Título:
Definición de tipos documentales en los productos de ECM. Implicaciones de la forma de implementación en SGBD.

Joaquín Hierro Torres

Resumen:
Este artículo reflexiona sobre las distintas alternativas elegidas por fabricantes de plataformas ECM para dar soporte en SGBD a la definición de los tipos documentales y las implicaciones de las diferentes soluciones desde el punto de vista de flexibilidad, rendimiento y evolución.

Palabras Clave:
Tipos Documentales, Bases de Datos, ECM, Modelado, Alternativas de Implementación, rendimiento, optimización

Abstract:
This work studies how to support in a DBMS document models and the implications of different implementations of ECM products.

keywords:
Document Types, Databases, ECM, Modelling, implementation alternatives, performance, optimization

Definición de tipos documentales en los productos de ECM. Implicaciones de la forma de implementación en SGBD.

1- Introducción.

En general, cuando se plantea la selección de un producto ECM para una entidad, de forma que sirva de infraestructura para los diferentes proyectos y departamentos de la misma, una de las preguntas básicas es :

“¿Permite la definición de tipos documentales?”

Y una respuesta habitual es:

“Si, por supuesto”.

De acuerdo a esta respuesta, se asume que, por tanto, podremos adaptarnos a las necesidades que puedan surgir en los proyectos futuros. La información necesaria (tanto las definiciones como los metadatos correspondientes a cada documento) se almacena en un SGBD. Cómo se estructure internamente esa información es responsabilidad del fabricante y en principio no parece que sea importante.
Se define un tipo documental, se empieza a almacenar información en el mismo, y cada vez que se introducen los metadatos, estos se almacenan como uno o varios registros de BBDD de forma transparente. Cuando se consulta, las búsquedas se convierten en una o varias consultas SQL sobre diversas tablas y se recupera finalmente un conjunto de documentos.

El problema es que, para permitirnos manejar esas definiciones, el fabricante/diseñador

debe elegir una forma de almacenar y estructurar las mismas y los metadatos correspondientes a cada documento. Esa elección afecta a las posibilidades reales y al rendimiento del producto final.

Es posible que finalmente se puede conseguir “lo mismo” con dos productos muy distintos, pero la diferencia puede ser de varios meses de tiempo de construcción, bastantes segundos de diferencia en la respuesta a una consulta, o varios servidores más para conseguir el mismo rendimiento.

Este artículo analiza la forma en que diversos productos estructuran internamente la información para comparar las ventajas e inconvenientes de cada uno.

2- Alcance del artículo.

Este artículo se centra exclusivamente en las posibilidades de definir los tipos documentales necesarios en herramientas de ECM (1, 2, 3). No abarca productos específicos que incluyen de forma predeterminada una estructura, definiciones y comportamiento (Ej. Bibliotecas).

Se trata de productos ECM, en el sentido utilizado por consultoras de referencia como Gartner o Forrester en sus estudios (3) como Documentum (4,5), Filenet (6,7), Oracle UCM (8), OpenText (9, 10) o IBM Content Manager(11, 12) , sobre los cuales, a partir de una parametrización o desarrollo se podrá construir aplicaciones documentales más o menos complejas o integrar los procesos documentales con procesos de negocio.

Dentro de los productos de propósito general, no se pretende estudiar las funciones de manejo de estructuras/o agrupaciones de documentos (carpetas, expedientes, ..) ni los tratamientos sobre los mismos (ciclos de vida, flujos de trabajo) ni la arquitectura general del producto, aspectos todos ellos que requerirían artículos en profundidad sobre cada uno.

Por supuesto, la selección de un producto debe hacerse teniendo en cuenta todos los requerimientos específicos del proyecto o de la entidad, incluyendo requerimientos funcionales, técnicos, de seguridad, escalabilidad, cumplimiento de normativas, etc. Sin embargo, las reflexiones de este artículo serían válidas como ayuda para las tareas de selección de un producto o diseño de uno propio.

