Reingeniería de "Presentes": Una Base de Datos sobre el Accionar del Terrorismo de Estado

Autor: Carlos E. Barcia
cebarcia@gmail.com
Director: Dr. Franco M. Luque
francolq@gmail.com

FaMAF, Universidad Nacional de Córdoba

Resumen "Presentes" es un software utilizado por el Archivo Provincial de la Memoria para la investigación del accionar del terrorismo de estado en la provincia de Córdoba. Este sistema informático contiene una base de datos que condensa y organiza información recopilada a lo largo de años de constante investigación.
El sistema no fue desarrollado mediante un proceso de producción de software adecuado y necesita un conjunto importante de modificaciones y extensiones. En consecuencia, en este trabajo proponemos realizar una reingeniería de "Presentes" teniendo en cuenta requerimientos relevados y planteados por el Archivo de la Memoria.
Como parte del proceso de reingeniería realizamos una ingeniería inversa de la base de datos existente, para esclarecer su funcionamiento actual, analizar sus falencias y proponer soluciones para ellas. Además, hicimos una migración de tecnología de base de datos a un motor más adecuado a los requisitos del sistema. Finalmente, diseñamos e implementamos una nueva base de datos utilizando conceptos de los modelos de bases de datos objeto-relacional y documental. En el proceso, mejoramos la calidad de los datos, eliminando inconsistencias y redundancias para adaptarlos a los cambios en la estructura y a las nuevas restricciones introducidas en cada etapa de la reingeniería.

Palabras clave: Sistema Informático Presentes, Ringeniería de Software, Ingeniería Inversa, Bases de Datos, SQL, Modelo de Dato Relacional, Modelo de Dato Objeto-Relacional, Modelo de Dato Documental, Mapeo Objeto Relacional, PostgreSQL, MySQL.

Nota: este documento es una síntesis del trabajo final de grado de la Licenciatura en Ciencias de la Computación de la FaMAF UNC, el informe completo se puede ver aqui[1].

[1] https://drive.google.com/file/d/0B8MYemwNUk0NeDZ5SDVPcnlLRmc/view?usp=sharing

1. Archivo Provincial de la Memoria y Sistema Informático Presentes

El Archivo Provincial de La Memoria (APM) se creó a partir de una Ley promulgada por la Legislatura de la Provincia de Córdoba en la que establece el marco para el desarrollo de sus funciones, entre las que destacamos la responsabilidad de recopilar y organizar documentación relacionada con violaciones a los derechos humanos y el accionar del terrorismo de estado en la Provincia de Córdoba durante la última dictadura militar (1976-1983).

La sede del APM se encuentra en el lugar donde funcionaba el ex Departamento de Inteligencia de la Policía de la Provincia de Córdoba, uno de los principales Centros Clandestinos de Detención. Actualmente en esa dependencia opera el Área de Informática y Digitalización del APM donde utilizan el Sistema Presentes. Este consta de una base de datos que almacena información sobre personas, documentos, causas judiciales y lugares vinculados a la última dictadura militar. Se comenzó a desarrollar en el año 2001 por Marcelo Yornet y la asociación H.I.J.O.S. Regional Córdoba, en 2011 fue donado al APM.

Al momento de comenzar este trabajo la base de datos de Presentes que utiliza el gestor de base de datos MySQL, consta de 102 tablas que están relacionadas entre si mediante el uso de 24 claves foráneas. La parte del sistema encargada de interactuar con la base de datos y la interfaz gráfica de usuario está desarrollada en Visual Basic. Por lo que se pudo apreciar[2] sus componentes no están correctamente modularizados.

La labor de campo realizada en este trabajo especial consta de tres etapas: ingeniería inversa de la base de datos, migración del gestor de base de datos y por último su completa reingeniería.

2. Ingeniería Inversa de Presentes y Análisis de Problemas

El objetivo de Presentes es coleccionar y relacionar información de damnificados y represores de la última dictadura militar en Córdoba. Para ello se utilizan diversos tipos de documentos e información de casos judiciales que los atañen.

La información es almacenada de forma estructurada en una base de datos que está implementada en el gestor de base de datos relacional [3] MySQL.

La base de datos no fue desarrollada a partir de un diseño conceptual específico y bien definido, fue realizada y extendida a partir de diversos requerimientos que surgieron a través del tiempo. Esta metodología no fue respaldada con documentación que explicara la estructura de la información almacenada, su propósito, ni funcionalidad.

Como única documentación y punto de partida se dispone del manual de usuario del sistema y el código fuente para la creación de las tablas de la base de datos, por lo que para comenzar a comprender el diseño subyacente fue necesario realizar la ingeniería inversa de la misma.

[2] No se tuvo acceso al código de la Interfaz.

2

2.1. Ingeniería Inversa

Como primer paso se utilizó la herramienta de diseño, desarrollo y administración de bases de datos MySQL Workbench[3], la cual cuenta con un proceso automático de ingeniería inversa que genera una representación gráfica a partir del código fuente de una base de datos implementada en MySQL. En la Figura 1[4] se puede observar el diagrama que da como resultado aplicar este proceso sobre la base de datos de Presentes.

Para comprender mejor este diagrama se organizaron las tablas en siete grupos conceptuales descritos a continuación:

Víctimas: Información vinculada a los damnificados.

Represores: Información asociada a represores.

Relación Victimas-Represores: Tablas que relacionan víctimas con represores e información de centros clandestinos de detención.

Códigos: En este grupo se reunieron tablas que son utilizadas para almacenar los posibles valores de atributos de otras tablas.

Causas: Datos asociados a causas judiciales.

Tareas: Datos que corresponden a los usuarios del sistema (nombre usuario, clave de ingreso, etc.).

Fuentes: Documentos legales, diarios, testimonios, etc.

Una vez obtenida la representación gráfica de la base de datos se dilucidaron variados problemas de diseño e implementación que posee. En la siguiente sección se hace una descripción y análisis de los principales problemas a la vez que se presentan soluciones y mejoras a los mismos.

