SERVICIO DE ALMACENAMIENTO

PARA

SISTEMAS DE GESTION DE MENSAJES

Tesis Doctoral de: JAIME M. DELGADO MERCE

Director de Tesis: MANUEL MEDINA LLINAS

5.1.5.1.5. MODIFICACION DE MENSAJES

Los MIPs, por su concepto y definición, no pueden ser realmente (físicamente) modificados. Una vez ha sido enviado un MIP, y almacenado en la base de datos de mensajes, no tiene sentido modificar ninguna parte de su información, pues dejaría de ser el mensaje que se envió.

Sin embargo, y desde un punto de vista lógico, y que puede reflejarse en la base de datos de mensajes, sí que hay varias formas de modificar un MIP.

El mismo originador de un mensaje puede querer modificar un mensaje ya enviado y almacenado, y las recomendaciones del CCITT permiten hacerlo de alguna forma, pero nunca modificando realmente el mensaje primeramente enviado.

Además, un nuevo mensaje recibido puede cambiar información sobre otros mensajes, como veremos seguidamente, aunque no debemos olvidar que la base de datos a que nos referimos pertenece a quien recibe el mensaje, y ésta no puede modificarse sin su consentimiento.

Asimismo, un propietario de una base de datos puede querer modificar o anular información contenida en su almacén. Eso se puede hacer, por ejemplo, añadiendo información que la base de datos sea capaz de asociar con la eliminación o sustitución de un

valor previo, permitiéndose, de cualquier forma, el recuperar siempre el mensaje originalmente enviado.

Por supuesto, siempre es posible borrar un mensaje en su totalidad, e incluso algunos de sus atributos, pero nunca éstos parcialmente.

Otra cosa a tener en cuenta es el hecho de que en la base de datos podemos tener almacenados mensajes que aún no han sido enviados, por lo que en este caso sí tiene sentido modificarlos.

Los métodos permitidos por las recomendaciones X.400 para hacer modificaciones en un MIP son:

- Utilización de una referencia mediante el campo "obsoletes".

- Mensaje multicuerpo encapsulando mensajes anteriores.

Una nueva forma de modificar información de atributos de un MIP sería, como ya hemos apuntado, mediante el uso de campos extra que anulen información de otros campos. Un nombre para estos campos (propuesto en [PALM-86b]) es "anticampos" (o, también, "antienlaces").

Estos anticampos podrían ser una modificación de las X.400, o campos a añadir directamente en el almacén.

En el primer caso, serían campos del envoltorio y del encabezamiento de un MIP que cambian o eliminan información de otros campos. Ya existe un anticampo en X.400: "obsoletes" del encabezamiento P2.

En el segundo, serían nuevos atributos que se añadirían a los campos de almacenamiento, manteniendo una estructura similar a los campos del encabezamiento P2.

5.1.5.1.5.1. IMPLEMENTACION DE LOS ANTICAMPOS EN X.400

Como un usuario no tiene porque permitir que se le borre o modifique información de su base de datos desde fuera, los anticampos pueden ser vistos como añadidos a la base de datos. El cómo se traspasa al usuario la información contenida en los anticampos es un problema del interfaz local.

Posibles anticampos (extensión del encabezamiento P2):

- "AntiReplyToUsers" y "AntiPersonalReplyTo": para eliminar un nombre de O/D de los campos correspondientes.

- "AntiPrimaryRecipient", "AntiCopyRecipient" y "Anti-BlindCopyRecipient": cancelan los campos correspondientes.

- "AntiGroupRecipient": en caso de que se haya añadido previamente el campo "GroupRecipient" al encabezamiento P2.

Formalmente, en notación X.409, estos campos podrían ser:

- AntiPersonalReplyTo:

IMPLICIT SEQUENCE OF ORDescriptor OPTIONAL

- AntiReplyToUsers,
  AntiPrimaryRecipient,
  AntiCopyRecipient,
  AntiBlindCopyRecipient,
  AntiGroupRecipient:

IMPLICIT SEQUENCE OF Recipient OPTIONAL

El cómo se gestiona la información de estos anticampos en la base de datos (por ejemplo, cómo cancelar campos de destinatario ("recipient") en mensajes previamente distribuidos y en mensajes internos a otros) es un problema local de la base de datos, que no debería estar sujeto a estandarización.

Los anticampos se añadirían enviando un nuevo mensaje igual al original (sin cuerpo, si es posible) excepto en que tiene un nuevo encabezamiento externo en el que para cada campo que se quiere modificar hay dos campos, el anticampo con el valor viejo y el campo con el valor nuevo. Este encabezamiento externo puede tener valores de otros campos sin anticampo distintos a los del

encabezamiento interno (por ejemplo, originador, si la persona que hace la modificación no es la misma que envió el mensaje original (esto puede ocurrir, por ejemplo, cuando hay listas de por medio)).

5.1.5.1.6. IMPLEMENTACION DE LOS NUEVOS CAMPOS EN EL ALMACEN

Hemos visto una primera forma de implementar los nuevos campos o atributos consistente en extender el envoltorio y/o encabezamiento de un MIP con campos adicionales dentro de la misma estructura.

Una segunda forma es añadiendo un encabezamiento de almacenamiento adicional. Ahora, la estructura de un mensaje sería:

<mensaje> ::= <envoltorio> <encabezamiento_almacenamiento>
<encabezamiento P2> <partes_de_cuerpo>

Algunos campos que se necesitan en esa cabecera de almacenamiento adicional, y aún no mencionados, podrían ser:

- "ConjuntoCaracteres" ("CharacterSet"): Daría la variante nacional de caracteres que se usa en los cuerpos con texto.

- Todos los campos relacionados con documentos que veremos

en la siguiente sección.

5.1.5.2. EXTENSIONES PARA DOCUMENTOS

Aquí vamos a tratar los posibles campos adicionales que serían útiles en el almacén de mensajes en caso de que éstos contuviesen documentos en su cuerpo. Los campos o atributos que vamos a ver están basados en el estándar que para definición e intercambio de documentos está desarrollando la ISO, y que es conocido como ODA/ODIF ("Office Document Architecture/Office Document Interchange Format", o "Arquitectura de los Documentos de Oficina/Formato de Intercambio de los Documentos de Oficina" en castellano) ([ODA-E-101] y [ODA-I-8613]).

Un documento ODA está compuesto, básicamente, por tres elementos:

- Contenido: el texto y/o gráficos.

- Perfil de documento (información semántica): atributos asociados al contenido que, además, dan su estructura lógica.

- Aspecto ("layout" en inglés) del documento (información posicional): información que da la disposición en el papel de los distintos elementos del contenido del documento; es decir, da la estructura de aspecto.

Veremos, a medida que desarrollemos estos nuevos atributos, que muchos de ellos ya existen en las normas X.400 (por lo que en muchos casos ni siquiera los mencionamos) o ya los hemos añadido para otra función que no tiene que ver con documentos, pero que también sirven para éstos.

5.1.5.2.1. NUEVOS ATRIBUTOS PARA DOCUMENTOS

Agruparemos los nuevos atributos en función de su cometido.

5.1.5.2.1.1. ATRIBUTOS DE DESCRIPCION DEL DOCUMENTO

- Título.

- Tema.

- Tipo de documento: sus valores pueden ser "memorandum", "carta", "informe", etc.

- Resumen.

- Tabla de contenidos.

El número de versión puede considerarse como un atributo de

documento, o como un atributo de mensaje tal como hemos definido anteriormente en 5.1.4.1.

5.1.5.2.1.2. ATRIBUTOS DE FECHAS Y HORAS

Es bastante discutible cuántas fechas y horas distintas necesitamos para un documento. Además, algunas de ellas están más relacionadas con el sistema de transferencia de mensajes ("tiempo de entrega") y con la base de datos ("tiempo de almacenamiento") que con el documento en sí. Algunos atributos sólo aplicables a documentos pueden ser:

- Fecha y hora de inicio del documento.

- Fecha y hora de finalización de la versión actual del documento.

También podríamos tener un atributo con la historia de la revisión del documento, es decir, la de sus distintas versiones.

5.1.5.2.1.3. ATRIBUTOS DE ORIGINADORES

- Organización.

- Propietario.

- Autor.

Los cuatro campos anteriores pueden tener más de un valor.

5.1.5.2.1.4. ATRIBUTOS CON OTRA INFORMACION DE USUARIO

- Información de "copyright".

- Estatus del documento.

- Códigos específicos del usuario.

5.1.5.2.1.5. ATRIBUTOS DE REFERENCIAS EXTERNAS

- Referencias a otros documentos.

Los atributos de este tipo podrían confundirse con algunos campos del encabezamiento P2 que se refieren a relaciones de respuestas entre Mensajes InterPersonales (MIPs).

5.1.5.2.1.6. ATRIBUTOS DE ARCHIVO Y RECUPERACION

- Palabras clave.

- Referencia de documento: este es el identificador único dentro de ODA/ODIF. ODA/ODIF no dice cómo hacerlo globalmente único.

- Referencia de archivo local: indica dónde se puede encontrar el documento.

5.1.5.2.1.7. ATRIBUTOS DE CONTENIDO

- Longitud del documento: en caracteres, o dando el tamaño de página y combinándolo con el siguiente atributo.

- Número de páginas.

- Lengua en que está escrito el documento.

Algunos de estos atributos pueden estar relacionados con los que añadimos en 5.1.4.2., propios del contenido del mensaje en la base de datos.

5.1.5.2.1.8. ATRIBUTOS DE SEGURIDAD

Realmente, los atributos de seguridad de un documento no tienen porque ser añadidos a los del MIP que lo contiene. En

principio, los atributos de seguridad ya existentes en P2 ("authorizingUsers", "sensitivity", ...) y los que hemos añadido nosotros ("lista de acceso", ...) son suficientes para controlar la seguridad de acceso a un documento.

5.1.5.2.2. ALGUNOS ATRIBUTOS UTILES PARA DOCUMENTOS YA EXISTENTES EN X.400

- Fecha y hora de expiración: está en el encabezamiento P2.

- Documentos a los que sustituye: "obsoletes" es un campo similar en P2.

- Autorización: podría ser equivalente a "authorizingUsers" del encabezamiento de un MIP.

- Clasificación de seguridad: es similar a la indicación de sensibilidad ("sensitivity") de los MIPs.

5.1.6. EJEMPLO DE UNA ESTRUCTURA DE BASE DE DATOS DE MENSAJES: ALMACEN DE MENSAJES DEL PROTOCOLO P7

El almacén de mensajes (AM) definido para el protocolo P7 (véase 2.5.2. y 3.4.) es un buen ejemplo de base de datos de mensajes, aunque no reúne todas las características que nosotros

pretendemos.

Como ya vimos, el contenido de los mensajes que guarda dicho almacén no es de tipo restringido, aunque el protocolo está también definido para el caso en que estos mensajes sean mensajes interpersonales, como los definidos en la recomendación X.420 del CCITT [MHS-C-X420].

La información contenida en el AM comprende un conjunto de mensajes y un conjunto de "entradas" ("entries" en inglés) de mensaje, correspondientes a cada uno de los mensajes.

Cada mensaje, por supuesto, comprende un envoltorio y un contenido, no teniendo restricción el tipo de este último.

Una entrada de mensaje es una descripción de un mensaje. La crea el AM cuando se le entrega el mensaje y contiene información extraída, en general, del envoltorio y del contenido del mensaje. Para poder extraer esta información, el AM debe conocer la sintaxis y semántica del contenido. Cada AM será capaz de tratar mensajes cuyo contenido pertenezca a cero, uno o varios tipos determinados. Si el contenido de un mensaje es de un tipo que no conoce el almacén, entonces la entrada correspondiente no contiene información relacionada con el contenido, sólo con el envoltorio.

En el caso de almacenarse mensajes interpersonales del tipo

que conocemos, las entradas de mensaje contendrán información extraída del encabezamiento de los mensajes.

Una entrada de mensaje es un conjunto desordenado de atributos, cada uno de los cuales representa información sobre el mensaje. Un atributo consta de lo siguiente:

- Tipo de atributo: una clase de información que tiene significado para el usuario del AM. Por ejemplo, para mensajes P2, "prioridad", "tema", "sensibilidad", etc.

- Interpretación de atributo: una clase de información que tiene significado para el mismo AM (por ejemplo, "entero" o "cadena de caracteres"), permitiéndole comparar inteligentemente dos atributos (uno en la entrada de mensaje y otro proporcionado por el usuario en una interrogación, por ejemplo).

- Valor de atributo: es la información misma. Por ejemplo, "urgente" para prioridad, o "personal" para sensibilidad.

Algunos tipos de atributo pueden repetirse dentro de una misma entrada. Por ejemplo, un atributo "destinatarios primarios" (de P2) representa un único destinatario primario del mensaje; si el mensaje tiene tres destinatarios primarios, su entrada contendrá tres atributos de este tipo.

Aparte de los atributos correspondientes al envoltorio y al

contenido de un mensaje, el servicio de buzón que ofrece el protocolo P7 añade unos atributos de almacenamiento a las entradas de todos los mensajes del AM. Estos, ya explicados en 3.4.2.1., son (no repetimos aquí los detalles):

- Número de secuencia del AM.

- Clase de mensaje de buzón.

- Estatus del AM.

5.1.7. "SUPRAESTRUCTURAS"

Por "supraestructuras" entenderemos una agrupación lógica o física de mensajes con "algo" común que los relacione. Ese "algo" será generalmente el tener una serie de atributos con valores iguales o similares.

Lo que hemos llamado agrupación física equivaldría a tener la base de datos subdivida en subbases de datos que almacenen cada una mensajes que no tendrán nunca nada en común con los de otra subbase de datos. Cada subbase de datos, que podríamos llamar "cabinet" o "armario", haciendo una analogía con la nomenclatura más utilizada actualmente, se accedería independientemente, lo que equivaldría a tener una base de datos distinta para cada grupo independiente de mensajes. Como no tiene

demasiado interés para nosotros este tipo de supraestructura, ya no la volveremos a considerar. A partir de ahora, todas las supraestructuras que tratemos serán lógicas.

Las supraestructuras lógicas se pueden jerarquizar en diferentes niveles, conteniendo cada una a una o varias del nivel inferior. La formación de una supraestructura de un nivel se realiza a través de lo que podemos llamar "filtro" o "condición de búsqueda". Un filtro es una condición (expresión formada por una combinación, mediante operadores lógicos, de atributos y valores, asociados todos ellos mediante operadores de relación) que distingue mensajes de interés de otros mensajes, dentro de una supraestructura concreta o en toda la base de datos.

Una estructura básica de un almacén de mensajes con supraestructuras podría tener un solo nivel de jerarquía que consistiría en agrupar "lógicamente" los mensajes en lo que podemos llamar "carpetas", siguiendo también la terminología más usada.

Una carpeta (o "folder" en inglés) será una colección de mensajes. Tendrá asociado un "criterio de inserción", que será una expresión booleana que seleccionará los mensajes que pertenecen a esa carpeta (antes, en general, a esto lo hemos llamado "filtro").

Cada mensaje podrá pertenecer a cero, una o varias carpetas

en función de los criterios de inserción que cumpla. Esta estructura en carpetas será "lógica", es decir, independiente de la estructura física de la base de datos, por lo que el utilizar más o menos carpetas, o cambiarlas continuamente no afectará a la estructura de la información. La agrupación "física" de los mensajes de una carpeta se hará únicamente en el momento en que se solicite, o se quiera "ver", una carpeta.

Tal como se ha definido la estructura carpeta, es claro que es necesario conocer toda la información existente de atributos del mensaje (envoltorio, encabezamiento, atributos de almacenamiento, anticampos, atributos de documento, etc.) para poder crear las carpetas en función de los valores de algunos de esos atributos.

5.2. OPERACIONES A REALIZAR

5.2.1. OPERACIONES DEL USUARIO SOBRE LA BASE DE DATOS

Una vez definida la estructura de la base de datos de mensajes (BDM), ahora vamos a plantear las operaciones que necesitaremos sobre el almacén que hemos definido. Nos basaremos en qué operaciones va a querer realizar el usuario sobre los mensajes almacenados en la base de datos.

Una vez definidas las operaciones básicas a realizar,

trataremos en detalle cada una de ellas, aunque siempre desde el punto de vista de los mensajes interpersonales que va a almacenar.

Las operaciones básicas de que va a disponer el usuario sobre la base de datos de mensajes son:

- Almacenar.

- Recuperar.

- Borrar.

- Modificar.

5.2.2. ALMACENAR

Evidentemente, será necesaria una operación de este tipo para poder introducir información en la base de datos. La operación "almacenar" guardará información en la base de datos manteniendo la estructura que hemos definido en la sección anterior, es decir, los mensajes se almacenarán con su contenido y sus atributos de envoltorio, encabezamiento, y los que hemos añadido de almacenamiento, documentos, etc.

