ρDB: Refactorizando Bases de Datos Mediante Ingeniería Inversa

Jessica Mariana Villar-García[1], Miguel Ehécatl Morales-Trujillo[1,2] y Guadalupe Ibargüengoitia[1,2]
[1]*Facultad de Ciencias, Universidad Nacional Autónoma de México, Ciudad de México, México*
[2]*Grupo de Investigación KUALI-KAANS, Universidad Nacional Autónoma de México, Ciudad de México, México*
{jess, migmor, gig}@ciencias.unam.mx

Resumen

Dada la estrecha relación entre los sistemas de software y las bases de datos, se vuelve imprescindible que su interacción sea precisa. Uno de los atributos necesarios para que esta interacción se dé, es contar con una base de datos bien diseñada. Sin embargo generar un "buen" diseño que atienda las necesidades y restricciones particulares de un determinado problema, no es una tarea trivial, mucho menos cuando el sistema se encuentra en operación. Por esta razón, en este artículo se describe a ρDB, un proceso de ingeniería inversa para mejorar el diseño e implementación de bases de datos inmersas en un sistema de software existente y en operación. La validación de ρDB se llevó a cabo mediante un caso práctico, el cual permitió describir y validar su utilidad y pertinencia e identificar sus ventajas y desventajas.

Palabras clave: Ingeniería inversa, refactorización, bases de datos, integridad

1. Introducción

En la actualidad son muchas las actividades en donde están involucrados los sistemas de software, resulta difícil imaginar alguna en la que no se requiera la automatización de sus procesos. Aunado a esto, y dada la relación que guardan los sistemas de software con las bases de datos, es imposible concebir el uno sin el otro. Por ello, las bases de datos se han convertido en un componente esencial para los sistemas de software, en consecuencia no sólo es importante construir buen software sino también diseñar buenas bases de datos.

Por una parte el uso de una base de datos bien diseñada tiene grandes ventajas, pues un buen diseño permite controlar los datos, recuperarlos, ordenarlos, analizarlos, resumirlos e incluso elaborar informes que aporten mayor y mejor información, otorgando así ventajas competitivas y certidumbre al individuo que interpreta los datos.

Por el contrario, las consecuencias de un mal diseño de la base de datos se ven reflejadas de muchas maneras, por ejemplo: el almacenamiento puede no ser el óptimo, la consistencia, integridad y precisión de los datos puede verse afectada o los usuarios pueden tener dificultades a la hora de acceder a los datos, provocando, probablemente, que se produzca información errónea y descontento entre los usuarios y administradores de la base de datos.

Dada la problemática que genera un mal diseño de una base de datos, este artículo tiene como objetivo describir y validar un proceso de ingeniería inversa que permita mejorar el diseño e implementación de una base de datos de un sistema de software en operación. Este proceso ha sido llamado *ρDB*.

La meta será que *ρDB* permita realizar modificaciones o mejoras sin que éstas afecten o alteren las necesidades para las que la base de datos y el sistema de software fueron creados. Además, *ρDB* deberá posibilitar el conservar la mayor cantidad de datos ya almacenados, deshaciéndose de aquellos que se podrían llamar "basura".

La validación se llevará a cabo mediante un caso práctico, es decir, mejorando el diseño de la base de datos de un sistema en operación. Se analizará su comportamiento, se identificarán fallas y con base en ello se definirán las mejoras que posteriormente serán aplicadas al diseño de la base de datos; de esta manera se describirá y validará la utilidad y pertinencia de *ρDB*.

Este artículo está organizado de la siguiente manera, en la sección 2 se exponen los antecedentes y la metodología seguida para la definición del proceso de ingeniería inversa. En la sección 3 se describen cada una de las prácticas que constituyen a *ρDB*. En la sección 4 se presenta el contexto en el cuál se

desarrolló l caso práctico mediante el cual se validó a *ρDB*. Los resultados obtenidos se concentran en la sección 5. Finalmente las conclusiones y el trabajo futuro se presentan en la sección 6.

2. Antecedentes

En esta sección se definirán conceptos importantes como lo son la ingeniería inversa y la refactorización. Además, se definirá la metodología empleada y el marco de trabajo utilizado para la definición de *ρDB*.

