnibus etc.) y extraer de todos estos los comportamientos comunes que nos importen. En una palabra, elegiremos una abstracción FUNCIONAL, en cuanto no manejaremos imágenes de vehículos sino sus comportamientos frente a determinadas operaciones. Habrá operaciones que permitan verificar propiedades del objeto. En nuestro caso: ¿está encendido?, ¿está en movimiento?, ¿qué trayectoria sigue? Existen otras que alteran estas propiedades, encender, arrancar, frenar, apagar.

En nuestro ejemplo trataríamos por esta vía de definir vehículo automotor sin hacer referencia a formas o imágenes, sino apenas a comportamientos.

Esto es sumamente importante, porque independiza las realizaciones particulares estableciendo apenas aquellas operaciones que pueden ser aplicadas al objeto en cuestión.

Está claro que estas funciones podrán implementarse luego de diversas formas, lo cual constituye una discusión de cómo efectuar una tarea determinada. Lo que se intenta entonces es separar la definición de la tarea, lo que hay que hacer, de cómo efectivamente se hace.

Vamos a presentar un ejemplo de un objeto pila, la pila podría asimilarse a un paquete de pastillas, en el cual el contenido del paquete está «apilado» y sigue el criterio del «ultimo en entrar es el primero en salir».

Ejemplo Objeto Pila

Una especificación algebraica del tipo PILA (de pastillas) sería la siguiente:

1)
Nueva: Pila
Poner: Pila X pastilla ————> Pila
Sacar: Pila ————————> Pila
Arriba: Pila ————————> pastilla

2)
Sacar(Nueva)=indefinido Sacar(Poner(pila,pastilla))=pila
Arriba(Nueva)=indefinido Sacar(Poner(pila,pastilla))=pastilla

Sin profundizar en el estudio de esta especificación algebraica, solo se señala que en **1)** se definen los dominios y recorridos de cuatro funciones: nueva, poner, sacar, arriba. Estas serían las operaciones aplicables al tipo pila. Los efectos de estas operaciones se vinculan en **2)**. A la parte **1)** la llamaremos especificación sintáctica y a la **2)** especificación semántica del objeto.

Al separar la especificación de la representación logramos dividir un gran problema en otros menores. El problema de elegir una implementación puede atacarse en profundidad con el objetivo de administrar lo más eficientemente posible los recursos disponibles.

La ventaja de esta aproximación es especialmente notable cuando nos enfrentamos a problemas de gran porte. En estos casos, mezclar discusiones sobre como hacer las cosas con discusiones sobre que hay que hacer, suele conducir a productos que realizan cosas diferentes a las propuestas.

Consideraciones generales

Todo lo expuesto anteriormente se apoya en una base formal, el álgebra y en particular el álgebra de tipos. Informalmente se puede decir que un álgebra de tipos elige un tipo de datos (un dominio de valores y un conjunto de operaciones aplicadas sobre ese dominio) y lo declara como tipo de interés (TOI-type of interest). El TOI (el conjunto de objetos cuyo comportamiento queremos definir) participa ya sea como dominio o como recorrido en todas las operaciones involucradas. Retomemos el ejemplo de la pila, la pila está presente en todas las operaciones ya que ha sido elegida como TOI.

Los otros conjuntos intervinientes son definidos separadamente. Veamos que en ninguna parte decimos que son las pastillas, ya que de ello nos ocuparíamos al definir el tipo pastillas. De esta manera se genera una sucesión de definiciones de tipos que debe comenzar con algún tipo primitivo. Este tipo primitivo es el booleano (un conjunto con dos valores distintos, generalmente

se asimila a verdadero y falso). La ventaja de elegir el álgebra de tipos y concentrarse en la definición de un tipo a la vez es la nitidez ganada en la definición de un tipo en particular, ya que se define un tipo complejo a partir de otros que o bien son conocidos o bien se definen separadamente.

El conjunto de axiomas debe ser consistente y completo. Consistente significa que dada una operación aplicada al mismo elemento el resultado será siempre el mismo. Resulta obvia la necesidad de obtener especificaciones consistentes. Una especificación resulta suficientemente completa si la combinación de los axiomas cubre todos los casos posibles. Es decir, todos lo resultados posibles están especificados en los axiomas.

Tipos Abstractos de Datos

Cuando se nos pide la definición de un objeto, el primer impulso suele ser dar una definición del tipo físico, haciendo énfasis en un montón de detalles del mismo, lo cual generalmente complica y oscurece la aproximación al objeto como entidad abstracta y poseedora de una serie de funciones que si, nos acercan más a la esencia del objeto.

