Anexo III: Diseño físico

Entorno de desarrollo

Para la construcción y las pruebas se hará uso de contenedores virtuales, concretamente de Docker. De esta forma no es necesario instalar PostgreSQL en el ordenador; además, se garantiza que el sistema funcionará en cualquier otra plataforma donde simplemente se instale Docker.

Se procede a detallar la instalación de Docker y la configuración del sistema al respecto, se ejecutan los siguientes comandos:

yum install docker

sudo firewall-cmd --permanent --zone=trusted --add-source=127.0.0.1/32

sudo firewall-cmd --permanent --zone=trusted --add-port=5432/tcp

sudo firewall-cmd --reload

Con esto queda Docker instalado en un sistema Fedora25 y el cortafuegos configurado para admitir acceso al puerto donde sirve PostgreSQL. A continuación se crea un directorio, en este se ubicará la BBDD, y se le configura el contexto SELinux –lo cual solo es necesario si este servicio está en ejecución– que utiliza Docker:

sudo mkdir -p /opt/dockyard/postgres

sudo chcon -Rt svirt_sandbox_file_t /opt/dockyard/postgres

Se procede con la creación del contenedor, sin entrar en detalle, cabe decir que Docker utiliza una imagen que contiene un pequeño sistema de archivos donde está instalado el *software*, PostgreSQL 9.6 en este caso. Con el comando *docker run* se inicializa la instancia, se le asigna un nombre y ciertos parámetros de configuración del servicio, como el puerto y la contraseña; y, lo más importante, se le indica un directorio físico en el que escribir los datos, concretamente el creado anteriormente, el cual se mapea con el que el SGBD utiliza por defecto. De esta forma, aunque el servicio PostgreSQL está virtualizado la BBDD está físicamente en el sistema de archivos del anfitrión. Una vez creada la instancia con los parámetros comentados, simplemente hay que arrancar el servicio Docker y luego la instancia utilizando su nombre.

docker run --name postgresql-tfg-uoc \
-p 5432:5432 \
-v /opt/dockyard/postgres:/var/lib/postgresql \
-e POSTGRES_PASSWORD=palocariz \
-d postgres:9.6.2

A continuación se crea un segundo contenedor con PgAdmin, en el corre la aplicación web que se utiliza para las pruebas y la gestión del SGBD.

docker run --name pgadmin4-tfg-uoc \
-p 5050:5050 \
-e POSTGRES_PASSWORD=palocariz \
-d fenglc/pgadmin4

A este segundo contenedor le falta un detalle, la imagen de Docker cada vez que arranca está tal cual se descargó del repositorio DockerHub; para el uso de esta aplicación se hace necesario configurar la conexión al servidor de base de datos, pero para cada nueva iniciación de la imagen se habrán perdido los datos; por tanto, se inicia la instancia, se configura la conexión y se ejecuta el siguiente comando con la maquina en funcionamiento:

docker commit pgadmin4-tfg-uoc

Con él se graban los cambios realizados en la instancia virtual del proceso PgAdmin a la imagen con la que trabaja, de forma que ya no será necesario configurar la conexión al servidor por cada arranque.

A partir de este momento, para poner en funcionamiento el entorno de desarrollo se ejecutan los siguientes comandos:

systemctl start docker.service

docker start postgresql-tfg-uoc

docker start pgadmin4-tfg-uoc

Con la opción *stop* se pueden parar los contenedores.

docker stop pgadmin4-tfg-uoc

docker stop postgresql-tfg-uoc

Finalmente, antes de crear los *tablespaces,* hay que crear los directorios que se les configura a cada uno, para ello se accede mediante la terminal al contenedor Docker donde se ejecuta PostgreSQL; desde el mismo se crean los directorios –incluido el dedicado al modelo estadístico– y se establece al usuario 'postgres' como propietario. Se puede observar que se crean en un directorio paralelo al que utiliza PostgreSQL por defecto para los datos. Ambos directorios residen en el mapeado en el sistema de archivos del anfitrión. Por otro lado, se debe realizar desde dentro porque el usuario 'postgres' fuera no existe. A continuación se muestra el orden en que se ejecutan los comandos de creación, incluido el *prompt* para marcar la diferencia entre uno y otro entorno de ejecución.

[root@localhost uoc]# *docker exec -it postgresql-tfg-uoc /bin/bash*
root@85c9fd8b0723:/# *mkdir -p /var/lib/postgresql/tablespaces/tb_general*
root@85c9fd8b0723:/# *mkdir -p /var/lib/postgresql/tablespaces/tb_documentos*
root@85c9fd8b0723:/# *mkdir -p /var/lib/postgresql/tablespaces/tb_letras*
root@85c9fd8b0723:/# *mkdir -p /var/lib/postgresql/tablespaces/tb_idx*
root@85c9fd8b0723:/# *mkdir -p /var/lib/postgresql/tablespaces/tb_dwh*
root@85c9fd8b0723:/# *chown -R postgres:postgres /var/lib/postgresql/tablespaces*

Para usar las utilidades del SGBD normalmente se accedería al contenedor, pero esto restringe el acceso al sistema de archivos del anfitrión. Para poder gestionar el servicio como si se tratase de un servicio remoto hay que instalar el paquete postgres96 en el anfitrión, disponible para Red Hat y derivados en la URL yum.postgresql.org/repopackages.php#pg96, el cual solo contiene aplicaciones de cliente. Una vez instalado se puede ejecutar el *script* 00_creacion.sh donde se alude en orden a todos los *scripts* SQL de creación de tablas y procedimientos almacenados.

Nivel virtual

Teniendo en cuenta que la creación y gestión de los *tablespaces* se centra en como el SGBD hace uso del sistema de archivos físico de la maquina en que se ejecuta, se plantea tener las tablas de mayor porcentaje de campos de tipo texto de longitud indeterminada, concretamente Documentos y Letras, en dos *tablespaces* separados entre ello y del resto de la BBDD, que estará en un tercero. Los dos primeros reciben el nombre de dichas tablas y el tercero se ha denominado "global". En el *script* SQL llamado **01_database.sql**, están las sentencias DDL necesarias para crearlos, presuponiendo que existen los directorios a tal efecto y con los permisos adecuados, tal como se detalla en el primer punto del presente anexo..

Con este esquema de distribución de las tablas se pretende aumentar la eficiencia global mediante:

- La extracción de la tabla Documento, con el mayor número de atributos nulos posibles, los cuales además son todos de tipo texto de longitud indeterminada; de hecho, todos son propensos a contener un mínimo de un párrafo.
- La extracción de la tabla Letra que, aunque solo contiene un campo de texto de gran longitud, concentrará la mayor parte de las búsquedas en este tipo de columnas.

De esta forma se paralelizarán los accesos a disco por parte del SGBD en caso de acceder a estas tablas tan pesadas. La ganancia de eficiencia no es tanto la que se introduce en las búsquedas mencionadas, como la que se aporta al rendimiento global mediante el aislamiento del resto de tablas en del tercer *tablespace,* el denominado *tablespace* "global".

Decisiones de diseño

A continuación se detallan las características del diseño físico de cada tabla, y las correspondientes a relaciones de cardinalidad 1:N y N:M se comentan junto con las que guardan mayor relación. Por otro lado, se ordenan igual que en el modelo lógico; esto es así porque, a lo largo de esta descripción, el detalle que se da está centrado en las decisiones tomadas para la codificación SQL y el *script* resultante, el llamado **03_tablas.sql**, necesita de dicho orden para que su ejecución no se vea afectada por las dependencias.

Para definir las restricciones de integridad referencial se tienen ciertas condiciones que se hacen explícitas durante esta fase, pero que ya existían. Por lo general se tiene la restricción de actualización y borrado en las tablas referenciadas, excepto:

- Siempre que se cuente con tablas intermedias para las relaciones de tipo N:M se hará actualización y borrado en cascada cuando desaparezca el registro dominante; por ejemplo, en los casos en que las entidades débiles actúan de forma adjetiva.
- En las tablas Entidad, AgrupaciónAnio, Documento y Obra donde se admite la modificación del identificador, por la misma razón que el cliente solicita poder indicar dicho identificador. Toda columna dependiente de los identificadores de estas tablas será actualizada en cascada, aunque se continua con la restricción en caso de borrado.

Dentro del proceso de implementación de los procedimientos almacenados surgió la necesidad de contar con identificadores para las tablas que representan las relaciones de multiplicidad N:M, con objeto de poder modificarlas, ya que si la identificación se basaba en una clave primaria compuesta se hacía necesario recibir los valores antiguos y los nuevos para poder modificarlas. La decisión se basa en la presunción de que una futura interfaz de usuario solo envíe el nuevo par que representa el vínculo entre tablas o registros; de esta manera, enviará el nuevo vinculo y el identificador sintético del que era el par anterior.

Para todos los identificadores sintéticos autogenerados, los comentados y los ya existentes, se crean estructuras de tipo secuencia con objeto de poder gestionarlas posteriormente; además, de esta forma se fuerza que sigan la misma estructura de nombres que se le esta dando al resto de componentes lógicos –PostgreSQL crearía secuencias para los tipos *serial* igualmente–. Por otro lado, los identificadores basados en secuencias declaradas explícitamente admiten el establecimiento manual del identificador; razón por la que, donde se admite la introducción de los mismos por parte del usuario, se inicializan las secuencias con un valor alto.

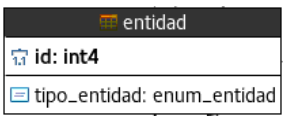
En lo referente a restricciones posibles en el DDL, se tienen las restricciones de integridad referencial ya indicadas en el diseño lógico y se incluyen las relativas a los campos; para los tipos numéricos se controlan los intervalos y para los alfanuméricos que no admiten nulos se controla que no contengan cadenas vacías –cosa que PostgreSQL admite como valor no nulo–.