2.1. Ingeniería Inversa y Refactorización

Ingeniería inversa y refactorización son dos conceptos que se han estudiado y aplicado de manera efectiva en disciplinas tales como la Ingeniería de Software y las Bases de Datos [2][3][4]. Para este artículo, resulta relevante proveer una definición concreta para cada uno de estos términos. Por una parte definiremos a la Ingeniería Inversa como:

"El proceso de descubrir los principios tecnológicos de un dispositivo, un objeto o sistema, mediante el análisis de su estructura, funcionamiento y operación" [1].

Por otra parte, dentro de la literatura existen varias menciones referentes a métodos o procesos para la refactorización de sistemas y/o bases de datos, entre ellas podemos encontrarla siguiente definición:

"La refactorización es el proceso de modificar un sistema de software de tal manera que dichos cambios no alteren su funcionamiento, en esencia se mejora el diseño de un código que ya ha sido escrito" [2].

Con ambos conceptos definidos, para este artículo, la ingeniería inversa será el medio para lograr una refactorización, es decir, se mejorará una base de datos en operación que carece de documentación, realizando adecuaciones basadas en la extracción de conocimiento e información a nuestro alcance.

Una refactorización a la base de datos permitirá mejorar la estructura del esquema sin alterar la semántica de la información de éste. Así el esquema proporcionará la misma información antes y después de realizar la refactorización [3][4].

Entre las metodologías más utilizadas para la refactorización de bases de datos relacionales podemos mencionar al Desarrollo Guiado por Pruebas de Bases de Datos Relacionales (TDD-RD por sus siglas en inglés) [5]. El objetivo de TDD-RD es realizar pruebas para validar el diseño de la base de datos mediante un proceso iterativo en donde se ejecutan las pruebas durante cada etapa, además de que el diseñador realiza pequeñas modificaciones (refactorización) para mejorar el diseño del código sin alterar la semántica de éste [5].

En la Figura 1 se muestra el proceso de refactorización utilizado por el método TDD-RD, mismo que fue utilizado para la definición del ciclo de vida que seguirían las prácticas de *ρDB*.

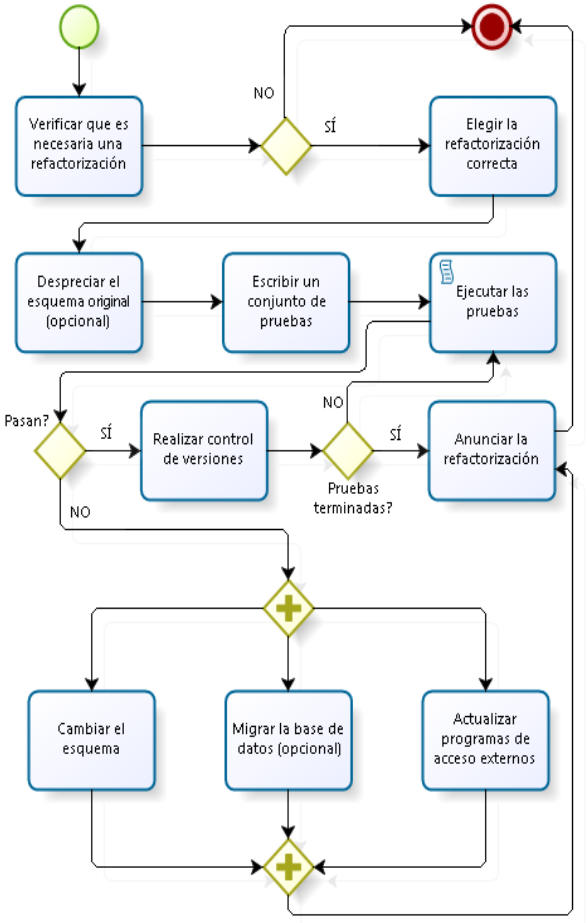

Figura 1. Actividades de TDD-RD, adaptado de [5].

Una oportunidad de mejora encontrada a este método, fue la necesidad profundizar y detallar en qué consiste la actividad de *elegir la refactorización correcta*. La definición de esta actividad en TDD-RD no aporta mayores detalles de cómo ejecutarla y tampoco define a qué aspectos habría que prestar atención para decidir cuál es la correcta.

Por esta razón, la propuesta presentada en este artículo resulta ser un complemento para TDD-RD, pues el objetivo es aportar una guía que permita realizar modificaciones o mejoras a una base de datos sin que éstas cambien o alteren las necesidades para las que fue diseñada, es decir, se pretende proveer de una serie detallada de prácticas que permitan elegir y ejecutar una "refactorización correcta" sobre una base de datos.

