VNIVERSIDAD NACIONAL
AVTNOMA DE
MEXICO

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE CIENCIAS

"Proceso de Ingeniería Inversa para la Mejora de Bases de Datos de Sistemas de Software en Operación: Un Caso Práctico."

T E S I S

QUE PARA OBTENER EL TÍTULO DE:

ACTUARIA

P R E S E N T A:

JESSICA MARIANA VILLAR GARCÍA

DIRECTOR DE TESIS:
M. EN I. MIGUEL EHÉCATL MORALES TRUJILLO

MÉXICO, D.F. **2015**

1. Datos del alumno
Villar
García
Jessica Mariana
Universidad Nacional Autónoma de México
Facultad de Ciencias
Actuaría
308279393

2. Datos del tutor
M. en I.
Miguel Ehécatl
Morales
Trujillo

3. Datos del sinodal 1
Dra.
Hanna Jadwiga
Oktaba

4. Datos del sinodal 2
M. en I.
Gerardo
Avilés
Rosas

5. Datos del sinodal 3
Dr.
Javier
García
García

6. Datos del sinodal 4
M. en C.
María Guadalupe Elena
Ibargüengoitia
González

7. Datos del trabajo escrito
Proceso de Ingeniería Inversa para la Mejora de Bases de Datos de Sistemas de Software en Operación: Un Caso Práctico.
76 p
2015

Agradecimientos

Agradezco sinceramente a mi asesor, el M. en I. Miguel Morales, por todo el apoyo que me ofreció durante el desarrollo de la tesis; por ser, además de excelente profesor, una persona tan entregada y motivadora. Muchas gracias Miguel por el conocimiento, no sólo académico, que has compartido conmigo. Gracias también a cada uno de los sinodales por sus comentarios y observaciones que sin duda hicieron de la tesis un mejor trabajo.

Aprovechando el espacio también quiero agradecer a mi equipo de trabajo durante gran parte de mi carrera: Beto, Diego, Brayam, Emma, Angie y Liz, muchas gracias por hacer de la carrera la más amena etapa de mi vida, agradezco mucho su apoyo en aquellos momentos de estrés y también por convertirlos en algo tan divertido y grato. Amigos, llegamos juntos al final y espero nuestra relación no quede ahí…

Finalmente quiero dedicar este trabajo a mis abuelos, que sin duda me hubiera encantado compartir este gran logro a su lado.

"A journey of a thousand miles begins with a single step"
Lao Tzu

"Don't be afraid to give up the good to go for the great"
John D. Rockefeller

Índice General

1. Introducción

En la actualidad las actividades en donde están involucrados los sistemas de software son muchas, resulta difícil imaginar alguna en la que no se requiera la automatización de sus procesos. Esta abundante presencia convierte a los sistemas en un elemento crítico para el buen funcionamiento de muchos ámbitos de la sociedad.

Sin embargo, 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 éstos.

En consecuencia no sólo es importante construir un 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 ésta.

Es por ello que cuando un mal diseño de la base de datos llega a ser implementado dentro de un sistema de software, las consecuencias afectarán directamente, y de manera negativa, al negocio en el que se utiliza. Sin embargo el generar un buen diseño, que atienda las necesidades y restricciones particulares de un determinado problema, no es una tarea trivial, lo que ocasiona que un buen número de bases de datos presenten problemas o deficiencias en su diseño.

Para llevar a cabo la explotación y un mejor aprovechamiento de una base de datos, normalmente éstas interactúan con sistemas de software. Los sistemas de software están presentes en un sinnúmero de entornos: en instituciones bancarias, en una empresa para la ayuda de toma de decisiones, para automatizar procesos y un largo etcétera. Sin duda las consecuencias de una mala aplicación o fallas en el sistema de software, originadas por un mal diseño, pueden llegar a ser cruciales dentro del negocio en que esté inmerso.

En particular, para esta tesis, resulta relevante un contexto en particular, el Departamento de Cómputo de la División de Estudios de Posgrado de la Facultad de Medicina[1]. Este Departamento está a cargo de la administración de distintos sistemas de software. Dichos

[1] División de Estudios de Posgrado de la Facultad de Medicina, http://www.fmposgrado.unam.mx/

sistemas llevan a cabo diversas operaciones, tales como administración de usuarios de las redes del Posgrado, servicio de atención y apoyo para alumnos y profesores, reporte para corrección de fallas y soporte técnico, entre otras.