2.2. Problemas y Soluciones Propuestas

Algunas de las soluciones o mejoras propuestas en esta sección son ejecutadas mediante el uso de distintos *script*, desarrollados en el lenguaje Structured Query Language (SQL) [4]. La meta es poder ejecutar estos *scripts* sobre una instancia de la base de datos, como pasos que van solucionando distintos problemas. Cada uno lleva la base de datos a un estado preparado para poder ejecutar el siguiente. De esta forma, el proceso puede ser fácilmente adaptado y reproducido en etapas bien definidas sobre distintas instancias de la base de datos.

A continuación se presentan los principales problemas que se encontraron.

2.2.1. Tablas y Vistas Obsoletas Como comienzo del relevamiento, se hizo un recorrido general por toda la base de datos, se localizaron veintisiete tablas y doce vistas que eran obsoletas.

[3] http://www.mysql.com/products/workbench/

[4] Los diagramas de la base de datos presentados en este trabajo no buscan mostrar en detalle su estructura ya que solo cumplen la función de ilustrar los cambios que van surgiendo en las distintas etapas.

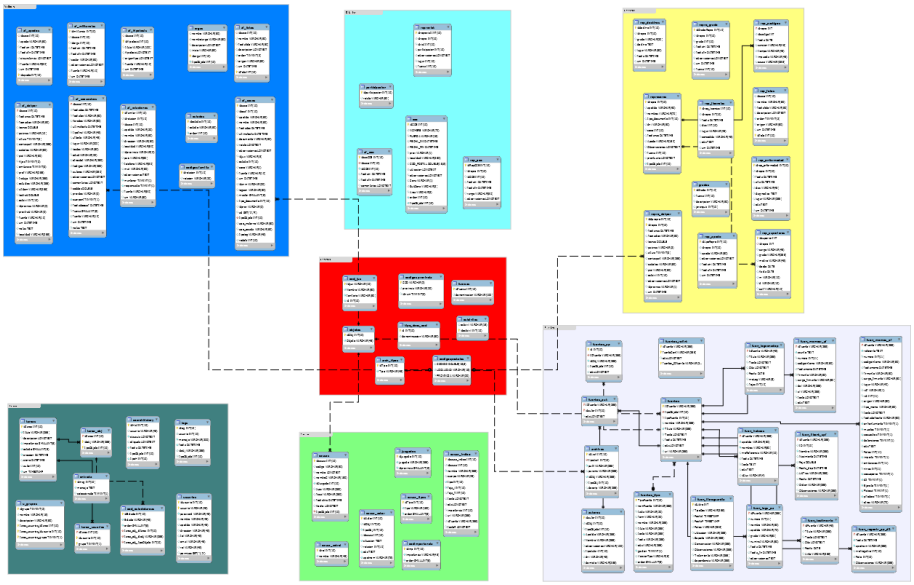


Figura 1. Diagrama de las tablas y restricciones de la base de datos de Presentes organizado según los grupos conceptuales: **Víctimas**, **Represores**, **Relación Victimas-Represores**, **Códigos**, **Causas**, **Tareas y Fuentes**.

2.2.2. Falta de Relaciones Explícitas Mediante el uso de Claves Foráneas

Varias de las tablas utilizan una columna que almacena valores de una columna de otra tabla. Este tipo de relación se puede establecer explícitamente mediante el uso de clave foránea [9].

Para poder implementarlas fue necesario alcanzar la integridad referencial. Para ello se tuvo que hacer un análisis y consecuente modificación de sus datos y estructura.

En total se agregaron 67 nuevas claves foráneas, llevando a un total de 91 las utilizadas en la base de datos (ver Apéndice E).

2.2.3. Relaciones Implícitas Entre Múltiples Tablas

El diseño conceptual de la base de datos se desarrolla en torno a siete entidades conceptuales principales, declaradas explícitamente en los registros de la tabla `objetos` (Tabla 1).

Cada una de estas entidades conceptuales esta representada por una tabla en la base de datos, estas quedan relacionadas entre sí por medio de claves foráneas a tablas que almacenan información sobre dichas relaciones. Sin embargo existen relaciones que no fueron posibles de implementar mediante el uso de claves foráneas. Una de estas se encuentra en la tabla `ap_fuentes` que relaciona Fuentes con cualquiera de las siete entidades conceptuales antes mencionadas. Para ello, la tabla cuenta con el atributo `tipoObjeto` que indica la entidad según la clasificación dada por la tabla `objetos` con que está relacionado cada registro

idObj	Objeto
1	Victimas
2	Represores
3	Causas
4	Fuentes
5	Organizaciones
6	CCD
31	Acto Procesal

Cuadro 1. Registros de la tabla `objetos` de la base de datos de Presentes.

y el atributo `idObj` que contiene el identificador del registro en la tabla que representa a esta entidad.

Como solución a este problema se aplicó en el nuevo diseño expuesto en la Sección 4.1 la técnica de **Mapeo Objeto Relacional** (ORM[5]). En [2] se define al ORM como la persistencia automatizada y transparente de los objetos de una aplicación en una base de datos relacional, utilizando metadatos que describen el mapeo entre los objetos y la base de datos.

2.2.4. Uso de Tablas Específicas para Cada Tipo de Documento La entidad "Fuente" representa a las distintas fuentes documentales que almacena el sistema. Esta entidad está representada por la tabla `fuentes` que además de almacenar información como su título y fecha, incluye un campo donde se guarda la concatenación de las distintas partes que componen al documento. A su vez, estos mismos documentos están almacenados en tablas auxiliares que representan a cada tipo de documento en particular. Estas tablas se ocupan de estructurar la información, empleando atributos específicos. Cada registro representa un documento y está relacionado mediante clave foránea a su versión en la tabla `fuentes`.

Si se desea comenzar a almacenar un nuevo tipo de documento, es necesario implementar una nueva tabla con los distintos campos específicos que lo componen y referenciarla a la tabla `fuentes`.

Se decidió emplear en el nuevo diseño (Sec. 4.2) un modelo de datos documental [10] [7] utilizando el tipo de dato `jsonb` que permite definir campos arbitrarios.