Esta operación, en la que no vamos a entrar en detalle

debido a su simplicidad, deberá permitir almacenar tanto mensajes completos como partes o atributos de los mensajes.

5.2.3. RECUPERAR

5.2.3.1. INFORMACION A RECUPERAR

La utilidad de una base de datos de mensajes es poder recuperar la información sobre mensajes que previamente hemos almacenado en ella de la forma más cómoda y eficiente posible. Un usuario potencial de este almacén podría querer realizar sobre la base de datos las siguientes operaciones generales de recuperación:

- Obtener información sobre qué mensajes hay en la BDM: equivale a recuperar ciertos campos o atributos de los mensajes.

- Obtener mensajes completos o partes de mensajes.

- Obtener información general sobre algunos mensajes: también equivale a recuperar ciertos atributos.

Cada una de las operaciones generales anteriores debería poder restringir el rango de mensajes que entren en ella, es decir, un usuario puede querer obtener alguna de las informaciones anteriores pero restringida a un grupo de mensajes

seleccionado. Esta selección se debe poder realizar expresando de alguna manera las condiciones que han de cumplir atributos concretos de los mensajes para que sean selecionados. Dicho de otra manera, se deberá poder añadir una condición de búsqueda, basada en expresiones booleanas con atributos y valores, para selecionar la información que se solicita.

Las solicitudes de recuperación de mensajes o información sobre ellos, deberán dar la siguiente información:

- Campos que se quiere obtener: la información que se solicita, que puede ser mensajes completos.

- Condiciones que han de cumplir los mensajes que nos han de proporcionar la información solicitada: estas condiciones pueden ser inexistentes.

- Cuántos mensajes queremos obtener (o de cuantos mensajes queremos obtener información): esta información de solicitud es opcional, y puede ser que no se cumpla en la respuesta si no hay suficientes mensajes que cumplan las limitaciones (condiciones) impuestas.

5.2.3.2. BUSQUEDAS

En el apartado anterior hemos expuesto las reglas generales

que deberían seguir las operaciones de recuperación. A continuación, vamos a centrarnos un poco más en la cuestión de las búsquedas de información en la base de datos de cara a recuperarla.

Como ya adelantamos, las condiciones de búsqueda son expresiones booleanas compuestas de:

- Comparaciones de atributos y valores (deberían ser válidos todos los operadores de comparación: =, <, >, <=, >= y <>).

- Operadores AND y OR (también debería admitirse el operador NOT) uniendo comparaciones y otras subexpresiones con comparaciones, operadores AND y OR (y NOT).

En las comparaciones mencionadas se debe permitir la presencia de todos los
atributos del envoltorio, del encabezamiento y los añadidos en la sección 5.1. El texto, o contenido propiamente dicho de los mensajes, es un problema aparte.

En la mayoría de los estudios y protocolos relacionados con almacenamiento, se propone que el criterio para recuperar un mensaje o información sobre él no se refiera a la información que sólo puede encontrarse explorando el contenido o texto completo. Es decir, se considera que no se puede incluir información interna al texto en las condiciones de búsqueda.

De todas formas, el planteamiento anterior es casi siempre de cara a la estandarización de protocolos de almacenamiento y recuperación, ya que esta característica complicaría mucho las implementaciones. Sin embargo, a nivel local no se debe descartar de entrada esta posibilidad.

5.2.4. BORRAR

La operación "borrar", al igual que "almacenar", es claramente necesaria para el manejo básico de la base de datos. Esta operación, simplemente, debe permitir eliminar un mensaje completo de la base de datos, del que ya no se podrá disponer nunca más.

Algunas variedades de esta operación podrían ser, por ejemplo:

- Borrar un mensaje sólo de forma "lógica", lo que permitiría recuperar un mensaje antes de ser suprimido definitivamente (posteriormente, con otra operación).

- Borrar sólo parte de un mensaje o algunos atributos de él: esta operación plantea bastantes problemas que veremos en la próxima sección cuando hablemos de "modificar", ya que borrar parcialmente un mensaje equivale a modificarlo.

Con la operación "borrar" deberían tenerse en cuenta los problemas de derechos de acceso, es decir, no todos los usuarios pueden tener derecho a borrar un mensaje. Este tema del control de acceso se trata detalladamente más adelante en la sección 6.5.

5.2.5. MODIFICAR

El concepto de mensaje interpersonal implica que éste no se debe modificar nunca una vez ha sido enviado, ya que si se hiciese, su información dejaría de ser real, debido a que no sería la que se ha enviado (este problema ya se planteó en 5.1.5.1.5.). Claramente, no tiene sentido modificar el atributo "originador" de un MIP, por ejemplo.

Ahora bien, algunos campos de los que hemos añadido, con información de almacenamiento o de documento, por ejemplo, sí que tiene sentido modificarlos (por ejemplo, "tamaño de los datos", "leído por", etc.). Incluso, ciertos atributos de los definidos en las X.400 también podrían querer modificarse por alguna razón (por ejemplo, "destinatarios de copia", "usuarios de respuesta", etc.). Para estos últimos podríamos utilizar lo que se definió como "anticampos" en 5.1.5.1.5. e implica una modificación "lógica" de la información de un mensaje manteniendo intacta la información real original. Algunos campos de almacenamiento, por supuesto, sí que pueden modificarse realmente (físicamente).

Por todo ello, podemos concluir que la operación "modificar" será equivalente, según lo que se quiera modificar, a:

- Una operación real de modificación, con una selección de los campos a modificar y una condición de búsqueda de los mensajes sobre los que hacer la modificación.

- Una operación de "almacenar" con una condición de busqueda.

Ambas condiciones de búsqueda serán de tipo similar a la de la operación "recuperar".

5.2.6. OPERACIONES DE MANTENIMIENTO Y GESTION

Aparte de las operaciones básicas mencionadas hasta ahora, también pueden ser útiles otras operaciones extra orientadas a la gestión y mantenimiento de la base de datos. Estas operaciones deberían ser realizadas por usuarios especiales privilegiados.

Dichas operaciones las podemos clasificar en dos grandes grupos:

- Operaciones que modifiquen derechos de acceso a la información de la base de datos.

- Operaciones que modifiquen campos muy concretos de la información de un mensaje, como, por ejemplo, "propietario" (de un documento), etc.

5.2.7. EJEMPLOS DE OPERACIONES EXISTENTES

5.2.7.1. P7

En 2.5.2. y 3.4. hemos tratado con bastante detalle el protocolo P7. Como allí establecimos, nos hemos basado en el quinto borrador de trabajo de dicho protocolo (siempre que no se haya especificado lo contrario). Ahora, al reconsiderar las operaciones del P7 sobre el Almacén de Mensajes (AM) nos basaremos en el sexto borrador [MHS-E-109], principalmente por dos razones. En primer lugar, porque esta última versión corrige algunas inconsistencias de su antecesora, y, en segundo lugar, para extender y concretar en alguna medida la información dada ya en los apartados mencionados.

Vamos a desglosar las operaciones sobre el AM del P7 en dos grupos en función del subconjunto del protocolo al que pertenece.

A) Para el subconjunto general, mensajes sin restricción en el contenido, son:

- Solicitar estado del AM: devuelve un pequeño resumen de los mensajes del AM, haciendo especial hincapié en los mensajes nuevos (aún no leídos).

- Enumerar mensajes del AM: obtiene las entradas de mensajes seleccionados. Esta selección se puede hacer, en primer lugar, con un rango de mensajes delimitado por el número de secuencia y/o el tiempo de entrega. Además, se puede hacer una selección basada en valores de los atributos almacenados en las entradas del AM. Asimismo, es posible seleccionar qué atributos de las entradas de los mensajes se obtienen.

- Traer mensaje del AM: trae un mensaje del AM. El mensaje a traer se puede especificar por su número de secuencia. Es también posible, sin embargo, especificar un criterio de selección de los mensajes que se desea traer, en cuyo caso sólo se trae el mensaje más antiguo de los que cumplen el criterio de selección y, además, una lista de los demás mensajes que satisfacen el criterio.

- Borrar mensajes del AM: borra mensajes del AM especificados por su número de secuencia, o un rango de ellos.

B) Para el subconjunto específico de Mensajes InterPersonales (MIPs), las operaciones son:

- Solicitar estado del AM para MIPs: extiende algo la

operación similar del otro subconjunto, permitiendo que el análisis de los mensajes del AM se haga en función de algunos atributos de MIP.

- Traer mensaje del AM para MIPs: recupera MIPs completos o partes seleccionadas de ellos. Se ha de especificar el número de secuencia del mensaje junto con las partes componentes que se solicitan.

- Mostrar "sinopsis" de un MIP: muestra la estructura interna de un mensaje interpersonal del AM. A partir de un número de secuencia devuelve una lista de las partes de cuerpo del mensaje correspondiente, incluyendo el anidamiento de las partes de cuerpo.

5.2.7.2. APLICACION DE ARCHIVO Y RECUPERACION DE DOCUMENTOS

En 3.2.1. dimos una introducción de las propuestas existentes para una Aplicación de Archivo y Recuperación de Documentos (AARD). Ahora vamos a tratar un punto concreto de dicha aplicación, que omitimos allí expresamente. Nos referimos al tema de las operaciones (o servicios) que ofrece la aplicación.

Estos servicios se agrupan en función de los tres subconjuntos del protocolo que se habían definido. Estos era

"Kernel", "Random Access" y "Find Set", dando los términos ingleses.

Pero no todos los servicios definidos en el protocolo nos interesan aquí y ahora, sino sólo los relacionados con la manipulación de ficheros. Estos últimos serán los únicos que trataremos. Los enumeraremos y explicaremos parcialmente siguiendo la agrupación antes mencionada.

5.2.7.2.1. SERVICIOS DE "KERNEL"

1) Servicios de gestión de acceso:

Estos servicios no nos interesan en principio. Sin embargo, mencionaremos brevemente un aspecto relacionado con los permisos de acceso a los documentos o ficheros. Estos están basados en la forma de acceso. Para ello, define cinco formas de hacerlo, que son: leer, escribir, propietario, añadir y eliminar.

2) Servicios de almacenamiento y recuperación:

- Almacenar un nuevo fichero.

- Recuperar un fichero.

- Crear un fichero vacío.

- Reemplazar el contenido de un fichero existente.

- Serializar ficheros.

- Deserializar ficheros.

3) Servicios de gestión de ficheros:

- Copiar ficheros y directorios de ficheros.

- Mover ficheros y directorios de ficheros.

4) Servicios de localización de ficheros:

- Enumerar ficheros.

- Enumerar ficheros seleccionados.

- Localizar ficheros.

5) Servicios de acceso y modificación de atributos:

- Acceder a atributo.

- Modificar atributo.

- Unificar controles de acceso del usuario.

5.2.7.2.2. SERVICIOS DE ACCESO ALEATORIO

Este subconjunto permite el acceso a porciones aleatorias de ficheros. Asimismo, es también posible leer y modificar ficheros sin necesidad de que el usuario recupere el fichero entero.

Las operaciones definidas son:

- Recuperar contenido de fichero aleatorio (rango de octetos).

- Modificar contenido de fichero aleatorio.

5.2.7.2.3. SERVICIOS DE "FIND SET"

Un conjunto de búsqueda ("find set") es una agrupación lógica de ficheros en la que cada uno de los ficheros satisface un conjunto particular de criterios especificados por el usuario. Una vez especificado un conjunto de búsqueda, el usuario puede tratar los ficheros individual o colectivamente.

Los criterios de selección son idénticos a los de la operación "enumerar ficheros seleccionados", mencionada en

5.2.7.2.1. Dichos criterios los comentaremos en el siguiente apartado.

Las operaciones definidas en este subconjunto son:

- Crear un conjunto de búsqueda.

- Reducir un conjunto de búsqueda.

- Enumerar elementos de un conjunto de búsqueda.

- Borrar un conjunto de búsqueda.

5.2.7.2.4. CRITERIO DE BUSQUEDA

El criterio de búsqueda está basado en el concepto de ámbito ("scope" en inglés), que es un argumento que describe cómo ha de proceder el proceso de búsqueda.

Los ámbitos permiten al usuario especificar:

- Dirección de enumeración (hacia atrás o hacia adelante).

- Restriciones sobre valores de atributos.

- Número máximo de ficheros que pueden satisfacer el

conjunto de restricciones.

- Profundidad en la jerarquía del armario ("cabinet") hasta la que ha de proceder la enumeración o la búsqueda.

Las operaciones de enumeración o búsqueda se pueden aplicar a ficheros estructurados a cualquier nivel de una jerarquía de ficheros de un armario.

Seguidamente, como ejemplo, damos la definición formal, usando notación X.409 [MHS-C-X409], de los ámbitos:

```
ScopeSequence ::= SET OF Scope

Scope ::= CHOICE {
        Count:  límite  para el número de documentos o carpetas
                encontrados,
        Direction: dirección de búsqueda,
        Filter: criterio de búsqueda real,
        Depth: profundidad de la subcarpeta a buscar}

Filter ::= CHOICE {
        less,  lessOrEqual,  equal,  notEqual,  greaterOrEqual,
        greater  --> Relation,
        and, or  --> LogicalBin,
        not  --> LogicalMon,
        noneOrAll  --> LogicalConstant,
```

```
        matches  --> Pattern)

    Relation  ::= SET {
        Información de atributo,
        Interpretación)

    LogicalBin ::= SEQUENCE of Filter

    LogicalMon  ::= Filter

    Pattern  ::= Attribute
```

La información de atributo ("attribute") es "tipo de atributo" y "valor(es) de atributo(s)".

5.3. PROTOCOLOS DE ACCESO

En los dos apartados anteriores de este capítulo hemos tratado la estructura de la información y las operaciones que debería tener una base de datos de mensajes. Esta base de datos era para servir como almacén de mensajes de un servicio de almacenamiento que debía cumplir ciertas características (véase 4.2.2.) y responder a alguna de las arquitecturas que habíamos propuesto (véanse secciones 4.4. a 4.7.).

Un aspecto que nos queda por tratar, antes de pasar a temas relacionados con la implementación de la base de datos de mensajes, es el de los protocolos de acceso a los almacenes o bases de datos de nuestro servicio de almacenamiento.

Sin embargo, éste va a ser un tema que vamos a dejar un poco de lado, ya que el objetivo central del presente trabajo es la base de datos en sí, junto con la arquitectura global del sistema.

5.3.1. REVISION DE LOS MODELOS PROPUESTOS

Cada uno de los 4 modelos propuestos necesitaba un tipo de protocolo particular de acceso a la base de datos, con problemáticas distintas en cada caso, y, muchas veces, totalmente diferentes unas de otras.

Analizaremos brevemente a continuación cada uno de los protocolos necesarios.

5.3.1.1. CODIFICACION EN P2

Este caso corresponde a las propuestas 1 y, parcialmente, a la 4.

Vimos que una forma sencilla de hacer llegar a la base de datos las operaciones que queremos que realice es codificándolas dentro de un protocolo ya existente. Ese mismo protocolo servirá después para devolver los resultados, si los hubiese.

En el modelo 1 ocurre claramente esta situación: las operaciones se codifican como mensajes P2 (exactamente P2+, aunque, por lo que respecta al punto que estamos tratando ahora, es indiferente uno u otro). Y la información de la base de datos se transmite también dentro de mensajes P2.

La solución número 4, que era una mezcla de la 1 y la 3, también utiliza esta técnica de codificación, aunque, en esa solución, el protocolo en el que se codificaba lo habíamos llamado P2++.

Por lo tanto, en este caso no necesitamos realmente un nuevo protocolo de acceso a la base de datos, sino una pequeña modificación de un protocolo ya existente (el P2 del CCITT [MHS-C-X420]) (véase 3.6.1.) para adaptarse al hecho de que almacenes de mensajes puedan funcionar como agentes de usuario interpersonales normales.

Aparte de las modificaciones del protocolo mencionadas, deberemos dar un formato a las operaciones que pueden realizar las propuestas 1 y 4 para que la base de datos pueda gestionarlas. Sin embargo, este no es realmente un problema de

protocolos de comunicaciones, sino de lenguajes de bases de datos, quedando fuera de nuestros objetivos.

5.3.1.2. USO DEL P7

Esto corresponde a la propuesta a la que hemos dado el número 2, en la que extendíamos el protocolo P7.

Aquí, está muy claro el tema del protocolo de acceso a la base de datos, que no es más que la parte del P7 (la actual y las mejoras que proponemos) que accede al almacén de mensajes. Este almacén es la base de datos de mensajes del P7.