La mayoría de los sistemas que controla el Departamento de Cómputo fueron desarrollados dentro del Posgrado pero no por el Departamento, lo que ha resultado en problemas de operación y mantenimiento para la actual administración.

La problemática de la División, recae en que algunos de sus sistemas presentan un número considerable de fallas relativas a las bases de datos, lo que impacta directamente en la calidad de los datos que los sistemas de software manejan, por ejemplo, existe redundancia de información, inconsistencia de datos y en general un mal diseño de éstas, provocando malestar en los usuarios y el personal que los administra.

Tomando en cuenta lo anterior, esta tesis tiene como objetivo describir y validar un proceso de ingeniería inversa – el cual llamaremos ρDB – para mejorar el diseño e implementación de las bases de datos de sistemas de software existentes y en operación.

La pertinencia y utilidad del método anterior será validada a través de un caso práctico dentro de la División de Estudios de Posgrado de la Facultad de Medicina, es decir, se analizarán los sistemas de software en operación de la División, de los cuales se elegirá uno, para el cual se propondrá una mejora que será implementada en su base de datos con el fin de obtener un buen diseño.

Con el objetivo de esta tesis definido, a continuación se describirán a mayor detalle aquellos conceptos básicos necesarios para la realización de la misma.

1.1. Bases de datos

Las bases de datos juegan un papel muy importante en casi cualquier área en donde se utilice una computadora, están presentes en distintos negocios, como lo son la ingeniería, la medicina, el derecho, la educación y la ciencia. Dicho término es fundamental para este trabajo por lo que se comenzará definiendo qué es una base de datos.

"Una base de datos es un conjunto de datos relacionados entre sí que contienen información relevante para alguna empresa." (Silberschatz et. al., 2010).

La manera de acceder a los datos se hace mediante un Sistema Manejador de Bases de Datos (SMBD), el cual tiene como objetivo principal el almacenamiento y recuperación de la información de una manera eficiente y práctica.

Consideraremos a un SMBD como una colección de programas que le permite al usuario crear y administrar una base de datos; éste facilita el proceso de definición, creación y manipulación de la base de datos entre varios usuarios y aplicaciones (García-Molina et. al., 2009).

La creación de una base de datos debe realizarse desde el modelado de la misma. Los modelos de bases de datos se definen como una *colección de herramientas conceptuales para describir los datos, las relaciones, la semántica y las restricciones de consistencia* (Silberschatz et. al., 2010). Entre ellos se encuentran:

- *Bases de datos transaccionales:* Su fin principal es recolectar y recuperar datos a la mayor velocidad posible. Están dirigidas al entorno de análisis de calidad, datos de producción e industrial (Netronycs).
- *Bases de datos jerárquicas:* Este modelo consiste en organizar los datos en una forma similar a un árbol, en donde un nodo padre de información puede tener varios hijos. Estas bases de datos son muy útiles para manipular un gran volumen de información, su principal limitación es la poca eficiencia en la redundancia de datos (Netronycs).
- *Bases de datos relacionales:* En este modelo un grupo de tablas representan los datos y las relaciones entre ellos. Las tablas se componen por varias columnas, y cada columna tiene un nombre único (Silberschatz et. al., 2010).
- *Base de datos Entidad – Relación:* Consta de una colección de objetos básicos, llamados *entidades,* – «cosa» u «objeto» en el mundo real que es distinguible de otros objetos – de *relaciones* entre estos objetos – asociación entre varias entidades – y las entidades se describen mediante un conjunto de *atributos* (Silberschatz et. al., 2010).

En particular para este trabajo, los modelos utilizados serán el Entidad-Relación y el Relacional, pues el objetivo será transformar una base de datos *problemática* en una base de datos bien diseñada. Para ello primero se descubrirá el diseño con el que cuenta, mismo que será modelado mediante el modelo Entidad-Relación y posteriormente se irá modificando y transformando hasta convertirse en un modelo Relacional. Por lo anterior ambos modelos serán de interés y serán detallados en las siguientes subsecciones.

1.1.1 Modelo Entidad-Relación

El Modelo Entidad-Relación fue desarrollado por el Dr. Peter Chen, éste fue publicado en el artículo *"The Entity Relationship Model - Toward A Unified View of Data"* (Chen, 1976), en 1976.

Una característica relevante de este modelo es que supone, de manera natural, que todo lo que podamos modelar del mundo real se basa en entidades y relaciones (Chen, 1976); cabe mencionar que éste está sustentado por teoría de conjuntos.