Se puede argumentar que aunque la estructura no sea muy eficaz, los SGBD tienen diversas formas de optimización de las consultas (basadas en costes o reglas, creación de índices, caché, etc.). Pero una parte, el rendimiento será aún mejor si se “ayuda” al SGBD estructurando la información y por otra, hay necesidades (como asegurar la unicidad de valores de metadatos) que son muy complejas, sino imposibles, de cumplir si no los asegura el SGBD.

Hay que destacar que los criterios que se manejan se refieren principalmente a productos basados en SGBD relacionales (que son la gran mayoría del mercado). Los criterios para los SGBD basados de forma nativa en XML u orientados a objetos tienen un comportamiento diferente y en algunos casos evitan parte de los problemas o limitaciones expuestos más adelante. El articulo se centre en los productos basados en SGBD Relacionales por varios motivos:

1) Todos los productos líderes del mercado (3), así como la mayoría de los productos comerciales o de fuente abierto, se basan en producto de ese tipo.
2) Si se pretende construir una solución, la gran mayoría de los SGBD instalados en

cualquier organismo o empresa relacionales.

3) El principal problema de los SGBD no relacionales es su falta de portabilidad.

No se acota el análisis a ningún producto en concreto porque son limitaciones inherentes al propio modelado de los metadatos. Menos aún se centra en un SGBD concreto porque además de que la mayoría de los productos ECM soportan varios productos, el modelado en si es evidentemente independiente del SGBD relacional utilizado (salvo que utilicemos elementos propietarios como tipos de datos XML)

Una última reflexión; hay sistemas para los que no es posible definir tipos documentales. Incluyen una definición de partida a la que hay que atenerse con una serie de metadatos más o menos adecuados. En mi opinión, y como siempre recordando que no hay "sistema mejor" y que debe tenerse en cuenta todos los requisitos, son válidos solo para casos muy sencillos.
Otra variante es aquella en que puede añadirse cualquier número de etiquetas, pero sin estructurar. En este caso tenemos más posibilidades de describir la información, pero no es posible sistematizarla ni acotarla, por lo que no son aplicables a muchos escenarios, especialmente los ligados a procesos.
Por supuesto el artículo no aplica a ninguno de ellos

3- Definición de un supuesto

Para poder estudiar mejor el problema se plantea un escenario muy simplificado pero con necesidades reales:

Una institución X desea almacenar y manejar los documentos relativos a los expedientes de becas. Los documentos tiene distintos orígenes (digitalización de documentos aportados por los becarios, documento compuesto dentro de la institución, etc.), distntos formatos (PDF/A, tiff, Documento ofimático OpenOffice o Microsoft, ..) y distintos tipos documentales.
Se asume que se desean almacenar en el gestor dentro de una "carpeta" por solicitante, donde se almacenará todos los documentos recibidos.

Solo se manejaría 4 tipos documentales, que serían los siguientes:

1- Solicitud		
Documento Preimpreso cumplimentado a mano por el solicitante con información de diverso tipo. De la información recogida en el formulario solo se desea introducir en el gestor documental la información:		
Metadato	**Tipo de dato**	**Notas**
Código	Varchar(30)	Obligatorio, Código único de solicitud asignado al recogerlo.
Fecha	Fecha	Obligatorio.
Identificador Solicitante	Varchar(20)	Obligatorio, Nif / Número tarjeta residencia/ Número pasaporte.
Tipo Beca	Numérico	Obligatorio, a elegir de una lista.

2- Documento Identidad
Documento acreditativo de la identidad (DNI, tarjeta de residencia u otros). Se

digitalizará desde el original al recoger el formulario.		
Metadato	**Tipo de dato**	**Notas**
Identificador Solicitante	Varchar(20)	Obligatorio, Nif / Número tarjeta residencia/ Número pasaporte.
Tipo documento identidad	Numérico	Obligatorio, a elegir de una lista (1-DNI, 2-NIE,...)
Fecha	Fecha	Obligatorio.