¿Qué es un libro?, podríamos preguntar. Seguramente nos vamos a embarcar en determinar si es una monografía aislada o pertenece a una colección, que cantidad de páginas tiene a los efectos de ver si no entra en la categoría de folleto y un sinfín de detalles más que nos van a complicar más que aclarar la calidad del libro mismo.

Yo les propongo otro enfoque: digamos cuales son las operaciones que se pueden realizar con un libro y desentendámonos, al menos parcialmente de su estructura física.

El libro pasa a ser aquel objeto con el cual se pueden hacer ciertas cosas y por lo tanto descentramos el problema de definirlo a través de una enumeración extensiva de sus detalles físicos.

Como bibliotecólogos las operaciones que nos van a interesar son aquellas que involucran la recuperación de información; dado un libro determinar su autor, su título, la editorial, el tema, etc.

Para realizar cualquier operación necesitamos definir el dominio sobre el cual se realiza la operación. Se denomina dominio a un conjunto de valores. Si hablamos de la operación suma debemos decir sobre que dominio la realizamos, los números enteros, los números reales, etc. No es lo mismo sumar números enteros, que sumar quebrados, por lo tanto las operaciones deben definir sobre que dominio se realizan para quedar bien establecidas.

En nuestro caso el dominio sobre el cual trabajaremos es un poco especial, porque también nuestras operaciones son especiales. Podríamos decir que un libro es una concatenación de valores; un autor, un título, una cierta cantidad de páginas y un tema. Las operaciones sobre el objeto libro son, determinar, dado un libro, su autor, su título, su editorial, las páginas y el tema. Ahora bien, el

dominio del libro es un dominio compuesto por la suma de todos aquellos dominios componentes. El autor mapea sobre un conjunto que está formado por todos los autores posibles; el título, la editorial, las páginas y el tema se comportan en forma análoga. Un libro es entonces un producto cartesiano (X) de autor, título, editorial, páginas, tema. El producto cartesiano de n conjuntos son todos los elementos formados por n componentes con todas las combinaciones posibles.

Conjunto A = (a1, a2, a3)
Conjunto B = (b1, b2)
A X B = { (a1,b1),(a1, b2),(a2, b1), (a2, b2),(a3, b1),(a3, b2) }

Poniendo un ejemplo:

Título = (La divina comedia, Romeo y Julieta, Hamlet)
Autor = (Shakespeare, Dante)
Título x Autor = { (La divina Comedia, Shakespeare), (La divina Comedia, Dante), (Romeo y Julieta, Shakespeare), (Romeo y Julieta, Dante), (Hamlet, Shakespeare), (Hamlet, Dante)}

Lo que antecede es la definición del producto cartesiano. Obsérvese sin embargo que hay elementos del producto cartesiano que no corresponden a la realidad –e. g. (Romeo y Julieta, Dante)–.

TAD Libros

Crear libro			
Autor X Titulo X Editorial X Páginas X Tema		-> Libro	**(1)**
Mostrar Autor:	Libro	-> Autor	**(2)**
Mostrar Titulo:	Libro	-> Título	**(3)**
Mostrar Editorial:	Libro	-> Editorial	**(4)**
Mostrar Páginas:	Libro	-> Páginas	**(5)**
Mostrar Tema:	Libro	-> Tema	**(6)**

Acá está presentada la especificación formal de la sintaxis del TAD libro. Se distingue una operación generadora **(1)**, que dice que dado un elemento significativo del producto cartesiano se genera un libro.

Crear libro
Autor X Titulo X Editorial X Páginas X Tema -> Libro **(1)**

Una materialización de esta operación podría ser:

Crear libro
Benedetti x La tregua x Arca x 123 x novela uruguaya= (Benedetti, La tregua, Arca, 123, novela uruguaya)

Podría haber elementos como (Shakespeare, La Divina Comedia, Kapelusz, 40, Química orgánica) que pertenecen al producto cartesiano pero no son libros.

Lo que aquí se indica es que la operación generadora toma elementos del producto cartesiano y los constituye en libros, pero no se indica cómo. Otro asunto con el cual no nos involucramos es en la definición estricta de los dominios, es decir el dominio autor, es de la forma apellidos, nombres o al revés, tiene un largo establecido, existe un autor nulo –cuando se trata de un libro anónimo–, etc. Estos detalles no hacen a la delimitación del objeto libro y corresponden a una etapa de implementación más que a una especificación formal.