En el modelo Entidad-Relación hay tres conceptos básicos: *entidad*, *relación* y *atributo*, enseguida se definirán cada uno de ellos.

1. Entidad: este concepto lo podemos definir como cualquier cosa u objeto que podamos identificar o distinguir fácilmente.
2. Relación: dado un conjunto de entidades podemos crear asociaciones o *relaciones* entre éstas.
3. Atributo: un atributo nos permite expresar las características o información que guarda una entidad o una relación.

Una peculiaridad de este modelo es el *Diagrama Entidad-Relación*, éste nos permite modelar visualmente la base de datos. Las características de este diagrama son:

- Cada Entidad será representada por un rectángulo,
- Las Relaciones las modelaremos con rombos y
- Los Atributos estarán representados por óvalos.

En la Figura 1 se ejemplifican las características antes mencionadas; se describe un caso sencillo en donde se tienen dos entidades: *ALUMNO* y *CARRERA,* y entre éstas la relación *ESTUDIA*, lo cual indica que un alumno estudia alguna carrera, mientras que *#Cuenta* y *Nombre* son atributos para cada entidad respectivamente.

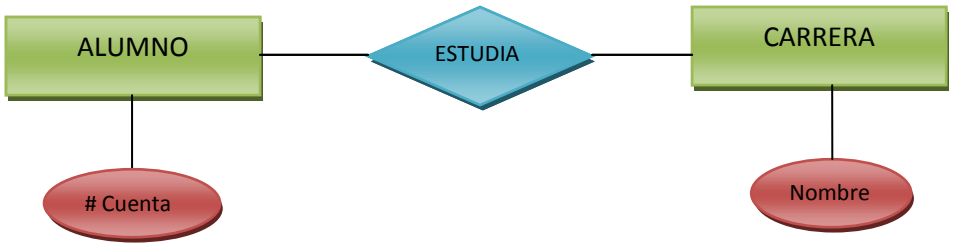


Figura 1. Diagrama Entidad-Relación

Este modelo se utilizará debido al alto grado de expresividad, su simplicidad y por la fácil comprensión por parte de los clientes y usuarios que proporciona.

1.1.2 Modelo Relacional

El Modelo Relacional fue propuesto por Edgar F. Codd en 1969, la idea básica de dicho modelo es que los datos están representados por ***tuplas*** y éstas a su vez están agrupadas en ***relaciones*** (que no se debe confundir con el término homónimo presentado en el modelo Entidad-Relación)

Codd asume que todos los datos serán definidos por una ***relación n-aria*** y define a ésta como un subconjunto del producto cartesiano de n conjuntos – no necesariamente distintos (S_1,…, S_n) en donde S_j será el j-ésimo ***dominio*** (Codd, 1969). Las relaciones las podemos representar mediante arreglos, y con ello definir qué es una ***tupla,*** la cual la entenderemos como el n-ésimo renglón de dicho arreglo.

Dada una relación definida sobre un conjunto de dominios, se tendrán valores de un dominio (o combinación de varios) que identifiquen de manera única a cada tupla, dichos valores se definen como ***llave primaria.*** Por otro lado, si existen relaciones con referencias en una relación diferente, <<referencias cruzadas>>, una ***llave foránea*** nos permitirá acceder a éstas referencias; definiremos ***llave foránea*** como el conjunto de dominios de una relación R que no son llaves primarias de dicha relación pero sí lo son de alguna otra relación S.

Este modelo se utilizará principalmente por los sólidos fundamentos teóricos que posee, además de ser ampliamente utilizado en la industria.

1.2. Sistemas de software

Un sistema de software es un conjunto de programas que administran, controlan y facilitan el acceso a los recursos de una computadora y que además funge como intermediario entre el usuario y el hardware (Weitzenfeld, 2005).

Dado el gran avance de la tecnología, hoy en día hay una gran lista de negocios en donde los sistemas de software son empleados, algunos ejemplos de éstos son: la medicina, las finanzas – como pueden ser casas de bolsa o bancos – compañías de comunicación e incluso en agencias espaciales.

Durante la creación de un sistema de software resaltan varios detalles a los que se debe prestar atención, de no ser así éstos pueden generar problemas o fallas cuando el sistema se encuentre en uso, lo que claramente no resultará satisfactorio para el negocio en donde éste esté inmerso. Es por ello que la labor de diseñar correctamente un sistema de software cobra importancia, a

continuación se definirá y analizará a la disciplina encargada de ellos, la ***Ingeniería de Software.***

La **Ingeniería de Software** es la disciplina que comprende todos los aspectos de la producción de software, desde la especificación del sistema hasta el mantenimiento de éste después de que se utiliza (Weitzenfeld, 2005).