5.3.1.3. NUEVOS PROTOCOLOS

Este último caso corresponde al modelo 3 y, parcialmente, al 4.

Ahí habíamos visto la necesidad de definir dos nuevos protocolos, que eran:

- Protocolo de Acceso al Almacenamiento (PAA): protocolo por medio del cual accedía un Agente de Usuario de Almacenamiento (AUA) a un Agente de Sistema de Almacenamiento (ASA), que era donde realmente se almacenaban los mensajes.

- Protocolo de Sistema de Almacenamiento (PSA): protocolo de comunicación entre ASAs.

5.3.1.3.1. PROTOCOLO DE ACCESO AL ALMACENAMIENTO

Existe una gran libertad a la hora de diseñar este protocolo, que convendría fuese lo más directo e interactivo posible.

Ya existe alguna propuesta de protocolo de este tipo, como puede ser el Protocolo de Acceso a Bases de Datos Remotas (PABDR) de ECMA [DB-E-30] e ISO [DB-I-1243]. Este es, simplemente, un protocolo de acceso a bases de datos relacionales remotas. En nuestro caso, necesitamos también un protocolo de acceso a bases de datos remotas, pues el ASA no es más que una base de datos de mensajes que está remota al AUA. Sin embargo, no estamos obligados a que la base de datos de mensajes del ASA sea relacional y, además, que cumpla el estándar SQL (lo que se exige por ahora en el PABDR).

Ya hablaremos ampliamente en el próximo capítulo del lenguaje estándar de interrogación de bases de datos relacionales llamado SQL (véase 6.2.).

A pesar de estas restricciones del PABDR, veremos más

adelante que el almacén de mensajes deberá ser una base de datos relacional que, seguramente, cumplirá el estándar SQL o, al menos, un subconjunto de éste. Por tanto, no es disparatado que nuestro PAA sea el PABDR. Sin embargo, será necesario, al menos, un programa de aplicación que transforme nuestras operaciones sobre la base de datos de mensajes en operaciones del protocolo estándar.

De todas formas, el tema de este protocolo queda abierto, siendo también función, posiblemente, del tipo de implementación.

Antes de terminar el apartado, haremos un breve repaso de lo que ha de manejar el protocolo. Básicamente, el PAA debe llevar al ASA solicitudes de operaciones emitidas por el AUA, junto con posibles mensajes a almacenar. A su vez, el ASA devolverá al AUA las respuestas a sus solicitudes, las cuales serán mensajes completos o parciales e información sobre éstos.

De cualquier forma, los protocolos de acceso a implementar deberían seguir la normas para diseño de protocolos de acceso a aplicaciones para la oficina distribuida del modelo correspondiente de ECMA e ISO (véase 2.4.).

5.3.1.3.2. PROTOCOLO DE SISTEMA DE ALMACENAMIENTO

Este protocolo ha de ser apreciablemente diferente al

anterior. Ha de cumplir funciones similares al protocolo P1 de los sistemas de gestión de mensajes [MHS-C-X411], o al protocolo de sistema de directorio [DS-C-Xdsx]]. En las más recientes versiones del modelo para la oficina distribuida de ECMA e ISO se utiliza también ahora el concepto de protocolo de sistema (al igual que el más antiguo de protocolo de acceso).

Lógicamente, será un protocolo de almacenamiento y reenvío que se encargará de la comunicación entre ASAs. Esta comunicación consistirá en el pase de solicitudes de mensajes o información sobre ellos (en general, interrogaciones sobre mensajes).

Cuando un ASA reciba una solicitud de una AUA y no pueda atenderla porque no disponga de la información suficiente, deberá traspasar la solicitud a otro ASA, y así sucesivamente, hasta que se encuentre la información o se decida que no se puede encontrar. Finalmente, la información solicitada, si se encontró, se entregará al AUA solicitante.

Este tipo de protocolo puede ser muy complejo si se hace general. Por el momento no hay ninguna propuesta al respecto. Una de las ideas que se manejan es que este protocolo se debería basar en otros ya existentes, como el P1 del CCITT. Sin embargo, nosotros consideramos en principio que el protocolo se define sobre un Servicio de Transferencia Fiable (STF), similar al definido por el CCITT en la recomendación X.410 [MHS-C-X410], o sobre el Servicio de Operaciones Remotas (SOR), definido en [MHS-

I-501] y [MHS-I-502]. (El protocolo de acceso también se puede definir sobre el SOR).

En la propuesta 4, existen dos versiones de este protocolo PSA. Una igual que la mencionada hasta ahora, entre dos ASAs normales, que hemos llamado PSA/STF, es decir, protocolo de sistema de almacenamiento sobre un servico de transferencia fiable. La segunda versión es para comunicar dos entidades de doble funcionalidad ASA/AU-IP (ASA y agente de usuario interpersonal) (véase 4.7.), en la que el protocolo PSA ha de ser sobre P1 (lo llamamos PSA/P1) para mantener la compatibilidad con el resto del sistema de gestión de mensajes, ya que la entidad ASA/AU-IP también podía ser accedida mediante un protocolo P2 o P2 extendido, que va sobre P1.

6. REPRESENTACION DE MENSAJES EN BASES DE DATOS

6.1. BASE DE DATOS DE MENSAJES RELACIONAL

En el capítulo precedente se ha planteado la necesidad de disponer de una base de datos para implementar el almacén de mensajes que queremos para resolver el problema del almacenamiento de mensajes interpersonales en un entorno X.400.

Allí hemos visto la estructura que debería tener la información a almacenar en dicha base de datos, así como las operaciones que se deberían poder realizar sobre ella. Pero todo ello sin concretar para nada la implementación.

Al contrario, este capítulo sí que va a pretender mostrar una posible implementación de una base de datos de mensajes.

Antes que nada, deberíamos elegir qué modelo de datos preferimos para implementar nuestra base de datos. La elección es el modelo relacional porque es el más claro conceptualmente y dispone de un lenguaje estándar (o en vías de estandarización) para trabajar con él. Debemos aclarar en este punto, que lo que pretendemos no es una implementación eficiente, sino un mapeo

entre las necesidades generales de una base de datos de mensajes como la definida en el capítulo 5 y las estructuras y operaciones de una base de datos, y para ello, lo mejor es elegir una base de datos relacional, aparte de que las tendencias actuales van dirigidas principalmente hacia las bases de datos relacionales, sobre todo en lo que se refiere a ordenadores personales que es uno de nuestros objetivos dentro del sistema de almacenamiento que estamos desarrollando.

Una vez tomada la decisión de implementar el almacén de mensajes en una base de datos relacional, esta claro que:

- Debemos representar nuestra estructura de memsajes a almacenar en tablas (o relaciones).

- El lenguaje a utilizar para expresar las operaciones es el lenguaje para bases de datos relacionales estandar de la ISO (originalmente de IBM) llamado SQL ("Structured Query Language", o "Lenguaje de Interrogación Estructurado" en castellano).

El presente capítulo continúa con una explicación de los puntos más importantes del SQL orientados al uso que vamos a hacer de él.

Después se desarrolla un posible mapeo de la estructura de un mensaje interpersonal a una estructura relacional. Para simplificar, utilizaremos una estructura puramente X.400, aunque

después la extenderemos a nuestra estructura completa del capítulo anterior.

El siguiente apartado discutirá cómo acceder a la base de datos definida, es decir, cómo se implementarán las operaciones.

La sección 5 planteará el problema de los derechos de acceso que ya se ha mencionado alguna vez en capítulos previos.

La última sección tratará sobre los mensajes complejos y cómo realizar búsquedas en ellos.

6.2. LENGUAJE ESTANDAR DE BASES DE DATOS RELACIONALES SQL

El estándar SQL ha sido desarrollado por el grupo X3H2 de ANSI [DB-A-127] y enviado al subcomité de ISO ISO/TC97/SC21 [DB-I-849] para su estudio y aprobación. La última versión es [DB-I-117]. Es el actual estándar de trabajo de ISO sobre lenguajes de bases de datos relacionales. La propuesta original de ANSI está basada en el lenguaje del mismo nombre desarrollado y comercializado por IBM [DATE-81].

El SQL especifica la sintaxis y semántica de dos lenguajes de bases de datos:

- Definición de esquema: este lenguaje es para declarar las

estructuras de la base de datos y las restricciones de integridad. Es un lenguaje de definición de datos [DATE-81].

- Módulo y manipulación de datos (lenguaje módulo): es el lenguaje utilizado para declarar las sentencias ejecutables de una aplicación de base de datos específica. Es un lenguaje de manipulación de datos [DATE-81].

Este estándar define la funcionalidad de los interfaces a un Sistema de Gestión de Bases de Datos (SGBD) para la definición y el acceso (incluyendo interrogación y alteración) de bases de datos relacionales.

El lenguaje módulo puede ser sustituido por sentencias SQL incluidas en un lenguaje de programación estándar (Pascal, FORTRAN, COBOL o PL/I) o por un programa de utilidad que invoque directamente las sentencias del lenguaje de manipulación de datos.

Este estándar especifica dos niveles: el nivel 2 es el lenguaje de bases de datos SQL completo, y el nivel 1 es un subconjunto del nivel 2.

6.2.1. CONCEPTOS BASICOS DEL SQL UTILES PARA ALMACENAMIENTO Y RECUPERACION DE MENSAJES

6.2.1.1. TIPOS DE DATOS

Los tipos de datos definidos en el estándar SQL son los siguientes:

- Cadena de caracteres.

- Número exacto: Numérico y Decimal (ambos con definiciones de precisión y escala), Entero y "Entero Pequeño" ("Smallint").

- Número aproximado (mantisa y exponente): Flotante (con definición de precisión), Real y Doble Precisión.

6.2.1.2. COLUMNAS

Una columna está compuesta de valores. Una valor de una columna es la unidad de datos más pequeña que se puede seleccionar o actualizar en una tabla.

Una columna está definida por una descripción y una posición ordinal dentro de una tabla.

6.2.1.3. TABLAS

Una tabla está compuesta de filas. La fila es la unidad de datos más pequeña que puede ser insertada o borrada en una tabla.

El grado de una tabla es el número de columnas de la tabla, y la cardinalidad es la cardinalidad de cada una de sus columnas, es decir, el número de filas de la tabla.

Una tabla tiene una descripción basada en la descripción de sus columnas.

Clases de tablas:

- Tabla base: tabla con nombre con representación de almacenamiento persistente y descripción persistente. Se define usando el lenguaje de definición de esquema.

Tabla derivada: tabla que existe sólo durante la ejecución de un programa de aplicación. Se deriva de una o más tablas mediante la ejecución de lo que se llama "especificación de interrogación" ("query specification") (véase 6.2.3.1).

- Tabla "vista" ("view"): tabla derivada con nombre con descripción persistente. Se define por medio del lenguaje de definición de esquema.

Una tabla puede ser actualizable o de sólo-lectura. Esta característica se le da también usando el lenguaje de definición de esquema.

Es también posible definir "tablas agrupadas" que son conjuntos de multiconjuntos de filas en las que todos los valores de una columna o columnas particulares (columna(s) de agrupación) son iguales. Realmente, estos multiconjuntos son tablas. Por lo tanto, una tabla agrupada es un conjunto de tablas.

6.2.1.4. RESTRICCIONES DE INTEGRIDAD

Definen los estados válidos de la base de datos restringiendo los valores en las tablas base, y se comprueban efectivamente al ejecutarse cada sentencia SQL.

Las restricciones están relacionadas con unicidad y valores nulos.

6.2.1.5. ESQUEMAS

Un esquema, definido utilizando el lenguaje de definición de esquema, es un objeto persistente que consta de:

- Una autorización de esquema.

- Cero o más definiciones de tabla (tablas base): incluyen las definiciones de columna, opcionalmente con sus restricciones correspondientes ("Unica" ("Unique") o "No Nula" ("Not Null")).

- Cero o más definiciones de vista: una tabla vista se crea a partir de una especificación de interrogación (véase 6.2.3.1).

- Cero o más definiciones de privilegio: dan autorización a cierto número de usuarios para realizar operaciones (seleccionar, insertar, borrar y/o actualizar) sobre tablas del esquema.

Las tablas, vistas y privilegios especificados son propiedad del identificador de autorización de la autorización de esquema.

La colección de todos los datos definidos por los esquemas en un entorno forman la base de datos.

6.2.2. LENGUAJE MODULO

Un módulo SQL incluye:

- Una cláusula de nombre de módulo.

- Una cláusula de autorización de módulo.

- Cero o más declaraciones de cursor.

- Uno o más procedimientos.

Un módulo estará asociado con un programa de aplicación durante su ejecución. Este programa de aplicación, que puede estar asociado con un módulo como máximo, usa el módulo como interfaz para ejecutar operaciones sobre la base de datos. Esas operaciones se definen en el módulo.

Un cursor es una tabla derivada con una ordenación y una posición de cursor (que indica una fila en esa tabla) asociadas. La posición puede ser antes de una fila o después de una fila.

Un procedimiento tiene una secuencia de declaraciones de parámetros y una única sentencia que especifica una acción sobre la base de datos, que puede relacionar los parámetros con el cursor o con la base de datos en general. Así, una sentencia SQL es o bien una operación de base de datos o bien una operación de cursor.

La comunicación entre un programa de aplicación y un módulo se realiza por medio de sentencias de llamada en un programa a procedimientos SQL, utilizando los parámetros necesarios para el intercambio de valores y resultados (con una correspondencia correcta). La sentencia SQL del procedimiento llamado empieza su ejecución en ese momento.

6.2.2.1. CURSORES

Un cursor puede estar en dos estados: abierto y cerrado. El estado inicial es el cerrado. Existe una sentencia especial para abrir un cursor.

Un cursor en el estado abierto designa, como dijimos antes, una tabla, una ordenación de las filas de la tabla y una posición relativa a ese orden.

Una declaración de cursor necesita un nombre de cursor y una especificación de cursor, que contiene una especificación de interrogación (véase 6.2.3.1) combinada con operaciones de unión sobre tablas de la base de datos, y una cláusula opcional de ordenación.

Los cursores se utilizan para definir comunicación específica por filas, en vez de la comunicación específica por tablas de las operaciones que actúan directamente sobre la base de datos.

6.2.2.2. SENTENCIAS SQL

Las sentencias SQL existentes se pueden clasificar en tres grupos, como sigue:

A) Operaciones sobre tablas:

- Seleccionar ("Select"): asigna a los parámetros del procedimiento valores de alguna fila de una tabla derivada usando una especificación de interrogación (véase 6.2.3.1).

- Insertar ("Insert"): inserta nuevas filas en una tabla.

- Buscar-actualizar ("Update-search"): modifica todas las filas especificadas mediante una condición de búsqueda (véase 6.2.3.3).

- Buscar-borrar ("Delete-search"): borra todas las filas especificadas mediante una condición de búsqueda.

B) Operaciones sobre cursores:

- Posicionar-actualizar ("Update-position"): modifica la fila actual de la tabla del cursor. La tabla no debe ser de sólo-lectura.

- Posicionar-borrar ("Delete-position"): borra la fila actual de la tabla del cursor (no debe ser una tabla de sólo-lectura).

- Declaración de cursor ("Cursor declaration"): se explicó en 6.2.2.1.

- Traer ("Fetch"): asigna valores de la fila actual de la tabla de cursor a los parámetros del procedimiento y posiciona el cursor en la siguiente fila.

- Abrir cursor ("Open cursor"): abre un cursor cerrado y lo posiciona antes de la primera fila de la tabla (por lo que la primera fila se convierte en la fila actual).

- Cerrar cursor ("Close cursor"): cierra un cursor abierto.

C) Operaciones sobre transacciones:

"Commit": finaliza la transacción actual confirmando cualquier cambio hecho a la base de datos. Se inicia entonces otra transacción. Entendemos por transacción una secuencia de operaciones de base de datos que es atómica respecto a restablecimiento y concurrencia.

"Rollback": finaliza la transacción actual cancelando cualquier cambio hecho a la base de datos, e inicia otra transacción.

6.2.3. FORMULACION DE INTERROGACIONES

Pensando en una interrogación como en una pregunta a la base de datos, para obtener valores almacenados, que tenga alguna

capacidad para seleccionar esa información, la formulación de interrogaciones en el estándar SQL está basada en los tres elementos siguientes:

- "Especificación de interrogación" ("query specification") (se explica en 6.2.3.1), que se usa implícitamente en la operación Seleccionar, y explícitamente en las sentencias de Declaración de Cursor e Insertar (las tres del lenguaje módulo (véase 6.2.2.2.)), y en la Definición de Vista del lenguaje de definición de esquema (véase 6.2.1).