Las operaciones de extracción de elementos son:

Mostrar Autor:	Libro	-> Autor	**(2)**
Mostrar Titulo:	Libro	-> Título	**(3)**
Mostrar Editorial:	Libro	-> Editorial	**(4)**
Mostrar Páginas:	Libro	-> Páginas	**(5)**
Mostrar Tema:	Libro	-> Tema	**(6)**

La operación generadora *Crear libro* **(1)** produce mi primer objeto libro:

Crear libro(a, ti, e, p, te) donde a es un autor, ti es un título, e es una editorial, p son las páginas, te el tema genera un libro. Entonces aplica la operación *Mostrar autor* **(2)** sobre el objeto:

Crear libro(a, ti, e, p, te)

Las operaciones de extracción permiten, dado un objeto libro, determinar cada uno de sus componentes: autor, titulo, tema, editorial o páginas. La semántica de las operaciones de 2) a 6) son análogas, de modo que se detalla solamente la operación Mostrar Autor (2):

Mostrar Autor (Crear libro (a, ti, e, p, te)) = a

Esta operación da como era de esperarse un resultado trivial.

Una vez definido el TAD libro podemos definir el TAD biblioteca que es un conjunto de libros (Set of libros)

TAD Biblioteca (Set of libros)

Biblioteca-Nueva :		:	Biblioteca **(1)**
Agregar:	Biblioteca X Libro:		Biblioteca **(2)**
Pertenece:	Biblioteca X Libro:		booleano **(3)**
Quitar:	Biblioteca X Libro:		Biblioteca **(4)**

Aquí se ha presentado la especificación sintáctica de cada una de las operaciones, es decir, dada una operación, sobre qué dominio toma los valores para operar y sobre qué dominio toma los resultados, pero no se dice nada de cómo se realiza la operación (especificación semántica).

Veámoslo detalladamente.

La operación **(1)**, Biblioteca-Nueva, toma los valores de la «nada» y da como resultado una biblioteca. Intuitivamente se puede interpretar como el obtener un local vacío, sin libros para la biblioteca. La operación **(2)**, *Agregar,* realiza la inserción de libros en la biblioteca. Una biblioteca cualquiera se irá formando por sucesivas operaciones «Agregar».

Agregar **(biblioteca-Nueva, libro1)**
Agregar ((***Agregar***(**biblioteca-Nueva,libro1**)), libro2)
Agregar ((***Agregar***((***Agregar***(**biblioteca-Nueva,libro1**))**,libro2**)), libro3)

Y así seguiríamos indefinidamente agregando nuevos libros.

Cabe una aclaración sobre el dominio denominado booleano, el cual se considera predefinido, y es que este dominio tiene dos valores: verdadero y falso. Veamos la operación *Pertenece* **(3)**

Pertenece: Biblioteca X Libro: booleano **(3)**

La operación Pertenece **(3)**, lo que hace es: dada una biblioteca y un libro, determina si ese libro pertenece o no a la biblioteca, dando como resultado verdadero o falso respectivamente. Veamos la sintaxis. Notar que puedo aplicar la operación Pertenece sobre dos elementos: a) Biblioteca-Nueva o b) *Agregar (biblioteca-Nueva, libro1)*

a) Para una *Biblioteca-Nueva*

Pertenece (*Biblioteca-Nueva*, libro1) = falso

Una *Biblioteca-Nueva* no contiene ningún libro y por lo tanto libro1 no pertenece a *Biblioteca-Nueva.*

b) Para *Agregar (biblioteca-Nueva, libro1)*

Pertenece (***Agregar(biblioteca,libro1***),libro2) =
Si libro1 = libro2 ===> verdadero sino Pertenece (biblioteca,libro2)

Esto se interpreta de la siguiente manera: si realizo la operación sobre una biblioteca que se formó agregando a biblioteca el libro1, sencillamente nos preguntamos si libro1 es efectivamente libro2, donde libro2 es el libro sobre el cual estamos averiguando la pertenencia. Si efectivamente libro1=libro2, entonces resulta claro que este libro pertenece a la biblioteca y se responde verdadero. Si este no fuera el caso, es decir que el último libro agregado a la biblioteca no es el que preguntamos si pertenece, entonces debemos preguntar si libro2 –el libro buscado– pertenece a la biblioteca sobre la cual se agregó libro1. Esta es una estructura que llamamos RECURSIVA, es decir, la operación se sigue realizando

o bien hasta encontrar que algún libro agregado con anterioridad es efectivamente libro2 –y respondemos verdadero– o bien hasta llegar a la biblioteca vacía (biblioteca-nueva) donde ya sabemos de acuerdo a lo visto que nos da falso.