Generalmente las fases que componen a un proceso de desarrollo de software son:

- *Análisis:* se basa en el análisis de los requerimientos y necesidades del cliente, tiene como objetivo definir y analizar cuáles serán las funcionalidades del sistema de software.
- *Diseño:* En este paso se toman las decisiones estratégicas sobre cómo organizar la funcionalidad del sistema en torno al ambiente de implementación (Weitzenfeld, 2005). Es aquí cuando se especifica la arquitectura del sistema, el lenguaje de programación y el sistema manejador de las bases de datos.
- *Implementación:* una vez que se conocen las decisiones estratégicas en la etapa anterior, es momento de generar el código del sistema. En esta fase se traducirán todos los términos utilizados en el diseño a términos y propiedades del lenguaje de programación.
- *Integración:* al concluir la generación del código es momento de conjuntar a todos los módulos construidos en la etapa anterior. Durante la integración se implementan las interfaces entre ellos y se verifica que el sistema en conjunto funciona correctamente.
- *Pruebas:* en esta etapa se revisará la calidad del sistema, se llevaran a cabo pruebas de *verificación* – debemos revisar que la construcción del sistema se esté llevando a cabo correctamente – y de *validación* – analizar si el sistema cumple con los requerimientos establecidos –.
- *Operación:* una vez que se ha creado el sistema, éste debe ponerse en marcha, el principal objetivo será realizar correcciones de todos los errores que surjan durante la ejecución del sistema, así como llevar a cabo mejoras de implementación.
- *Mantenimiento:* el objetivo será considerar las posibles extensiones del sistema, básicamente se realizarán las etapas anteriores a partir de un sistema ya existente.

Además de contar con todo un proceso de acuerdo a lo definido por la Ingeniería de Software, también es necesario tomar en cuenta y analizar aspectos relevantes que nos permitan tener un mejor diseño, como lo son:

- *Complejidad:* Este concepto estará siempre relacionado con el tamaño del sistema, pues entre más grande sea éste mayor será su complejidad, y ésta, a su vez, está afectada por dos factores:

1. **El Problema:** Se refiere al número de requerimientos que deberá satisfacer el sistema, es claro que entre más sean las funcionalidades el sistema será más grande y difícil de comprender. Si quisiéramos reducir la complejidad se deberá reducir la funcionalidad con la que el sistema cuente.
2. **La Solución:** Este aspecto está relacionado con el diseño del sistema y éste deberá satisfacer los requerimientos del problema (Weitzenfeld, 2005). Cuando contamos con una complejidad muy grande y que es difícil de reducir, tendremos que enfocarnos en disminuir la complejidad de la solución.

- *Confiabilidad:* La confiabilidad nos permitirá saber qué tan *bueno* y a prueba de fallas es el sistema (Weitzenfeld, 2005). Ésta dependerá de la cantidad de errores que produzca el sistema, la confiabilidad será mayor mientras dicha cantidad sea pequeña.
- *Robustez:* Este aspecto describirá la capacidad que tiene el sistema para responder ante situaciones anormales (Weitzenfeld, 2005). Cabe mencionar que es difícil diseñar un software 100% confiable y robusto, es decir, perfecto; pero sí es importante disminuir lo más que se pueda la cantidad de errores y/o fallas que éste genere.
- *Software suficientemente bueno:* En primera instancia podríamos decir si un sistema es bueno o no si cumple cada uno de los requerimientos del problema y mejor aún si cumple eso y más, sin embargo existen dos factores que caracterizan el éxito de un sistema, los cuales son:

    1. **Factores externos***:* Como usuario, lo que se espera del sistema es que se tengan resultados rápidos, sea fácil de utilizar, que tenga robustez, que sea confiable y muchas cosas más (Weitzenfeld, 2005).
    2. **Factores internos:** En este caso, como administrador se espera que el sistema se pueda extender, exista la facilidad de modificarlo, que no sea difícil de comprender y que exista la posibilidad de migrarlo a diferentes ambientes computacionales (Weitzenfeld, 2005).