2.2. Metodología

Con la intención de establecer una metodología que permitiera identificar las prácticas que compondrían al proceso de ingeniería inversa, se establecieron una serie de pasos que fueron agrupados en las siguientes fases:

1. **Pre-análisis de los sistemas:** Analizar el contexto, necesidades y restricciones de los sistemas candidatos a ser intervenidos.
2. **Selección del sistema:** Tomando en cuenta criterios como la viabilidad y el impacto, elegir un sistema de los analizados previamente para llevar a cabo el proceso de ingeniería inversa.
3. **Análisis del sistema:** Identificar los objetivos para los cuales fue creado el sistema, en particular aquellos ligados al almacenamiento y persistencia de datos. Para ello se debe estudiar la base de datos con la intención de crear una lista de supuestos generales para respondernos la pregunta ¿qué se deseaba modelar?
4. **Diseño de la mejora:** Una vez que se analizó el sistema y se conoció su propósito específico, validar lo que realmente se modeló contra lo que se deseaba modelar. Es en esta fase en la que se proponen las modificaciones necesarias a la base de datos.
5. **Implementación y prueba de la mejora:** Con las modificaciones a la base de datos identificadas, implementar el modelo mejorado en un ambiente de prueba que permita validar su integración con el sistema de software y los datos.

Con el objetivo claro y los pasos a seguir definidos surgió la necesidad de contar con un mecanismo que permitiera expresar, con el nivel de detalle adecuado, cada una de las prácticas que compondrían a *ρDB*.

2.3. Marco de Trabajo para la Definición del Proceso

El marco de trabajo elegido fue KUALI-BEH [6]. Este marco de trabajo está diseñado para permitir a los equipos de trabajo transformar el conocimiento tácito en explícito, al proveerles de un conjunto de conceptos comunes mediante los cuales puedan expresar sus maneras de trabajo y así poderlas compartir, comparar y mejorar.

KUALI-BEH está compuesto por dos vistas y se definen de la siguiente manera:

- **Vista Estática:** define los conceptos comunes utilizados en proyectos de software por los practicantes de Ingeniería de Software. Se basa en dos conceptos fundamentales: método y práctica. Un método será un conjunto de prácticas con un propósito definido, mientras que una práctica representará una secuencia de actividades que indica cómo se desarrollará la manera de trabajo durante un proyecto, cada actividad puede ser detallada con tareas, aunque éstas no son obligatorias.
- **Operacional:** esta vista representará a los practicantes en operación, esto es, se definirá un equipo de trabajo con el objetivo de instanciar métodos y prácticas para cumplir con los objetivos definidos en el proyecto.

Para entender de mejor manera cómo funciona, a manera de ejemplo, en la Figura 2 se muestra la práctica 4 de *ρDB*, expresada usando los conceptos comunes y la plantilla de *Práctica* definidos en KUALI-BEH.

P 4	**Práctica**		
Validación de hipótesis			
Objetivo			
Aclarar detalles importantes respecto a los supuestos establecidos de las características de las entidades así como del modelo relacional.			
Entrada		**Resultado**	
PT: Entrevista [grabada] PT: Diagrama E/R [borrador] PT: Descripción_tabla_atributo [Actualizado] PT: Listado de supuestos [inicial]		Listado de supuestos [Revisado] Diagrama E/R [esbozado] Descripción_tabla_atributo [Detallado]	
Guía			
Actividad 1	Comparar los supuestos y la realidad (Información proporcionada por el diseñador).		
Entrada		**Resultado**	
Entrevista [grabada] Descripción_tabla_atributo [Actualizado]		Listado de supuestos [Revisado] Descripción_tabla_atributo [Detallado]	
Tareas (opcional)	**Herramientas (opcional)**	**Conocimientos y Habilidades**	**Métricas**
Analizar la entrevista detenidamente.	Reproductor de audio		
Actividad 2	Actualizar el diagrama E/R basados en la nueva información		
Entrada		**Resultado**	
Diagrama E/R [borrador] Descripción_tabla_atributo [Detallado]		Diagrama E/R [esbozado]	
Tareas (opcional)	**Herramientas (opcional)**	**Conocimientos y Habilidades**	**Métricas**
	Herramienta de Diagramación (ER Editor)		

Figura 2. Práctica de Validación de Hipótesis.

En ella se pueden observar los elementos propios de la propuesta como son el Objetivo, Entrada, Resultado, Actividades, Tareas y Herramientas, conceptos que resultaron útiles para la definición de *ρDB*.