Nótese que si bien esta especificación semántica resulta un poco engorrosa, define perfectamente lo «que» hace la operación (sin decir cómo lo hace). Ordenadamente se va recorriendo la biblioteca y mirando si el último libro agregado es libro2 en cuyo caso contestamos verdadero. Si no fuera el caso, debemos repetir la operación sobre la biblioteca a la que se le agregó el libro1. De esta manera vamos revisando la biblioteca libro por libro hasta encontrar el libro2 o hasta vaciar la biblioteca.

Veamos la operación quitar

Quitar: Biblioteca X Libro: Biblioteca **(4)**

La operación *Quitar* consiste en sacar de la biblioteca un libro y se especifica así:
Notar que nuevamente puedo aplicar la operación *Quitar* sobre dos elementos: a) *Biblioteca-Nueva* o b) *Agregar (biblioteca-Nueva, libro1)*

a) Quitar (Biblioteca-Nueva, libro1)= Biblioteca-Nueva

b) Quitar(Agregar(biblioteca, libro1),libro2) =
Si libro2 = libro1 ➔ biblioteca
sino Agregar(*Quitar(*biblioteca,libro2),libro1)

Acá nuevamente tenemos una definición recursiva. Si queremos quitar a la biblioteca el último libro agregado entonces nos da la biblioteca. Si el último libro agregado no es el que queremos quitar entonces quitamos el libro2 a la biblioteca y luego le agregamos el libro1 que no es el que queremos sacar. Así seguimos sucesivamente hasta encontrar el libro a quitar y lo quitamos, o bien hasta llegar a Biblioteca-Nueva, es decir que el libro que queríamos quitar no estaba en la biblioteca, y por lo tanto no hacemos nada.

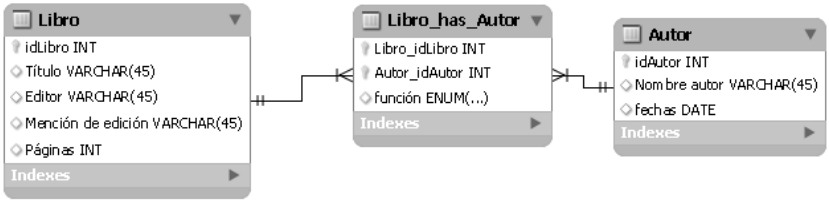
Hemos terminado de especificar nuestro TAD biblioteca de acuerdo a las operaciones que queremos realizar con ella pero no hemos implementado las operaciones sino solamente hemos dicho lo que queremos que haga la biblioteca. Las ventajas de este enfoque ya han sido ampliamente promocionadas, no se insistirá en ello.

Otras Miradas

Los elementos que aparecen en un producto cartesiano (autor, título, etc.), parecen estar en una estructura plana, pero esto es solo una forma de mostrarlos. En realidad los datos tienen su estructura y nos aproximamos a ellos en su complejidad a través de otras herramientas de modelado de datos como el

modelo entidad relación (MER) o el diagrama de clases de UML (Unified modeling language) y podemos entonces distinguir la forma en que se articulan.

Por ejemplo una estructura posible sería:



Entidad Libro	**Relación Libro-Autor**	**Entidad Autor**
Identificador del libro	Identificador del libro	Identificador de autor
Título	Identificador del autor	Nombre autor
Editor	Función del autor	Fechas
Mención de edición		
Páginas		

Instanciemos el modelo, es decir probemos con datos:

Libro
Identificador del libro = 1
Título = Romeo y Julieta
Editor= Kapelusz
Mención de edición= 2a.
Páginas=82

Entidad Autor
Identificador de autor=40
Nombre autor= Shakespeare
Fechas=1564-1616

Relación Libro-autor
Identificador del libro=1
Identificador de autor=40
Función del autor=autor

Este modelo de datos es aproximadamente el que usa el software integral de bibliotecas PMB.