3- Pasaporte		
Documento acreditativo de la identidad para extranjeros no residentes. Se digitalizará desde el original al recoger el formulario. Es un subtipo de **Documento de Identidad** que debido a la normativa X, debe contener el País.		
Metadato	**Tipo de dato**	**Notas**
Identificador Solicitante	Varchar(20)	Obligatorio, Nif / Número tarjeta residencia/ Número pasaporte.
Tipo documento identidad	Numérico	Obligatorio, a elegir de una lista (1-DNI, 2-NIE,...)
Fecha	Fecha	Obligatorio.
País	Numérico	Obligatorio, código telefónico de país

4- Hacienda		
Documento justificativo de los ingresos (IRPF, justificante retenciones, etc.) Se digitalizará al recoger el formulario.		
Metadato	**Tipo de dato**	**Notas**
Identificador Solicitante	Varchar(20)	Obligatorio, Nif / Número tarjeta residencia/ Número pasaporte.
Fecha	Fecha	Obligatorio

4- Expresando formalmente unas definiciones “teóricas”.

4.1 – Metadatos

Antes de centrarse en la definición de los tipos, hay que hacer una “parada” en los metadatos que componen la definición del tipo documental.

En esta fase ya surgen las primeras divergencias importantes que van a afectar tanto a la forma de estructurar los tipos como a la forma de implementar por debajo en BBDD.

La pregunta es: al definir un tipo documental, ¿se definen los metadatos directamente o elijo los metadatos desde un diccionario? O planteado de otra forma ¿los metadatos se normalizan y reutilizan o deben definirse para cada tipo?

Tenemos entonces estos enfoques:

A- Normalización/diccionario de metadatos

B- Metadatos específicos de cada tipo

Desde luego puede existir una normalización “por procedimiento”, disponiendo de un “libro de estilo” y reglas de creación muy claras o bien un equipo de normalización, pero las ventajas en cuanto a facilidad y disminución del número de errores de un sistema automático son grandes.

Con el enfoque A, el proceso sería:

A.1 - Definición de Metadatos

Identificador Solicitante	Varchar(20)
Código	Varchar(30)
Fecha Solicitud	Fecha
...	

A.2 - Creación de tipos.
A.3 - Asociación de metadatos a los tipos.

Con el enfoque B, se definiría los tipos documentales 1 a 4 “desde cero” (salvo que contemos con herencia, tal como se detalla en el apartado posterior “**Orientación a objetos”**).

La opción A, incluye una normalización alta, y permite una estructura de BBDD con tablas por metadato (que luego se comenta), pero puede implicar limitaciones a la evolución posterior. Si en un futuro se desea “redefinir” la información contenida en un tipo concreto (no en todos), como podría ser en el ejemplo ampliar el Identificador de solicitante solo para las Solicitudes, de forma que admitan otros tipos de documentos de identificación, el modelo está acoplado y repercute en todos los tipos que reutilizan el metadato. Modificar la definición del metadato afecta a todos, aunque no sean de la misma familia/clase de tipos documentales.
La opción B, aunque se presta a proliferación de definiciones desnormalizadas, tiene más libertad y combinada con Orientación a Objetos y una normalización por procedimientos o equipo centralizador puede permitir un funcionamiento más ágil.

4.2 – Orientación a Objetos
La otra pregunta, en mi opinión mucho más importante es ¿El producto está orientado a objetos? Aquí los matices son mucho mayores, ya que incluso en el ámbito de los lenguajes de programación que se auto-califican como orientados a objetos, hay diferencias, pero en este caso los requisitos mínimos serían que se pueda definir los tipos documentales como objetos con herencia y polimorfismo.

- En este contexto, se considera que los tipos documentales tienen herencia si los tipos definidos a partir de otro heredan dinámicamente de este sus metadatos, seguridad, ciclo de vida, y restricciones. Por tanto, si se define un tipo de documento **Informe**, que contenga como atributos **Titulo, Autores, Fecha, Resumen** y **Palabras Clave**, un subtipo Informe Médico tendría automáticamente los metadatos anteriores, y solo sería necesario definirle el metadato **Especialidad**.