3. Proceso de Ingeniería Inversa: *ρDB*

En esta sección se presentan las características esperadas de *ρDB*, las prácticas que lo componen, así como la relación entre éstas.

3.1. Características esperadas de *ρDB*

Después de llevar a cabo las fases de la metodología descrita en la sección 2.2, se definieron las siguientes características esperadas para *ρDB*:

- Que al liberar la versión mejorada de la base de datos se asegure que ésta aún cumple con las restricciones y necesidades para las que fue creada.
- Que en conjunto, base de datos y sistema de software, operen de manera correcta.
- Obtener un proceso genérico de ingeniería inversa que permita mejorar una base de datos en operación.

Además dicho proceso deberá:

- Permitir que la intervención se dé de manera iterativa e incremental.
- Cumplir con los principios de buen diseño.
- Minimizar la redundancia en los datos.
- Asegurar la integridad en los datos.
- Satisfacer las necesidades del cliente/usuario
- Atender los supuestos y restricciones establecidos.

3.2. Prácticas de *ρDB*

Con las características del proceso y el marco de trabajo para expresar las prácticas establecidos, se procedió con la definición de *ρDB*. A continuación se presenta el listado de las prácticas que componen a *ρDB* junto con el objetivo de cada una de ellas.

1. **Análisis y búsqueda de supuestos y restricciones**: Conocer los supuestos y restricciones bajo los que fue construida la base de datos. A partir de una copia de la base de datos en operación se genera un documento y un diagrama E/R en el que se describen tanto las tablas como los atributos que conforman a cada una.
2. **Búsqueda de relaciones:** Encontrar las relaciones que existen entre las entidades.
3. **Entrevista:** Entrevistarse con el diseñador u administrador de la base de datos para obtener información de interés.
4. **Validación de hipótesis:** Aclarar detalles importantes respecto a los supuestos establecidos de las características de las entidades así como del modelo relacional.
5. **Búsqueda de errores de integridad:** Analizar cada una de las tablas para determinar cuáles cuentan con errores de integridad.
6. **Análisis y propuesta de integridad:** Analizar el por qué y cómo se generan errores de integridad en cada una de las tablas.
7. **Preparación del ambiente de trabajo:** Se replica el ambiente del sistema en operación en un ambiente de desarrollo.
8. **Explotación de la base de datos en operación:** Interactuar con la base de datos en operación para observar el comportamiento e identificar errores específicos.
9. **Análisis de la navegación del sistema y su relación con las tablas de la base de datos:** Mediante el análisis de la navegación que se realizó, identificar qué tablas son utilizadas dentro de éste y de qué manera.
10. **Actualización del modelo de la base de datos:** Mejorar las hipótesis que se tenían en el Diagrama E/R y/o agregar características faltantes, ya sea de las tablas o de los atributos.
11. **Primera intervención de la base de datos (Tablas inútiles):** Identificar y eliminar las tablas inútiles de la base de datos.
12. **Segunda intervención de la base de datos (Mejoras de integridad):** Implementar las restricciones de integridad a la base de datos.
13. **Liberar la base de datos intervenida para su integración con el sistema en operación:** Liberar la versión mejorada de la base de datos asegurando que cumple con las restricciones y necesidades para las que fue creada.

En la Figura 3 se muestran las relaciones entre las 13 prácticas que conformar a *ρDB*.

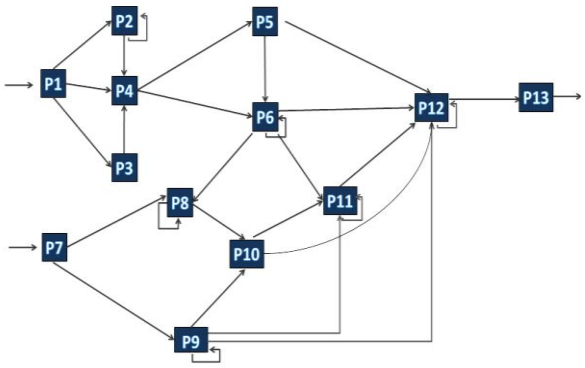

Figura 3.Diagrama de *ρDB*.