¿Como es que se transforma un diagrama entidad relación en una base de datos relacional, es decir un conjunto de tablas? Existe un procedimiento por el que las entidades y las relaciones se transforman en tablas y los atributos de las entidades en campos. No detallaremos este proceso que tiene algunos aspectos

de complejidad, puesto que los datos que provienen del MER deben quedar normalizados, es decir sufrir una transformación que asegure su integridad y consistencia. En este caso, las entidades y las relaciones se transforman en tablas, los atributos en campos y los atributos determinantes en claves primarias.

Quedaría así:

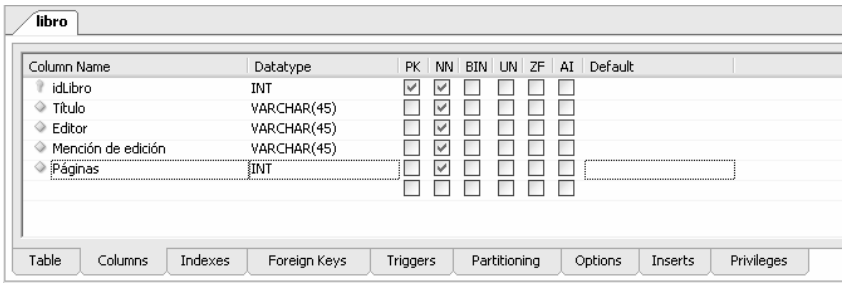


Tabla Libro
Identificador del libro – clave primaria
Título
Editor
Mención de edición
Páginas

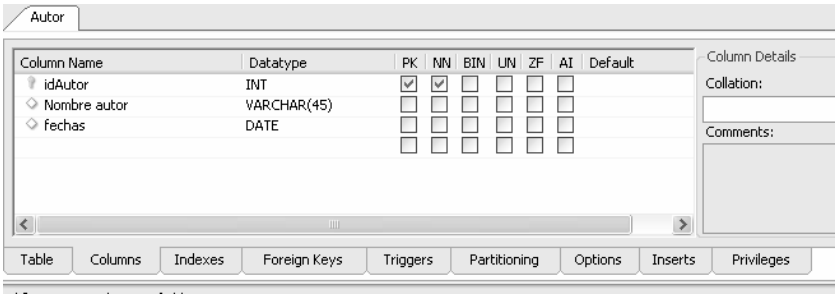


Tabla Autor
Identificador de autor – clave primaria
Nombre autor
Fechas

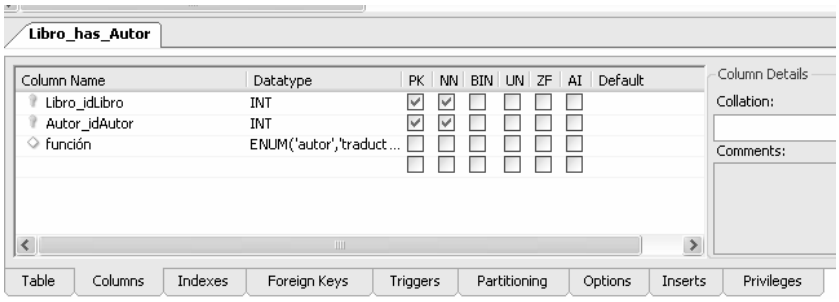


Tabla Libro-autor
Identificador del libro –clave primaria
Identificador de autor – clave primaria
Función del autor

El PMB (software integral para gestión de bibliotecas) trabaja con formato UNIMARC, pero esto no quiere decir que tiene una única tabla con todos los campos UNIMARC, sino que trabaja con campos en sus tablas que se corresponden con los tags (o etiquetas) del UNIMARC, por ej. en el caso de Shakespeare, el campo nombre de autor con la función autor, corresponden a la etiqueta 700 del UNIMARC o a la etiqueta 100 de MARC. Los formatos como MARC o UNIMARC establecen una serie de elementos que son indicaciones de almacenamiento para uso de la aplicación. Los demás elementos corresponden a conceptos. Los elementos se identifican a través de etiquetas numéricas. El instructivo del formato detalla claramente estas etiquetas, indicando el alcance de cada una. Algunas etiquetas albergan elementos de información que llaman subcampos y que se anteceden de un delimitador, por ej. el tag 700 subcampo f alberga las fechas extremas del autor principal lo que en nuestro caso correspondería al campo fecha de la tabla autor. Como cuando se exporta en este formato los datos aparecen en una secuencia del tipo: **...700$aShakespeare, William$d1564-1616 ...** existe una confusión generalizada de que los datos están en una única tabla. Posiblemente esta confusión provenga del uso extendido de Winisis con su tabla de definición de campos FDT que tiene una estructura lineal.