Como pautas para la denominación de tablas, columnas y otras estructuras se tiene que:

- Todas las denominaciones utilizadas hasta ahora se convierten a minúsculas y, en los casos donde se componen de varias palabras se utiliza un guión bajo como separador. Esto se decide porque para usar en PostgreSQL la nomenclatura Camel y Pascal, como se viene haciendo, hay que establecer todos los nombres entrecomillados; lo cual complicará el uso y administración de la BBDD, así como resultará en un código más ilegible.
- Se utilizan como prefijos las partículas: ‘enum_’ para las enumeraciones, ‘seq_’ en las secuencias, ‘pk_’ para las claves primarias, ‘fk_’ para las externas, ‘uq_’ en las restricciones de unicidad, ‘chk_’ en las restricciones de tipo *check* a nivel de columna e ‘idx_’ para índices.
- Por último, las estructuras recién detalladas se denominan mediante el prefijo correspondiente junto con el nombre de la tabla seguido del nombre de las columnas involucradas.

En cuanto a los gráficos en que se basa la siguiente descripción, los genera automáticamente la aplicación DBeaver. Simplemente configurándole la conexión al servidor PostgreSQL la aplicación genera el gráfico completo, incluidas las relaciones de clave externa; para esto, previamente se han creado las tablas. DBeaver, en cuanto al diseño de los gráficos, define dos zonas: la superior contiene las claves primarias en negrita y en la inferior está el resto de los atributos, indica en cursiva los que admiten valores nulos y muestra el tipo específico junto al atributo. Se procede a comentar las tablas.

Entidad

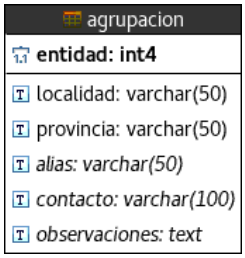


El identificador hace uso de una estructura de tipo secuencia para generar los valores ordenados automáticamente; no obstante, de recibir alguno lo admitirá.

Como ventaja se tiene que las estructuras secuencia pueden ser actualizadas en cuanto al próximo valor a asignar y al número de unidades que incrementa.

Luego, desde este punto aclarar que todos los campos de tipo enumeración son obligatorios; para posibilitarlo, en los modelos anteriores, se les propusieron campos de tipo “se desconoce”. En ciertos casos podría flexibilizarse al respecto, pero para los atributos discriminantes de herencia es vital que sean cumplimentados; no obstante, esto es parte de la operativa interna de la BBDD.

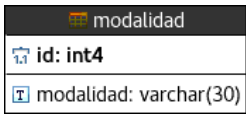
Agrupacion



En este caso el identificador, entidad, se obtiene al crear un registro en la anterior; por tanto, no hace uso de secuencias ni del tipo de dato *serial* de PostgreSQL, que es análogo a dicha estructura.

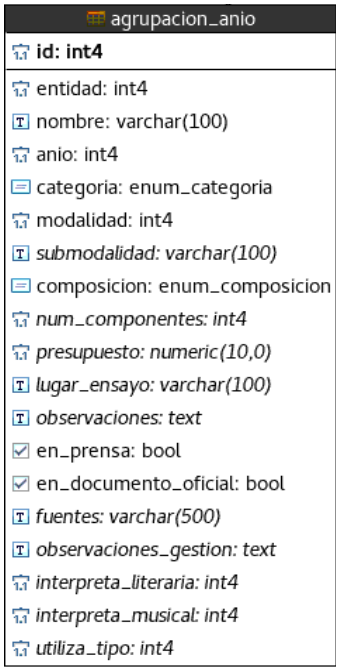
El campo alias tiene una restricción de tipo *unique.*

Modalidad



El campo modalidad tiene una restricción de tipo *unique.*

AgrupacionAnio



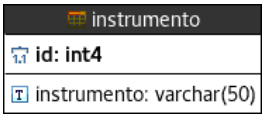
Se crean las enumeraciones necesarias, llamadas enum_categoria y enum_composicion.

Se le incluye el identificador propio, puesto que no va a ser compartido por varias entidades. Además, se le crea una secuencia inicializada a alto valor para admitir la inserción por parte del usuario.

La clave externa es entidad para la herencia y modalidad a modo adjetivo; además, se tienen con restricción de unicidad a nombre y entidad, nombre.

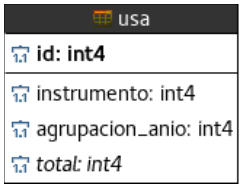
Se modelan restricciones de tipo *check* para los numéricos, principalmente porque son todos naturales; más, en el caso del año, que tenga 4 dígitos.

Instrumento



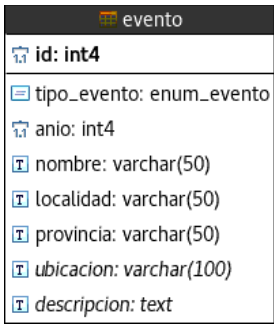
El campo instrumento tiene una restricción de tipo *unique.*

usa



Con la combinación de las claves externas definida como *unique*, por haberle introducido el identificador sintético; y, en cuanto al campo total, es establece que sea por defecto 0, más la restricción de que no sea negativo.

Evento



El identificador hace uso de una estructura de tipo secuencia, al igual que Entidad.

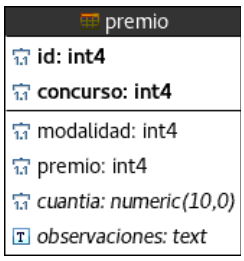
Se crea una restricción de tipo *unique* sobre anio, nombre y localidad.

Y, de nuevo, se controla que el año tenga cuatro dígitos.

Concurso

En este caso el identificador, evento, se obtiene al crear un registro en la anterior que, a su vez, es clave foránea.

Premio

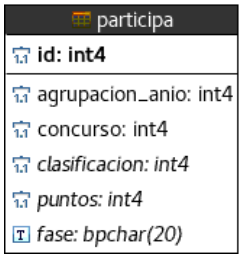


Se le añade un identificador propio, y la clave primaria es compuesta del mismo más la clave externa, la cual representa la relación de composición respecto de Concurso.

Se crea una restricción de unicidad sobre el conjunto concurso, modalidad y premio.

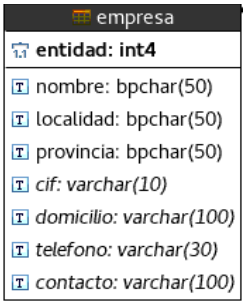
También se crea una restricción para controlar que los numéricos sean positivos.

participa



La clave primaria es compuesta de las externas y se añaden restricciones de tipo *check* sobre los numéricos.

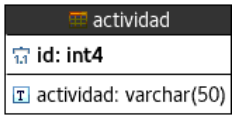
Empresa



La clave primaria la hereda de Entidad, la cual es generada por la secuencia asociada a la misma, y es clave externa para el modelado de la herencia.

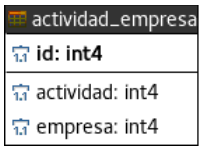
Además, tiene restricciones de unicidad sobre las que serían sus claves candidatas nombre, localidad y provincia más el cif, que admite nulos.

Actividad



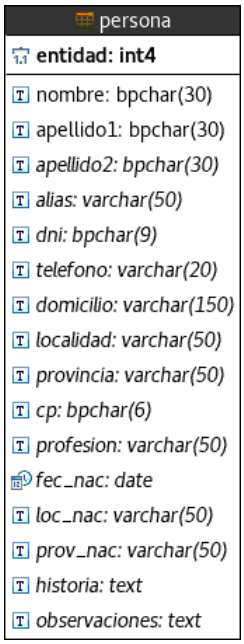
Como el resto de tablas de tipo *lookup*, el campo actividad es de tipo *unique.*

ActividadEmpresa



El par de claves externas se define como clave única, por la necesidad de crear el identificador sintético para la edición de la tabla.

Persona

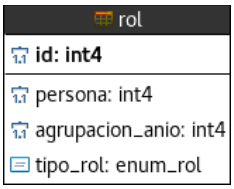


En este caso el identificador, entidad, se obtiene al crear un registro en dicha relación y es clave foránea a la misma.

Por otro lado, se crea una restricción de tipo *unique* sobre el DNI, la cual admitirá nulos que es condición de este campo.

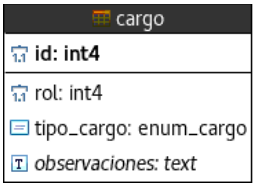
En cuanto a la longitud de los tipos carácter se entienden suficiente.

rol



Se incluye el identificador y se define una restricción de unicidad sobre el conjunto persona, agrupacion_anio y tipo_rol.

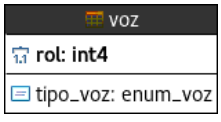
Cargo



Tiene como restricción de unicidad el identificador sintético de rol y el tipo de cargo, y se le añade una clave primaria paralela.

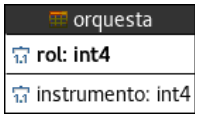
Para facilitar la gestión de modificaciones en el tipo de cargo de una persona en concreto se ha introducido un identificador en esta tabla, ya que para cambiar el tipo de cargo había que poder identificar el par adecuado sin conocer el valor anterior del cargo; es decir, no se puede cambiar parte de la clave primaria de una relación si no se la conoce completa.