Todos estos aspectos nos permiten contar con un sistema de software que cumpla de manera eficiente cada una de las necesidades para las que fue diseñado. Como ya se mencionó, en la práctica no es posible garantizar un sistema perfecto, por lo que es necesario responder preguntas como: ¿dados los requerimientos del problema, hasta qué punto un sistema de software es suficientemente *bueno*? y esto nos lleva a preguntarnos, ¿qué tanto vale la pena invertir tiempo omitiendo o eliminando errores del sistema? Al responder estas preguntas podremos definir una ruta para discernir cuando un sistema de software satisface mejor las

necesidades para las que fue construido y así ocasionar que los costos de las fallas o errores que éste produzca se minimicen.

1.3. Calidad de los datos

Otro concepto importante para este trabajo es la calidad de los datos, cuando una empresa manipula información y se toman decisiones a partir de ésta, sin duda la calidad de los datos juega un papel fundamental para obtener resultados de manera eficaz y eficiente, por ello es importante el estudio y análisis de ésta. Comencemos dando una definición para *calidad de dato*.

Según la Real Academia Española de la Lengua[2]:

> **Calidad** es "*la propiedad o conjunto de propiedades inherentes a algo, que permiten juzgar su valor*"
>
> **Dato** es *"la información dispuesta de manera adecuada para su tratamiento por un ordenador"*

Siendo así, definimos ***calidad de datos*** como las propiedades necesarias con las que debe contar la información dispuesta para darle un tratamiento adecuado.

Existen varias definiciones para calidad de datos dentro de la literatura, para este trabajo resultarán relevantes las siguientes:

> *Decimos que un **dato** tiene alta **calidad** si es adecuado para su uso en operaciones, toma de decisiones y planeación* (Kerr et. al., 2007).
>
> *Un **dato** es **apropiado** para algún uso específico cuando cuenta con características como precisión, integridad, consistencia y validez* (Data Quality Working Group).

En la práctica hay distintas formas en las que se puede deteriorar la calidad de los datos, por ejemplo pensemos en una empresa en donde día a día se registran datos personales de los clientes, si la persona encargada de registrar dicha información cometiera errores ortográficos o simplemente errores humanos (*error de dedo* como comúnmente se dice) entonces la calidad de los datos se verá afectada, pues no habría precisión en éstos.

La calidad de los datos está asociada con la integridad de los mismos, en base de datos la calidad de un dato quiere decir que éste cumple con las reglas de integridad. Los procesos que se llevan a cabo para identificar y medir la calidad de los datos son distintos y en algunos casos

[2] Real Academia Española, http://www.rae.es/

depende del diseño empleado en la base de datos. Pensemos en el modelo Relacional, en donde es parte esencial del diseño contar con integridad de los datos así como calidad en el esquema.

Las restricciones de integridad protegen a la base de datos contra los daños accidentales, pues asegura que cualquier modificación que se realice a la base de datos no provoque la pérdida de la consistencia en los datos (Silberschatz et. al., 2010).

A continuación se presentan los diferentes tipos de integridad:

- *Entidad:* Para contar con este tipo de integridad es importante asignar a cada entidad del modelo una ***llave primaria*** (PK, por sus siglas en inglés). Una llave primaria es un conjunto mínimo de atributos que nos permite identificar de forma única a cada tupla y con ello evitamos duplicidad en los datos.
- *Dominio:* Este tipo de restricción permite asociar a cada atributo un domino de valores posibles (Silberschatz et. al., 2010). Con ello cualquier dato que no pertenezca a dicho dominio simplemente no podrá ser registrado.
- *No nulidad*: Se refiere a definir restricciones a todos aquellos atributos que no pueden ser nulos, ello nos permitirá contar con información precisa u omitir datos relevantes al momento del registro. Esta restricción es esencial para una PK o FK. Recordemos que un atributo puede ser ***nulo aplicable****,* es decir, si pensamos en el apellido materno de un norteamericano dicho atributo puede ser nulo ya que no todos cuentan con éste, sin embargo si pensamos en el atributo edad, éste será ***nulo no aplicable****,* ya que todos deberán tener asignada una edad dentro de la base de datos. El objetivo será minimizar el número de *nulos no aplicables*.
- *Referencial:* Permite asegurar que un valor que aparece en una relación para un conjunto de atributos determinado, aparezca también en otra relación para un cierto conjunto de atributos (Silberschatz et. al., 2010). Típicamente se asigna una ***llave foránea*** (FK, por sus siglas en inglés) – conjunto de atributos del esquema de una relación que forma la clave primaria de otro esquema (Silberschatz et. al., 2010) – y se asegura que cualquier modificación que se realice en la relación de referencia se haga también en la relación referenciada.
- *Integridad de Usuario:* Se refiere a la integridad de un conjunto de reglas de negocio especificadas por el usuario, las cuales no pertenecen a ninguno de los tipos de integridad antes mencionados. Sin embargo, los tipos de integridad antes mencionados resultan fundamentales para soportar las reglas de integridad de usuario (Microsoft SQL Server 2000).