2.2.5. Tablas Utilizadas para Restringir Valores de Atributos Las tablas del grupo "Codigos" (Fig. 1) son utilizadas para restringir posibles valores de atributos en otras tablas, esto implica la utilización de una gran cantidad de tablas y claves foráneas. En su lugar se optó por la utilización de "Dominios Definidos por el Usuario" y "Tipos de Datos Enumerados". Este cambio es desarrollado en la Sección 5.2.

En el Apéndice E se listan los dominios y tipo de datos enumerado que fueron definidos en este diseño.

[5] ORM son las siglas de *"Object Relational Mapping"*

2.2.6. Tablas, Atributos e Información Redundante Para almacenar los datos personales de las Víctimas y Represores se utilizan tablas que almacenan los mismos datos (salvo unos pocos atributos), estructurados de la misma forma. También la base de datos cuenta con otras tablas que almacenan datos de personas relacionados con causas judiciales, autores de los documentos e individuos relacionados con víctimas. Una persona puede cumplir con más de uno de estos roles, por lo que sus datos pueden verse repetidos en estas tablas.

En la Sección 4.3 se presenta una mejora a este diseño, en la cual todos los datos de las personas están almacenadas en una única tabla.

2.2.7. Múltiples Criterios para la Asignación de Nombres de Tablas y Columnas En el desarrollo del sistema no se siguió una convención de nombres bien definido para tablas y columnas. Esto dificulta la comprensión del código y su mantenimiento. Por este motivo se propuso, en el rediseño de la base de datos, reglas para una convención de nombres bien definidos (ver Apéndice D).

2.3. Diagrama Conceptual de la Base de Datos de Presentes

Como resultado de esta etapa se llegó a comprender la mayor parte del funcionamiento de la base de datos y su estructura. Aunque existe una gran cantidad de relaciones entre distintas tablas que, dado la manera en que se implementaron, no se ven reflejadas en este diseño. Más bien pertenecen a un diseño conceptual subyacente que no se desarrolló correctamente como es el caso de las relaciones entre múltiples tablas (Sec. 2.2.3).

3. Migración del Gestor de Base de Datos

Una vez hecho el análisis y superados los inconvenientes iniciales, el siguiente paso fue migrar Presentes del gestor de base de datos relacional MySQL al gestor de base de datos objeto relacional [5] PostgreSQL.

Una de las principales razones para llevar a cabo esta migración es que los motores de almacenamiento de MySQL [8] cuentan con distintas características, la que más nos concierne es que no se permite la utilización de claves foráneas y Full Text Search a la vez. Además, la implementación de búsqueda de texto completo de PostgreSQL [6] tiene mejores prestaciones y permite una gran variedad de ajustes para adaptar la herramienta a las particularidades de las búsquedas requeridas.

Si bien esta fue la principal razón, también se analizaron otros aspectos pensando en futuras mejoras y que entorno se adapta mejor a los requerimientos actuales del sistema.

3.1. Metodología utilizada

El proceso de migración se diseñó siguiendo el objetivo de poder aplicar cambios paso a paso, a partir de distintos *script* ejecutados sobre la instancia de

la base de datos que da como resultado el aplicar lo desarrollado en la primer etapa. De esta forma, en la instancia de la base que utiliza el APM se puede primero aplicar las soluciones de la etapa anterior y luego migrar a PostgreSQL, empleando en un orden establecido estos *scripts*. La secuencia de los pasos que realizan estos *scripts* es la siguiente:

Paso 1: Crear una versión en PostgreSQL de solo las tablas de Presentes.
Paso 2: Migrar los registros de Presentes almacenadas en el gestor MySQL a la versión de PostgreSQL.
Paso 3: Agregar a la versión en PostgreSQL las claves primarias y foráneas, índices, etc.

3.2. Problemas que se Presentaron en la Migración de Gestor de Base de Datos

La gran cantidad de tablas, relaciones entre ellas, datos de gran tamaño y las diferencias entre los gestores generaron muchas complicaciones para esta migración. A esto se suma la necesidad de poder realizar este proceso en otras instancias de la base de datos. Los inconvenientes más importantes que presenta el desarrollo del proceso de migración son:

3.2.1. Constantes Modificaciones en Presente Mientras realizamos el trabajo la base de datos utilizada en el APM era constantemente modificada. Por este motivo fue necesario ir actualizando los *scripts* a medida que ocurrían estas modificaciones.

3.2.2. Caracteres no Reconocidos: Varios de los documentos almacenados en las tablas fueron generados a partir de imágenes escaneadas, utilizando una herramienta conocida como *Optical Character Recognition*. Algunos de los caracteres de estos documentos digitalizados no fueron bien reconocidos por esta herramienta, lo que causó que se almacenarán caracteres con un mal formato en su codificación.

3.2.3. Cambios en la Base de Datos en MySQL Antes de Migrar Los Datos: Para poder implementar algunas restricciones de claves primarias y foráneas que tiene Presentes, fue necesario realizar algunos cambios. Por ejemplo, en PostgreSQL para declarar una clave foránea ambas columnas deben poseer el mismo dominio y la columna referenciada debe tener una restricción que haga que sus valores sean únicos para cada registro de la tabla. Por ello, en algunas columnas se tuvo que agregar la restricción de unicidad y cambiar el tipo de dato.

3.3. Problemas que se Presentaron para Adaptar la Interfaz a la Base de Datos Implementada en PostgreSQL

Si bien no nos encargamos de adaptar la interfaz de Presentes a la base de datos implementada en PostgreSQL, se prestó ayuda para ajustar la sintaxis SQL

de las consultas más complejas. Además se adecuaron funciones y características específicas del gestor MySQL que no pertenecen al estándar SQL, utilizando funciones equivalentes provistas por PostgreSQL.

Cuando se intentó realizar esta adaptación se presentaron una serie de dificultades, a continuación se describen las más importantes.

3.3.1. Búsquedas Insensibles a Mayúsculas y Acentos A diferencia de PostgreSQL[6], las búsquedas que se pueden realizar en MySQL son sensibles a los acentos y a las mayúsculas.

Si bien en nuestro idioma un acento puede cambiar la semántica de una palabra, en este caso es deseable que una consulta devuelva todas las formas de la palabra buscada, ya que puede estar mal escrita o simplemente ser una variación gramatical de ésta. Para lograrlo se desarrollaron funciones en PostgreSQL que permiten búsquedas insensibles a mayúsculas y acentos.