Voz



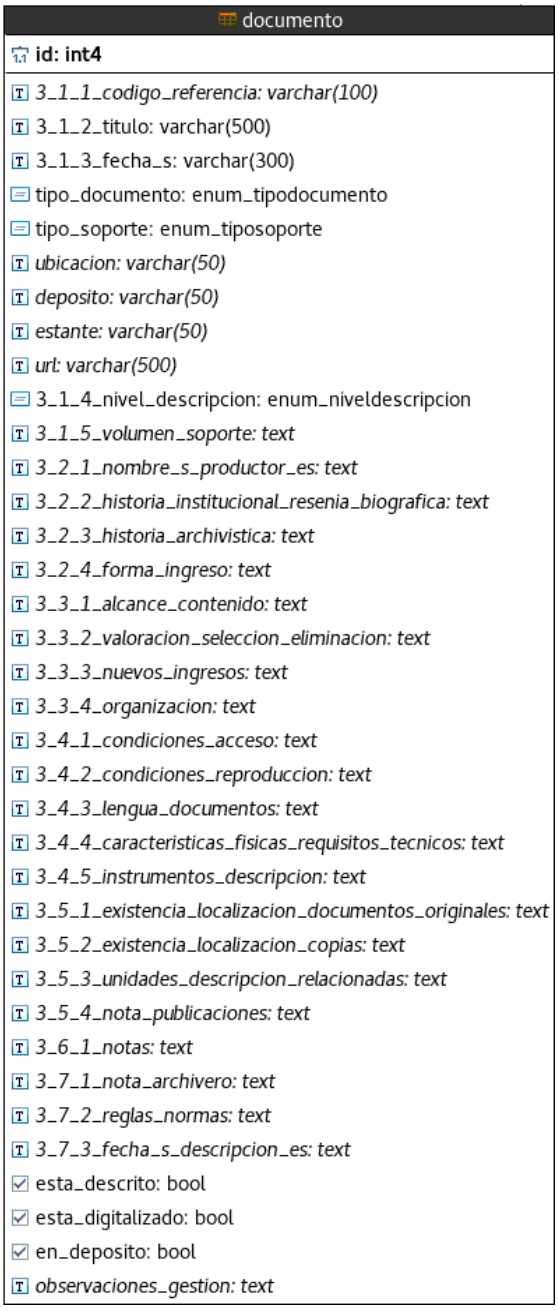
Se utiliza el identificador heredado de rol como clave primaria y externa al mismo. No necesita admitir más de un registro por cada uno en rol.

Orquesta



Similar a los anteriores en cuanto a la clave primaria y la herencia; luego tiene el instrumento como clave foránea a dicha entidad.

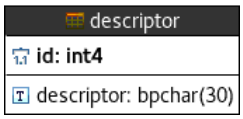
Documento



En este caso la clave primaria hace uso de una secuencia dedicada, inicializada en 100000 con objeto de admitir las signaturas que los usuarios del Aula hayan ido asignando a los documentos a lo largo de los últimos años.

Al campo 3.1.1 Código de Referencia de la ISAD(G) se asignan 100 caracteres por la complejidad que se ha descrito en los modelos anteriores. Además, tiene una restricción de tipo *unique.*

Descriptor



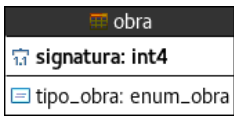
Se crea una restricción de unicidad sobre el atributo descriptor.

DepositoLegal

Su clave primaria es necesariamente compuesta por la relación de dependencia de multiplicidad 1:M; sin embargo, tal cual se expuso, requiere una restricción de tipo *unique* sobre el campo de deposito_legal a nivel de tabla.

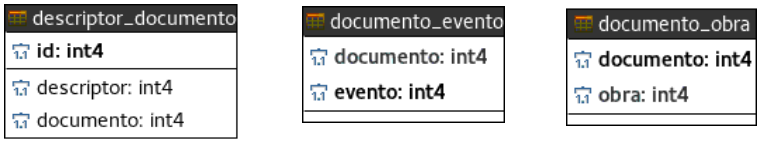
Por otro lado, se añade un identificado propio por no incluir a la clave primaria un campo de tipo texto.

Obra



Al igual que Entidad, para controlar la unicidad de los identificadores de los subtipos de Obra se hace uso de una secuencia; además de imponer el valor del discriminante tipo_obra.

DescriptorDocumento, **DocumentoEvento** y **DocumentoObra**,

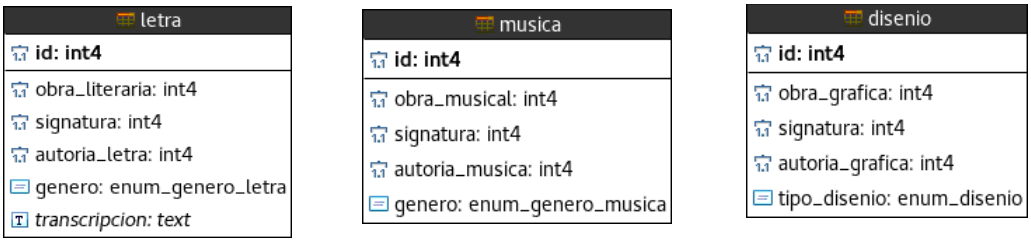


Por ser todas ellas relaciones de cardinalidad N:M se describen en conjunto. Tienen como clave primaria el compuesto de las dos externas que involucran a Documento con Descriptor, Evento y Obra. En el caso concreto de Descriptor, por la posibilidad de modificar los descriptores asociados a los documentos, se añade el identificador paralelo y se define el par de claves foráneas como clave única.

Literaria, **Musical** y **Grafica**

Por ser todas subtipos de Obra se diseñan de igual manera, y se comentan en conjunto. Las claves primarias las heredan, generada por la secuencia asociada la signatura de Obra, y son clave externa para el modelado de la herencia.

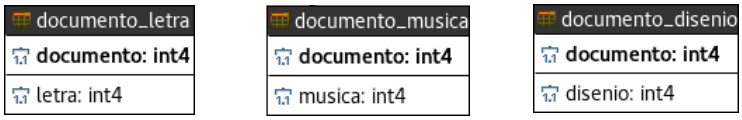
Letra, **Musica** y **Disenio**



Aun teniendo diferencias semánticas, estos tres componentes de las Obras implican las mismas decisiones de diseño. Se introduce un identificador sintético para facilitar las referencias desde otras tablas; por otro lado, como parte del modelado de la relación de composición entre estas y Literaria, Musical y Grafica, respectivamente, se les crean restricciones de tipo *unique* sobre el par obra y signatura.

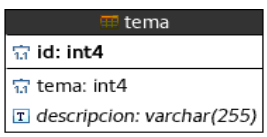
Por otro lado, respecto al valor de signatura, se establece el tipo *integer* sin secuencia asociada porque se gestionará la secuencialidad por cada obra diferente; lo cual, como se desarrolló en el modelo conceptual, se gestionará mediante los procedimientos almacenados de alta y modificación. Cabe señalar que por esta misma razón ha sido necesario añadir un segundo identificador paralelo al compuesto por el de la signatura de la obra, como compuesto, y la propia del componente.

DocumentoLetra, **DocumentoMusica** y **DocumentoDisenio**



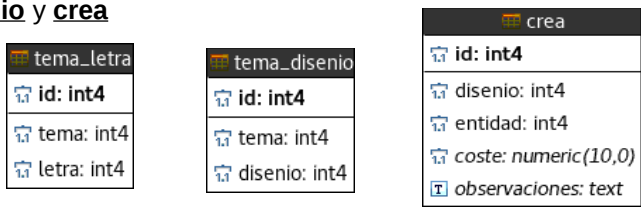
Tal como se hizo anteriormente, se comentan las estas relaciones que vinculan las tres tablas recién comentadas respecto de Documento; la particularidad en este caso es que modelan relaciones de multiplicidad 1:N, por tanto la clave primaria no es compuesta. En este caso se controla que cada documento pueda apuntar a un único componente de Obra.

Tema



Al igual que el resto de las tablas de tipo adjetivo, solo necesita la restricción de unicidad sobre el campo tema, inicialmente propuesto como clave candidata.

TemaLetra, **TemaDisenio** y **crea**



Por último, se dan estas tres tablas intermedias para el diseño de relaciones de cardinalidad N:M; nuevamente, por haber incluido el identificador sintético no se tiene sus claves primarias compuestas por las foráneas, sino que se define el par de tipo único.

Índices

Los índices los crea PostgreSQL implícitamente sobre las claves primarias y las únicas; para el presente proyecto, de forma explícita, se crean sobre: las claves externas, las columnas dedicadas a fechas, concretamente años, y las que serán foco de búsquedas, las cuales hacen uso en su mayor parte de tipos enumerados. Por otro lado, por ser clave primaria, no se crean índices en las claves externas que modelan las herencias. Por último, en todas las tablas que representan relaciones de cardinalidad N:M ya se cuanta con un índice sobre la clave primaria compuesta, por lo que se desestima uno por cada foránea. Todas las definiciones están incluidas en el *script* **03_indices.sql**.

A continuación se listan los campos objetivo según las premisas indicadas, según la convención *tabla.columna*, luego por la composición que se viene utilizando para las denominaciones de los elementos de la BBDD se omite la indicación acerca del destino.