Por otro lado, existe un concepto más que resulta importante dentro de la calidad de los datos, éste es la normalización, la cual se detalla a continuación.

Para poder comprender y llevar a cabo una normalización en base de datos, se requiere de un concepto relevante, las *dependencias funcionales,* comencemos definiendo éstas*:*

> Un atributo Y ***depende funcionalmente*** de otro atributo X ($X \rightarrow Y$) si y sólo si X define de forma única a Y.
>
> Una ***forma normal*** define condiciones sobre un conjunto de dependencias funcionales, o restricciones semánticas, que han sido especificadas en el esquema de la base de datos. Estas condiciones se utilizan para verificar que el diseño de la base de datos cumple ciertas propiedades y, si no fuera el caso, dichas condiciones podrán utilizarse para mejorar el diseño de la base de datos y obtener un buen diseño equivalente al anterior (Liu & Özsu, 2009).
>
> Para verificar que una base de datos está normalizada se deberá analizar cada una de las tablas inmersas en ésta y discernir en qué *forma normal* se encuentra cada una.
>
> Las formas normales más importantes son:
>
> - *Primera Forma Normal (1FN):* Todos sus atributos son atómicos
> - No hay orden de arriba-abajo en las tuplas.
> - No hay orden de izquierda-derecha en atributos.
> - No hay filas duplicadas.
> - Cada intersección fila – columna contiene exactamente un valor del dominio.
> - *Segunda Forma Normal (2FN):* Atributos no llave dependen de alguno de los atributos llave. Si la PK es de sólo un atributo automáticamente está en 2FN.
> - *Tercera Forma Normal (3FN)*: Una relación R está en 3FN con respecto a un conjunto de dependencias funcionales F si, para cada dependencia funcional en F^+ de la forma $\beta \rightarrow \alpha$, donde $\beta \subseteq R$ y $\alpha \subseteq R$, al menos una de las siguientes afirmaciones se cumple (Silberschatz et. al., 2010):
> - $\alpha \rightarrow \beta$ es una dependencia funcional trivial, esto es, $\beta \subseteq \alpha$
> - α es una superllave de R
> - Cada atributo A en $\beta - \alpha$ está contenido en una llave candidata de R.
> - *Forma Normal Boyce-Codd (BCFN):*Una relación R está en BCFN con respecto a un conjunto de dependencias funcionales F si, para cada dependencia

funcional en F^+ de la forma $\beta \rightarrow \alpha$, donde $\beta \subseteq R$ y $\alpha \subseteq R$, al menos una de las siguientes afirmaciones se cumple (Silberschatz et. al., 2010):

- $\alpha \rightarrow \beta$ es una dependencia funcional trivial, esto es, $\beta \subseteq \alpha$
- α es una superllave de R.

Consideraremos que una base de datos está bien diseñada cuando ésta se encuentre en BCFN, esto nos permitirá evitar inconsistencias protegiendo la integridad de los datos y disminuir problemas de actualización de los datos en las tablas.

En conjunto, todas estas características ayudan a que la calidad de los datos mejore significativamente y es deber del diseñador de la base de datos asignar las restricciones adecuadas para lograr dicho objetivo.

Por otro lado, las métricas que definen la calidad de los datos varían dependiendo de las decisiones que se tomen para determinar la metodología que se utilizará, como sabemos todo tiene un costo y medir la calidad de datos no suele ser algo económico.