3.3.2. Diferencias de Sintaxis SQL entre MySQL y PostgreSQL La sintaxis que utiliza MySQL para interpretar sentencias SQL tiene varias diferencias con respecto PostgreSQL. Principalmente debido a las distintas formas de utilizar comillas para escribir valores de diferentes tipos de datos o nombres de tablas, columnas, etc.

3.3.3. Funciones Específicas del Gestor MySQL En las consultas que genera la interfaz, constantemente se encuentran funciones propias de MySQL que no pertenecen al lenguaje de consulta SQL. Cada una de estas funciones es reemplazada por una equivalente provista por PostgreSQL.

3.4. Mejoras Aplicadas a la Base de Datos en PostreSQL

Luego de concretar la migración se introdujeron dos mejoras a la base de datos. La primera consiste en implementar Full Text Search sobre campos de las tablas de Fuentes y Causas. La siguiente agrega un sistema que lleva un registro meticuloso de los datos que son insertados, borrados o modificados en la base de datos.

3.4.1. Full Text Search en Presentes Las tablas que almacenan los distintos tipos de documentos contienen registros con textos de gran tamaño. Una manera de mejorar las búsquedas en estos documentos es emplear la herramienta Full Text Search (FTS) en Presentes, que identifica de mejor forma las palabras y sus derivaciones en las consultas. También agrega una mayor rapidez o la posibilidad de ordenar los resultados según su relevancia.

[6] Nota: PostgreSQL provee el operador `ILIKE` que es insensible a mayúsculas pero fue un requerimiento del APM que no se cambiaran los operadores SQL.

3.4.2. Historial de Cambios en Registros de Tablas en PostgreSQL

La siguiente mejora que fue introducida en esta etapa es un historial de la información de la base de datos que es agregada, modificada o borrada por los distintos usuarios del sistema. Para ello se utilizó Audit trigger 91plus[7], que es un sistema de auditoría que crea una tabla, funciones y `TRIGGERS` que permiten llevar un detallado registro de los cambios que se realizan.

4. Reingeniería de Presentes: Nuevo diseño de la Base de Datos

Varias de las soluciones y mejoras que requería el sistema no pudieron ser aplicadas en fases previas ya que implican grandes cambios en la estructura de la base de datos o no se adaptan al modelo relacional utilizado en su diseño. Por lo tanto se decidió realizar una reingeniería de la base de datos de Presentes. En este capítulo se explican las principales decisiones que determinaron el desarrollo del nuevo diseño.

4.1. Relaciónes Múltiples entre Tablas

Para satisfacer la necesidad de relacionar las distintas entidades conceptuales[8] entre si, es necesario contar con un identificador unívoco para cada fila entre las tablas que representan a estas entidades.

Para ello se utilizó la técnica Mapeo Objeto Relacional (MOR) en particular el mapeo de herencia de clases a modelo relacional expuestos por Scott Wambbler en [1]. Se pensó a estas entidades como clases que heredan de la super clase denominada `Objeto` (Fig. 2). De esta forma las tablas que representan las entidades contienen los atributos persistentes de los objetos de esas clases. En particular el atributo que interesa para este propósito es el Identificador de Objeto Persistente, que permite tener un identificador común para los registros que representan instancias de las entidades. Al utilizar este método, cada clase es mapeada a una tabla y la herencia es representada a través de claves foráneas entre los identificadores POIds de las tablas de las subclases y la superclase.

Esto permite relacionar por ejemplo una Fuente con cualquier otra entidad conceptual, simplemente creando una relación entre las tablas `fuente` y `objeto`.

4.2. Almacenamiento de los Distintos Tipos de Documentos

En este diseño la información almacenada en las tablas específicas de cada tipo de Fuente es recopilada de forma semiestructurada en una única columna llamada `documento`, perteneciente a la tabla `fuente` mediante el uso del tipo de dato `jsonb`.

[7] https://github.com/2ndQuadrant/audit-trigger

[8] Víctima, Represores, Centro Clandestino de Detención, Organización, Fuente, Causa y Acto Procesal

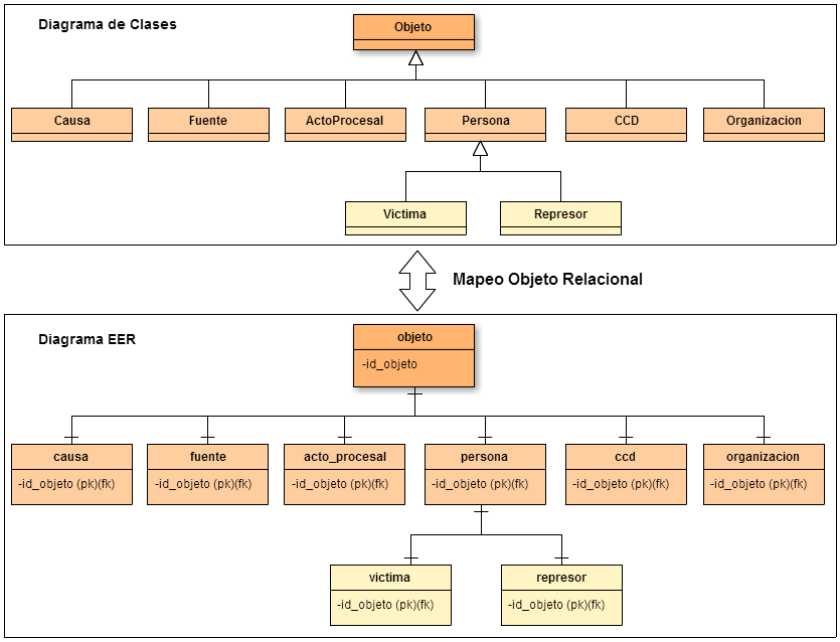


Figura 2. Utilización del MOR en las entidades conceptuales de Presentes. Para simplificar el diagrama solo se muestra el atributo POId.