3.3. Productos de Trabajo de *ρDB*

Los productos de trabajo necesarios durante la ejecución del método fueron:

- **Diagrama E/R:** Diagrama Entidad/Relación asociado a la base de datos inmersa en el sistema.
- **Descripción de tablas y atributos:** Nociones de lo que modela cada tabla y atributo.
- **Entrevista:** Grabación de la entrevista realizada al diseñador de la base de datos.
- **Reportes de Integridad:** Información obtenida al realizar el análisis de integridad de entidad, dominio, nulidad y referencial de la base de datos, así como la forma normal en la que se encontraba cada tabla.
- **Factores de error:** Posibles factores que dan lugar a los errores de integridad en cada tabla.
- **Propuesta de mejoras:** Listado de propuestas a modificar en la base de datos.
- **Respaldo:** Respaldo de la base de datos en operación.
- **Errores en la aplicación:** Errores encontrados durante el uso de la aplicación.
- **Diagramas de estado:** Diagramas que modelan el comportamiento del sistema.
- **Tablas inútiles:** Script para el borrado de tablas inútiles.
- **Interacción SQL:** Listado de archivos (clases) que interactúan con la base de datos.
- **Búsqueda Relaciones SQL:** Script con consultas para adquirir información sobre las relaciones existentes entre las distintas tablas.
- **Integridad SQL:** Script con consultas para analizar la integridad de dominio, entidad, nulidad y referencial de la base de datos, así como la forma normal en que se encontraba cada tabla, ver Figura 4.
- **Scripts de conteo y borrado:** Script con consultas para el conteo y/o borrado de tuplas de cada tabla.
- **Respaldo Integridad SQL [1, 2, 3, 4]:** Respaldo de la base de datos después de aplicar las propuestas de integridad de entidad (1); después de aplicar las propuestas de integridad de nulidad (2); después de aplicar las propuestas de integridad de dominio (3); o después de aplicar las propuestas de integridad referencial (4).
- **Datos de Acceso:** Permisos requeridos para manipular la base de datos.
- **Reporte de Consultas:** Reporte de las consultas realizadas para obtener información de las relaciones existentes entre las tablas de la base de datos.
- **Preguntas para entrevista:** Listado de preguntas a realizar durante la entrevista con el diseñador de la base de datos.
- **Listado de supuestos:** Lista de supuestos realizados respecto a la base de datos, como pueden ser las características de tablas y atributos.
- **Respaldo Intervención SQL [1, 2, 3]:** Respaldo de la base de datos previo a la primera intervención (1); previo a la segunda intervención (2); o previo a la tercera intervención (3).

4. Caso Práctico

Con *ρDB* documentado, se procedió a validarlo. Dicha acción se llevó a cabo utilizando un sistema en operación de la División de Estudios de Posgrado de la Facultad de Medicina de la Universidad Nacional Autónoma de México. El contexto de esta dependencia se detalla a continuación.

4.1. Contexto

La División de Estudios de Posgrado de la Facultad de Medicina está encargada de administrar las Especialidades Médicas que forman parte de su plan de estudios. En la actualidad la División cuenta con una serie de sistemas de software administrativos para solventar sus necesidades de funcionamiento, el área encargada de administrar cada uno de estos sistemas es el Departamento de Cómputo e Informática (DCI) del mismo Posgrado de Medicina.

La problemática más importante del DCI es que muchos de los sistemas que actualmente tiene a su cargo son sistemas "heredados" de otras administraciones o subdirecciones por lo que hay una carencia de documentación desactualizada o, aún más grave, inexistente.

Hoy en día sólo cinco de los sistemas que administra han sido creados por el DCI. Entre los principales usuarios que interactúan con sus sistemas se tienen a:

- Personal administrativo, propio de la División de Estudios de Posgrado
- Jefes de Enseñanza del posgrado
- Alumnos
- Profesores.

Con el contexto anterior es importante no perder de vista el nivel de satisfacción de los usuarios de los sistemas, ni las necesidades tecnológicas propias de la

disciplina, situación que obligan a mantener a los sistemas en una constante evolución y mejora.

Sin embargo dicha mejora de los sistemas se dificulta de sobremanera cuando no existe la documentación adecuada que soporte un proceso de mantenimiento sobre éstos.

Considerando estas características tan específicas, *ρDB* intenta aprovechar las ventajas que ofrece la ingeniería inversa y actuar como herramienta para que el DCI refactorice sus bases de datos y en consecuencia mejore sus sistemas de software en operación.

4.2. Ejecución de *ρDB*

Para llevar a cabo la validación de *ρDB* se realizó un análisis exploratorio para cada uno de los sistemas que se encontraban en operación en el DCI, de esta manera fue posible definir las características con las que contaba cada sistema y así seleccionar aquel que fuese más relevante para este trabajo.