- "Condición de búsqueda" (se explica en 6.2.3.3), usada en las sentencias Buscar-actualizar y Buscar-borrar (véase 6.2.2.2), y en las cláusulas Teniendo ("Having") y Donde ("Where") de las expresiones de tabla definidas después (véase 6.2.3.1).

- "Subinterrogación" (se explica en 6.2.3.1), que se usa explícitamente en algunos predicados de condiciones de búsqueda (véase 6.2.3.3).

6.2.3.1. ESPECIFICACION DE INTERROGACION Y SUBINTERROGACION

Agrupamos las explicaciones de Especificación de Interrogación (EI) y subinterrogación ya que una EI está compuesta de subinterrogaciones.

En una notación tipo BNF, una EI puede expresarse como:

<especificación de interrogación> ::= SELECT [ALL | DISTINCT]
<lista de selección>
<expresión tabla>

<lista de selección> ::=
["*" | (<expresión valor> {<expresión valor>})]

(siendo el asterisco "*" el carácter de "comodín"),

y una subinterrogación como:

<subinterrogación> ::= SELECT [ALL | DISTINCT]
<especificación de resultado>
<expresión tabla>

<especificación de resultado> ::= ["*" | <expresión valor>]

(siendo el asterisco "*" el carácter de "comodín").

Por lo tanto, es posible decir que una subinterrogación es un ejemplo particular de una EI, y que una EI está compuesta de subinterrogaciones.

Lo que quieren decir las dos definiciones anteriores es que

una EI o una subinterrogación (interrogación en general) es el resultado de aplicar una "expresión valor" (o algunas expresiones valor) a una "expresión tabla". Una expresión valor será, en general, una especificación de columna o una especificación de función conjunto. Lo que el estándar SQL llama especificación de función conjunto no es más que un valor obtenido al aplicar a un argumento una de las funciones que permite el SQL.

Una expresión tabla es un poco más complicada, siendo realmente el centro del problema. Es el resultado de la aplicación secuencial de las siguientes cláusulas a una o más tablas base (con nombre) del esquema:

- Cláusula "Desde" ("From"): es la selección de las tablas base.

- Cláusula "Donde" ("Where"): es el resultado de aplicar una condición de búsqueda (discutida en 6.2.3.3) a la tabla obtenida antes (con la cláusula "Donde").

- Cláusula "Agrupar por" ("Group by"): obtiene una partición de la tabla previa en un conjunto de grupos (véase "tablas agrupadas" en 6.2.1.3).

- Cláusula "Teniendo" ("Having"): elimina los grupos (obtenidos con la cláusula "Agrupar por") que no cumplen una condición de búsqueda expresada en la cláusula.

Sólo la primera cláusula es obligatoria. Bastante a menudo, sólo se utilizan las dos primeras.

6.2.3.2. LA SENTENCIA SQL "SELECT"

Esta sentencia es como una EI, pero:

- La cardinalidad de la tabla obtenida no debe ser mayor que uno. Es decir, el resultado es como máximo una fila.

- Existe una cláusula para asignar la fila obtenida a los parámetros correspondientes.

6.2.3.3. CONDICION DE BUSQUEDA

Una condición de búsqueda se expresa mediante una combinación de operadores lógicos OR, AND y NOT, aplicados sobre predicados (funciones booleanas), y otras condiciones de búsqueda.

Una condición de búsqueda no devuelve realmente un valor booleano. Los valores devueltos por una condición de búsqueda incluyen "verdadero" ("true"), "falso" ("false") y "desconocido" ("unknown"). Este último valor se puede obtener de algunos

predicados. Los resultados producidos por los distintos operadores lógicos sobre estos tres valores están especificados en el estándar [DB-A-127].

Cuando se aplica sobre una tabla, una condición de búsqueda crea una tabla derivada que cumple la condición de búsqueda (su valor es "verdadero").

El estándar SQL define siete predicados:

- Comparación: los operadores normales de comparación aplicados entre expresiones valor y subinterrogaciones.

- Entre ("Between"): se refiere a un rango de expresiones valor.

- En ("In"): mira si una expresión valor pertenece a una subinterrogación o a un conjunto de valores.

- Como ("Like"): hace una comparación, para ver si son iguales, entre cadenas de caracteres (una especificación de columna y una especificación de valor).

- Nulo ("Null"): comprueba si una especificación de columna tiene un valor nulo.

- Existe ("Exist"): comprueba si un conjunto está vacío. Se

aplica a una subinterrogación.

- Cuantificado: es una comparación cuantificada (ALL ("todo") o SOME ("algo")) entre una expresión valor y una subinterrogación.

6.3. REPRESENTACION EN SQL

La base de datos relacional que pretendemos diseñar ha de almacenar la estructura completa definida en el capítulo anterior. Sin embargo, para facilitar el diseño y su explicación nos restringiremos inicialmente a Mensajes InterPersonales (MIPs) tal como están definidos en las recomendaciones X.400. Más adelante extenderemos la solución a la estructura completa.

En lo que sigue, desarrollaremos paso a paso la estructura en tablas (relaciones) que necesitamos para representar un MIP.

6.3.1. ESTRUCTURA A REPRESENTAR DE UN MENSAJE INTERPERSONAL

Un MIP que siga las recomendaciones X.400 del CCITT [MHS-C-X4xx] (véase también [BABA-85]), como ya hemos descrito, se puede describir inicialmente, utilizando una notación tipo BNF, como:

```
<mensaje>                ::= envoltorio  <contenido>
<contenido>              ::= encabezamiento
                             <parte_de_cuerpo> {<parte_de_cuerpo>}
<parte_de_cuerpo>        ::= texto  | <mensaje-IP_reenviado>
<mensaje-IP_reenviado>   ::= informaciónEntregaPrevia
                             <contenido>
```

En esta descripción, los elementos terminales "envoltorio", "encabezamiento", "texto" e "informaciónEntregaPrevia" no representan exactamente campos simples, sino que, a su vez, están estructurados en otros campos.

El SQL define un interfaz estándar para acceder a una base de datos relacional, y esta es la razón por la que lo hemos elegido como lenguaje.

Para utilizar la formulación de interrogaciones del SQL sobre un almacén de MIPs, será necesario expresar una base de datos de mensajes en función de una base de datos relacional. Esto se puede conseguir definiendo los mensajes en función de tablas (relaciones) SQL. De esta manera, una base de datos de mensajes podrá ser relacional.

Si queremos almacenar los mensajes en sólo una tabla (que represente una relación) con la estructura presentada, existen muchos problemas para definir los campos o atributos (columnas, en SQL) de la tabla, ya que:

- El número de partes de cuerpo del contenido de un mensaje es variable.

- La definición del contenido es recursiva.

Existen más problemas que aparecerán al detallar los campos del envoltorio y del encabezamientos aún no descompuestos.

6.3.2. PRIMERA REPRESENTACION DE UN MENSAJE INTERPERSONAL EN UNA BASE DE DATOS RELACIONAL

Los hechos antes mencionados, junto con la estructura de un mensaje descrita, sugieren una posible estructura directa para el almacén de MIPs utilizando tres tablas, una para mensajes, otra para contenidos y la última para partes de cuerpo. De esta manera, los problemas antes explicados se pueden resolver, puesto que los números desconocidos de partes de cuerpo y contenidos están ahora relacionados con el número de filas (no de columnas) de las dos últimas tablas.

Al almacenar un mensaje, se descompone para poder adaptarse a las tres tablas, dependiendo de su estructura particular. Al recuperar este mensaje debemos ser capaces de reconstruirlo a partir de las tres tablas. Para que el usuario no tenga que preocuparse de la estructura de tres tablas, necesitamos una

aplicación que traduzca la vista del mensaje del usuario a la vista del mensaje de la base de datos relacional.

Presentamos a continuación una estructura factible de las tres tablas que podría obtener las características antes mencionadas.

La tabla de mensajes necesita dos campos básicos: envoltorio y contenido. El primero, como ya vimos, está estructurado a su vez en otros campos [MHS-C-X411]. En esta primera representación supondremos que cada campo corresponde a una columna de la tabla SQL. El otro campo básico de la tabla de mensajes será una columna con una referencia a la tabla de contenidos (los valores de la columna de referencia de contenido son transparentes al usuario). Podrían ser, por ejemplo, los campos identificador de MIP del encabezamiento).

La tabla de contenidos necesita los siguientes campos:

- Referencia de contenido: una columna (cuyos valores son transparentes para el usuario) que relaciona esta tabla con la tabla de mensajes.

- Encabezamiento: al igual que en el envoltorio, este campo está también estructurado [MHS-C-X420] y necesita (en esta primera representación) algunas columnas de la tabla de contenidos. Veremos más tarde que esta primera suposición debe

ser refinada.

- Partes de cuerpo: como mencionamos anteriormente, no es posible utilizar uno o más campos de una tabla general para almacenar las partes de cuerpo de un mensaje. Hemos adoptado la solución de usar una tabla únicamente para partes de cuerpo. Por tanto, la información de partes de cuerpo en la tabla de contenidos debería constar de dos columnas, una referenciando la primera parte de cuerpo, y la otra referenciando la última. Los valores de estas referencias de partes de cuerpo son transparentes al usuario. Al almacenar un mensaje se rellenan de manera adecuada estas columnas en paralelo con los campos de la tabla de partes de cuerpo de forma que el mensaje pueda ser fácilmente reconstruido en el momento de la recuperación.

Finalmente, la tabla de partes de cuerpo se puede componer de:

- Referencia de parte de cuerpo: esta columna se usará para relacionar esta tabla con la de contenidos.

- Texto: aquí, "texto" significa la información real de un mensaje normal. Podría ser una o más columnas con la información, o bien con referencias a otras tablas o ficheros opcionales. Esta última alternativa queda fuera de los objetivos de este trabajo, por lo que supondremos que "texto" es simplemente una columna de la tabla de partes de cuerpo.

- Mensajes reenviados: en caso de que la parte de cuerpo sea un mensaje reenviado, necesitamos dos campos más,

* Información de entrega previa (que, en esta primera solución, estará compuesta de varias columnas [MHS-C-X411]).

* Referencia a contenido (una columna cuyos valores referencian la tabla de contenidos).

La estructura de tres tablas propuesta para el almacén de MIPs se puede resumir, por tanto, como indica la tabla 6.1.

TABLA	COLUMNAS
mensajes	envoltorio, referencia_contenido
contenidos	referencia_contenido, encabezamiento, primera_parte_de_cuerpo, última_parte_de_cuerpo
partes_de_cuerpo	referencia_parte_de_cuerpo, texto, información_entrega_previa, referencia_contenido

Tabla 6.1: Estructura de tres tablas propuesta.

A continuación presentamos un ejemplo mostrando cómo se pueden almacenar los mensajes en esta estructura relacional de tres tablas.

6.3.2.1. EJEMPLO

Supóngase que tenemos los siguientes mensajes:

M1:	E1	B1	T1		
M2:	E2	B2	T21	T22	
M3:	E3	B3	IEP3	C(M1)	
M4:	E4	B4	IEP4	C(M3)	T4

donde M = mensaje, E = envoltorio, B = encabezamiento, T = texto, IEP = informaciónEntregaPrevia, y C(Mi) = contenido del mensaje "i".

Los valores almacenados en las tablas podrían ser los de las tablas 6.2, 6.3 y 6.4.

En la tabla 6.2, que representa la tabla "mensajes", los valores de la columna "referencia_contenido" corresponden a los identificadores de MIP de los mensajes; ya se explicó anteriormente el significado de la columna "envoltorio".

envoltorio	referencia_contenido
E1	1
E2	2
E3	3
E4	4

Tabla 6.2: Tabla "mensajes".

refer._contenido	encabez.	prim._parte_c.	últ._parte_c.
1	B1	1	1
2	B2	2	3
3	B3	4	4
4	B4	5	6

Tabla 6.3: Tabla "contenidos".

En la tabla "contenidos", "refer._contenido", "encabez.", "prim._parte_c." y "últ._parte_c." son las abreviaturas de "referencia_contenido", "encabezamiento", "primera_parte_contenido" y "última_parte_contenido", respectivamente. Los valores de las columnas "prim._parte_c." y "últ._parte_c." son referencias a partes de cuerpo; ya se explicó anteriormente el significado de la columna "encabez.".

refer._parte_c.	texto	inform._entrega_pr.	refer._cont.
1	T1	-	-
2	T21	-	-
3	T22	-	-
4	-	PDI3	1
5	-	PDI4	3
6	T4	-	-

Tabla 6.4: Tabla "partes_de_cuerpo".

En esta última tabla, "refer._parte_c.", "inform._entrega_pr." y "refer._cont." son las abreviaturas de "referencia_parte_de_cuerpo", "información_entrega_previa" y "referencia_contenido", respectivamente. Ya se explicaron anteriormente los significados de las columnas "texto" e "inform._entrega_pr.".

6.3.3. CONSIDERACIONES PARA UNA MEJOR REPRESENTACION DE UN MENSAJE INTERPERSONAL EN UNA BASE DE DATOS RELACIONAL

De acuerdo con las recomendaciones X.400 [MHS-C-X4xx], algunos de los campos del envoltorio y del encabezamiento de un

MIP, e incluso de la información de entrega previa, pueden tener más de un valor (un número variable de ellos). En general, esos valores pertenecen a dos dominios principales: mensajes y nombres. En otras palabras, algunos campos del envoltorio y del encabezamiento de un mensaje son relaciones entre mensajes y mensajes, o entre mensajes y nombres. En particular, y tratando sólo con campos variables, las relaciones mensaje-mensaje del encabezamiento de un mensaje son "obsoletiza" ("obsoletes") y "referencia" ("crossReferences"), y algunas relaciones mensaje-nombre podrían ser "destinatariosPrimarios" ("primaryRecipients") y "destinatariosDeCopia" ("copyRecipients"), por ejemplo. El envoltorio de un mensaje puede tener también relaciones mensaje-nombre. Además, los campos de tipo "mensaje" o "nombre" con sólo un valor pueden ser también vistos como relaciones.

Cada mensaje completo tiene información de P1 [MHS-C-X411] y de P2 [MHS-C-X420]. Un campo del encabezamiento P2 es el identificadorDeMensajeIP, pero las recomendaciones X.400 actuales no aseguran que este identificador sea único, por lo que no puede utilizarse para este fin.

Para poder distinguir mensajes diferentes, usaremos las siguientes reglas:

- Suponer que cada usuario no duplicará identificadores.

- La diferenciación entre mensajes P2 se hará usando una

combinación de los campos "originador" e "identificadorDeMensajeIP" del mensaje.

- El identificador único completo de un mensaje se construirá utilizando los valores de originador, identificadorDeMensajeIP y tiempoDeEntrega. De esta manera, será posible distinguir envoltorios P1 diferentes conteniendo el mismo mensaje P2 (reenvío, listas de distribución, ...). Lo que hemos llamado tiempoDeEntrega corresponderá al campo correspondiente de la operación de entrega ("deliver") definida en el protocolo P3 [MHS-C-X411].

Las explicaciones anteriores, y otras consideraciones que aparecerán enseguida, sugieren la siguiente modificación de la estructura de tres tablas antes presentada.

Un primer paso con una ligera modificación, basada en el esquema actual, podría ser la transformación de las columnas del envoltorio y del encabezamiento que se refieren a posibles multivalores en parejas de columnas que referencien el primer y el último mensaje o nombre perteneciente a ese campo.

Los valores de las columnas "primer_mensaje" y "último_mensaje" serían referencias a nuevas tablas mensaje-mensaje (una por cada relación mensaje-mensaje), y los valores de las columnas "primer_nombre" y "último_nombre" serían referencias a nuevas tablas mensaje-nombre (una por cada relación mensaje-

nombre existente). Es decir, deberíamos modificar la estructura de las tres tablas y añadir tantas tablas como relaciones mensaje-mensaje y mensaje-nombre haya en los campos de P1 y P2.

Hasta ahora, la estructura de la base de datos se ha hecho tomando como punto de partida la definición de MIP de las recomendaciones X.400 y buscando soluciones directas. Ahora, podemos proceder de una "manera más relacional" e intentar obtener una estructura multitabla más eficiente y lógica.

En primer lugar, podemos unir las tablas de mensajes y contenidos en una única tabla (tabla de mensajes), ya que no hay ninguna razón para mantener dos tablas separadas. La clave primaria de esta nueva tabla será un identificador de MIP único, construido tal como se explicó anteriormente en esta misma sección.