De esta forma los registros de las tablas que almacenan los distintos tipos de Fuentes y los de la tabla `fuentes`, de la implementación original, quedan fusionados en una única tabla. Esto evita el uso de tablas específicas para cada tipo de documento y permite almacenar nuevos, sin necesidad de crear otras tablas y relaciones.

4.3. Personas

En diversas tablas se almacena información de personas que pueden ser Víctimas, Represores, individuos que están vinculadas a Causas o a Víctimas, etc. y en algunos casos una persona puede cumplir más de un rol.

En este diseño se agrupan los datos de las personas (a excepción de los usuarios del sistema) en una única tabla llamada `persona`. Los atributos de esta tabla contienen los datos personales más comunes, tales como nombre, apellido, etc. Además cuenta con una columna de tipo `jsonb` llamada `otros_datos`, en la que se almacena información particular según cada persona.

En el diseño original tanto las Víctimas como los Represores pueden tener relacionados apodos y fotos. En el nuevo diseño estas relaciones son unificadas y asociadas a las Personas en general.

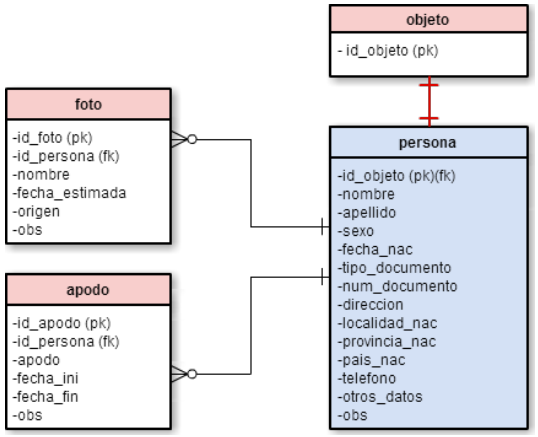


Figura 3. Diagrama Rediseño Personas.

Unificando las tablas que contienen información de Personas que cumplen distintos roles, se logra utilizar menos tablas y columnas que en el diseño original. Se evita repetir datos de Personas que cumplen con más de un rol.

4.4. Usuarios

En la base de datos original el nombre y clave de ingreso de los usuarios al sistema son almacenados en la tabla `usuarios`. La interfaz accede a la base de datos y utiliza los datos de esa tabla para establecer los niveles de acceso asignados a cada usuario. Esta forma de administrar la clave de ingreso y los niveles de acceso es reemplazada en el nuevo diseño por el sistema de usuarios nativo de PostgreSQL. Asimismo se siguen manteniendo las tablas que guardan los datos personales de los usuarios, tareas asignadas y relaciones entre estas y cualquier entidad conceptual.

5. Proceso de Implementación del Nuevo Diseño

La implementación del nuevo diseño de la base de datos de Presentes fue planificada para ser adaptada al resto del sistema en cuatro etapas. De esta forma se divide el trabajo a la vez que permite ir utilizando las mejoras que agrega cada etapa del proceso en el que está dividido.

5.1. Etapa 1: Documentos

Como primer etapa se implementa sobre la base de datos original la parte del rediseño explicada en la Sección 4.2, donde se unifican las tablas que almacenan los distintos tipos de documentos. Con esta etapa implementada se puede adaptar la interfaz de Presentes para usar los documentos almacenados en la tabla

`fuentes` y dejar de utilizar las tablas específicas de los distintos tipos de documentos. La adaptación puede seguir siendo aprovechada a medida que se avanza en las siguientes etapas (ver Apéndice F).

5.2. Etapa 2: Dominios y Tipos de Datos enumerados

En la segunda etapa se cambia el sistema de tablas utilizado para restringir posibles valores de atributos en los registros en la base de datos original[9], por dominios y tipos de datos enumerados. Son reemplazados los campos que almacenan los identificadores a estas tablas por otros con el mismo nombre que contienen el valor correspondiente del registro al que apuntaba.

Gracias a esto las tablas utilizadas por la base de datos disminuyeron de 75 a 61 y las claves foráneas, necesarias para relacionar los registros, se redujeron de 91 a 64 (ver Apéndice G).

5.3. Etapa 3: Reemplazo Completo de la Base de Datos

El primer paso del proceso para reemplazar la base de datos original por la desarrollada en función del nuevo diseño (ver Apéndice H), es crear un nuevo esquema. Este es un contenedor que utiliza el gestor para generar un espacio de nombres de los objetos de la base de datos. De esta forma puede tener en distintos esquemas, tablas y otros objetos con el mismo nombre. Lo que permite tener funcionando en la misma base de datos la versión original y la nueva, facilitando así la migración de datos entre estas y la adaptación al sistema.

El siguiente paso es realizar la migración de datos, pero antes es necesario modificar en gran medida los registros que contienen las tablas de la base de datos original para que cumplan con las características y restricciones de la nueva versión. Una vez realizado este proceso, la base de datos del nuevo diseño ya se encuentra en condiciones de ser adaptada al sistema.

5.4. Etapa 4: FTS, Historial y Configuración de Usuarios

En esta última etapa se implementa el sistema de historial y FTS. Esto se realiza como paso final dado que es necesario que las nuevas tablas con los datos migrados desde la base de datos original se encuentren en el esquema definitivo. Además, los usuarios y sus niveles de acceso son configurados en el sistema de PostgreSQL con los datos que se encuentran en la tabla `usuarios` del diseño original.

[9] La tabla `objetos` también es parte de este sistema, pero es sustituida en el nuevo diseño por la tabla `objeto` que almacena identificadores en vez de valores que determinan a que entidad conceptual corresponde.

5.5. Migración de Datos y Relaciones

Previo a migrar los datos de la base de datos original a la nueva, es necesario modificar la mayoría de los registros por diversos motivos. Algunos de los cambios se realizan sin la necesidad de un análisis detallado de los datos pero en otros casos se debe buscar y examinar los valores que deben ser modificados.

Por otro lado para poder conservar las relaciones entre los datos de las distintas tablas, es necesario almacenar de forma paralela los identificadores originales de las entidades conceptuales y los asignados de forma automática por la nueva base de datos.