Para este trabajo los atributos que mediremos son los siguientes:

- **Precisión:** éste nos permitirá saber la relación que hay entre el valor exacto de un dato y su valor estimado, esto es, responderá la pregunta ¿qué tan exactos son los datos registrados?
- **Nivel de confianza:** éste será un valor cuantificable y nos ayudará a saber qué tanto podemos confiar en la certeza de los datos almacenados.
- **Trazabilidad:** este atributo será útil para identificar el origen y las diferentes etapas en las que se han hecho modificaciones al sistema o a los datos.
- **Actualidad:** representa el nivel de confianza que tiene un dato durante el período de tiempo en que será utilizado, es aplicable cuando a pesar de que el dato fue registrado de manera adecuada en un instante de tiempo dado, al avanzar el tiempo, el valor real del dato ha cambiado, lo que ocasiona que el valor almacenado no corresponda con el real.
- **Completitud:** es el grado de confianza – que tiene todo un conjunto de datos – necesario para el uso que se le pretende dar.
- **Formato:** será el resultado del proceso de traducir, embalar, comprimir y organizar un conjunto de datos seleccionados para ser distribuidos a un sistema específico. Este proceso nos dará como resultado una estructura de datos que cumple las características de calidad de dato.

1.4. Consecuencias

Toda mala implementación en un sistema de software, así como en una base de datos, trae consigo distintas consecuencias, éstas son muy variadas, una lista de las más comunes es:

- Si el sistema o la base de datos nos permite obtener estadísticas, reportes, resultados o algún dato que implique números, al no tener calidad en éstos los resultados generados no serán confiables.
- Cuando una base de datos está mal diseñada y hay un sistema que la manipule, aún cuando éste sea bueno, la mala calidad en los datos no nos permitirá explotar la información de manera eficaz y eficiente. De manera análoga sucederá lo mismo, cuando la calidad de los datos es buena pero el sistema no cumple con los requerimientos o está mal diseñado.
- Si el principal objetivo será la toma de decisiones y tanto el sistema como la base de datos no son buenos, se tendrán consecuencias negativas ya sean monetarias o de planeación dentro de la empresa.
- Que el mal diseño haga que los procesos sean más lentos. Por ejemplo si hay redundancia en la información o hay un mal diseño en la base de datos (tener muchas tablas que describan lo mismo, atributos inútiles, tablas o atributos necesarios que no existen, etc.). También cuando el sistema no está bien diseñado (más que nada cuestiones de código, o decisiones tomadas durante la programación)

Esta tesis está organizada de la siguiente manera, en el capítulo 2 se presenta el planteamiento del problema, el objetivo de este trabajo, la metodología seguida, así como los resultados esperados. En el capítulo 3 se profundiza en cada una de las fases que se llevaron a cabo para generar el proceso de ingeniería inversa ρDB. Mientras que en el capítulo 4 se muestra el caso práctico en el que éste se aplicó. Los resultados y lecciones aprendidas se concentran en el capítulo 5. Finalmente las conclusiones y el trabajo futuro se presentan en el capítulo 6.

2. Planteamiento del problema

En este capítulo se presentará el contexto de esta tesis, su objetivo, la metodología seguida, los resultados esperados y el mecanismo de validación de los mismos.

2.1. 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. Ésta división depende de la Facultad de Medicina de la Universidad Nacional Autónoma de México.

En la actualidad la División cuenta con una serie de sistemas de software administrativos para su funcionamiento, el área encargada de administrar cada uno de éstos es el Departamento de Cómputo e Informática del Posgrado de Medicina quien, entre otras actividades, está encargado del soporte técnico, instalación de software y hardware, diseño y desarrollo de sistemas, diseño de bases de datos, generación de estadísticas, manejos de información, administración de la red, telefonía y páginas web, entre otras actividades.

El Departamento ha heredado muchos de los sistemas que actualmente tiene a su cargo, hoy en día sólo cinco de éstos han sido creados por éste. Los principales usuarios que interactúan con sus sistemas son: personal dentro de la misma División de Estudios de Posgrado, los Jefes de Enseñanza del posgrado, alumnos y profesores.

Los sistemas más importantes a cargo del Departamento de Cómputo son los siguientes:

- *Profesores.2.0:* Sistema diseñado para la Administración de Profesores en Cursos de Especialidad y Alta Especialidad Médica
- *Jornadas:* Sistema cuyo objetivo es el registro de carteles a presentar en las Jornadas Anuales de Investigación de Alta Especialidad Médica por parte de los alumnos.
- *Carteles Jornadas:* Este sistema se utiliza para la consulta e impresión de acuses de carteles que son presentados en las Jornadas Anuales de Investigación de Alta Especialidad Médica.
- *Sistema de Razonamiento Ético en la Clínica:* El objetivo de éste sistema es fortalecer las debilidades éticas de los médicos y así mejorar la práctica de su profesión a través de un curso en línea.