Los datos, sin embargo, tienen una estructura de cierta complejidad, no lineal, que refleja la riqueza de su articulación, pero a los efectos de intercambiarlos se presentan como una cadena en donde el dato específico se antecede del número de la etiqueta o el prefijo del subcampo.

En el pasado, el deseo de usar un formato estándar prevaleció, con el objetivo de poder hablar un mismo lenguaje y compartir datos, pero en la actualidad el paradigma de compartir los datos ha evolucionado y la idea que todos usen el mismo formato, ha sido sobrepasada por la idea de que todos puedan interoperar con los datos. Esta tendencia se generalizó en el campo de

los sistemas y en el de los datos. En los sistemas se programan cañerías que vinculan una aplicación con otra y en cuanto a los datos, lo importante es establecer tablas de equivalencia de un formato a otro para que se puedan intercambiar[1]. Existen muchos programas específicos que convierten de un formato a otro[2] y también dentro de las propias aplicaciones de biblioteca hay módulos que se ocupan de conversión de datos –como es el caso de PMB–.

Otra forma de intercambiar los datos en el ámbito de las bibliotecas es a través estándar Z39.50 que se ha vuelto una herramienta de amplio uso. El Z39.50 permite la extracción de datos de un generador de registros bibliográficos que los ofrece y que aclara el formato que usa. Con esta modalidad el cliente recupera la descripción del generador.

De modo que encontramos muchas formas de compartir datos: podemos exportar los datos de la base de datos y a través de una tabla de equivalencias tratarlos, para importarlos en otra base de datos con otro formato. Seguramente exportar e importar a través del Z39.50 será más sencillo, y es por eso que se recomienda verificar que la aplicación soporte este protocolo.

El XML es el lenguaje de intercambio de datos más difundido en todos los ámbitos y aporta otro modo de trabajar, más claro, más sencillo y muy extensible. El dato se encapsula entre dos etiquetas que indican donde empieza y termina el dato:

<título**> Romeo y Julieta </**título>
<autor> **Shakespeare, William** </autor>
<fechas> **1564-1616** </fechas>

El XML permite definir la estructuración del dato, señala que carácter es obligatorio, cual se puede omitir, de cual puede haber más de una ocurrencia, como es el orden, el tipo de datos y mucho más.

El que sigue es un ejemplo del documento que acompaña el archivo de datos XML (DTD) y que define la estructura:

<!ELEMENT Biblioteca (Libro)+>
<!ELEMENT Libro (Titulo, Autor+, Fecha?, ISBN)>
<!ELEMENT Titulo (#PCDATA)>
<!ELEMENT Autor (#PCDATA)>
<!ELEMENT Fecha (#PCDATA)>
<!ELEMENT ISBN (#PCDATA)>

La biblioteca –dice este DTD– es un conjunto de elementos libro, y el libro es una sucesión ordenada, de título, al menos un autor, fecha, que puede omitirse, e ISBN.

A su vez tenemos la combinación de presentar un formato conocido como MARC en XML en lo que se conoce como MARCXML.

Cada etiqueta de MARC se representa con un *datafield* y los subcampos con un *subfield*

He aquí un ejemplo de cómo se vería MARCXML:

```
<datafield >
   <datafield  tag 700>
         <subfield cod=»a»>Shakespeare, William</subfield cod=»a»>
         <subfield  cod=»d»>1564-1616</subfield  cod=»d»>
   <datafield tag 700>
</datafield>
```