Claves externas	Campos de búsqueda	Fecha
agrupacion_anio.entidad	persona.apellido1	agrupacion_anio.anio
agrupacionanio.modalidad	persona.alias	evento.anio
agrupacion_anio.interpreta_literaria	agrupacion_anio.categoria	deposito_legal.anio
agrupacion_anio.interpreta_musical	evento.tipo_evento	
agrupacion_anio.utiliza_tipo	cargo.tipo_cargo	
cargo.rol	voz.tipo_voz	
orquesta.instrumento	documento.tipo_documento	
deposito_legal.documento	documento.tipo_soporte	
letra.obra_literaria	letra.genero	
letra.autoria	musica.genero	
documento_letra.letra	disenio.tipo_disenio	
musica.obra_musical		
musica.autoria		
documento_musica.musica		
disenio.obra_grafica		
disenio.autoria		
documento_disenio.disenio		

Excepciones

Para las enumeraciones utilizadas por las superclases de las herencias se omite la creación del índice, teniendo en cuenta principalmente que el valor es meramente informativo, los accesos y búsquedas se hará principalmente en los subtipos y los índices ya existen sobre todas las claves primarias, que son a su vez externas, de los elementos participantes en las generalizaciones.

Respecto a la generalización referente a los roles de persona, si se ha creado un índice en la tabla Cargo porque a esta se le creó una clave primaria sintética que permitiera más de un cargo por persona y agrupación.

Por otro lado, a pesar de lo expuesto, si se crea el indice en el atributo discriminante de Evento teniendo en cuenta que aún no se ha especializado todo lo posible y, además, contiene gran parte de los datos de las que serán sus herederas.

Otro caso obviado es el campo documento.3_1_4_nivel_descripcion, este contendrá un alto porcentaje de repeticiones y es en cierto grado una información accesoria; además, su objetivo está más vinculado a las recomendaciones de la ISAD(G) que a las propias del dominio y necesarias para la usabilidad del sistema final.

Finalmente, se ha desestimado crear un índice en la categoría de los concursos, en este caso prima el año, luego pocos hay en que haya varias categorías.

Procedimientos almacenados

Se procede a detallar los procedimientos almacenados de alta, baja y modificación implementados. Para el desarrollo general se tienen y siguen las siguientes premisas y pautas:

- Se desarrolla un conjunto por cada entidad editable y otro por cada atributo multivaluado, incluidas en estos últimos las relaciones de tipo N:M.
- En cuanto a la gestión de las generalizaciones, en todos los casos donde la superclase de la herencia es abstracta no se crean procedimientos almacenados, se crean los registros previamente a la creación de los registros del subtipo y se indica el tipo de la subclase, lo que evita el posible error humano; luego se hace uso de las secuencias y las funciones dedicadas a las mismas. De esta forma, además se controla que un subtipo no cambie de subtipo dentro de la relación de herencia.
- Para eliminar redundancias en el código se han creado funciones auxiliares asociadas a las entidades Documento y Obra; a raíz de esta reestructuración del código, en todos los casos en que los procedimientos son de uso estrictamente interno se establece como prefijo a los nombre de dichas funciones dos guiones bajos para distinguirlos.
- También como convención de nombres, se dan todas las variables de entrada de las funciones con el prefijo ‘in_’ y, en los casos necesarios, se hace uso de variables ‘auxInteger’, ‘auxCharacter’ y ‘was’; las dos primeras se utilizan en función del tipo de datos a tratar, la última se utiliza para vectores de tipo registro en las ocasiones en que se utilizan los valores anteriores del registro en edición.
- Todos los procedimientos de baja y modificación requieren de recibir el identificador del registro a gestionar, o todos los implicados cuando se trata de conceptos interdependientes.
- En lo relativo a la gestión de errores, siempre se devuelve una cadena de caracteres con el valor ‘OK’ o el error en concreto; y, en la mayoría de los casos, solo se validan las restricciones semánticas, las no declaradas como parte de la creación de las tablas y las relaciones. Lo que se valida en todos los casos de modificación y baja es la recepción de los identificadores; además, cuando son claves compuestas, que exista la combinación recibida. No se validan cuestiones que ya controla el SGBD mediante restricciones SQL definidas mediante DDL; como los campos que no admiten valores nulos, los de tipo único, los que contienen un *check* o los que tienen restringida la eliminación por estar referenciados desde otras tablas; para todos estos casos se captura la excepción y se muestra el mensaje sin procesar.
- Dicho esto, lo que se controla mediante los procedimientos almacenados son las restricciones semánticas, o de usuario, y las relaciones que son indirectas; por ejemplo, la que se da entre la modalidad de los premios y la de las agrupaciones, que está a un par de tablas.
- Los bloques BEGIN/END de PostgreSQL implican la atomicidad en la ejecución del procedimiento, por lo que solo se anidan estos bloques para el control de muy bajo nivel de las operaciones de tipo CRUD; esto, a su vez, es necesario para controlar cuestiones que no son afectadas por los procesos de *rollback*, donde las secuencias no vuelven al valor previo, por ejemplo. Así mismo, cuando se llama a una de las funciones descritas como internas, en lugar de gestionarlas con el bloque BEGIN/END se evalúa el valor de retorno; en caso de ser ‘OK’ se continúa y en caso contrario se lanza una excepción con el mensaje recibido.
- Finalmente, se da una restricción de nivel de usuario a gestionar mediante las siguientes funciones. Por la labor documental ya realizada por el cliente se tiene el requisito de admitir la inserción y edición de los identificadores de las agrupaciones, como entidad y las anuales, de las obras, sus

componentes y los documentos. Por esta razón las secuencias de estos identificadores se inicializaban en valores altos y, en este punto, se hacen necesarias funciones dedicadas a la modificación de los mismos.

Con base en todo esto se procede a detallar la implementación de los procedimientos almacenados tomando como estructura la utilizada para la denominación de los *scripts* SQL.

05_SP_01_tablasLookup.sql

En primer lugar se desarrollaron los dedicados a las tablas de tipo *lookup,* principalmente por la simplicidad y la independencia de estas tablas.

- Alta_Modalidad, Modifica_Modalidad y Baja_Modalidad
- Alta_Descriptor, Modifica_Descriptor y Baja_Descriptor
- Alta_Tema, Modifica_Tema y Baja_Tema
- Alta_Instrumento, Modifica_Instrumento y Baja_Instrumento
- Alta_Actividad, Modifica_Actividad y Baja_Actividad

Para todos estos, en el alta se verifica que el valor no sea una cadena vacía –aunque lo controla la restricción *check* asociada a la columna– y que no exista considerando la capitalización del mismo. Los procedimientos dedicados a la modificación de estas tablas verifican que se ha recibido el identificador del registro a editar y que el campo único que tienen no reciba una cadena vacía. Como se ha indicado respecto a las premisas generales de control de errores, para las bajas no se verifica que se estén utilizando los registros a eliminar desde las relaciones en que participan, cabe señalar que esto ya es controlado por la restricción *on delete restrict* de las restricciones de clave externa.

En el caso de Modifica_Tema se reciben tres parámetros, el identificador y los dos campos que tiene la tabla, en el se hace el primer uso de la función *coalesce*() de PostgreSQL. Para no volver a repetir la funcionalidad en lo siguiente, esta función se utiliza para asignar a los campos el valor recibido como parámetro de la función y, en su defecto, el que ya estaba –previamente cargado en la variable *was* que se comentaba más arriba–. Mediante el uso de esta función se obtiene un patrón homogéneo para todas sentencias UPDATE, las cuales serán ejecutadas previa verificación de los parámetros de entrada.

También se da por hecho que toda modificación requiere de la recepción del identificador del registro en lo que sigue, de hecho en algunos casos de todos los identificadores participes de la modificación; por ejemplo, para los componentes o las obras se requiere del identificador, la signatura y de uno de los documentos asociados, el cual será afectado si procede.

05_SP_02_Documento.sql

Seguidamente, por el uso que se hace desde varios conceptos de la tabla Documento, se procede a desarrollar el conjunto de procedimientos almacenados que gestionan el contenido de dicha tabla. En este caso los procedimientos internos son todos utilizados desde las funciones dedicadas a Obra y Evento, incluso desde los centrados en los componentes de obras; es decir, desde todos los lugares en que se hace necesaria la actualización de la tabla Documento se utilizan estas tres funciones.

La gestión de los documentos como vía de descripción de obras y sus componentes lleva a tomar ciertas decisiones de uso que, a su vez, tienen cierto impacto en la implementación de nivel técnico. Al igual que en los casos donde se hace uso de las tablas de *lookup* con sentido adjetivo, se restringe la modificación de los registros solo en el campo que actúa como adjetivo; en este caso, se impide que los documentos puedan cambiar sus asociaciones entre los posibles conceptos a los que se vinculan –eventos, obras, letras, músicas y diseños–. Esto, a nivel técnico, solo implica omitir la definición de procedimientos almacenados de modificación sobre las tablas que relacionan en 1:N y N:M los conceptos mencionados. Lo curioso, que debe hacerse notar, es que se ha hecho uso de nombres de función que pueden llevar a confusión, pero esto se detallará donde sean utilizados.

Por otro lado, continuando con el contenido del *script* dedicado a Documento, como regla general en el uso de estas funciones internas, todos los que crean documentos gestionan la secuencia de identificadores de Documento y componen su Código de Referencia antes de la llamada; de forma que se controla la composición de signaturas según la CDU en función del tipo de entidad. En caso de recibir este valor se debe controlar que contenga las signaturas, en caso de lanzar el error se indican cuales se espera recibir.

Todos los campos con valor por defecto se gestionan en caso de recibir nulos, para evitar excepciones del SGBD en casos en que se diseñan los atributos con un valor inicial; por ejemplo, en el caso de los booleanos. En otros casos, respecto a las columnas de tipo texto, se deja actuar a las restricciones del SGBD, pero en este se han gestionado por tratarse de comunicación entre las funciones implementadas.