En segundo lugar, las columnas dobles que hemos añadido para relacionar la tabla de mensajes con las tablas mensaje-mensaje y mensaje-nombre no son necesarias, ya que las dos últimas colecciones de tablas pueden tener un identificador de MIP único como clave primaria, relacionando de esta manera las tablas mensaje-mensaje con la tabla de mensajes y también las tablas mensaje-nombre con la tabla de mensajes. De esta forma, es posible obtener, por ejemplo, todos los nombres de "PrimaryRecipient" de un MIP mirando sólo en la tabla mensaje-nombre de "primary recipient" (tiene una columna de clave

primaria llamada "ID-MIP-U", por "identificador de MIP único", que corresponde con el definido al principio de la sección), por lo que no tenemos que acceder a la tabla de mensajes para saber cuáles son las filas de la tabla mensaje-nombre correspondiente relacionadas con un MIP particular.

En tercer lugar, es posible eliminar las columnas "primera_parte_de_cuerpo" y "última_parte_de_cuerpo" de la tabla de mensajes arguyendo las mismas razones que se dieron en el párrafo anterior, y dando a la columna "referencia_parte_de_cuerpo", de la tabla de partes de cuerpo, valores de ID-MIP-U. El nombre de esta columna podría ser "ID-MIP_parte_de_cuerpo".

Finalmente, podemos resolver un pequeño problema que podría complicar bastante la gestión de las tablas. Nos referimos al hecho de que cada relación mensaje-mensaje y mensaje-nombre necesite una tabla propia, lo que hace muy elevado el número de tablas e implica tener que definir una nueva tabla por cada nuevo atributo que queramos añadir (pensemos en los campos extra del capítulo anterior). La solución es utilizar únicamente dos tablas (una para relaciones mensaje-mensaje y otra para relaciones mensaje-nombre) de forma que cada una de las tablas tenga tres columnas. La primera será el identificador de mensaje único, la segunda el valor de la relación con el mensaje identificado en la primera columna (será un mensaje o un nombre en función de la tabla de que se trate) y la última será una indicación del tipo de relación mensaje-mensaje o mensaje-nombre de que se trate. Los

valores que puede tomar esta columna son los distintos tipos de relación posible (los distintos atributos existentes), es decir lo que antes eran nombres de tabla. Ahora, cuando añadamos un nuevo atributo no será necesario crear una nueva tabla, sino que bastará con aumentar los valores del dominio de la tercera columna de las dos únicas tablas existentes para relaciones.

6.3.4. IMPLEMENTACION DE UNA BASE DE DATOS DE MENSAJES INTERPERSONALES RELACIONAL

Las explicaciones dadas en la última sección, y otras cuestiones que irán apareciendo, nos llevan a una estructura de almacenamiento para el almacén de MIPs que utiliza cuatro tablas, una para mensajes (no el mensaje completo, por supuesto), dos para relaciones de mensajes (mensaje-mensaje y mensaje-nombre) y la última para partes de cuerpo.

6.3.4.1. TABLAS

Para obtener las características antes mencionadas, una estructura factible de las cuatro tablas podría ser la que sigue.

6.3.4.1.1. TABLA DE MENSAJES

La tabla de mensajes se usará para almacenar la identificación de los mensajes y los campos de un solo valor, del envoltorio y del encabezamiento, que no se pueden almacenar adecuadamente en las otras tablas.

Para ello, esta tabla necesita tres campos básicos: envoltorio, encabezamiento e identificador de MIP único (ID-MIP-U). Los dos primeros, como se discutió anteriormente, e incluso el último (véase sección 6.3.3.), están estructurados a su vez en otros campos ([MHS-C-X411] y [MHS-C-X420]).

Casi todos los campos del envoltorio y del encabezamiento con un valor como máximo permanecerán en esta tabla con una columna SQL para cada campo, aunque todos estos campos los representaremos a partir de ahora con sólo dos columnas, para simplificar, a las que llamaremos "envoltorio_fijo" y "encabezamiento_fijo".

Las partes del envoltorio y del encabezamiento que son relaciones mensaje-mensaje o mensaje-nombre (con uno o más valores) se dejan a las tablas de relaciones de mensajes. Por ejemplo, campos que permanecerán en esta tabla son "tipoDeContenido" ("contentType") y "prioridad" ("priority") del envoltorio, y "tema" ("subject") y "fechaDeExpiración" ("expiryDate") del encabezamiento.

El último campo básico, el ID-MIP-U, constará de tres campos (como se mencionó en la sección 6.3.3.):

- "IdentificadorDeMensajeIP" del encabezamiento P2.

- "Originador", del encabezamiento P2. Aunque este campo sea realmente una relación mensaje-nombre con un solo valor, lo utilizaremos dentro de la identificación.

- "TiempoDeEntrega", de la operación "entrega" ("deliver") de P3. Aunque no se utilice el protocolo P3, siempre deberemos almacenar esta información.

El campo ID-MIP-U es la clave primaria de esta tabla, y se usará en todas las demás tablas para identificar mensajes. Sin embargo, en la mayor parte de los casos, sólo se necesitarán una o dos de las tres columnas que la forman para identificar o recuperar mensajes o información parcial sobre ellos.

6.3.4.1.2. TABLAS DE ENLACES

La tabla de relaciones mensaje-mensaje (o tabla enlaces_mensaje) y la tabla de relaciones mensaje-nombre (o tabla enlaces_nombre) tienen una estructura muy similar una a otra. Ambas necesitan los siguientes campos:

- Identificador de MIP único (ID-MIP-U): este campo es para identificar el mensaje con el que está relacionada la información de una fila. Un mensaje (con un ID-MIP-U) puede tener muchas filas en esta tabla, una para cada campo y valor del encabezamiento P2 y del envoltorio P1 con este tipo de enlace.

- Mensaje (para la tabla enlaces_mensaje) o nombre (para la tabla enlaces_nombre): los valores de este campo son los identificadores de MIP únicos de los mensajes, o de los nombres de Originador/Destinatario, con los que el mensaje de la columna ID-MIP-U está relacionado.

- Tipo de relación ("tipo_relación"): este campo es para identificar qué clase de relación (es decir, qué campo del encabezamiento o del envoltorio de un mensaje) existe entre los valores de las dos primeras columnas de cada fila. Algunos posibles valores de esta columna en la tabla enlaces_mensaje son "obsoletiza" ("obsoletes"), "referencia" ("crossReferences") o "enRespuestaA" ("inReplyTo"), y para la tabla enlaces_nombre son "destinatariosPrimarios" ("primaryRecipients"), "destinatariosDeCopia" ("copyRecipients") o "usuariosDeRespuesta" ("replyToUsers"). Cuando extendamos esta implementación a la estructura completa del capítulo 5, esta columna podrá tomar también como valores los identificadores de las atributos que allí añadimos a los ya definidos para mensajes en P1 y P2.

6.3.4.1.3. TABLA DE PARTES DE CUERPO

La tabla de partes de cuerpo está compuesta de:

- Identificador de MIP único (ID-MIP-U): esta columna se usará para identificar cuál es el mensaje que posee la parte de cuerpo.

- Texto: tal como explicamos en 6.3.2., "texto" significa el contenido real de una parte de cuerpo de tipo distinto a "mensajeReenviado". Normalmente, este contenido será texto. Mantendremos la suposición, tal como planteamos allí, de que "texto" es simplemente una columna de la tabla de partes de cuerpo.

- Mensajes reenviados: en caso de que la parte de cuerpo sea un mensaje reenviado, ya dijimos (véase 6.3.2.) que necesitamos dos campos más, que en este caso corresponderán a la información de entrega previa y al identificador de mensaje reenviado. Sin embargo, la información de entrega previa tiene características particulares.

Si un mensaje reenviado ha sido recibido previamente, la mayor parte de la información del campo informaciónEntregaPrevia estará ya en la base de datos de mensajes, debido a que ya estaba en el envoltorio del mensaje previo. De este modo, no necesitamos

almacenar otra vez esa información, sólo se necesita una referencia al mensaje previo. Esta referencia será el identificador de MIP único de ese mensaje. Sin embargo, la información nueva ha de ser almacenada en las tablas y necesitará una identificación diferente.

El identificador único para la nueva información estará formado por el identificadorDeMensajeIP de P2, el originador de P2 y el tiempoDeEntrega del mensaje (tiempo de entrega del envoltorio en el que está incluido el mensaje reenviado, no el del envoltorio de la remisión previa). Por lo tanto, el campo de mensaje reenviado será este identificador único (tres columnas, realmente).

6.3.4.1.4. RESUMEN DE LAS TABLAS

La estructura final de cuatro tablas propuesta para el almacén de MIPs se puede resumir como indica la tabla 6.5.

A continuación presentamos un ejemplo que muestra cómo se pueden almacenar mensajes en la estructura relacional de cuatro tablas que hemos definido.

TABLA	COLUMNAS
mensajes	ID-MIP-U, envoltorio_fijo, encabezamiento_fijo
enlaces_mensaje	ID-MIP-U, mensaje, tipo_relación
enlaces_nombre	ID-MIP-U, nombre, tipo_relación
partes_de_cuerpo	ID-MIP-U, texto, ID-MIP-U_reenviado

Tabla 6.5: Estructura final de cuatro tablas propuesta.

6.3.4.2. EJEMPLO

Supongamos que tenemos los mensajes ejemplo de la sección 6.3.2.1.

Como se apuntó anteriormente, la información de entrega previa está duplicada en su mayor parte. En este ejemplo, "IEP3" equivale aproximadamente a "E1", y "IEP4" a "E3".

Ahora, supóngase que tenemos los encabezamientos detallados de la tabla 6.6. En dicha tabla omitimos la información de envoltorio y restringimos el ejemplo a tres relaciones mensaje-mensaje ("obsoletes", "crossReferences" e "inReplyTo") y a dos relaciones mensaje-nombre ("primaryRecipient" y "copyRecipient"). Suponemos también que "Mi" representa el identificador de MIP único del mensaje "i".

mensaje	OBS.	CR.R.	I.R.T.	PR.R.	CO.R.
M1	-	-	-	N1	-
M2	M1	-	M6	N1,N2	N3
M3	-	M2,M5	M1	N4	-
M4	-	-	-	N3	-

Tabla 6.6: Encabezamientos detallados del ejemplo.

En la tabla 6.6, "M" significa "mensaje" y "N" significa "nombre". Obsérvese que no siempre serán necesarias las tres columnas del ID-MIP-U para identificar un mensaje. Las abreviaturas de la tabla corresponden: "obs." a "obsoletes", "cr.r." a "crossReferences", "i.r.t." a "inReplyTo", "pr.r." a "primaryRecipient" y "co.r." a "copyRecipient".

Suponemos que los campos del encabezamiento sin valores en el ejemplo, o no existen o no nos importan ahora. La extensión del ejemplo para tratar mensajes completos reales es directa, será lo mismo con más información (habrá más filas en la tabla).

El almacenamiento de los mensajes en la base de datos relacional de cuatro tablas sería como muestran las tablas 6.7 a 6.10.

ID-MIP-U	envoltorio_fijo	encabezamiento_fijo
M1	parte de E1	parte de B1
M2	parte de E2	parte de B2
M3	parte de E3	parte de B3
M4	parte de E4	parte de B4

Tabla 6.7: Tabla "mensajes".

ID-MIP-U	mensaje	tipo_relación
M3	M2	crossReferences
M3	M5	crossReferences
M2	M1	obsoletes
M2	M6	inReplyTo
M3	M1	inReplyTo

Tabla 6.8: Tabla "enlaces_mensaje".

ID-MIP-U	nombre	tipo_relación
M1	N1	primaryRecipient
M2	N1	primaryRecipient
M2	N2	primaryRecipient
M3	N4	primaryRecipient
M4	N3	primaryRecipient
M2	N3	copyRecipient

Tabla 6.9: Tabla "enlaces_nombre".

ID-MIP-U	texto	ID-MIP-U_reenviado
M1	T1	-
M2	T21	-
M2	T22	-
M3	-	M13
M4	-	M34
M4	T4	-

Tabla 6.10: Tabla "partes_de_cuerpo".

En la tabla 6.10, Mij de la columna ID-MIP-U_reenviado representa el identificadorDeMensajeIP y el originador del mensaje Mi junto con el tiempoDeEntrega del mensaje Mj.

6.3.5. EXTENSION DE LA ESTRUCTURA MULTITABLA DEFINIDA PARA LA ESTRUCTURA COMPLETA DE LA BASE DE DATOS DE MENSAJES

Hemos intentado mostrar que es posible tener una base de datos de mensajes relacional. Nuestro objetivo era almacenar mensajes interpersonales utilizando una estructura relacional. Para hacer eso estamos usando el estándar SQL como lenguaje de definición y manipulación de bases de datos relacionales.

Las secciones 6.3.1. y 6.3.4. estudian la estructura de un MIP y cómo mapear esta estructura en una base de datos relacional.

Pero lo anterior lo hemos restringido inicialmente a mensajes que cumplan estrictamente las recomendaciones X.400 [MHS-C-X4xx]. Sin embargo, la extensión para poder representar la estructura de base de datos definida en el capítulo anterior es bastante sencilla.

Vimos en dicho capítulo que un mensaje interpersonal en un almacén de mensajes debería constar de la siguiente información: atributos del envoltorio P1, atributos del encabezamiento P2, atributos de almacenamiento, atributos de extensiones a las recomendaciones X.400, atributos para documentos y contenido propiamente dicho del mensaje (normalmente texto). Asimismo, el

almacén debería ofrecer información acerca de las "supraestructuras" en las que se pueden agrupar los mensajes, aunque esta información no se almacene físicamente en la base de datos.

De toda la información anterior, en nuestra implementación multitabla sólo hemos tenido en cuenta los atributos P1 y P2 y el contenido del mensaje (aunque realmente no hemos entrado mucho en detalle en este último aspecto). Para completar la información que deseamos, falta almacenar tres tipos de atributos: de almacenamiento, extensiones a X.400 y de documentos. Todos estos los podemos agrupar, según sus características orientadas al almacenamiento, en:

a) Atributos con valor único.

b) Atributos que son relaciones mensaje-mensaje.

c) Atributos que son relaciones mensaje-nombre.

Ejemplos de los tres grupos son:

A) Valor único:

- Almacenamiento:

    * Versión.

* Tamaño de los datos.

* Almacenador (también podría considerarse una relación mensaje-nombre, aunque sólo tenga un valor posible).

- Extensiones a X.400:

* conjuntoDeCaracteres.

- Documentos:

* Título.

* Estatus del documento.

* Número de páginas.

B) Relaciones mensaje-mensaje:

- Extensiones a X.400:

* revisiónPropuestaDe.

* soluciónA.

C) Relaciones mensaje-nombre:

- Almacenamiento:

    * Lista de acceso.

- Extensiones a X.400:

    * respuestaPersonalA.

    * gruposDestinatarios.

    * antiDestinatariosPrimarios.

La forma de almacenar cada uno de estos tres grupos es:

- Valores únicos: hay dos soluciones que son, o bien añadir nuevas columnas a la tabla de mensajes, una por cada atributo distinto, o bien añadir una nueva tabla que tendrá una columna por cada atributo más una columna con el identificador de MIP único para relacionar los valores de los atributos con los mensajes a que corresponden. En principio, la primera solución será la más normal.

- Relaciones mensaje-mensaje y mensaje-nombre: se almacenarán en las tablas "enlaces_mensaje" y "enlaces_nombre". Para ello, la columna "tipo_relación" de dichas tablas tendrá

como nuevos valores posibles todos los identificadores de los nuevos atributos.

6.4. ACCESO A UNA BASE DE DATOS DE MENSAJES SQL

6.4.1. OPERACIONES SOBRE LA BASE DE DATOS DE MENSAJES

En el capítulo 5 habíamos definido cuatro operaciones básicas: "almacenar", "recuperar", "borrar" y "modificar". De todas ellas, la que presentaba más interés, por su dificultad, era la de recuperar. Una recuperación equivale a lo que se suele llamar consulta o interrogación en cualquier lenguaje de bases de datos. En nuestro caso, este lenguaje de interrogación será el lenguaje módulo del SQL (véase 6.2.).

Las operaciones "almacenar", "borrar" y "modificar" no presentan ningún problema para la base de datos de mensajes relacional, por lo que en el resto de la sección 6.4. trataremos únicamente de la operación de recuperación.

"Recuperar" será una sentencia "seleccionar" ("select") de SQL, la cual permite todas las características que habíamos requerido de tal operación. Básicamente, esas características eran la selección de la parte de un mensaje a recuperar y la selección del conjunto de mensajes sobre los que hacer la búsqueda para la recuperación. La potencia que deseábamos en las

condiciones de búsqueda también nos la puede proporcionar directa o indirectamente el SQL.