5.6. Migración de Entidades Conceptuales y Datos Relacionados

Para poder migrar los registros que representan a las distintas entidades conceptuales, se agregaron en la nueva base de datos columnas auxiliares que son utilizadas para almacenar identificadores de los registros que representan las entidades conceptuales en el diseño original. De esta forma cada fila en la tabla `objeto` contiene un identificador único entre todos los registros de las entidades conceptuales y el identificador que tenían en la base de datos original. Cada vez que se debe guardar un registro en una tabla con campos que se relacionan mediante clave foránea a las entidades conceptuales, es necesario consultar en la tabla `objeto` su antiguo identificador. En base a este, se cambia el valor del identificador que hace referencia por el nuevo que fue asignado en la tabla `objeto`.

La nueva estructura contiene relaciones entre tablas que, a diferencia de la base de datos original, están implementadas mediante el uso de claves foráneas. Cuando se migraron los datos a tablas con estas nuevas restricciones se encontró una gran cantidad de registros huérfanos.

6. Conclusiones y Trabajo Futuro

Si bien se alcanzó el objetivo de realizar la reingeniería de la base de datos de Presentes que es completamente funcional y está documentada, aún no se concretó la adaptación de esta a la interfaz. En el momento de presentar este trabajo se encuentra implementado en el sistema en uso solo lo descrito en el Sección 2, que abarca la primer etapa. Esto es consecuencia de la complejidad que presenta para los responsables del sistema adaptar los cambios para migrar de gestor de base de datos, a lo que se suma la limitación del tiempo que disponen para efectivizarlo.

A continuación y salvando las dificultades mencionadas, se describen las mejoras que considero presenta el trabajo de reingeniería de la base de datos de Presentes.

6.1. En Cuanto al Diseño

Como resultado de este trabajo hay evidentes mejoras con respecto al cambio de diseño. La base de datos en un comienzo contaba con 75[10] tablas que se vieron reducidas a 32. La cantidad de relaciones implementadas mediante claves foráneas disminuyeron de 91[11] a 41.

Si bien hay una gran disminución en el número de tablas y relaciones, se alcanzó mayor organización y estructuración de la información. Se unificaron tablas que almacenan información similar y se añadieron relaciones explícitas faltantes que, como es el casos de las de entidades conceptuales, eran muy complejas de implementar en la base de datos original. Además, se añadieron restricciones a columnas que evitan información redundante o inconsistente. También es más sencillo borrar y actualizar registros asociados mediante clave foránea, dado que estas acciones se propagan en cascada.

Se quitaron de varias tablas las columnas que almacenan datos usados por el sistema para presentar formularios de consultas u organizar resultados en la interfaz gráfica de usuario. De esta forma se logra una mayor independencia con el resto de la aplicación.

La nueva base de datos posee documentación donde se exponen diagramas de su diseño y la explicación de los mismos. También, como parte de la documentación, existe un archivo que muestra donde se almacenan los valores de cada campo de la base de datos original en el nuevo diseño.

6.2. En Cuanto a los Datos

A lo largo de las tres etapas de este trabajo fue mejorando la calidad de los datos. Se elimina información redundante, principalmente por la unificación de tablas que tenían datos o estructuras similares. Se consiguió una mayor integridad referencial de los datos que se relacionan, desechando en el proceso una gran cantidad de registros huérfanos. Muchos datos fueron modificados para que pertenezcan al dominio asignado a las columnas en el nuevo diseño.

Por medio del tipo de dato `jsonb` se sumó la posibilidad de almacenar documentos e información particular de las personas sin necesidad de agregar o modificar tablas específicas para ello.

6.3. En Cuanto a la Funcionalidad

La base de datos resultante cuenta con la capacidad de realizar búsquedas mediante el sistema FTS de PostgreSQL en los documentos y causas judiciales.

Otra funcionalidad que se agrega es el historial implementado mediante un sistema que permite llevar un registro minucioso de todos los cambios en los datos de forma automática y que es totalmente independiente de la interfaz.

También se añade la utilización del sistema nativo de PostgreSQL para la administración de los usuarios que aporta mayor seguridad para el acceso a la base de datos, dado que la clave es controlada por el gestor.

[10] Después de borrar las tablas obsoletas (Sec. 2.2.1).
[11] Después de agregar las claves foráneas faltantes (Sec. 2.2.2).

A. Vistas y Tablas Eliminadas

A.1. Vistas:

- qryrepre_dest
- af_qryfuente
- listado
- listado sin fecha
- profesion
- af_listado
- #mysql50#listado sin fecha
- estadisticas
- listaupdate
- consulta1
- fuen_qrytest
- est_por_meses

A.2. Tablas:

- actosproc
- actualizaciones
- af_tareas
- af_testimonios
- caidas
- codifuentes
- codnomjuz
- destinos
- fuen_boletines_pc
- fuen_carpmemos_pf
- fuen_docvar_pf
- fuen_ent_lg
- fuen_entrevista_apm
- fuen_libcop_pf
- fuen_libroguardia_pc
- fuen_morgue
- fuen_ordendia_pc
- fuentes_autrem
- fuentes_jud
- fuentes_test
- legales
- permisos
- repre_dest
- roles
- u_usuarios
- #mysql50#codigos familia
- #mysql50#codigos provincia

B. Claves Foráneas Implementadas

af_apodos.idcaso ⟶ af_casos.idcaso
af_casos.idprov ⟶ ‘codigos provincia.COD
af_casos.estado ⟶ estados.idestado
af_casos.tipo_documento ⟶ tipo_docs_cod.id
af_cce.idcaso ⟶ af_casos.idcaso
af_cce.idCCE ⟶ cce.idCCE
af_datper.estcivil ⟶ estciviles.idestcivil
af_datper.idcaso ⟶ af_casos.idcaso
af_datper.idprovnac ⟶ ‘codigos provincia‘.COD
af_datper.provtrab ⟶ ‘codigos provincia‘.COD
af_fotos.idcaso ⟶ af_casos.idcaso
af_hipotesis.idcaso ⟶ af_casos.idcaso
af_militancias.idcaso ⟶ af_casos.idcaso
af_militancias.idorga ⟶ orgas.idorga
af_relaciones.idcaso ⟶ af_casos.idcaso
af_relaciones.idprovincia ⟶ ‘codigos provincia‘.COD
af_relaciones.idrelacion ⟶ ‘codigos familia‘.idrelacion