- __Alta_Documento, __Modifica_Documento y __Baja_Documento
  - En las altas y modificaciones se controla que de ser verdadero algún booleano, de los dedicados a gestión de documentos, los campos url o los relativos a la ubicación física contengan algún dato. Además, se evalúa el nivel de descripción en función del tipo de documento.
  - Para crear un documento se requiere indicar un descriptor, esto solo puede ser controlado de forma externa a las tablas; es decir, que en la tabla DescriptorDocumento haya un mínimo de una referencia por cada registro de la tabla Documento. Por tanto, una vez insertado el documento se crea el vínculo entre este y un primer descriptor.

    En la modificación del documento ya no interviene este concepto, para gestionar los descriptores se debe acceder mediante procedimientos específicos dedicados a la actualización de la tabla DescriptorDocumento; de hecho, mediante el de alta en dicha tabla se ha hecho la primera inserción.
  - La eliminación controlará que no se eliminen documentos de todas las relaciones donde participan, de quedar solo una referencia en alguna de las tablas que implementan las relaciones, será la obra o componente el que haya que eliminar; es decir, se entenderá como el documento mínimo asociado necesario para la descripción del involucrado. Cabe decir que para esto, el que llama a la función __Baja_Documento ya habrá gestionado la baja en su tabla de relación específica, por esto la igualdad se centra en 0.

    Si va a proceder con la eliminación, antes verifica que no haya documento tangible asociado al registro descriptor, ya sea electrónico o físico. Es decir, que tipoSoporte sea igual a “No disponible”, más que estaDigitalizado y enDeposito sean falsos.
- Modifica_ID_Documento
  - Es función se implementa específicamente para cambiar el identificador de los documentos, puesto que es un valor que el usuario puede aportar hay que contar con que pueda equivocarse y deba rectificarlo.
  - Además de actualizar el identificador, si es menor al intervalo gestionado por la secuencia, gestiona el valor de 3_1_1_codigo_referencia.
- Alta_DepositoLegal_Documento, Modifica_DepositoLegal_Documento y Baja_DepositoLegal_Documento
  - Estos tres se centran en la tabla dependiente, que representa al atributo multivaluado DepositoLegal. Para el alta se verifica que exista documento tangible indicado en el registro de Documento al que se pretende vincular el nuevo de DepositoLegal. El resto de validaciones están explicitas en el DDL de declaración de la tabla.
  - La modificación y la baja solo verifican la coherencia entre el identificador de esta tabla y del documento implicado.
- Alta_Descriptor_Documento, Modifica_Descriptor_Documento y Baja_Descriptor_Documento
  - Estas son las funciones dedicadas a gestionar la tabla que relaciona los documentos con sus descriptores. Simplemente verifican que no exista la relación del par indicado, para el alta, y lo contrario para la modificación; por otro lado, para la eliminación, verifica que no sea el último registro que alude al documento, por su carácter adjetivo obligatorio.

05_SP_03_Entidad.sql

Este *script* contiene los procedimientos almacenados relativos a la jerarquía que parte de Entidad, la cual solo tiene ninguno explícito en su nivel.

- Alta_Empresa, Modifica_Empresa y Baja_Empresa
  - En el caso del alta se reciben todos los atributos de dicha tabla más el identificador de la actividad que la misma desarrolla, el cual se impone así como obligatorio; no obstante, en la edición posterior, mediante la función Modifica_Empresa, no se editará dicho valor por ser multivaluado; es decir, una vez creada la empresa para añadirle actividades se utilizarán las tres que dedicadas a este concepto adjetivo.
  - Para la modificación, como en todos los casos, se requiere el identificador de la entidad a la que pertenece para poder editar; pero, este no se editará mediante este procedimiento almacenado. Obviamente sería imposible puesto que de recibir otro se editaría el registro indicado por el otro.

Solo la entidad de tipo Agrupacion admite el cambio de identificador, por restricciones del cliente.

- En el caso de la baja de las empresas todas las restricciones están dentro del DDL de las tablas, lo único que se gestiona explícitamente en caso de permitirse la eliminación es el consecuente borrado de la Entidad a la que apuntaba.

- Alta_Actividad_Empresa, Modifica_Actividad_Empresa y Baja_Actividad_Empresa
  - Para las altas no se hace ninguna verificación, ya que todas las restricciones están definidas por el DDL de la tabla ActividadEmpresa.
  - Para la modificación, solo se admite la actualización del campo actividad, teniendo en cuenta que no tiene sentido cambiar una actividad de empresa, sino a las empresas cambiarles las actividades. Para esto, si se recibe un identificador de empresa distinto al del registro indicado, se lanza un error.
  - En el caso de las bajas, solo se eliminan relaciones de tipo adjetivo en caso de existir varias, de ser la última se lanza una excepción.

  Este es un caso de relación entre registros con multiplicidad N:M donde se ha introducido un identificador sintético que permite alterar la combinación de forma unívoca; obviamente, no se da la gestión de la secuencia para estos casos.
- Alta_Persona, Modifica_Persona y Baja_Persona
  - Para las altas se implementa, mediante este procedimiento almacenado, la restricción de clave alternativa, la cual no se ha podido generar como parte de la creación de las tablas: de no existir segundo apellido o DNI, debe existir el domicilio con localidad. Esto se verifica también para la función de modificación de la entidad. Por otro lado, se verifica que el DNI, de ser recibido, sea de longitud 0, 8 o 9.
  - Para la baja, si existe el registro con el identificador recibido se intenta, si es eliminada, porque no participe de ninguna relación, se borra seguidamente el registro en Entidad.

    Por el carácter atómico de los procedimientos, si la eliminación de la tabla Entidad genera algún error, no se eliminará el registro solicitado en Persona. De esta forma, si la persona tiene rol de autoría en su carácter de Entidad, se evitará su eliminación.
- Alta_Agrupacion, Modifica_Agrupacion y Baja_Agrupacion
  - Se gestiona el identificador a partir de la secuencia asociada a la tabla Entidad. Por el contrario que en los casos anteriores, este concepto admite la recepción de un identificador dado por el usuario, por tanto se gestiona de no ser recibido; más, como se indicó al principio de este apartado, en caso de error en la inserción se debe restar la unidad incrementada a la secuencia, si es que la misma fue utilizada.
  - La actualización es sencilla, no hay verificaciones más allá de la existencia del identificador recibido, lo cual es un requisito mínimo en todas las funciones de modificación, y el borrado es similar a los dos subtipos anteriores.
- Modifica_ID_Agrupacion
  - Como se ha indicado anteriormente se hace necesaria una vía por la que modificar el identificador de entidad de las agrupaciones, no de las personas ni las empresas que hasta ahora el cliente no había contemplado; no obstante, para que funcione la actualización en cascada a través de todas las referencias a la Entidad en cuestión, se edita esta última tabla, aunque la consulta previa de existencia se hace en Persona, para verificar que efectivamente de ese tipo de entidad se trata. Como en el resto de estos casos, durante la actualización de los registros no se edita el identificador del mismo, unicamente es requerido para localizar el registro del que se solicita la edición.

  En todos estos casos se hace gestión de la secuencia que genera los identificadores de la tabla Entidad. La complejidad de esta gestión radica en decrementar la secuencia durante el manejo de las excepciones que se dan una vez hecho uso de dicha secuencia, razón por la que la acción se da en más de un lugar; siempre que no se obtenga el valor de la secuencia al principio del procedimiento habrá que decrementarla atendiendo a esta condición, puesto que a veces se lanzan errores antes de utilizar la secuencia no debe decrementarse en la última, o más externa, captura de excepciones.

05_SP_04_Evento.sql

De nuevo en este *script* se contienen todas las funciones que actualizan las tablas participantes en la jerarquía de Evento y todas las directamente vinculadas.

- Alta_Evento, Modifica_Evento y Baja_Evento
  - El alta evaluá la no existencia de la que es clave única atendiendo a la capitalización de los valores indicados.
  - Luego la de modificación y baja se restringen a los eventos que no son de tipo “Concurso”, estos deben gestionarse mediante el siguiente conjunto de procedimientos almacenados.
- Alta_Concurso, Modifica_Concurso y Baja_Concurso
  - Se hace uso de la anterior en cuanto al alta y de la secuencia asociada al identificador de Evento; aunque, en este caso, no se gestiona el decremento en caso de errores.
  - Para las modificaciones, por tratarse de una herencia en la que los atributos están repartidos entre generales y específicos, se cargan dos variables de tipo registro, una para editar la tabla Evento y otra para los valores en Concurso.

    Por otro lado, de tener agrupaciones ya inscritas, no se admite cambiar la categoría ni el año, lo que crearía una inconsistencia semántica en la asociación.
  - La baja es sencilla, está gestionada completamente por las restricciones de integridad referencial declaradas por DDL.
- Alta_Premio_Concurso, Modifica_Premio_Concurso y Baja_Premio_Concurso
  - El alta es sencilla, para la modificación y la baja se verifica que no hay ninguna agrupación asociada al mismo mediante Concurso y por la Modalidad, teniendo en cuenta que esta información es indirecta; es decir, se podría eliminar un Premio que aporta información siempre que no se elimine el Concurso, el cual si será bloqueado por la existencia de un registro que lo referencie desde participa.
  - Por otro lado, la modificación lo que se verifica según lo anterior es que no se pueda modificar la modalidad, que es el concepto vinculante entre la agrupación y el premio. En caso de admitirse la actualización del registro, no se edita la referencia a Concurso; de esta forma, se impide que un premio cambie de concurso, habría que dar la baja y el alta en el adecuado.
- Doc_Nuevo_A_Evento y Modifica_Documento_Evento,
  - Gestionan registros de Documento asociados a Evento. En el primero se crea y asocia un registro al evento indicado, en el segundo se edita un registro existente de Documento que está ya asociado al evento que se tenga activo –o visualizado si se considera la futura interfaz–.