Por su parte, la operación "almacenar", de la que ya no volveremos a hablar, puede realizarse en SQL de dos formas: 1) mediante la sentencia "insertar" ("insert"); o 2) utilizando la sentencia "buscar-actualizar" ("update-search").

6.4.2. FORMULACION DE INTERROGACIONES Y ACCESO A LA BASE DE DATOS

Una vez hemos expresado el almacén de MIPs como una colección de tablas (véase 6.3.4.), el problema es cómo expresar las interrogaciones sobre el almacén de MIPs como interrogaciones de SQL, y cómo obtener mensajes que están en dicho almacén.

Ahora, por tanto, el objetivo es realizar interrogaciones sobre el almacén utilizando el lenguaje módulo del estándar SQL y la estructura multitablas propuesta.

Como se explicó anteriormente, es necesario tener un programa de aplicación entre el usuario del almacén (vista de mensajes del usuario) y el lenguaje SQL (vista de mensajes de la base de datos relacional). El programa de aplicación es el encargado de traducir las interrogaciones del usuario a interrogaciones SQL de una manera transparente. El programa de

aplicación debería mantener también la consistencia de las tablas, por ejemplo no repitiendo filas en una tabla.

Esta traducción de interrogaciones a la base de datos de mensajes (la del capítulo 5) a módulos SQL es relativamente fácil de realizar utilizando la estructura multitabla propuesta. En 6.4.3. se muestran algunos ejemplos de recuperaciones simples. El cómo automatizar la traducción de interrogaciones es un posterior tema de estudio que queda fuera de los objetivos de este trabajo.

Al igual que en 6.3., vamos a considerar que los mensajes que almacenamos son estrictamente MIPs X.400, por lo que las operaciones de acceso que desarrollemos (principalmente recuperaciones) se referirán sólo a los campos de P1 y P2 y al contenido, no a los nuevos campos de almacenamiento, documento, etc.

Podemos suponer, por ejemplo, que el almacén de MIPs es un Agente de Almacenamiento (AA) especial que utiliza un protocolo P2+, como se propone en la sección 4.4. Por lo tanto, las interrogaciones expresadas en el lenguaje módulo se pueden codificar en mensajes P2+, ya que el formato SQL de un módulo es una cadena de caracteres. Los resultados de una interrogación pueden ser mensajes (1 o más), parte o partes de mensajes, o información relacionada con mensajes. Por ello, necesitamos un protocolo P2+ capaz de manejar partes de mensajes de forma adecuada, tal como se mencionó en 5.1.5.1.4.

Debido a la descomposición de los mensajes en tablas, el almacén tendrá que reconstruir el mensaje (o mensajes) o parte (o partes) de un mensaje para reenviarlo(s) al agente de usuario que lo(s) solicite. Este trabajo lo realiza el programa de aplicación ya discutido.

La gestión de parámetros se puede dejar como problema local (para el agente de usuario que solicita al AA). Los mensajes remitidos por los AAs serán MIPs cuyos contenidos serán MIPs P2+ (o parte de ellos), haciéndose la asignación de parámetros una vez se hayan recibido los mensajes desde el almacén.

Sin embargo, otros esquemas de comunicación, como un protocolo interactivo directo de acceso al almacén (véase 4.6., 4.7. y 5.3.), son posibles, e incluso más fácilmente implementables.

6.4.3. EJEMPLOS DE RECUPERACIONES

A continuación se presentan algunos ejemplos de operaciones de recuperación. La interrogación se escribe primero en lenguaje natural y se traduce después a un lenguaje SQL más o menos formal. En la interrogación de lenguaje natural la vista de los mensajes es la de las recomendaciones X.400 [MHS-C-X4xx], la que hemos llamado vista de usuario. La interrogación SQL no se

formaliza totalmente para no distraer la atención y poder concentrarnos en nuestro problema principal que es la conversión a la estructura multitabla.

Como ya mencionamos, una operación "recuperar" equivale a una operación "select" de SQL.

Las interrogaciones ejemplo en lenguaje natural vienen a ser "informalizaciones" de interrogaciones sobre mensajes en un posible lenguaje formal de interrogación a una base de datos de mensajes relacional. La especificación de este lenguaje formal está fuera de los objetivos de este trabajo, aunque es una buena posible línea de continuación. Las interrogaciones en lenguaje natural pueden dar una idea de la potencia de la base de datos definida y de su posible lenguaje formal de interrogación.

Repetimos que lo que tratamos aquí es de comprobar que la estructura multitabla que hemos definido permite realizar las interrogaciones que pretendemos, y para ello las traducimos al lenguaje relacional de interrogación SQL.

6.4.3.1. EJEMPLO 1

Interrogación: Recuperar los destinatarios primarios ("primaryRecipient") de todos los mensajes que son respuesta al mensaje con identificador de mensaje IP (del encabezamiento P2)

IPMId1 originado por ORIG1.

Interrogación en SQL:

```
SELECT
  nombre
FROM enlaces_nombre
WHERE
  (enlaces_mensaje . identificadorDeMensajeIP = "IPMId1") AND
  (enlaces_mensaje . originador = "ORIG1") AND
  (enlaces_mensaje . tipo_relación = "inReplyTo") AND
  (enlaces_nombre . tipo_relación = "primaryRecipient") AND
  (enlaces_nombre . ID-MIP-U = enlaces_mensaje . mensaje))
```

(Recuérdese que identificadorDeMensajeIP y originador son columnas del ID-MIP-U. Asimismo, cuando hablamos de ID-MIP-U nos referimos en realidad a tres columnas (véase 6.3.3.), aunque lo expresamos como una para simplificar).

La interrogación en SQL selecciona una tabla (realmente una columna) con la información solicitada. Se supone la existencia de un programa de aplicación que presenta la información al usuario. La misma idea se aplica en los restantes ejemplos.

6.4.3.2. EJEMPLO 2

Interrogación: Recuperar el texto de todos los mensajes que son respuesta al mensaje IPMId2 (identificación P2) originado por ORIG2 y entregado en el instante de tiempo TE2.

Interrogación en SQL:

```
SELECT DISTINCT
  texto
FROM partes_de_cuerpo
WHERE
  (enlaces_mensaje . ID-MIP-U = "ID-MIP-U2") AND
  (enlaces_mensaje . tipo_relación = "inReplyTo") AND
  (partes_de_cuerpo . ID-MIP-U = enlaces_mensaje . mensaje)
```

(Suponemos que "ID-MIP-U2" es la composición de "IPMId2", "ORIG2" y "TE2").

Recordemos que estamos haciendo la suposición, no real, de que "texto" es únicamente una columna SQL.

6.4.3.3. EJEMPLO 3

Interrogación: Recuperar el encabezamiento completo de todos los mensajes que tengan la cadena de caracteres C1 en su campo

tema.

No hay una columna para cada campo del encabezamiento, excepto en lo que hemos llamado "encabezamiento_fijo", sino que las columnas "mensaje" (de la tabla enlaces_mensaje) y "nombre" (de la tabla enlaces_nombre) contienen los diferentes campos del encabezamiento, de acuerdo con los valores de las columnas "tipo_relación", por lo que es necesaria una interrogación SQL para cada uno de los campos del encabezamiento almacenados en la tabla de enlaces. Por ejemplo, la interrogación SQL para obtener los valores de "crossReferences" ("referencia") podría ser:

```
SELECT
  mensaje                          (* Aquí, "mensaje" se refiere
                                      al nombre de columna *)
FROM enlaces_mensaje
WHERE
  (mensajes . tema  LIKE S1) AND
  (mensajes . ID-MIP-U = enlaces_mensaje . ID-MIP-U) AND
  (enlaces_mensaje . tipo_relación = "crossReferences")
```

(En esta interrogación SQL, "tema" es una de las columnas del "encabezamiento_fijo" de la tabla de mensajes. ID-MIP-U sigue siendo realmente tres columnas).

Las interrogaciones SQL para recuperar los restantes campos del encabezamiento son similares.

6.4.3.4. EJEMPLO 4

Interrogación: Recuperar y mostrar todos los mensajes, sin mensajes reenviados incluidos, enviados a RECIP4.

Aquí, otra vez serían necesaias distintas interrogaciones SQL para obtener los campos del encabezamiento, envoltorio y partes de cuerpo. No todas las interrogaciones tendrían la misma complejidad, pudiéndose, en algunos casos, obtener más de un campo en una sola interrogación SQL (por ejemplo, todos los campos del envoltorio fijo). Ahora, la interrogación ejemplo en SQL recuperará el originador:

```
SELECT
  originador
FROM enlaces_nombre
WHERE
  (partes_de_cuerpo . texto  IS NOT NULL) AND
  (partes_de_cuerpo . ID-MIP-U = enlaces_nombre . ID-MIP-U)
  AND (enlaces_nombre . tipo_relación = "primaryRecipient") AND
  (enlaces_nombre . nombre = "RECIP4")
```

(Recuérdese que originador es una de las tres columnas del ID-MIP-U, por lo que no aparece explícitamente la comparación "partes_de_cuerpo . originador = enlaces_nombre . originador" en

la cláusula "where". Por la misma razón, el originador se podía haber obtenido de la tabla "partes_de_cuerpo", en vez de "enlaces_nombre").

6.4.3.5. EJEMPLO 5

Interrogación: Recuperar el envoltorio de todos los mensajes con identificador de mensaje interpersonal P2 IPMId5.

Ahora, será necesaria una interrogación SQL para cada relación de mensaje del envoltorio, y otra para los campos de un solo valor. Presentamos a continuación esta última.

Interrogación en SQL:

```
SELECT DISTINCT
     <columnas del envoltorio_fijo (tabla de mensajes)>
FROM mensajes
WHERE mensajes . identificadorDeMensajeIP = "IPMId5"
```

(identificadorDeMensajeIP es una de las tres partes del ID-MIP-U).

6.5. DERECHOS DE ACCESO

El tema de los derechos de acceso podía haberse tratado ya en profundidad en alguno de los capítulos anteriores, pero para poder plantear juntas todas las cuestiones relativas a ellos, lo hemos dejado voluntariamente para este capítulo.

El control del acceso a la información es un problema importante en todos los sistemas informáticos, y más aún si se trata de sistemas distribuidos como en nuestro caso.

En un sistema distribuido y abierto, como es el sistema de gestión de mensajes definido en las recomendaciones del CCITT, en el que cualquiera podría acceder a cualquier máquina perteneciente a dicho sistema desde cualquier parte del mundo, el problema de controlar quién accede y manipula la información es muy importante, dependiendo esta importancia del grado de confidencialidad de la información que manejemos.

6.5.1. DERECHOS DE ACCESO EN ALMACENES DE MENSAJES

Por lo que respecta a este trabajo, al tratar con almacenes de mensajes, debemos tener en cuenta los siguientes tipos de acceso al almacén:

- Lectura de mensajes o información relacionada (recupera-

ción).

- Escritura y envío de mensajes (almacenamiento).

- Borrado de mensajes.

- Modificación de mensajes (ya vimos que podíamos considerar que esta operación equivale al almacenamiento de nueva información parcial).

En un almacén dado, puede ser necesario seleccionar qué personas están autorizadas a realizar cada una de las operaciones antes mencionadas, e incluso, para algunas operaciones, es posible que los derechos de acceso sean distinto para cada uno de los mensajes del almacén.

Todo lo anterior nos lleva a la necesidad de almacenar una nueva información en la base de datos de cara a controlar quién y cómo accede a la información contenida en la base de datos. Esta información podrá ser, básicamente, de dos tipos:

- Listas de nombres con el detalle de las operaciones que está autorizada a realizar la persona o personas asociadas a cada nombre.

- Nuevos campos en los mensajes que den información de quién está autorizado a manipular, y de qué forma, cada mensaje en

concreto.

El hecho de que cada usuario del almacén de mensajes esté autorizado a realizar sólo un número determinado de operaciones y, lo que es más importante, sobre un subconjunto concreto de la información contenida en la base de datos, hace que cada usuario vea la base de datos como si sólo contuviese la información que él está autorizado a ver.

Esto plantea una serie de problemas relacionados con la estructura y significado de los mensajes interpersonales. Uno de ellos es el relativo a las respuestas a mensajes "no visibles", que veremos en 6.5.5.

6.5.2. OPERACIONES SOBRE LOS DERECHOS DE ACCESO

Hemos visto que el tener en cuenta los derechos de acceso a los mensajes almacenados en una base de datos implica la necesidad de almacenar información extra relacionada con tales derechos. El contenido de esta información puede ser dado por:

- El originador del mensaje, que sólo quiere que vean su mensajes una serie determinada de usuarios.

- El propietario del almacén, que querrá controlar qué mensajes se almacenan y modifican en su base de datos.

Esto implica tener que extender la estructura de la base de datos que hemos definido en el capítulo 5 y, asimismo, añadir un conjunto de operaciones para manipular el control de acceso. Todo esto queda, sin embargo, fuera del alcance de este trabajo. De todas formas, en la dos próximas secciones vamos a hacer un pequeño estudio de cómo manejar los derechos de acceso en SQL (sección 6.5.3.) y de algunas operaciones necesarias para que el propietario de una base de datos de mensajes pueda controlar la información sobre derechos de acceso (sección 6.5.4.).

6.5.3. DERECHOS DE ACCESO EN SQL

En general, el acceso a la base de datos de mensajes relacional estará restringido a nivel de mensaje completo, de cierto conjunto de atributos (especialmente relaciones entre mensajes o entre mensaje y nombre) y de parte de cuerpo (un mensaje reenviado incluido o el contenido real del mensaje (texto, en general)).

Para gestionar este control de acceso en SQL tenemos las dos siguientes alternativas:

A) Control fuera del SQL:

Esta primera alternativa consiste en añadir un campo (o

varios) con información de derechos de acceso a cada tabla. Con esta solución, el verdadero control del acceso no está a nivel del SQL sino a nivel de la aplicación que maneje la estructura SQL multitabla. Lo que se hace en SQL en este caso es almacenar en las tablas la información que hará falta para gestionar el acceso.

Otra posibilidad, dentro de la misma alternativa, es, por ejemplo, almacenar toda la información de derechos de acceso en una o varias tablas, pero independientes de las otras tablas con la información real de mensajes.

En ambos casos se debería añadir a todas las interrogaciones SQL una cláusula "where" en la que se comprueben los derechos de acceso, lo que implica que el SQL se utiliza para algo más que almacenar la información de derechos de acceso.

De esta forma, como ya anticipamos, los derechos de acceso están controlados por la aplicación "Interfaz de usuario - SQL" en vez de por el SQL directamente, aunque seguimos utilizando el SQL y una base de datos relacional.

B) Control dentro del SQL:

Para que el control de acceso lo realice totalmente el mismo SQL será necesaria la creación de una vista SQL para cada usuario en el momento en que éste accede.

Estas vistas se crean de forma que cada tabla se transforma en una vista de la misma tabla pero sin los mensajes que el usuario no está autorizado a ver. La información para hacer esto está en campos especiales de derechos de acceso que hemos de añadir, por lo que esta solución puede considerarse una extensión de la anterior, en la que la tarea que habíamos dejado para la aplicación la realiza ahora el mismo SQL.

Pero para crear una tabla de vista hemos de crear primero un esquema donde podemos además añadir definiciones de privilegios, tal como define el estandar SQL [DB-A-127], donde el esquema es el nivel en el que se pueden definir controles de acceso, aunque los controles de acceso puedan ser a nivel de operaciones y columnas.

Pero no tiene mucho sentido, ni será nada práctico, definir un nuevo esquema cada vez que accede un usuario, por lo que esta alternativa no parece muy factible.

6.5.4. OPERACIONES DE CONTROL DE LOS DERECHOS DE ACCESO

En este apartado daremos simplemente una pincelada a unas cuantas operaciones posibles para modificar los derechos de acceso a un almacén de mensajes. Se supone que estas operaciones sólo las puede realizar, si no se especifica lo contrario en un

caso concretro, el propietario del almacén.

Las operaciones son:

- Dar derechos de acceso a un mensaje (también los podría dar el "propietario" (originador, en principio) del mensaje).

- Dar derechos de acceso a mensajes que cumplen ciertas condiciones, pudiendo estar estas condiciones explícitas en los mensajes (por ejemplo, los campos de destinatarios).

- Dar derechos de acceso a atributos de un mensaje.

- Dar derechos de acceso a atributos de un mensaje, pero con condiciones en función de las características del mensaje.

Una cuestión delicada, planteada en [PALM-86a], que relaciona derechos de acceso y modificaciones de mensajes es quién está autorizado a poner antienlaces (enviar mensajes con anticampos). Nos referimos a qué anticampos, de todos los que se pueden recibir, son procesables, en función de quién los envía. El tema sólo lo dejamos así planteado, por estar fuera del alcance de este trabajo.