- Se considera que tiene polimorfismo si para cualquier operación o proceso en que se espera un tipo de documento, puede aceptarse un documento de cualquiera de sus subtipos. En estas operaciones se incluiría tanto búsquedas (de forma que cuando busco un “Documento Identificativo” puedo recuperar los “Documentos

identificativos” “puros” o recuperar también cualquiera de los subtipos) como procesos automáticos (Ej. al recibir un informe, o subtipos, enviar un correo con texto de asunto el título del informe y crear una copia en la carpeta “Revisión”)

¿Cómo se plasmaría eso en nuestro ejemplo?

Podría definirse una jerarquía de tipos:

DocBeca
- Solicitud
- Documentos Identificativos
  - Pasaporte.
- Hacienda

Donde los metadatos de cada tipo serían:

DocBeca			
IdSolicitante	Fecha		
Solicitud			
		Codigo	Tipo Beca
Documentos Identificativos			
		TipoDoc Identidad	
Pasaporte			
			País
Hacienda			

Donde las celdas en gris indican los atributos heredados automáticamente de la clase padre.

Como se ha indicado antes, la herencia debe ser dinámica, así que cualquier cambio, después de la definición, sobre un tipo “padre” actualizará la definición de los tipos “hijos”.

4.3 – Implementación de los Tipos documentales

Con los criterios anteriores ¿Cómo modelar internamente los tipos?

Tabla única fija.

Una solución trivial es tener una única tabla que admite atributos de cadena, junto con un indicador del tipo documental, es decir:

Tabla1

Tipo Documental	**Attr1**	**Attr2**	**Attr3**	**Attr4**
Solicitud	12345	12/03/2010	AU89-34	23
Documento Identificativo	12345	12/03/2008	DNI	

Cada entrada de la tabla contiene los metadatos por orden, y un atributo adicional que es

el tipo documental, para poder interpretar el sentido de cada metadato. Para acceder a los metadatos, el producto debe “traducir” las posiciones para cada tipo documental. Es decir, la columna Attr1 de la tabla corresponde para cada tipo documental a un atributo diferente que se asigna por medio de las definiciones de los tipos.

Esta alternativa es muy limitada:

- No se puede añadir más columnas/metadatos de las definidas.
- No se puede superar la longitud máxima ya definida para cada columna.
- La comprobación y formato de los tipos debe hacerse por parte del producto, no por el SGBD
- No es posible crear índices de BBDD para optimizar las búsquedas o limitar valores, ya que la misma columna comparte tipos de metadatos diferentes con restricciones distintas.
- Se comparte una misma tabla, que crecerá mucho, bajando el rendimiento.

A las condiciones de búsqueda debe añadírseles siempre el tipo documental. Es decir, la búsqueda “Documentos de tipo='Solicitud' donde IdSolicitante='12345' ” se convertirá en una SELECT SQL sobre la Tabla1 donde TipoDocumental='Solicitud' and Attr1='12345' “.

Una tabla única para todos los tipos.
Esta alternativa, que puede parecer similar a la anterior es muy diferente. Se basa en crear una tabla a la que se añaden solo los metadatos o columnas necesarios al crear un tipo. La tabla resultante es entonces la unión de los metadatos de todos los tipos. Esto implica que quedarán muchos “huecos”, ya que se ha definido columnas para todos los metadatos pero cada documento usará los de su tipo:

Tabla1

Tipo Documental	IdSolicitante	Fecha	Código	Tipo Beca	TipoDoc Identidad	País
Solicitud	12345	12/03/2010	AU89-34	23		
Documento Identificativo	12345	12/03/2008			DNI	

Las principales ventajas son:

- Los tipos de datos corresponden al tipo, por lo que se refleja fielmente el modelo, aunque “sobran” internamente metadatos.
- Puede crearse índices de BBDD para optimizar
- Puede implementarse fácilmente un modelo orientado a objetos, pudiéndose buscar en varios tipos documentales simultáneamente.