    En el caso de Modifica_Documento_Evento –como todas las funciones que seguirán este patrón de nombre–, aunque contiene como parte de la firma el nombre de la tabla que implementa la relación N:M, Documento_Evento, no la edita. De hecho, controla que exista la relación entre el evento y el documento indicados antes de proceder con la actualización en Documento; aunque, desde una interfaz, sería extraño enviar estos identificadores tergiversados. Cabe señalar que si se pudieran alterar estos identificadores, el documento editado no sería el activo.
  - En el caso de creación de documentos mediante la primera de las funciones, Doc_Nuevo_A_Evento, se gestiona el identificador propio y el Código de Referencia de la ISAD(G) en caso de no recibirlos. De haberlos recibido se verifica su coherencia según todas las premisas ya descritas al respecto. Por el contrario que en el caso de las obras, no se hace uso de la función externa para la verificación y construcción, porque los eventos siguen un patron parcialmente diferente.

    Luego se hace uso de la función interna __Alta_Documento, descrita anteriormente; por último, se crea el registro en la tabla que gestiona la vinculación de tipo N:M, DocumentoEvento, sin más verificaciones que las que hará el SGBD. De haber error en alguna de estas dos últimas acciones, y de haberse utilizado, se decrementa la secuencia del identificador de Documento.
- Doc_Existente_A_Evento y Baja_Documento_Evento
  - Ambas centradas en la la tabla que implementa la relación N:M. En el primer caso no hay más verificaciones que las del SGBD, como se acaba de indicar en el detalle anterior.

- Para la baja, al igual que en el resto, se verifica que el par de identificadores existe en la tabla DocumentoEvento, luego se llama a la función __Baja_Documento y se gestionan los diversos mensajes que esta puede devolver –si devuelve alguno es porque se trata del último vínculo del documento dentro de la BBDD–. Si es el último y solo se asocia al evento indicado se informará de tener algo tangible descrito, en el resto de casos se da por eliminada la relación al evento.

05_SP_05_AgrupacionAnio.sql

Seguidamente, en el *script*, se incluyen todas las funciones relativas a las agrupaciones de carácter anual: las relaciones con instrumentos a través de usa; las relaciones con personas a través de rol, aunque mediante la gestión de los subtipos de dicha entidad; y, por último, la relación con los concursos a través de participa.

- Alta_AgrupacionAnio, Modifica_AgrupacionAnio, Baja_AgrupacionAnio
  - En las altas se establecen los campos booleanos a false si se recibe *null.* No se hace en la actualización de los registros porque por el uso de *coalesce* se quedará el valor no nulo de los que se gestionan en la actualización; pero, principalmente, porque la inserción únicamente debe haber sido realizada mediante el procedimiento Alta_AgrupacionAnio.
  - Luego, en las altas y modificaciones se controla que:
    - El nombre indicado para la agrupación no exista ya en la tabla, considerando la capitalización del valor.
    - De ser verdadero algún booleano, de los dedicados a gestión de información, el campo fuentes contenga algún dato. Y el tipo de soporte debe contener algún valor distinto a “No disponible”.
    - Y que de recibir el identificador de una obra, del tipo que sea, su título contenga el nombre de la agrupación. Para esto se cuenta con una función externa, dentro de este mismo *script*, implementada para reutilización y modularización general del código.
  - Si no se recibe el identificador de Agrupacion se debe recibir localidad y provincia, con esto se creará una nueva agrupación con carácter de entidad y se utilizará el identificador recién generado. Esto se diseña a modo de facilidad para el usuario, se admite la recepción de localidad y provincia para crear una agrupación en caso de no ser esta última indicada.
  - Por último se hace la inserción o la actualización del registro, con *castings* al tipo ENUM que proceda en cada caso. En caso de error se restará una unidad a la secuencia de Entidad si fue utilizada –para lo que se utiliza una variable auxiliar a modo de *flag*–, ya que PostgreSQL no lo hace como parte del *rollback* del bloque BEGIN/END como ya se ha expuesto.
  - En el caso de las modificaciones, además de las verificaciones dadas, se comprueba que de intentar cambiar la categoría o el año la agrupación no esté inscrita en ningún concurso, a través de la tabla participa. Este requisito es puramente semántico, las agrupaciones solo pueden inscribirse en concursos de su categoría y año.
  - Para la eliminación, hay que verificar que no tienen referencias a Obra ni a premios o participación; respecto al primer caso, porque la restricción de integridad referencial se daría en caso de intentar eliminar la Obra, en caso contrario la Obra se queda existente sin interprete, por lo que hay que controlar la eliminación del interprete en el mismo. Para la segunda restricción, la controla el SGBD por integridad referencial.

    En cuanto a los instrumentos vinculados mediante la tabla usa desaparecerán en cascada si se elimina un registro de la tabla Agrupacion_Anio.
- Modifica_ID_AgrupacionAnio
  - Como se ha indicado anteriormente se hace necesaria una vía por la que modificar el identificador de las agrupaciones, al igual que en su carácter de Entidad y en los documentos. Igual que en los casos análogos se verifica que el identificador no sea de los gestionados por la secuencia.
- Baja_Literaria_AgrupacionAnio, Baja_Musical_AgrupacionAnio y Baja_Tipo_AgrupacionAnio
  - Estas funciones se hacen necesarias porque no se puede eliminar una agrupación si tiene alguna obra asociada; más, por la imposición de actualizar los contenidos únicamente a través de los procedimientos almacenados, es imposible establecer a *null* uno de los valores mediante los procedimientos de modificación debido al uso de l función *coalesce* de PostgreSQL.

- Al igual que en la eliminación de la agrupación de carácter anual, se requiere recibir el identificador de los dos niveles dela agrupación, el de su entidad y el específico.

- __Verifica_AgupacionAnio_Obra
  - Se define este procedimiento de uso interno que se utilizará en los casos asignación de obras en los atributos interpretaLetra, interpretaMusica y utilizaTipo para verificar que el nombre de la agrupación está contenido en el título de la Obra.
  - La función toma el título y el tipo de obra según el identificador de la misma, que es el valor que reciben los procedimientos almacenados Alta_AgrupacionAnio y Modifica_AgrupacionAnio, y verifica la corrección respecto al tipo de obra y el nombre de agrupación que también recibe como parámetros. En caso de incorrección informa de cual se trata.
  - No se verifica la concordancia entre los años de Obra y AgrupacionAnio porque suele darse la reinterpretación de las mismas a modo de antología –aunque esto suele hacerlo la misma agrupación, dentro del concepto de entidad–.
- Alta_Agrupacion_Usa_Instrumento, Modifica_Agrupacion_Usa_Instrumento y Baja_Agrupacion_Usa_Instrumento
  - Como se ha ido exponiendo a lo largo de este diseño y del lógico, no se coordinará el contenido de esta relación de instrumentos con la de el rol Orquesta. Aunque en este caso se pueda sugerir como mínimo, se deja al diseño de la futura interfaz esta cuestión.
  - Los tres procedimientos acceden a la tabla usa y requieren recibir los identificadores de Instrumento y AgrupacionAnio como única restricción, más el identificador sintético para modificación y baja; luego verifican si existe en usa, para no crearlo o para poder modificarlos o borrarlos, según el procedimiento almacenado.
  - La implementación es sencilla, como en los casos anteriores de tipo adjetivo solo se permite modificar el instrumento, no la agrupación.
- __Alta_Rol
  - Se introduce esta función auxiliar para gestionar desde las siguientes, específicas de los subtipos de rol, las posibilidades de alta en los distintos roles para un par persona-agrupación, vistos desde este nivel de abstracción.
  - Para un par verifica el número de roles ya creados y según el tipo de rol del que se solicita el alta lo creará o no. En el caso de ser de tipo Cargo se admiten varios, pero esto lo gestionan las siguientes.
  - Para los siguientes, se fuerza la necesidad de recibir el identificador del rol a editar, más el de la persona y la agrupación aludidos por el anterior. Por un lado se hace por seguridad en las modificaciones y, por otro, se restringe la posibilidad de intercambiar las personas o agrupaciones en los registros de rol.

    Durante las pruebas se detectó que permitir esto generaba registros huérfanos en rol; por tanto, es una decisión de diseño físico, para simplificar la programación dedicada a la gestión de roles de personas.
- Alta_Cargo_AgrupacionAnio, Modifica_Cargo_AgrupacionAnio y Baja_Cargo_AgrupacionAnio
  - Para las altas y modificaciones, de recibir como cargo 'Presidente', 'Director' o 'Representante legal' se verifica que no exista ya uno para la agrupación indicada.
  - Luego, si durante la creación del rol para el par persona-agrupación se recibe como error '__CARGO__' este procederá a localizar el identificador de rol para el par y creará un registro con el nuevo cargo para dicho par.
  - Tanto las bajas como las altas implican la gestión previa o posterior del registro existente en rol. En el caso de las altas, de haber errores de inserción, se resta la unidad a la secuencia, aunque en este caso no es estrictamente necesario. En el caso de la baja, de recibir error en el intento de eliminación del registro en rol se ignora puesto que puede haber más cargos para el mismo identificador de rol.