Finalizaremos este apartado comentando brevemente un punto más sobre modificaciones. Ya explicamos con bastante extensión los problemas relacionados con la operación de modificación

(véase 5.1.5.1.5.), lo que no detallamos es que existen una serie limitada de modificaciones que nunca se deben autorizar, como son cambiar, ni siquiera con anticampos, los siguientes atributos de un mensaje:

- "identificadorDeMensajeIP" (encabezamiento P2).

- "originador" (encabezamiento P2).

- "tiempo de creación" (almacenamiento).

- "identificador interno del mensaje en el almacén" (almacenamiento).

6.5.5. PROBLEMAS ENTRE DERECHOS DE ACCESO Y MENSAJES INTERPERSONALES X.400

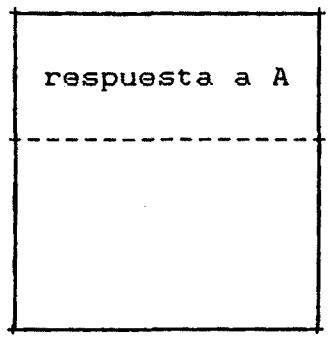
Al igual que el anterior, este apartado es un breve comentario sobre una cuestión concreta. Aquí, lo que tratamos son los problemas existentes entre los derechos de acceso tal como los hemos definido y los mensajes interpersonales (y el significado de sus campos según se definen en las X.400).

En concreto, vamos a tratar sólo uno de tales problemas, y es el relativo a las respuestas a mensajes "no visibles", es decir, aquéllos a los que no tiene derecho a acceder un

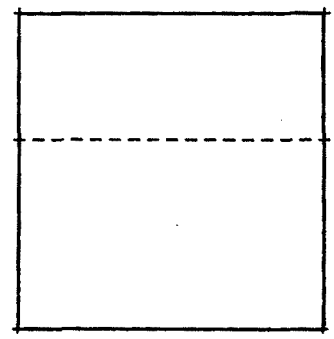
determinado usuario, por lo que para él no existen en la base de datos.

Supongamos el siguiente caso:

Mensaje B:



Mensaje A:



Es decir, tenemos los mensajes A y B siendo B una respuesta al mensaje A.

Por tanto, si un usuario tiene derecho a leer B, obtendrá ttoda la información de ese mensaje, incluido el dato de que es una respuesta a A, aunque ese usuario no pueda ver el mensaje A.

Al contrario, un usuario que tenga derecho a ver A no sabrá que existe una respuesta a ese mensaje, a no ser que también tenga derecho a acceder a B.

Este hecho nos lleva a concluir que haría falta un campo en el encabezamiento P2 de un MIP, que podríamos llamar "respuesta"

(o "ReplyIn"), que sea el inverso de "enRespuestaA" ("inReplyTo") y contenga los mensajes que son respuesta a un mensaje dado.

6.6. MENSAJES COMPLEJOS

6.6.1. ESTRUCTURA

Los mensajes con los que hemos operado directamente hasta ahora eran relativamente simples en su estructura, dentro de todo lo simple que puede ser la estructura de un mensaje interpersonal normal almacenado en la base de datos definida, nunca hemos tratado explícitamente mensajes que contengan otros mensajes en su estructura. Nos referimos al caso previsto en las recomendaciones del CCITT en que la parte de cuerpo del contenido de un mensaje es un mensaje reenviado, que será, a su vez, un mensaje estructurado.

Cuando hablamos de mensajes complejos nos referimos exactamente a esta estructura de mensajes anidados, es decir, mensajes incluidos en otros mensajes, donde el nivel de anidamiento no está expresamente limitado.

A continuación vamos a presentar un ejemplo de mensajes incluidos sobre el que vamos a trabajar.

6.6.1.1. EJEMPLO DE MENSAJES INCLUIDOS

Supongamos que la figura 6.1 representa el contenido (la parte P2) de un mensaje llamado M1.

En dicha figura, Mi = mensaje "i", Bi = encabezamiento del mensaje "i", Ci = contenido del mensaje "i", IEPi = información de entrega previa del mensaje "i", Ti = texto del mensaje "i" y M.R. = mensaje reenviado. Las letras A a F representan partes de cuerpo.

6.6.1.2. REPRESENTACION EN UNA TABLA

El contenido del mensaje M1 anterior se puede representar en una tabla en la que cada fila corresponda a lo que llamamos "bloque" (marcados con trazo continuo en la figura) que será realmente una parte de cuerpo, es decir, un mensaje reenviado (con su información de entrega previa correspondiente) o texto (en general).

En realidad, podríamos decir que los bloques A, B y D son los mensajes reenviados M2, M3 y M4, respectivamente, y que los bloques C, E y F son los textos de los mensajes M2, M3 y M4.

La tabla es la 6.11.

bloque	parte_de_c._de	texto	inf._entr._pr.	ID_M.R.
A	M1	-	IEP2	M2
B	M1	-	IEP3	M3
C	M2	T2	-	-
D	M3	-	IEP4	M4
E	M3	T3	-	-
F	M4	T4	-	-

Tabla 6.11: Tabla representando un mensaje complejo.

En la tabla ejemplo, "parte_de_c._de" es una abreviatura de "parte de cuerpo de" e indica el mensaje al que pertenece el bloque correspondiente. "inf._entr._pr." es la abreviatura de "información de entrega previa" e "ID_M.R." es el identificador del mensaje reenviado que está contenido en un determinado bloque.

Esta tabla es similar a la tabla de partes de cuerpo que definimos para representar mensajes en SQL (véase 6.3.2. y 6.3.4.1.3.).

6.6.1.3. REPRESENTACION EN ARBOL

La estructura de mensajes incluidos que estamos viendo (la estructura tabla anterior) se puede representar también mediante un árbol.

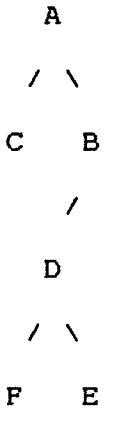
Además, podemos hacer que el árbol sea binario, con las ventajas de programación que ello conlleva, si tomamos los siguientes criterios:

- La raíz será el primer mensaje reenviado incluido en el mensaje que contiene a todos los demás (el bloque A del mensaje M1 en el ejemplo).

- El primer mensaje reenviado interno a cada bloque, en caso de que los tenga, será su hijo izquierdo.

- El siguiente bloque (mensaje reenviado o texto) dentro del mismo mensaje en que está incluido un mensaje será su hijo derecho.

El árbol binario correspondiente a nuestro ejemplo será (expresamos los elementos del árbol con lo que antes hemos llamado bloques (que eran partes de cuerpo)):



Obsérvese que las hojas del árbol (nodos sin hijos) corresponden a las partes de cuerpo que no son mensajes reenviados, es decir, a textos de mensajes. En este caso, son C, F y E.

No entraremos en más detalles en este tema por no estar dentro de los objetivos del presente trabajo.

6.6.2. OPERACIONES

En este apartado relativo a operaciones sobre mensajes complejos vamos a tratar básicamente tres cuestiones (véase [PALM-86a]):

- Respuestas recursivas: buscaremos un algoritmo para encontrar las respuestas, y respuestas a respuestas, recursivamente, a un mensaje dado.

- Mensajes incluidos: en muchas ocasiones interesará buscar qué mensajes están incluidos en uno dado, y viceversa.

- Mensajes incluidos similares: también será útil encontrar conjuntos de mensajes incluidos en otros que sean similares.

Los algoritmos para las tres operaciones (en realidad son cuatro, ya que la segunda son dos operaciones inversas), deberían tener en cuenta los derechos de acceso, pero no los consideraremos para claridad de los algoritmos.

La segunda y tercera operación mencionadas formarán parte usualmente de operaciones más complejas, como por ejemplo:

- Encontrar todas las respuestas, y respuestas a respuestas, recursivamente, a un mensaje dado, incluyendo todos los mensajes incluidos en ese mensaje, y todos los mensajes en que está incluido.

- Encontrar todos los destinatarios de un mensaje, incluyendo los destinatarios de todos los mensajes que contengan el mismo conjunto interno de mensajes interpersonales incuidos.

- Obtener el encabezamiento más interno de un mensaje dado que contenga las mismas partes de cuerpo de texto (este es el encabezamiento al que deben referirse las referencias "enRespuestaA" ("inReplyTo")).

A continuación presentamos unos posibles algoritmos para las operaciones mencionadas.

6.6.2.1. ALGORITMO: BUSCAR TODAS LAS RESPUESTAS A UN MENSAJE

```
procedimiento buscar_todas_las_respuestas (var mensaje);

  var existe: booleano;

     (* Las variables que representan información de la base
     de datos,  como mensajes o cursores,  no las declaramos
     *)

  comienzo
    existe:= verdadero;
    crear_cursor (mensaje, cursor_mensaje);        (* crea un
          cursor,  al que llamamos "cursor_mensaje", del que
          vamos a extraer los mensajes *)

    mientras existe hacer
    comienzo
      buscar_la_siguiente_respuesta (mensaje,
                                     cursor_mensaje,
                                     mensaje_encontrado,
                                     existe);      (*  busca
```

```
                un mensaje dentro del cursor *)

        si existe entonces
        comienzo
          dar_mensaje (mensaje_encontrado);     (* presenta al
                "usuario"  del  procedimiento  un  mensaje
                resultado *)

          buscar_todas_las_respuestas (mensaje_encontrado);
        fin;
      fin;
    fin;
```

Este primer algoritmo es general, independiente de la base de datos que utilicemos. Sin embargo, los procedimientos "crear_cursor" y "buscar_la_siguiente_respuesta", que vamos a definir seguidamente, sí deberán tener en cuenta el lenguaje SQL y la estructura multitabla definida (véase 6.3. y 6.4.). El procedimiento "dar_mensaje" no lo detallamos, ya que depende de la aplicación en la que esté implementado este algoritmo. Obsérvese que el algoritmo principal es recursivo.

```
    procedimiento crear_cursor (mensaje;
                                var cursor_mensaje);

            (*  Es  simplemente una declaración de cursor  SQL
```

```
        (véase 6.2.2.1.) *)

  comienzo
    declare cursor_mensaje cursor for
      select "ID-MIP-U"
        from "mensajes"
        where "inReplyTo" = mensaje;  (* "inReplyTo" es una
                                columna de "encabezamiento_fijo" *)
  fin;


procedimiento buscar_la_siguiente_respuesta
                                          (mensaje_entrada;
                                           var cursor_mensaje;
                                           var mensaje_salida;
                                           var existe);

        (* Devuelve el siguiente mensaje del cursor *)

  comienzo
    fetch cursor_mensaje into mensaje_salida;
    existe:= (SQLCODE<>100);      (* 100 es el código de
                                  cursor vacío *)
  fin;
```

6.6.2.2. ALGORITMO: BUSCAR TODOS LOS MENSAJES REENVIADOS INCLUIDOS EN UN MENSAJE

```
procedimiento buscar_mensajes_incluidos (var mensaje);

  var existe: booleano;

     (* Las variables que representan información de la base
     de datos, como mensajes o cursores, no las declaramos
     *)

  comienzo
    primero (cursor_mensaje, mensaje, mensaje_salida, existe);
                (* crea un cursor, al que llamamos
                "cursor_mensaje", del que vamos a extraer los
                mensajes, y busca el primero de ellos *)

    mientras existe hacer
    comienzo
      dar_mensaje (mensaje_salida);          (* presenta al
                "usuario" del procedimiento un mensaje
                resultado *)

      buscar_mensajes_incluidos (mensaje_salida);
      buscar_el_siguiente_mensaje (mensaje_entrada,
                                   cursor_mensaje,
                                   mensaje_salida,
```

```
                                        existe);        (* busca un
                mensaje dentro del cursor *)

    fin;
  fin;


procedimiento primero (var cursor_mensaje;
                       mensaje_entrada;
                       var mensaje_salida;
                       var existe);

          (*  Es una declaración de cursor SQL y la búsqueda
          del primer elemento del cursor *)

  comienzo
    declare cursor_mensaje cursor for
      select "ID-MIP-U_reenviado"
        from "partes_de_cuerpo"
        where "ID-MIP-U" = mensaje_entrada;
    fetch cursor_mensaje into mensaje_salida;
              (* Sólo debería seleccionar la primera fila que
                cumpla la condición *)
    existe:=  (SQLCODE<>100);       (*  100  es el código  de
                                    cursor vacío *)
  fin;
```

```
procedimiento buscar_el_siguiente_mensaje
                                        (mensaje_entrada;
                                        var cursor_mensaje;
                                        var mensaje_salida;
                                        var existe);

        (* Devuelve el siguiente elemento del cursor *)

  comienzo
    fetch cursor_mensaje into mensaje_salida;
    existe:= (SQLCODE<>100);        (* 100 es el código de
                                        cursor vacío *)
  fin;
```

6.6.2.3. ALGORITMO: BUSCAR TODOS LOS MENSAJES REENVIADOS EN QUE ESTA INCLUIDO UN MENSAJE

Este algoritmo es el inverso del anterior. Es similar a la búsqueda directa excepto en que:

- Las condiciones de búsqueda (cláusula "where" de la declaración de cursor del procedimiento "primero") son por el campo "ID-MIP-U_reenviado" en vez de por "ID-MIP-U" (ambos de la tabla "partes_de_cuerpo").

- Los resultados (cláusula "select" de la declaración antes mencionada) son "ID-MIP-U" en vez de "ID-MIP-U_reenviado".

Aunque en todos los algoritmos estemos utilizando el identificador de mensaje interpersonal único (tres campos en realidad (véase 6.3.3.)), éste no es necesario, ya que sería suficiente una identificación a nivel de P2 (con sólo dos de los tres campos de "ID-MIP-U" (o "ID-MIP-U_reenviado")). Es decir, nos bastarían los campos "identificadorDeMensajeIP" y "originador".

6.6.2.4. COMPROBACION DE QUE DOS MENSAJES TIENEN EL MISMO CONJUNTO DE MENSAJES INTERPERSONALES INCLUIDOS

En 6.6.2.2. vimos un algoritmo para encontrar todos los mensajes incluidos dentro de otro. Una forma de comprobar si dos mensajes tienen el mismo conjunto de mensajes incluidos es basándonos en ese algoritmo y hacer comparaciones a cada nivel de llamada recursiva mientras los mensajes incluidos que vayamos encontrando sean iguales, o hasta que se acaben los anidamientos (mensajes incluidos).

Otra forma de plantear el problema de los mensajes incluidos en otros es basándonos en las representación en árbol que hicimos en la sección 6.6.1.3.

Con esa representación, obtener todos los mensajes reenviados incluidos en otro equivale a obtener del árbol anterior la raíz y todos los hijos que tengan hijos, es decir, todos los nodos del árbol (o, dicho de otra manera, todos los elementos excepto las hojas).

En el ejemplo que vimos (véase 6.6.1.1. y 6.6.1.3.), estos miembros serán los bloques A, B y D, que corresponden a los mensajes reenviados (con sus informaciones de entrega previa correspondientes) M2, M3 y M4, respectivamente.

El problema estaría en la construcción del árbol, pero eso se podría hacer mediante el procedimiento definido en 6.6.2.2.

De todas formas, como ya dijimos, no entraremos en detalles en este tema por estar fuera de nuestros objetivos.

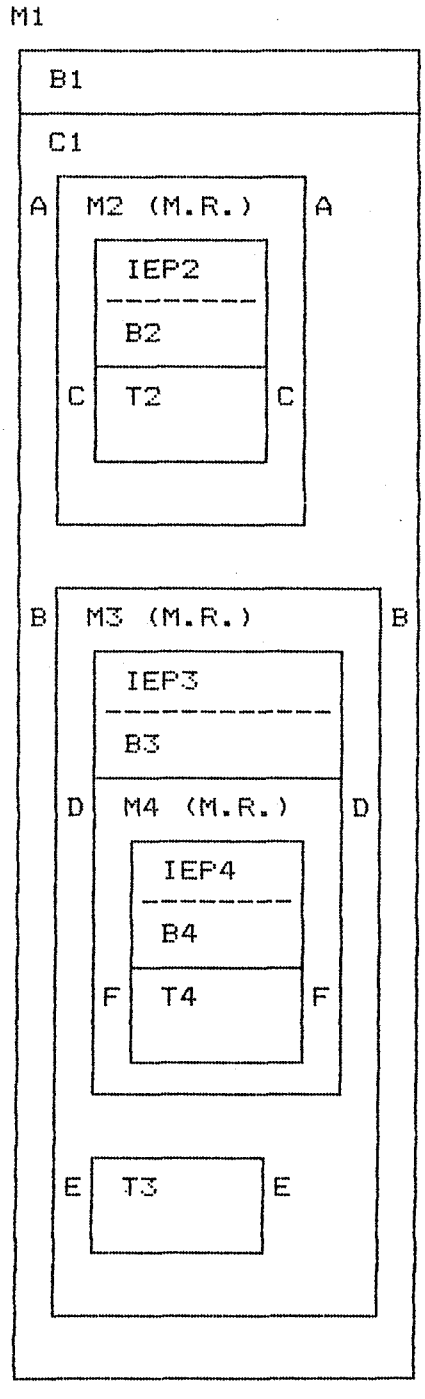


FIGURA 6.1: Ejemplo de mensaje complejo.