Los principales inconvenientes son:

- La tabla crecerá enormemente, tanto en número de columnas como en registros.
- No es posible exigir en BBDD elementos como metadato requerido o asegurar unicidad de valores concretos (Ej código de solicitud), ya que muchos documentos no lo cumplirán al ser de otro tipo.

Una tabla completa por tipo documental.
En este modelo, se crea una tabla por cada tipo documental, con los metadatos propios del mismo relacionados uno a uno a las columnas de la tabla.

Las principales ventajas del modelo son:

- Los tipos de datos y estructura de cada tabla corresponden al tipo, por lo que se refleja fielmente el modelo.
- Puede crearse índices de BBDD para optimizar y asegurar unicidad de valores concretos (Ej. código de solicitud)
- La información está repartida entre distintas tablas, por lo que las búsquedas son bastante eficientes.

Los principales inconvenientes son:

- Si desconocemos el tipo documental buscado, la búsqueda debe hacerse en múltiples tablas, lo que implica una operación de UNION entre muchas tablas, que puede hacerse muy lenta o superar los límites establecidos para el SGBD.
- Similarmente, si se desea implementar orientación a objetos, las búsquedas y otras operaciones pueden ser poco eficaces o imposibles.
- Puede llegar a crearse muchas tablas en la SGBD.

Una tabla parcial por tipo documental.

Esta opción es más compleja y está orientada especialmente a los productos orientados a objetos.

En ella, se estructura la información creando una taba por cada tipo documental, pero esta tabla solo contiene los metadatos que NO tiene la clase padre.

La asociación entre tablas para unificar todos los metadatos de un documento concreto se haría por medio de un campo clave interno común a todas las tablas (campo que por otra parte ya suele incluirse para uso interno en gran parte de los productos)

Es decir, se crearían tablas:

DocBeca		
IdSolicitante	Fecha	*IdInternoDocumento*
Solicitud		
Codigo	Tipo Beca	*IdInternoDocumento*
Documentos Identificativos		
TipoDoc Identidad		*IdInternoDocumento*
Pasaporte		
País		*IdInternoDocumento*

Las principales ventajas del modelo son:

- Puede crearse índices de BBDD para optimizar y asegurar unicidad de valores concretos (Ej. código de solicitud)
- Las tablas creadas son relativamente compactas y con pocas columnas, aunque para los tipos con muchos subtipos tendrán muchos registros.
- Puede hacerse búsquedas y operaciones con herencia fácilmente, ya que en cualquier tabla está la información de un Tipo Documental y todos sus subtipos.

Los principales inconvenientes son:

- Cuando se desea recuperar todos los metadatos de un elemento, debe accederse

a varias tablas.
- Puede llegar a crearse muchas tablas en la SGBD.

Una tabla por metadato.
Esta solución consiste en crear una tabla por cada metadato distinto. Al igual que en el caso anterior, la asociación entre tablas para reunificar todos los metadatos de un documento concreto se haría por medio de un campo clave interno común a todas las tablas.

Es decir, se crearían tablas, cada una con un campo:

IdSolicitante	*IdInternoDocumento*
Fecha	*IdInternoDocumento*
Código	*IdInternoDocumento*
Tipo_Beca	*IdInternoDocumento*
TipoDoc_Identidad	*IdInternoDocumento*
País	*IdInternoDocumento*

Las principales ventajas del modelo son:
- Todos los campos automáticamente admiten valores múltiples.
- Las tablas creadas son relativamente compactas y con pocas columnas, aunque tendrán muchos registros.

Los principales inconvenientes son:
- Siempre que se desea recuperar los metadatos de un elemento, debe accederse a varias tablas.
- Las búsquedas son siempre complejas y sobre varias tablas.
- No puede imponerse condiciones de unicidad ni índices de BBDD.