af_secuestros.idcaso ⟶ af_casos.idcaso
af_secuestros.provdes ⟶ ‘codigos provincia‘.COD
archivos.tipoObj ⟶ objetos.idObj
autores.tipoAutor ⟶ tipo_autor.id_tipodeautor
autores.tipoObjeto ⟶ objetos.idObj
causa_actor.idcausa ⟶ causas.idcausa
causa_actor.relacion ⟶ causa_actrel.idrel
causa_actor.tipoObj ⟶ objetos.idObj
causa_indice.idcausa ⟶ causas.idcausa
causa_indice.idfuente ⟶ fuentes.IDfuente
causa_indice.importancia ⟶ codimportancia.idImp
causa_indice.tipo ⟶ causa_tipos.idTipo
causas.idJuzgado ⟶ juzgados.idjuzgado
cce.FUERZA ⟶ fuerzas.idfuerza
cce.prov ⟶ ‘codigos provincia‘.COD
cce.tipoObjeto ⟶ objetos.idObj
fuentes_ap.tipoObjeto ⟶ objetos.idObj
fuentes_refint.idfuente ⟶ fuentes.IDfuente
grados.fuerza ⟶ fuerzas.idfuerza
logs.tipoObjeto ⟶ objetos.idObj
logs.usuario ⟶ usuarios.usuario
orgas.tipoObjeto ⟶ objetos.idObj
rep_apodo.idrepre ⟶ represores.idrepre
rep_cce.idCCE ⟶ cce.idcce
rep_cce.idrepre ⟶ represores.idrepre
rep_destinos.idrepre ⟶ represores.idrepre
rep_enfermedad.idrepre ⟶ represores.idrepre
rep_fotos.idcaso ⟶ represores.idrepre
rep_licencias.idrepre ⟶ represores.idrepre
repre_datper.estcivil ⟶ estciviles.idestcivil
repre_datper.idprovnac ⟶ ‘codigos provincia‘.COD
repre_datper.idrepre ⟶ represores.idrepre
repre_grado.fuerza ⟶ fuerzas.idfuerza
repre_grado.grado ⟶ grados.idgrado
repre_grado.idrepre ⟶ represores.idrepre
represores.fuerza ⟶ fuerzas.idfuerza
represores.tipo_documento ⟶ tipo_docs_cod.id
represores.tipoObjeto ⟶ objetos.idObj
reprevict.fuerza ⟶ fuerzas.idfuerza
reprevict.idrepre ⟶ represores.idrepre
reprevict.idvict ⟶ af_casos.idcaso
reprevict.lugar ⟶ cce.idCCE
reprevict.participacion ⟶ participacion.idparticipacion
searchhistory.tipoObjeto ⟶ objetos.idObj
searchhistory.usuario ⟶ usuarios.idusuario

tarea_obj.tipoObjeto ⟶ objetos.idObj
tarea_usuarios.grupo ⟶ u_grupos.idgrupo
tarea_usuarios.idusuario ⟶ usuarios.idusuario
tareas.estado ⟶ cod_estadotareas.idEstado

C. Dominios y Tipos de datos Enumerados

C.1. Dominios

Tipo de documentos de identidad: `tipo_documento`
Vínculos entre personas y victimas: `vinculo`
Relación entre personas y causas: `relacion_causa`
Los distintos tipos de autores: `tipo_autor`
Acto en el que un represor puede haber participado: `participacion`
Genero de una persona: `sexo`
Estado civil: `estado_civil`
País: `pais`
Provincia: `provincia`
Las distintas fuerzas armadas: `fuerza`
Estado Victima: `estado_victima`
Tipos archivos: `tipo_archivo`
Juzgados: `juzgado`
Tipo de acto procesal: `tipo_acto_procesal`

C.2. Tipos de Datos Enumerados

Estado en que se encuentran las tarea: `estado_tarea`
Importancia de las tareas: `importancia`

D. Reglas para la asignación de nombres a tablas y columnas

- Los nombres de tablas y columnas se escriben en singular, excepto que esto implique que pierdan su semántica.
- Todas las letras de los nombres de tablas y columnas se escriben en minúscula.
- Nombres de tablas o columnas que estén compuestas por más de una palabra se separan utilizando guión bajo "_".
- Al nombre de las columnas utilizadas como clave primaria o clave foránea se agrega el sufijo "id_".
- En las tablas se colocan primero las columnas que son claves primarias, seguidas de las claves foráneas y a continuación el resto.
- Los nombres utilizados deben ser descriptivos de su contenido.
- Los campos de distintas tablas que son utilizados para albergar la misma clase de información deben tener el mismo nombre.
- Los campos de distintas tablas que son utilizados para albergar distinta clase de información deben tener distinto nombre.