7. LINEAS DE CONTINUACION

En este último y más breve capítulo vamos a señalar algunas posibles líneas de continuación del trabajo realizado.

Ya hemos ido comentando a lo largo del texto en qué punto de algunos temas concretos nos quedábamos, indicando cómo se podía continuar el trabajo de investigación que habíamos hecho.

En concreto, algunas posibles vías de investigación futuras en el tema de los sistemas de almacenamiento son:

- Estudiar el problema del almacenamiento y recuperación del texto de los mensajes interpersonales.

- Desarrollar un lenguaje formal de interrogación de la base de datos que hemos definido. Sería un lenguaje no procedimental de alto nivel en el que el usuario sólo vería mensajes y sus atributos, siendo transparente para él la estructura interna de la base de datos. Este lenguaje estaría incluido en el interfaz de usuario.

- Desarrollar un traductor automático de interrogaciones de

usuario a la estructura multitabla SQL propuesta. Este traductor está íntimamente relacionado con el lenguaje anterior.

- Estudiar los posibles formatos de codificación de operaciones sobre la base de datos dentro de mensajes P2. Este punto también tendría relación con el lenguaje formal mencionado.

- Definir una nueva estructura y operaciones para los derechos de acceso.

- Definir un subconjunto del lenguaje SQL de cara a la estandarización.

- Desarrollar una base de datos local para combinar con un sistema basado en el protocolo P7.

REFERENCIAS

A continuación, presentamos una lista de referencias bibliográficas. La mayoría de ellas han sido mencionadas explícitamente a lo largo del texto. Las demás, aunque no referidas directamente, también han sido utilizadas, en mayor o menor medida, para el desarrollo del trabajo.

- [ALAB-82]

Alabau, A. et al.: "Teleinformática y redes de computadores", Serie Mundo Electrónico, Marcombo, Boixareu editores, 1982.

- [AMIG-86]

COST-11ter (CCE), "Summary of the workshop on Group Communication in MHS", 8 Abril 1986, Bruselas (organizado por el proyecto AMIGO de COST-11ter), Abril 1986.

- [BABA-85]

Babatz, R. y Bogen, T.: "Referable Documents: Semantic Relations in Message Handling Systems", Borrador del GMD, Mayo 1985.

- [CERI-84]

Ceri, S. y Pelagatti, G.: "Distributed Databases. Principles and Systems", Ed. McGraw-Hill, 1984.

- [CODD-70]

Codd, E. F.: "A relational model of data for large shared data banks", Communications ACM, volumen 13, número 6, Junio 1970.

- [CROS-86]

Cross, T. B. y Raizman, M. B.: "Networking. An electronic mail handbook", Ed. Scott, Foresman and Company, 1986.

- [DATE-81]

Date, C.J.: "An introduction to database systems" (tercera edición), Ed. Addison Wesley, 1981.

- [DATE-83]

Date, C.J.: "An introduction to database systems (volume II)", Ed. Addison Wesley, 1983.

- [DB-A-SUPL]

ANSI X3H2: "Supplemental information for draft proposed American National Standard X3.135-198X Database Language SQL", Marzo 1985.

- [DB-A-127]

ANSI X3H2-85-127: "Database Language SQL", October 1985 (It is the same document as [DB-I-849]).

- [DB-E-30]

ECMA TR/30: "Remote Database Access Services and Protocol", Diciembre 1985.

- [DB-I-117]

ISO/TC97/SC21/WG3 N117: "SQL", Febrero 1986.

- [DB-I-387]

ISO/TC97/SC18/WG4 N387 (procedente de SC21/WG3): "A remote database access protocol based on ROS", documento de trabajo, Junio 1986.

- [DB-I-849]

ISO/TC97/SC21 N849 (or WG3 N48): "Database Language SQL", October 1985 (It is the same document as [DB-A-127]).

- [DB-I-1243]

ISO/TC97/SC21 N1243: "First initial draft of ISO Remote Database Access Service and Protocol", Marzo 1986.

- [DELG-86]

Delgado, J. et al.: "Storage Agents in Message Handling Systems", Proceedings of the IFIP WG8.4 Working Conference on

Methods and Tools for Office Systems, Pisa (Italia), Octubre 1986.

- [DELG-86]

Delgado, J. et al.: "Use of SQL for Message Storage and Retrieval", Documento de trabajo del proyecto AMIGO, Abril 1986.

- [DOA-E-34]

ECMA/TC32-TG5/86/34: "Distributed Services Framework" (primer borrador de trabajo), Febrero 1986.

- [DOA-E-90]

ECMA/TC32-TG5/86/90: "Distributed Office Services Framework" (cuarto borrador de trabajo), Agosto 1986.

- [DOA-E-107]

ECMA/TC32-TG5/86/107: "Framework for Distributed Office Applications" (quinto borrador de trabajo), Octubre 1986.

- [DOA-E-275]

ECMA/TC32/85/275: "A Strawman Proposal Distributed Services Framework", Octubre 1985.

- [DOA-I-355]

ISO/TC97/SC18/WG4 N355: "Distributed office services (applications)", Primer documento de trabajo, Enero 1986.

- [DOA-I-440]

ISO/TC97/SC18/WG4 N440: "Distributed office services model" (primer borrador de trabajo), Junio 1986.

- [DOA-I-504]

ISO/TC97/SC18/WG4 N505: Es el mismo documento que [DOA-E-107].

- [DS-C-Xdsx]

CCITT STUDY GROUP VII, Directory Systems (Q35 / VII), (Version 2): "Draft Recommendation X.ds1 -- X.ds7", Enero 1986.

- [FR-E-83]

ECMA/TC32-TG5/86/83: "A proposal for filing and retrieval service and filing and retrieval service access protocol", Agosto 1986.

- [FR-E-94]

ECMA/TC32-TG5/86/94 (TC32/86/206): "Document filing and retrieval: scope of new work item". Septiembre 1986.

- [FR-E-98]

ECMA/TC32-TG5/86/98: "Document filing and retrieval service. General definition of functionalities". Octubre 1986.

- [FR-E-99]

ECMA/TC32-TG5/86/99: "Comment on document filing and

retrieval". Octubre 1986.

- [FR-I-578]

ISO/TC97/SC18 N578: "Text and Office Systems Standards Development Relating to Filing and Retrieval: WG 1 Review of Progress and Direction", Septiembre 1985.

- [HALS-85]

Halsall, F.: "Introduction to data communications and computer networks", Ed. Addison Wesley, 1985.

- [KAUF-85]

Kaufmann, P.: Model of a location-independent access to a distributed MHS", Second International Symposium on Computer Message Systems, IFIP, Washington, Septiembre 1985.

- [KILL-85]

Kille, S. E.: "Mapping the global naming tree onto a relational database", Second International Symposium on Computer Message Systems, IFIP, Washington, Septiembre 1985.

- [MEDI-86]

Medina M., Maude T. y Smith H.: "An Extended X.400 Architectural Model", ACM Computer Communication Review, volumen 16, número 2, Abril 1986.

- [MHS-C-MS]

CCITT Rapporteur Group on Message Handling (Q33/VII): "Conceptual Model of the Message Store", Melbourne (Australia), April 1986.

- [MHS-C-SDR]

CCITT Q33/VII OD-6: "Output from the editing group on Submission, Delivery and Retrieval", Octubre 1986.

- [MHS-C-WD1]

CCITT Special Rapporteur's Group on Message Handling Systems (Question 33/VII): "X.400-series working document", Septiembre 1985.

- [MHS-C-X4xx]

CCITT Study Group VII, Report R 38 and R 39, Recomendaciones X.400, X.401, X.408, X.409, X.410, X.411, X.420, X.430, "Message Handling Systems", Torremolinos (España), 1984.

- [MHS-C-X400]

CCITT Study Group VII, Report R 38: Recomendación X.400, "System Model - Service Elements", Torremolinos (España), 1984.

- [MHS-C-X401]

CCITT Study Group VII, Report R 38: Recomendación X.401, "Basic Service Elements and Optional User Facilities", Torremolinos (España), 1984.

- [MHS-C-X408]

CCITT Study Group VII, Report R 38: Recomendación X.408, "Encoded Information Type Conversion Rules", Torremolinos (España), 1984.

- [MHS-C-X409]

CCITT Study Group VII, Report R 38: Recomendación X.409, "Presentation Transfer Syntax and Notation", Torremolinos (España), 1984.

- [MHS-C-X410]

CCITT Study Group VII, Report R 38, Recomendación X.410, "Remote operations and reliable transfer server", Torremolinos (España), 1984.

- [MHS-C-X411]

CCITT Study Group VII, Report R 39, Recomendación X.411, "Message transfer layer Elements", Torremolinos (España), 1984.

- [MHS-C-X420]

CCITT Study Group VII, Report R 39, Recomendación X.420, "Interpersonal messaging user agent layer", Torremolinos (España), 1984.

- [MHS-C-X430]

CCITT Study Group VII, Report R 39, Recomendación X.430,

"Access Protocol for Teletex Terminals", Torremolinos (España), 1984.

- [MHS-E-31]

ECMA-TR/31: "Remote Operations: Concepts, Notation and Connection-oriented Mappings", Febrero 1986.

- [MHS-E-39]

ECMA/TC32-TG5/86/39: "Conceptual model of the message store", Mayo 1986.

- [MHS-E-59]

ECMA/TC32-TG5/85/59: "Distributed Application for Message Interchange (MIDA) Mailbox Service and Mailbox Service Access Protocol" (tercer borrador), Noviembre 1985.

- [MHS-E-87]

ECMA/TC32-TG5/86/87: "Distributed Application for Message Interchange (MIDA) Mailbox Service and Mailbox Service Access Protocol" (quinto borrador), Agosto 1986.

- [MHS-E-93]

STANDARD ECMA-93: "Distributed Application for Message Interchange (MIDA)", Septiembre 1984.

- [MHS-E-109]

ECMA/TC32-TG5/86/109: "Distributed Application for Message

Interchange (MIDA) Mailbox Access Service Definition and Protocol Specification" (sexto borrador), Noviembre 1986.

- [MHS-I-BARNA]

ISO/TC97/SC18/WG4, Documentos de salida de la reunión celebrada en Barcelona del 1 al 5 de Diciembre de 1986, Diciembre 1986.

- [MHS-I-299]

ISO/TC97/SC18/WG4 N299, "MOTIS: Functional description and service specification for MOTIS" (para DP 8505), Agosto 1985.

- [MHS-I-316]

ISO/TC97/SC18/WG4 N316: Es el mismo documento que [PALM-85].

- [MHS-I-319]

ISO/TC97/SC18/WG4 N319: Es el mismo documento que [MHS-E-59].

- [MHS-I-328]

ISO/TC97/SC18/WG4 N328: Es el mismo documento que [MEDI-86] (cuando era documento de trabajo del proyecto AMIGO).

- [MHS-I-436]

ISO/TC97/SC18/WG4 N436: The relationship of MOTIS to MHS", Julio 1986.

- [MHS-I-501]

ISO/TC97/SC18/WG4 N501 (documento de trabajo para DP 9072/1): "Information Processing Systems - Text Communications - MOTIS - Remote Operations Part 1: Model, Notation and Service Definition", Noviembre 1986.

- [MHS-I-502]

ISO/TC97/SC18/WG4 N502 (documento de trabajo para DP 9072/2): "Information Processing Systems - Text Communications - MOTIS - Remote Operations Part 2: Protocol Specification", Noviembre 1986.

- [MHS-I-505]

ISO/TC97/SC18/WG4 N505: Es el mismo documento que [MHS-E-109].

- [MHS-I-8824]

ISO/DIS 8824: "Specification of Abstract Syntax Notation One (ASN.1)", 1986.

- [MHS-I-8883]

"Message Oriented Text Interchange Systems - Message Transfer Sublayer, Message Interchange Service and Message Transfer Protocol", 1986.

- [MHS-I-9065]

ISO/TC97/SC18/WG4 N438 (texto revisado para DP 9065):

"Information Processing. Text Communication. Message Oriented Text Interchange Systems. User Agent Sublayer. Inter-personal Messaging User Agent. Message Interchange Formats and Protocols", Septiembre 1986.

- [MHS-I-9066]

ISO/TC97/SC18/WG4 N452 (texto revisado para DP 9066): "Information Processing. Text Communication. Message Oriented Text Interchange Systems. Reliable Transfer Service Element and Use of the Presentation Service", Agosto 1986.

- [MMS-86a]

MINI-MICRO SYSTEMS: "OSI standards spur product profusion", W. Rauch-Hindin, Junio 1986.

- [MMS-86b]

MINI-MICRO SYSTEMS: "Upper Level OSI protocols near completion", W. Rauch-Hindin, Julio 1986.

- [ODA-E-101]

Standard ECMA-101: "Office Document Architecture", Septiembre 1985.

- [ODA-I-8613]

ISO/TC97/SC18/WG3: "Information Processing - Text and Office Systems - Document Structures. Office Document Architecture/Office Document Interchange Format (ODA/ODIF)" (para

DP 8613), Abril 1986.

- [OSI-I-575]

ISO/TC97/SC16 N575: "Brief description of the reference model of open systems interconnection", 1979.

- [OSI-I-1494]

ISO/TC97/SC21 N1494: "Application Layer Structure", Septiembre 1986.

- [OSI-I-7498]

ISO 7498: "Information Processing Systems. Open Systems Interconnection. Basic Reference Model", Junio 1979.

- [PALM-85]

Palme, J.: "Amigo: Document storage and retrieval", Documento de trabajo del proyecto AMIGO, Noviembre 1985.

- [PALM-86a]

Palme, J.: "Examples of storage database operations", Documento de trabajo del proyecto AMIGO, Abril 1986.

- [PALM-86b]

Palme, J.: "Influence of MHS on user agent databases", Documento de trabajo del proyecto AMIGO, Mayo 1986.

- [PALM-86c]

Palme, J.: "Header structures in the storage system", Documento de trabajo del proyecto AMIGO, Mayo 1986.

- [SARA-86]

Saras, J. A. y Santo, H.: "Evaluation of existing concepts for modelling storage service agents", Documento de trabajo del proyecto AMIGO, Marzo 1986.

- [TANE-81]

Tanenbaum, A. S.: "Computer networks", Ed. Prentice-Hall, 1981.

- [TSIC-82]

Tsichritzis, D. C. y Lochovsky, F. H.: "Data models", Ed. Prentice-Hall, 1982.

Agradecimientos

En primer lugar, debo agradecer a mi director de Tesis, Dr. Manuel Medina, el haberme situado en el tema para poder realizar el trabajo, así como toda la inestimable ayuda prestada a lo largo del desarrollo de la labor.

También debo mencionar a todos los miembros del proyecto de investigación AMIGO, patrocinado por la acción COST11ter de la Comisión de las Comunidades Europeas, dentro del cual se han discutido y confrontado una buena parte de las ideas expuestas en la Tesis.

Dentro del grupo AMIGO quiero destacar a dos miembros del departamento de ordenadores del "Hahn-Meitner-Institut" de Berlín (R.F. Alemania) con quienes estuve trabajando allí durante más de dos meses sobre temas que dieron lugar a gran parte de la Tesis Doctoral. Estos son Berthold Butscher y Michael Tschichholz.

Relacionadas también con el proyecto AMIGO están las personas de la E.T.S.I. y la E.U.I.T. de Telecomunicación de la U.P. de Madrid con las que estamos trabajando dentro de un proyecto de la CAICYT, entre las que destacan el impulsor del

proyecto, Juan Riera, y Juan Antonio Saras, Justo Carracedo y Encarna Pastor, con quienes he comentado los artículos emanados del trabajo de esta Tesis.

Asimismo, agradezco a todos los miembros del departamento de "Ciencias de los Computadores" de la E.T.S.E. de Telecomunicació de la U.P.de Catalunya su colaboración y apoyo prestados.

Finalmente, debo expresar también agradecimiento a todos los miembros de mi familia por su ánimo y ayuda constante durante todo el tiempo que he dedicado al desarrollo de esta Tesis.