- Alta_Voz_AgrupacionAnio, Modifica_Voz_AgrupacionAnio y Baja_Voz_AgrupacionAnio
  - Se tienen las restricciones definidas mediante la declaración de las claves externas y primaria, donde solo se puede dar un par por persona y agrupación, y se incluye la verificación de que la persona no participe con rol de tipo Orquesta.
  - Como en el caso anterior, las bajas y las altas implican la gestión del registro existente en rol correspondiente al de Voz.
- Alta_Orquesta_AgrupacionAnio, Modifica_Orquesta_AgrupacionAnio y Baja_Orquesta_AgrupacionAnio
  - Similar al anterior, pero inverso, de tener un integrante de agrupación con rol de Voz no se dará el mediante el procedimiento Modifica_Orquesta_AgrupacionAnio, también se hace la gestión de la entidad superclase y de la secuencia de identificadores de la misma.
- Alta_Participa, Modifica_Participa y Baja_Participa
  - Para el alta en la tabla participa, tras verificar que no se haya asociado aún la agrupación al concurso, se verifica que la agrupación tenga el mismo año y categoría que el concurso en que se inscribe.
  - Seguidamente, si se asigna un premio se debe controlar que dicho premio no existe para otra agrupación participante en el concurso, del mismo año y con la misma modalidad. Cabe recordar que en esta tabla no se conoce la modalidad por la redundancia que introducía; por tanto, es una relación indirecta que debe controlarse entre los premios y las agrupaciones mediante una tercera capa de gestión de la integridad, concretamente la que se modela mediante este conjunto de procedimientos almacenados. Naturalmente esta comprobación se da para las altas y las modificaciones.
  - Por otro lado, de nuevo como decisión de diseño físico, no se admite cambiar una relación entre agrupación y concurso, ni viceversa, habría que dar la baja de participación; lo único que se pueden cambiar son el resto de atributos propios de la relación de asociación. Luego, la eliminación de participación solo se da a través de esta tabla y mediante el procedimiento Baja_Agrupacion_Concurso únicamente tras comprobar que el par agrupación y concurso existe.

05_SP_06_Obra.sql

Para la jerarquía definida desde Obra, se han definido varias funciones para reducir el código redundante que genera la gestión de documentos; casi todas son internas y utilizadas desde los procedimientos almacenados definidos en los próximos 3 *scripts* que se describirán, dedicados a cada uno de los subtipos de esta.

- __Alta_Obra, __Modifica_Obra y __Baja_Obra
  - El primero crea el registro en Obra y el documento mediante el procedimiento __Doc_Nuevo_a_Obra, definido en este mismo *script*; esta última función gestiona la relación entre la obra y el documento en la tabla DocumentoObra.
  - El segundo verifica la existencia del vínculo entre la obra y el documento solicitados para modificar, la corrección del Código de Referencia de la ISAD(G) y luego modifica únicamente el documento indicado por el identificador recibido, haciendo uso del procedimiento almacenado __Modifica_Documento detallado anteriormente. En este caso si se extrae el código necesario para verificar la corrección del valor 3_1_1_codigo_referencia, puesto que todas las obras comparten la composición del mismo; de hecho, se utilizará también para los componentes de las mismas.

    De nuevo no se gestiona el intercambio de documentos entre obras; la decisión de diseño al respecto es que habría que dar baja y alta respecto de la obra inicial y la elegida o simplemente asociarlo a ambas, según se pretenda. En cual quier caso, para no perder los datos ya registrados en Documento, habrá que dar alta en el elegido y luego baja en la que fuera errónea la vinculación.
  - Por último, para las bajas, siguiendo el comportamiento esperado por el procedimiento almacenado __Baja_Documento, si existe el registro en la tabla DocumentoObra se elimina y se llama a dicha función del *script* 05_SP_02_Documento.sql. Como se modeló en el caso de Evento, se lanza error en caso de ser el último vínculo del documento en la BBDD y tener referencias a un documento físico o electrónico; para el resto de mensajes recibidos se entiende

que el documento fue borrado o tiene vínculos a otros objetos, en cuyo caso se sale de este procedimiento sin error.

La eliminación en la tabla Obra la gestionan las funciones de los subtipos, porque antes deben retirarse los registros de los mismos para evitar la excepción por la restricción de integridad que participa en la generalización.

- __Modifica_Signatura_Obra
  - Se considera necesario ofrecer la posibilidad de editar la signatura de las obras, la de los componentes tiene menor repercusión porque no es afectado por componentes de control del SGBD. Como en los casos anteriores se verifica que no esté en uso el identificador que se solicita utilizar ni que sea parte de la secuencia.
  - La complejidad deriva de la necesidad de actualizar los valores del campo 3_1_1_codigo_referencia en los documentos asociados a la obra, atendiendo a la composición del código cambiar. Solo los que contienen la signatura de la obra tras el código de la institución –de momento se ha utilizado ES.ACCC//–, ya que en cualquier otro caso no habrá sido introducido a partir de la obra en edición.
- __Verifica_Codigo_Referencia y __Componer_Codigo_Referencia
  - Ambos reciben tres identificadores, el de la obra, el del componente si se está en ese nivel y el del documento; además, en el caso de la función de verificación, el valor dado para 3_1_1_codigo_referencia. Este último no es nulo si se llama a esta función, en caso contrario se llama a la segunda directamente.
  - El verificador comprueba que contenga el código de la institución como prefijo y los identificadores a partir del carácter 10, ya que 9 es la longitud del anterior. En el caso de las signaturas de componentes de obra y del identificador de documentos fuerza que estén precedidos de un guión medio; de esta forma, se garantiza que en la validación no corresponda a la obra sino a los adscritos a la misma, ya sean componentes o documentos.

    Por último, si el identificador recibido para el documento no está contenido en el valor a verificar, se permite la no inclusión del mismo en el Código de Referencia siempre que el provisto no exista.
  - La función destinada a componer el valor de 3_1_1_codigo_referencia forma el valor según las mismas especificaciones que verifica el anterior; y, si no recibe el identificador de Letra, Musica o Disenio, coloca el del documento en segundo lugar porque se trata del caso en que no se solicita desde dichos componentes.
- __Doc_Nuevo_A_Obra
  - Esta función es llamada por los diversos subtipos de Obra para añadir nuevos documentos asociados a las mismas y desde __Alta_Obra que, a su vez, es la llamada por los subtipos para el alta de registros en Obra. Dentro del modelo conceptual ya se explicitaba que cada obra y componente tendría un mínimo de un registro en Documento.
  - Verifica el Código de Referencia si lo recibe o lo construye en su defecto, luego llama a __Alta_Documento, finalmente crea el vínculo en DocumentoObra haciendo uso de la primera de las siguientes.
- __Alta_Documento_Obra y __Baja_Documento_Obra
  - Todas estas gestionan la tabla DocumentoObra que implementa la relación N:M entre estos dos conceptos.
  - Para el alta de esta relación se tiene en cuenta el Nivel de Descripción, según la ISAD(G), y el tipo de documento y de obra. En primer lugar verifica que no exista el vínculo entre los identificadores recibidos; de no existir, se accede a los atributos recién indicados desde las dos tablas –en las que ya existen los registros a vincular, sea por creación reciente o por vinculación de existentes–; por último, verifica que se cumplan las siguientes condiciones para la asociación de documentos a obras y, si se cumplen, inserta el par de identificadores.
    - Desde esta tabla dedicada a relacionar con multiplicidad N:M se pueden vincular documentos de nivel “unidad documental compuesta”, o “unidad documental simple” si es de tipo “video”.
    - Si es “texto” y “unidad documental compuesta” a Literaria y/o uno o más de Evento.

        - Si es “imagen” y “unidad documental compuesta” se puede vincular a Grafica, Musical (también por las partituras) y/o una o más instancias de Evento.
        - Si es “video” y “unidad documental simple” a Letra, Musica, Grafica y/o Evento.
        - Si es “audio” y “unidad documental compuesta” se puede vincular a Literaria, Musical y/o Evento.
        - Si es “video” y “unidad documental compuesta” a Literaria, Musical y/o Evento.
    - El caso de la baja en esta tabla requiere verificar que no se trate del último registro que apunta a la Obra, en cuyo caso se lanzará error -para esto se requiere de eliminar la Obra por decisión relativa al uso del modelo–. De ser posible la eliminación llama a __Baja_Documento y gestiona los diversos errores que puede generar este último; como en casos anteriores, de haber documento tangible asociado al Documento impedirá la eliminación.
- __Doc_Nuevo_a_Componente y __Baja_Documento_Componente
    - Es estos casos se reduce algo de código, pero no demasiado porque cada componente tiene sus peculiaridades, principalmente una tabla N:M por cada uno y distintas condiciones a cerca de los tipos de documento y soporte. Por esto mismo, no se ha definido la función __Doc_Existente_a_Componente.
    - El primero de estos es similar a __Doc_Nuevo_a_Obra, solo que no hace la vinculación entre el documento y el componente; y lo mismo para las bajas, lo que reducen en código es la verificación común y la gestión de errores generados por __Baja_Documento.

Naturalmente estas funciones surgen durante el desarrollo de las que se detallan a continuación, organizadas en un *script* por cada subtipo de Obra.

05_SP_061_ObraLiteraria.sql

- Alta_Literaria, Modifica_Literaria y Baja_Literaria
    - Ya en este nivel, el más específico, se reciben como parámetros todos los valores de las tablas participantes, Literaria y Documento, luego en la llamada a __Alta_Obra, __Modifica_Obra y __Baja_Obra se hace uso de la tabla adecuada y se indica el tipo de Obra en cuestión.
    - Como ya se comentó, para el alta se gestionan en este nivel los identificadores de obra y documento antes de solicitar la creación de estos; así como las secuencias de uno y otro en caso de haberlas utilizado.
    - En el caso de la modificación se comprueba la existencia del par en la tabla DocumentoObra, que no contempla la obtención de más de un registro, puesto que no debe existir más de un registro en esta tabla por cada documento.