Después de analizar los sistemas del DCI, resultó ser de interés el *Sistema de Razonamiento Ético en la Clínica*. Este sistema fue heredado al DCI y su objetivo es fortalecer las debilidades éticas de los médicos y así mejorar la práctica de su profesión a través de un curso en línea. Por ello está orientado tanto para alumnos de residencia como para profesores. El contenido del sistema, está coordinado por el Consejo Técnico de la Facultad de Medicina.

Con el sistema elegido, comenzó la ejecución de *ρDB*. La primera práctica a ejecutar fue *Análisis y búsqueda de supuestos y restricciones*, a través de la cual se logró obtener una primera apreciación de la base de datos y así fue posible describir algunos de sus atributos y las tablas inmersas en ésta. El entendimiento de los supuestos identificados se logró mejorar después de ejecutar la práctica 2 *Búsqueda de relaciones*, lo que permitió detectar relaciones existentes entre las entidades.

Al llevar a cabo la práctica 3, *Entrevista*, se tuvo una reunión en el DCI con el diseñador de la base de datos del Sistema de Ética, lo que permitió comprender varios aspectos del diseño de ésta y resolvió detalles al respecto.

Una vez que se realizó la entrevista con el diseñador de la base de datos fue posible precisar detalles tanto del modelo relacional como de las características de tablas y atributos, al ejecutar la práctica 4 *Validación de hipótesis*, que previamente el equipo de trabajo había establecido. Después de analizar la base de datos y conocer los objetivos para los que fue creada, así como las características de las entidades y atributos inmersos en ella, entre otras características; la práctica 5 fue en donde se comenzaron a identificar y analizar los errores de integridad con los que contaba el diseño de la base de datos.

Dentro de las oportunidades de mejora para el diseño de la base de datos se reportaron las siguientes:

- No se contaba con restricciones de integridad.
- Existía nula normalización debido a que ninguna de las tablas contaba con llave primaria (PK), en consecuencia el modelo no se encontraba en Primera Forma Normal (1FN).
- Ninguna de las dependencias funcionales que se identificaron estaba siendo respetada, debido a la inexistencia de restricciones que vigilaran cuestiones de integridad y a que la base de datos no estaba normalizada.
- Existían datos duplicados en varias de las tablas inmersas en la base de datos.
- No se contaba con un diagrama E/R que representara el diseño de la base de datos ni ningún otro documento que reflejara el modelado del sistema.

Durante la práctica 6 se hizo un análisis de las causas que podrían estar generando errores de integridad en la base de datos, con ello se mejoró el Diagrama E/R anteriormente modelado y se hizo una propuesta de mejoras a aplicar.

Al ejecutar la práctica 7 *Preparación del ambiente de trabajo*, se solicitaron los datos de acceso necesarios, como el acceso al Sistema de Ética, a la base de datos en operación, entre otros más; esto con el objetivo de contar con el ambiente de trabajo requerido para continuar con el diseño de la mejora. Una vez que se tuvo validado el ambiente de trabajo, el siguiente paso fue analizar la base de datos en operación, durante la práctica 8; ello permitiría detectar errores específicos en el funcionamiento de ésta.

En la figura 4 se muestra un ejemplo de las consultas utilizadas para llevar a cabo dicho análisis.

```
--INTEGRIDAD DE ENTIDAD

SELECT COUNT (DISTINCT clave)
FROM acceso;

SELECT clave, COUNT(*) AS repeticiones
FROM acceso
WHERE clave IS NOT NULL
GROUP BY clave
HAVING COUNT(*) > 1
ORDER BY repeticiones;

SELECT COUNT (DISTINCT anores)
FROM acceso;
SELECT anores, COUNT(*)
FROM acceso
GROUP BY anores;
```

Figura 4. Consultas de integridad en pgAdminIII.

En la práctica 9 *Análisis de la navegación del sistema y su relación con las tablas de la base de datos* se analizó el comportamiento entre la base de datos y el Sistema de Ética al navegar en éste. De esta manera se identificaron aquellas tablas que eran utilizadas y de qué manera (inserciones, consultas, actualizaciones o borrados). Enseguida, con la práctica 10 fue posible mejorar el modelo de la base de datos siendo éste el modelo refactorizado que se aplicaría durante la primera intervención a la base de datos en operación.

Durante la primera intervención, práctica 11, se detectaron y eliminaron las tablas que resultaban inútiles dentro del diseño de la base de datos. Para después aplicar una segunda intervención, encargada en la práctica 12 y así implementar las restricciones de integridad propuestas.