Otra forma de definir estructuras en XML es con XML Schema. El ejemplo anterior de biblioteca, en XML Schema se vería así:

```
<?xml version=»1.0"?>
<xs:schema  xmlns:xs=»http://www.w3.org/2001/XMLSchema»
          targetNamespace=»http://www.libros.org»
          xmlns=»http://www.libros.org»
         elementFormDefault=»qualified»>
  <xs:element name=»Biblioteca» type=»TipoBiblioteca»/>
  <xs:complexType name=»TipoBiblioteca»>
    <xs:sequence>
                  <xs:element  name=»Libro»  type=»TipoLibro»
maxOccurs=»unbounded»/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name=»TipoLibro»>
    <xs:sequence>
     <xs:element name=»Titulo» type=»xs:string»/>
     <xs:element name=»Autor» type=»xs:string» maxOccurs=»3"/>
     <xs:element name=»Fecha» type=»xs:string»/>
     <xs:element name=»ISBN» type=»xs:string»/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

En este caso la biblioteca se describió como un tipo de datos *TipoBiblioteca* y el tipo de datos *TipoBiblioteca* es una secuencia de elementos *libros* y a su vez los *libros* son un tipo (*Tipolibro*) que resulta ser una secuencia de datos: título, autor (con máximo de 3), fecha e ISBN.

Como se verá el rastro del TAD llega hasta aquí, aunque solo queda un suave perfume. Pero adviértase que aquí solo hablamos de estructura, no de operaciones.

Desde el punto de vista de FRBR se miraría el asunto desde una concepción aún más abstracta, aunque también práctica, pero con un punto de vista abarcativo, universal, agrupando por una parte por ejemplo todas las

ediciones y formatos y desagrupando por la otra hasta llegar al nivel de detalle deseado. Si interesa una obra de Shakespeare en todos los idiomas, ediciones y aún formatos nada más adecuado que un modelo FRBRizado y en ese sentido la OCLC nos brinda una propuesta interesante[3]. Por un lado se abarca y agrupa pero también se quiere llegar a un nivel de detalle del ítem en una biblioteca en particular.

Sin embargo el FRBR es una concepción teórica y reflexiva y no es trasladable sin ajustes a un modelo práctico. La entidad ítem se asimilaría al libro del TAD biblioteca pero con un enfoque de mayor generalidad.

Item= (identificador, huella tipográfica, procedencia, marcas/inscripciones, estado, historial tratamientos, tratamiento programado, restricciones de acceso)

Aún dejando de lado el hecho de que el ítem forma parte tal vez de una agregación obra-expresión-manifestación-ítem , el propio ítem adolece de un problema a la hora de encarar la obtención del mismo, carece de atributo determinante. El ítem es sin duda una entidad débil y toma su fortaleza de una entidad biblioteca, entendida biblioteca como un sitio que contiene ítems, y dentro de biblioteca el identificador del ítem se establece perfectamente con lo cual la operación **obtener** puede llevarse a cabo.

Podríamos ver la manifestación como el producto cartesiano de los siguientes atributos que marca FRBR:

título de la manifestación
mención de responsabilidad
designación de edición/emisión
lugar de publicación/distribución
editor/distribuidor
fecha de publicación/distribución
fabricante/productor
mención de serie
forma del soporte
extensión del soporte
soporte físico
modo de captura
dimensiones del soporte
identificador de la manifestación
fuente de adquisición/autorización para el acceso
condiciones de disponibilidad
restricciones de acceso a la manifestación
tipografía (libro impreso)
tamaño de la letra (libro impreso)
foliación (imprenta manual)

colación (imprenta manual)
estado de la publicación (publicación seriada)
numeración (publicación seriada)
velocidad de reproducción (grabación sonora)
anchura del surco (grabación sonora)
tipo de grabación (grabación sonora)
configuración de la cinta (grabación sonora)
tipo de sonido (grabación sonora)
características especiales de reproducción (grabación sonora)
color (imagen)
escala de la reducción (microfilm)
polaridad (microforma o proyección visual)
generación (microforma o proyección visual)
formato de presentación (proyección visual)
requisitos del sistema (recurso electrónico)
características del archivo (recurso electrónico)
modo de acceso (recurso electrónico de acceso remoto)
dirección del acceso (recurso electrónico de acceso remoto)

El identificador de la manifestación podrá ser el ISBN si es un libro, pero podría ser un URI si el ítem es un objeto digital. En cualquier caso una manifestación queda perfectamente identificada y se le puede aplicar la operación **seleccionar**. Y una vez seleccionado el objeto manifestación navegar al ítem específico en una biblioteca en particular estableciendo un préstamo.