      Por otra parte, se hace necesario verificar la coherencia entre el título de la obra, si se recibe un valor no nulo, y el de la agrupación que lo interpreta, si es que la obra está referenciada por alguna. En este caso el código de verificación es mucho más sencillo porque se cuenta con muchos de los valores, por lo que no se hace uso de la función definida a tal efecto en 05_SP_05_AgrupacionAnio.sql; por otro lado, es necesario hacer la misma verificación a un lado y al otro para implementar la restricción de forma completa.

      Tras estas comprobaciones se carga el registro de Literaria para la actualización y se actualizan sus valores con los recibidos no nulos, solo repertorio en este caso, y de haber éxito se llama a la interna __Modifica_Documento.
    - La baja es sencilla en este nivel, toda la lógica está en __Baja_Obra, que es la primera acción que se realiza. De eliminarse se elimina en Literaria y en Obra, tal como se mencionó en su momento solo se puede retirar el registro de Obra tras hacerlo desde su dependiente.
- Modifica_Signatura_Literaria
    - Tras verificar que el identificador recibido para sustituir, denominado in_signatura_vieja, está contenido en la tabla Literaria, simplemente se llama a __Modifica_Signatura_Obra.
- Doc_Nuevo_a_Literaria
    - Este procedimiento almacenado recibe el identificador de la Obra y todos los valores necesarios para crear un Documento. Solamente verifica que sea de tipo literario, gestiona el identificador del documento de no ser recibido y __Doc_Nuevo_a_Obra hace el resto.

- Doc_Existente_a_Literaria y Baja_Documento_Literaria
  - Estas, tras verificar que el una obra de tipo literaria, hacen uso de __Alta_Documento_Obra y __Baja_Documento_Obra, donde se concentra toda la lógica ya expuesta.

05_SP_062_ObraMusical.sql

Este *script* y el siguiente son totalmente idénticos al anterior, solo cambian las tablas objetivos, en este caso se trata de los procedimientos almacenados para la actualización de Musical.

- Alta_Musical, Modifica_Musical y Baja_Musical
- Modifica_Signatura_Musical
- __Doc_Nuevo_a_Musical, __Doc_Existente_a_Musical y __Baja_Documento_Musical

05_SP_063_ObraGrafica.sql

Al igual que el anterior, este es una réplica exacta del dedicado a Literaria, por tanto simplemente se listan las funciones.

- Alta_Grafica, Modifica_Grafica y Baja_Grafica
- Modifica_Signatura_Grafica
- __Doc_Nuevo_a_Grafica, __Doc_Existente_a_Grafica y __Baja_Documento_Grafica

05_SP_071_Letra.sql

- Alta_Letra, Modifica_Letra y Baja_Letra
  - Los tres editan el documento que se les indique si reciben datos al respecto; el de baja gestiona la eliminación del vínculo con el documento mínimo asociados y del mismo si queda huérfano. Por ende, las dos primeros reciben todos los valores posibles para poder cumplimentar la tabla Letra y la tabla Documento.
  - Para las altas, en primer lugar, se valida la recepción de la signatura de la obra, por su carácter de compuesto en la relación de composición requiere de este identificador. Junto con esta se comprueba que se haya recibido tema; al igual que en el caso de las actividades de las empresas, solo mediante un procedimiento almacenado puede imponerse la existencia de este concepto asociado a las letras.

    Seguidamente se comprueba que no haya más de una presentación o popurrí como género para la misma obra literaria; de esta forma se restringe que solo haya un componente para describir cada pieza, aunque se admite que una misma pieza tenga varios documentos tangibles asociados.

    Luego se gestiona la signatura del componente, de no ser indicada se selecciona el que siga al último registrado; es decir, la signatura mayor más uno. Para el identificador del documento se siguen las mismas pautas que en casos anteriores, de no recibirlo se toma de la secuencia asociada a este campo.

    Tras todo esto se llama a __Doc_Nuevo_a_Componente con todos los identificadores necesarios, si hay error se decrementa la secuencia de Documento y en caso contrario se inserta el registro en Letra, se vincula en DocumentoLetra, haciendo uso de una función interna que se ha generado a tal efecto, y se le vincula el tema mediante otra función, la de alta para TemaLetra que si forma parte de la API de la BBDD.
  - Modifica_Letra recibe los mismos argumentos que la anterior más el identificador del registro del que se solicita la actualización y del documento activo en ese momento, puesto que también implica la edición del documento mínimo por el que se describe el componente de tipo letra.

    Se hace una verificación en dos pasos, en primer lugar se consulta en DocumentoLetra para los identificadores recibidos; en segundo lugar, de existir, se verifica la existencia del par literaria y letra indicado en Letra. Si existe ya se habrá cargado el registro actual completo en una variable.

    A continuación se procede con la verificación de los géneros, de hacer un cambio solicitado respecto a este; luego, de recibir valor para el Código de Referencia o para la signatura de la letra, se verifica la coherencia del mismo. Finalmente se actualiza el registro de Letra y se llama a __Modifica_Documento.

  - Respecto a la baja en la tabla Letra se verifica que exista el conjunto de registros debidamente relacionados en las tablas Letra y DocumentoLetra, considerando todos los identificadores implicados. Luego, comprueba cuantos documentos tiene asociados la letra de la que se solicita la baja, de ser solo uno prosigue llamando al procedimiento de baja definido para la tabla DocumentoLetra; esta última, por el contrario, lanza error si solo existe uno, y de ser este el error se continúa con el proceso de eliminación en DocumentoLetra de forma directa; en Documento haciendo uso de __Baja_Documento_Componente; y, finalmente, en Letra.

- __Alta_Documento_Letra, Doc_Existente_a_Letra y Baja_Documento_Letra
  - Gestionan la tabla que modela la relación N:M entre letras y documentos. Para las altas y las nuevas vinculaciones entre elementos registrados se verifican las siguientes condiciones:
    - El nivel de descripción, según la ISAD(G), será en todo caso “unidad documental simple”.
    - Como tipo de documento admite “Texto”, “Audio” y “Video”.

    Cabe señalar que el segundo hace uso del primero tras verificar la no existencia del vínculo en DocumentoLetra.
  - En el caso de las bajas, como se expuso, en este *script* específico para las letras realiza la misma comprobación que para las obras en la tabla dedicada a relacionar letras y documentos. De existir más de un documento se hace uso de la función más genérica __Baja_Documento_Componente, en caso contrario lo que habrá que eliminar es el registro en Letra.

  Siguiendo la pauta descrita anteriormente, no se implementa el método que admita el intercambio de documentos y letras, de nuevo forzando al usuario a que cambie el documento de letra dando el alta en la letra deseada y luego la baja en la errónea.
- Doc_Nuevo_a_Letra
  - Análogo al desarrollado para las obras, requiere las signaturas de la letra y la obra a la que se adscribe, más el identificador del registro –que solo será utilizado para vincular los elementos–; y, de no ser indicado, se gestiona el identificador del documento. Tras esto, se procede como en el alta de las letras haciendo uso de __Doc_Nuevo_A_Componente y __Alta_Documento_Letra.
- Alta_Tema_Letra, Modifica_Tema_Letra y Baja_Tema_Letra
  - Estos últimos actualizan la tabla TemaLetra, la única verificación es la existencia o no del par indicado en dicha tabla; también como en el resto de conceptos considerados adjetivos, solo se modifica el tema en el procedimiento que implementa esta acción.

05_SP_072_Musica.sql

Los tres conjuntos que se listan a continuación son completamente iguales a los descritos para la tabla Letra, contenidos en el *script* anterior.

- Alta_Musica, Modifica_Musica y Baja_Musica
- __Alta_Documento_Musica, Doc_Existente_A_Musica y Baja_Documento_Musica
- Doc_Nuevo_A_Musica

  Una de las diferencias es que no es necesario comprobar los géneros, porque la restricción está definida en el DDL mediante una clave única entre la signatura de la obra musical y el género; por tanto, la no repetición de géneros por obra musical ya está controlada. La otra diferencia es la restricción en el tipo de documentos vinculables, la música además admite el tipo “Imagen”.

05_SP_073_Disenio.sql

- Alta_Disenio, Modifica_Disenio y Baja_Disenio
- __Alta_Documento_Disenio, Doc_Existente_a_Disenio y Baja_Documento_Disenio
- Doc_Nuevo_a_Disenio

  Como en el caso anterior se tiene una gran similitud en la implementación, aunque centrados sobre las tablas Disenio y DocumenoDisenio. Para los tipos de diseño admisibles por cada obra no se han definido verificaciones ni se tienen restricciones en la definición de la tabla, respecto a los tipos de documento no admite el “Audio”.

- Alta_Tema_Disenio, Modifica_Tema_Disenio y Baja_Tema_Disenio
  - Análogos a los desarrollados para las letras, estos actualizan la tabla TemaDisenio.
- Alta_Crea_Disenio, Modifica_Crea_Disenio y Baja_Crea_Disenio
  - Estos tres últimos procedimientos almacenados gestionan la segunda relación entre diseños y entidades. Para el alta simplemente se verifica la no existencia de la relación entre los pares de identificadores indicados, las restricciones sobre los valores ya están definidas en la tabla crea.
  - Respecto a la modificación, este es uno de los pocos casos en que se admite el cambio de los elementos relacionados; es decir, a un diseño se le puede modificar la entidad creadora sin alterarle el coste y las posibles observaciones y viceversa. Cabe señalar que por esta razón se incluye el identificador sintético a la tabla.
  - Por último, la eliminación, simplemente requiere que se den los identificadores de los registros vinculados mediante esta tabla.