Finalmente durante la práctica 13 se liberó la versión mejorada de la base de datos después de asegurar que cumplía con las restricciones y objetivos para los que fue creada.

El modelo resultante de la base de datos se muestra en la Figura 5.

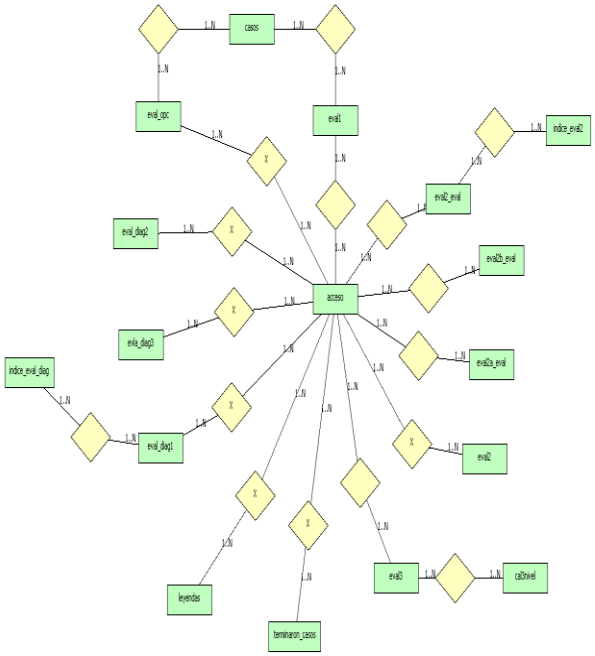

Figura 5. Diagrama E/R propuesto.

4.3. Mejoras Obtenidas

Entre las mejoras obtenidas más importantes se pueden mencionar las siguientes:

- Se eliminaron 21 tablas inútiles dado que no tenían relación con los requerimientos de la base de datos.
- Se agregaron 17 PKs y 17 FKs a las tablas ya que ninguna contaba con ellas.
- Se detectaron y eliminaron registros duplicados.
- Se eliminaron 1,638 tuplas espurias de las relaciones.
- Se implementaron 89 restricciones de no-nulidad y 46 restricciones de dominio a cada uno de los atributos que así lo requerían.
- Se generó un diagrama E/R que modela el diseño mejorado y real de la base de datos.
- Se realizaron y reportaron modificaciones a los archivos PHP de la aplicación, resultando en una mejora del sistema en operación.

Al llevar a cabo la refactorización, el diseño de la base de datos inmersa en el Sistema de Ética mejoró, obteniendo los siguientes efectos:

- La integridad de los datos aumentó.
- Las independencias de datos lógica y física fueron preservadas.
- No hubo pérdida de información.

5. Resultados

Al proponer de manera teórica a *ρDB*, éste resultó ser un proceso con una lista de productos de trabajo demasiado extensa, lo que auguraba cierta complejidad al momento de ejecutarlo. Al llevar a cabo el caso práctico fue posible notar que existían productos de trabajo que no resultaban relevantes para la toma de decisiones o cuyo esfuerzo para generarlos sobrepasaba el beneficio de contar con ellos.

Derivado de esta experiencia los productos de trabajo requeridos se redujeron. Además fue posible revalorar aquellos productos que se les estaba dando menor importancia que a otros cuando realmente éstos sí la tenían. De igual manera, reafirmamos el hecho de que contar con un diagrama E/R resulta ser bastante útil, por ejemplo, para el trabajo a futuro que se desee hacer con la base de datos; además de que generar un diagrama E/R no implica un esfuerzo grande, magnificando el retorno de inversión (ROI).

Por otro lado uno de los elementos de la propuesta de KUALI-BEH que no se utilizó fueron los criterios de verificación, porque al ejecutar cada una de las prácticas propuestas resultaba evidente si se había cumplido o no el objetivo de éstas, por ello no fue necesario especificar criterios que permitieran discernir si la meta se había logrado o no.

Finalmente el caso práctico permitió notar qué tan ágil resultó ser *ρDB*, es decir, fue posible detectar cuándo una práctica resultaba muy corta y podría unirse con alguna contigua a ésta o proponerse como actividad en otra práctica o por el contrario, detectar aquellas prácticas extensas y complejas por lo que

convenía dividirlas para que resultara más fácil su ejecución.