Conclusiones

Con respecto al tema que nos ocupa, los formatos bibliográficos, debe señalarse que éstos generalmente han sido pensados para ser utilizados en una cierta implementación, un software, pero subyace en ellos más allá de la implementación específica, lo que realizan las operaciones implementables y, los dominios y recorridos sobre los cuales las realizarían. Es decir, el TAD no se ve aquí como un paso previo a la implementación. Podríamos desentrañarlo de la implementación, pero por cierto no tendría generalidad. Las operaciones serían las que hubiesen sido implementadas, no las posibles o planteadas como significativas.

Hay dos aspectos a considerar: la definición de los dominios de valores sobre los cuales trabaja el formato, al que intentamos aproximarnos a través de un modelo lógico de datos (el modelo de entidad–relación) y las operaciones concretas que se pueden realizar con los datos, tengamos o no el software del cual nace el formato.

Establecidas claramente nuestras necesidades, planteando nuestra situación en forma general, y establecidos los datos y operaciones que se

pretenden realizar podemos enfocarnos con mayor claridad. Esta forma de abordar el tema no garantiza, sin embargo, una elección adecuada, ya que no es posible de ninguna manera demostrar la brecha realidad – abstracción.

No obstante, abordar la diversidad de soportes, presentaciones y los nuevos medios de trasmisión realizando un enfoque a la vez abstracto y funcional tiene la bondad de un abordaje más maleable a distintas miradas, despejando detalles que oscurecen el objetivo. En una instancia de cambio de paradigmas, un enfoque conceptual, una mirada abstracta, puede ser eficaz y aplicable a la web, al documento electrónico y a los repositorios digitales tanto como a las bibliotecas tradicionales.

Notas

[1] http://www.loc.gov/marc/unimarctomarc21.html
[2] http://www.loc.gov/marc/marctools.html
[3] http://oclc.org/developer/documentation/worldcat-search-api/using-api

Bibliografía consultada

Baillie, Jean. 1992. An Introduction to the Algebraic Specification of Abstract Data Types. <http://homepages.feis.herts.ac.uk/~comqejb/algspec/pr.html> [Consulta: 10 marzo 2009].

Federación Internacional de Asociaciones de Bibliotecarios y Bibliotecas. Requisitos funcionales de los registros bibliográficos: informe final. 2004. <http://archive.ifla.org/VII/s13/frbr/frbr-es.pdf> [Consulta: 1 agosto 2010].

Guttag, John V. 1977. Abstract data types and the development of data structures. En *Communications of the ACM*. Vol. 20, no. 6, 396-404.

Guttag, John V. 1980. Notes on Type Abstraction (Version 2). En *IEEE Transactions on Software Engineering*. Vol. 6, no. 1, 13-23.

Guttag, John V. 2002. Abstract data types, then and now. En *Software pioneers: contributions to software engineering*. New York: Springer-Verlag. p. 442-452.

Guttag, John V. y J. J. Horning. 1978. The algebraic specification of abstract data types. En *Acta Informatica*. Vol. 10, no. 1, 27-52.

Guttag, John V.; E. Horowitz y D. Musser. 1978. Abstract Data Types and Software Validation. En *Communications of the ACM*. Vol. 21, no. 12, 1048-1064.

IFLA. Grupo de Estudio sobre los Requisitos Funcionales de los Registros Bibliográficos. 2004. Requisitos funcionales de los registros bibliográficos: informe final. [Madrid]: Ministerio de Cultura. <http://archive.ifla.org/VII/s13/frbr/frbr-es.pdf> [Consulta: 1 agosto 2010].

Liskov, Barbara y Stephen Zilles. 1974. Programming with Abstract Data Types. En *ACM SIGPLAN Notices*. Vol. 9, no. 4, 50-59.

Moreno, Francisco; Jaime Echeverri y Roberto Flórez. 2008. La abstracción de datos y su proceso gradual de construcción. En *Dyna*. Vol. 75, no. 154. <http://redalyc.uaemex.mx/redalyc/src/inicio/ArtPdfRed.jsp?iCve=49615417> [Consulta: 18 noviembre 2009].

Sommerville, Ian. 2000. Algebraic specification 10. <http://www.comp.lancs.ac.uk/computing/resources/IanS/SE7/ElectronicSupplements/AlgebraicSpec.pdf> [Consulta: 8 mayo 2009].

Stotts, David. 1992. Algebraic Specifications and Abstract Data Types. <http://www.cs.unc.edu/~stotts/723/senotes/L06.html> [Consulta: 8 mayo 2009].

Stotts, David. 1992. Verification of Abstract Data Types. <http://www.cs.unc.edu/~stotts/723/senotes/L07.htm> [Consulta: 8 mayo 2009].

Varga, L. 1983. On the verification of abstract data types. En *Acta Cibernética*. Vol. 6, no. 1, 7-12. http://www.inf.uszeged.hu/actacybernetica/edb/vol06n1/pdf/Varga_1983_ActaCybernetica.pdf [Consulta: 8 mayo 2009].