4.4 – Manejo de versiones de los documentos
Es importante también la forma en que se almacena las versiones de los documentos (siempre referido a los metadatos, el almacenamiento del documento como tal tiene otras implicaciones). No todas las alternativas citadas anteriormente son compatibles con todas las formas de versionado, pero básicamente podemos plantear dos grandes variantes:

A- Almacenar las versiones en la misma tabla donde se almacena la versión “Actual” o
B- Almacenar en otra tabla las versiones anteriores.

Tabla Hacienda (Opción A):

IdSolicitante	**Fecha**	**Versión**
123456789A	12/05/2010	Actual
123456789A	20/05/2009	1.0

Tabla Hacienda (Opción B):

IdSolicitante	**Fecha**
123456789A	12/05/2010

Tabla Hacienda_V (Opción B):

IdSolicitante	Fecha	Versión
123456789A	20/05/2009	1.0

En la opción A, todas las versiones estarán juntas en la misma tabla. Si tenemos grandes volúmenes de documentos y se crean muchas versiones, las búsquedas serán mas lentas. Esto se debe, no solo a que la búsqueda se realiza sobre una tabla mas grande, sino a que se incluye internamente una condición adicional (…. y Versión="Actual"). A cambio, si se desea incluir en las búsquedas a versiones antiguas además de la actual, la búsqueda es más rápida.

En la opción B, la búsqueda es más eficiente, al ser sobre una tabla más pequeña, pero si se desea buscar sobre la versión vigente y SIMULTANEAMENTE sobre las antiguas, la búsqueda será menos eficiente al tener que acceder a dos tablas en lugar de una.

5- Conclusiones.

La forma en que se elija almacenar los metadatos de los documentos en un producto ECM no es "neutra", tiene bastantes implicaciones y es conveniente conocerla para no llevarse "sorpresas".

Confrontar los requerimientos y uso esperado del producto frente a la implementación del fabricante puede ayudar a la elección correcta del producto y no es un tema puramente tecnológico o de los responsables de SGBD.

No solo puede permitir disponer de un producto más eficiente y rápido sino que puede evitar encontrarse con la respuesta de "esto NO puede hacerse en este producto..", "hay que simularlo de la forma …. "

En cualquier caso, por supuesto, la elección debe hacerse siempre ponderando las necesidades y uso concreto frente a las especificaciones de los productos en todos los aspectos: funcionales, técnicos, de seguridad, escalabilidad, cumplimiento de normativas, etc.

Joaquín Hierro Torres

Anexo A-Glosario.

ECM: Enterprise Content Management. Productos de Gestión de Documentos Empresarial que permiten la parametrización y acceso desde múltiples sistemas para soportar necesidades de información o procesos de negocio. En esta categoría estarían productos como Documentum, Filenet, Content Manager, Oracle UCM y OpenText.

SGBD: Sistema Gestor de Base de Datos. Productos como Oracle, DB2 o MySQL que gestionan tablas con un modelo relacional.

Servidor de aplicaciones J2EE: Servidor basado en el estandar J2EE que permite la ejecución de aplicaciones java con tecnologías como servlets o jsp (13).

Bibliografía:

1) http://en.wikipedia.org/wiki/Enterprise_content_management)
2) http://en.wikipedia.org/wiki/Association_for_Information_and_Image_Management
3) Gartner. Publication Date: 23 September 2008 ID Number: G00160668 Magic Quadrant for Enterprise Content Management.
4) http://en.wikipedia.org/wiki/Documentum
5) http://spain.emc.com/products/category/subcategory/documentum-platform.htm

6) http://en.wikipedia.org/wiki/FileNet
7) http://publib.boulder.ibm.com/infocenter/p8docs/v4r5m1/index.jsp
8) http://www.oracle.com/technology/products/content-management/ucm/index.html
9) http://en.wikipedia.org/wiki/OpenText
10) http://www.opentext.com/
11) http://www-01.ibm.com/software/data/cm/cmgr/
12) http://publib.boulder.ibm.com/infocenter/cmx/v8r3m0/topic/com.ibm.sysadmin.hlp/gsi00011.htm
13) http://en.wikipedia.org/wiki/Application_server