Si bien somos conscientes de que el tiempo de ejecución del proceso variará de acuerdo a la naturaleza y características particulares de cada sistema en operación, en este caso práctico el esfuerzo de aplicar *ρDB* fue de 108 horas distribuidas en un periodo de 15 semanas. Valores que resultaron razonables con respecto al beneficio obtenido para el CDI y sus usuarios, considerando que tanto la base de datos como el sistema en operación se encontraron siempre disponibles para el usuario final, antes, durante y después de la intervención a la base de datos.

5.1 Ventajas y desventajas de *ρDB*

La ejecución del caso práctico hizo posible identificar algunas ventajas y desventajas de *ρDB*. Entre las ventajas encontradas se pueden mencionar:

- Se identifican errores en el diseño del sistema de software y la base de datos.
- Se determina qué tan adecuada es la interacción entre el sistema y la base de datos.
- Se verifica y valida que realmente se cumplen los requerimientos que el cliente y/o usuario solicitó.
- Se clarifican las razones de ser, tanto de la base de datos como del sistema de software.
- Se genera documentación actualizada, antes inexistente, a la par de la refactorización.

La desventaja más importante fue:

- El esfuerzo requerido para refactorizar o no una base de datos se hace visible hasta la práctica 8, lo que consideramos algo tardío, ya que en ese punto podría concluirse que resultará mejor diseñar una nueva base de datos en lugar de refactorizarla. Cabe mencionar que dicha decisión tampoco puede ser tomada antes.

6. Conclusiones y Trabajo Futuro

La motivación para realizar este trabajo surgió de la necesidad de contar con un buen diseño tanto de un sistema de software como de una base de datos para que el desempeño en conjunto de éstos resultara exitoso. El contexto elegido para este trabajo fueron las bases de datos inmersas en un sistema de software en operación y de esta manera como objetivo se planteó proponer un proceso de ingeniería inversa que permitiera al administrador realizar modificaciones, es decir una refactorización, al diseño de ésta sin alterar o modificar las necesidades para las que la base de datos y el sistema fueron diseñados.

Para lograr esta meta se propuso a *ρDB*, proceso de ingeniería inversa, al que mediante un caso práctico se pudo validar su pertinencia y utilidad, para así corroborar que dicho proceso cumpliera con su objetivo. Al llevar a cabo el caso práctico se confirmó la importancia que tiene el contar con un buen diseño de una base de datos, ya que éste impacta de manera directa en el funcionamiento integral del sistema.

En conclusión el trabajo que se realizó cumple con las expectativas esperadas, el objetivo propuesto se logró y ofrece al lector un manual (*ρDB*) que permite mejorar el diseño de una base de datos inmersa en un sistema de software en operación.

Finalmente, como trabajo futuro, se requieren de más casos prácticos para reafirmar la pertinencia y utilidad de *ρDB*, será conveniente probarlo en bases de datos donde existan interacciones más complejas. Además, el proceso obtenido puede ser mejorado e incluso complementado con algún otro método para hacerlo más ágil.

Agradecimientos

Los autores agradecen a Sergio Villa, David Velázquez y Erick Matla, equipo de trabajo del DCI, por el apoyo otorgado durante el desarrollo del caso práctico. Este trabajo fue posible gracias al apoyo del Posgrado en Ciencia e Ingeniería de la Computación (PCIC-UNAM) y al Programa de Apoyo a los Estudios de Posgrado (PAEP-UNAM).

Referencias

1. Juárez-Ramírez R., Licea G., Cristóbal-Salas, A. (2007). *Ingeniería Inversa y Reingeniería Aplicadas a Proyectos de Software Desarrollados por Alumnos de Nivel Licenciatura*. Sistemas, Cibernética e Informática, Vol. 4, No. 2, pp. 48-53.
2. Fowler, M. (1999) *Refactoring: Improving the Design of existing Code*. Boston, MA, EUA, Addison-Wesley Longman Publishing Co., Inc.
3. Ambler, S. (2003). *Agile Database Techniques: Effective Strategies for the Agile Software Developer*. John Wiley & Sons, Inc.
4. Ambler, S., Sadalage, P. (2006). R*efactoring Databases: Evolutionary Database Design*. Addison-Wesley Professional.
5. Ambler, S. (2007). *Test-Driven Development of Relational Databases*. IEEE Software, Vol. 24, No. 3, pp. 37-43.
6. Morales, M., Oktaba, H. (2012). *KUALI-BEH Kernel Extension. Annex B (Normative)* in: Essence – Kernel and Language for Software Engineering Methods. Object Management Group.