E. Diagrama Presentes con claves foráneas implementadas

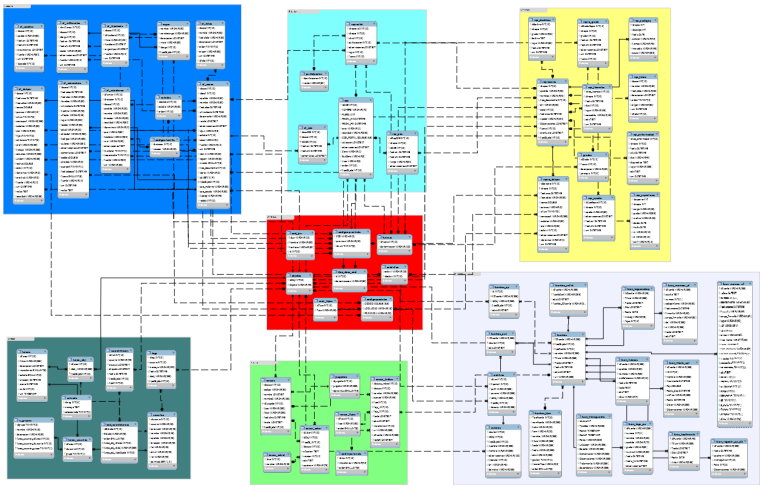
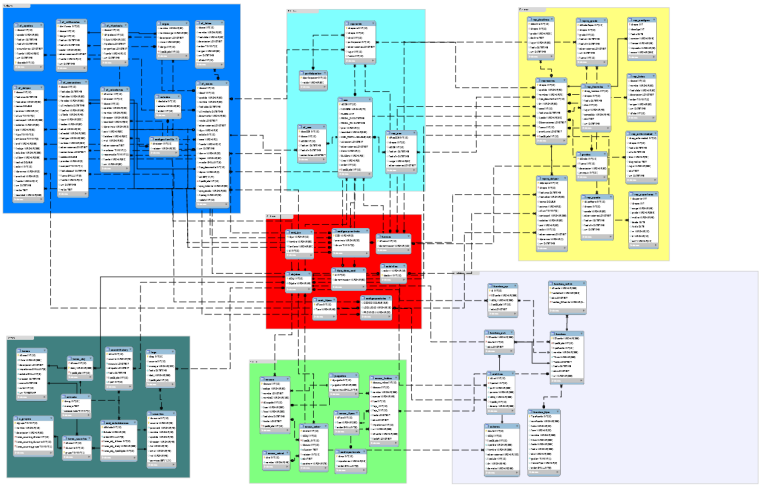


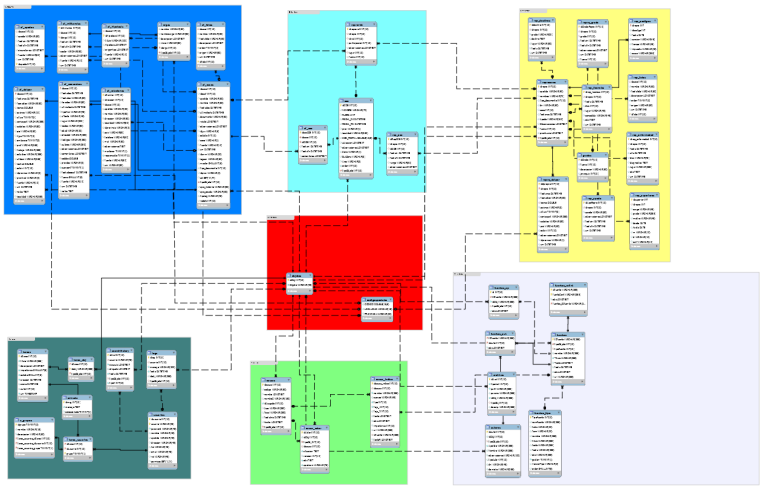
Figura 4. Diagrama Entidad Relación de la base de datos PRESENTES donde se muestran todas las tablas y relaciones entre ellas implementadas mediante el uso de claves foráneas.

F. Diagrama Presentes sin tablas especificas para cada tipo de documento



18

G. Diagrama Presentes sin tablas especificas para cada tipo de documento ni las tablas de códigos



H. Diagrama Rediseño de Presentes

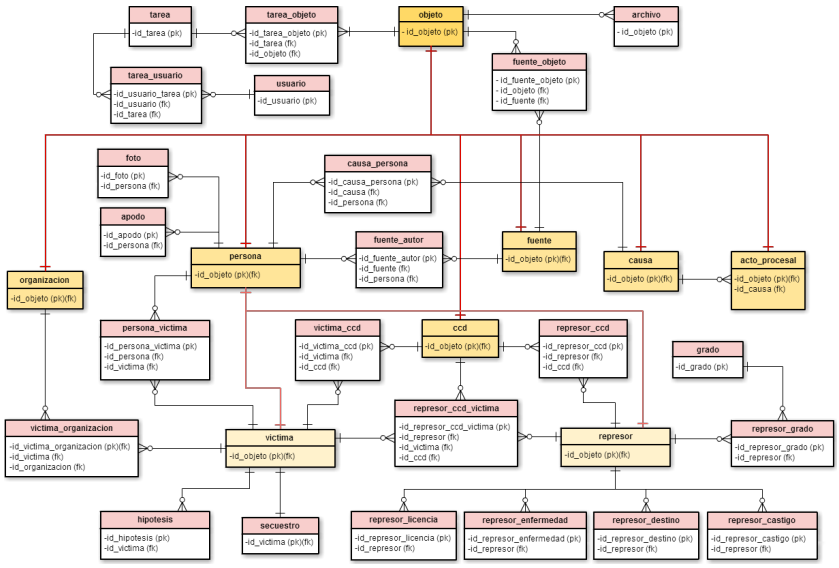


Figura 5. Diagrama Rediseño todas las tablas y solo las columnas que tienen clave primaria o clave foránea.

Referencias

1. Ambler, S.W.: Mapping Objects to Relational Databases (2002)
2. Bauer, C., King, G.: Hibernate in action. Manning Publications Co., 2th edn. (2004)
3. Codd, E.F.: A relational model of data for large shared data banks. Communications of the ACM 13(6), 377–387 (1970)
4. Eisenberg, A., Melton, J., Kulkarni, K., Michels, J.E., Zemke, F.: Sql:2003 has been published. SIGMOD Rec. 33(1), 119–126 (2004)
5. Elmasri, R., Navathe, S.: Fundamentos de Sistemas de Base de Datos. Addison-Wesley, 5th edn. (2007)
6. Group, P.G.D.: PostgreSQL 9.4beta2 Documentation (2014)
7. Mendelzon, A., Schwentick, T., Suciu, D.: Foundations of semistructured data (2001)
8. Oracle y/o afiliados: MySQL 5.6 Reference Manual: Including MySQL Cluster NDB 7.3-7.4 Reference Guide (2014)
9. Pare, R.C., , Santillan, L.A.C., Costa, D.C., Ginesta, M.G., Escofet, C.M., Mora, O.P.: Bases de datos. Fundacio per a la Universitat Oberta de Catalunya, first edition edn. (2005)
10. Silberschatz, A., Korth, H., Sudarshan, S., Pérez, F.: Fundamentos de bases de datos. McGraw-Hill, 4th edn. (2002)