Como ya hemos visto el mal diseño de una base de datos trae consigo consecuencias negativas, entre los trabajos similares podemos destacar los siguientes:

- *Refactorización de bases de datos:* este trabajo fue realizado por Román Arango Díaz, en donde describe una técnica de desarrollo de bases de datos que permite realizar cambios o mejoras al diseño de éstas – la refactorización de bases de datos –. Además se aplica el proceso de refactorización a una base de datos en operación empleando una aplicación web y la biblioteca Java Liquibase para dicho objetivo. Los resultados que se obtuvieron en este trabajo fueron: el esquema del diseño mejoró significativamente, la semántica de la información y comportamiento se logró conservar y adicionalmente se logró obtener consultas rápidas, redundancia mínima, consistencia en la información, así como integridad referencial (Arango, 2013).

- *Sistema Evaluador Universal de Calidad de Datos (SEUCAD):* proyecto a cargo de la Dra. María del Pilar Ángeles de la Facultad de Ingeniería de la UNAM, quien, junto con su equipo de trabajo, está desarrollando actualmente un sistema evaluador universal de calidad de datos, la fecha de liberación está programada para el año 2015. Sin embargo, ya han sido presentados algunos de los resultados obtenidos, dicho proyecto se ha realizado en Python y SQL.

  Uno de los objetivos de SEUCAD es el cálculo de indicadores de calidad de datos, además de mejorar y extender un software libre para la duplicación de registros, perteneciente a la Universidad Nacional de Australia, cuyos objetivos son la detección, clasificación y fusión de registros duplicados bajo diversos manejadores de base de datos y para cualquier esquema de base de datos (Ángeles, 2013).

Sin embargo, el estudio y análisis de esta problemática es un tema importante dentro de la literatura referente a la mejora de bases de datos, es por ello que en la sección 3.9 de este trabajo se discute y destacan aquellos que también han influenciado este trabajo y cuyo análisis se dio cuando ρDB ya estaba siendo desarrollado.

2.2. Objetivo

Dada la problemática que genera el mal diseño de una base de datos, este trabajo 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 ya existente y en operación*.

La meta será que dicho proceso permita realizar modificaciones o mejoras sin que éstas cambien o alteren las necesidades para las que la base de datos y el sistema de software fueron creados,

así como conservar la mayor cantidad de datos ya almacenados, deshaciéndose de aquellos que podríamos llamar "basura".
La validación se llevará a cabo mediante un caso práctico, es decir, mejorando el diseño de una base de datos de un sistema en operación, para ello utilizaremos alguno de los sistemas que se encuentra en operación a cargo del Departamento de Cómputo del Posgrado de Medicina. Analizaremos su comportamiento, identificaremos fallas y con base en ello definiremos las mejoras que posteriormente aplicaremos al diseño de la base de datos; de esta manera se describirá y probará la utilidad de ρDB.

2.3. Metodología

Los pasos a seguir para lograr el objetivo anteriormente definido serán los siguientes:

Pre-análisis de los sistemas
Se analizarán el contexto, necesidades y restricciones de los sistemas de software bajo operación del Departamento de Cómputo de la División de Estudios de Posgrado de la Facultad de Medicina.

Selección del sistema
Tomando en cuenta criterios como la viabilidad y el impacto se elegirá un sistema de los analizados previamente para llevar a cabo el proceso de ingeniería inversa.

Análisis del sistema
Se identificarán los objetivos para los cuales fue creado el sistema, en particular aquellos ligados al almacenamiento y persistencia de datos. Para ello se estudiará la base de datos con la intención de crear una lista de supuestos generales para respondernos la pregunta ¿qué se deseaba modelar?

Diseño de la mejora
Una vez que se haya analizado el sistema y se conozca su propósito específico, se validará lo que realmente se modeló contra lo que se deseaba modelar. Siendo esta fase en la que se propongan las modificaciones necesarias a la base de datos.

Implementación y prueba de la mejora
Con las modificaciones a la base de datos identificadas, se implementará el modelo mejorado en un ambiente de prueba que permita validar su integración con el sistema de software y los datos

almacenados. En caso de que el sistema de software requiera cambios para su correcta integración con la base de datos, éstos serán reportados y documentados para que el personal del Departamento de Cómputo los tome en cuenta y planee su realización.

Liberación de la mejora

Después de llevar a cabo las fases anteriores, se podrá liberar la versión mejorada de la base de datos asegurando que cumple con las restricciones y necesidades para las que fue creada. Cabe resaltar que la integración con el sistema de software para la liberación definitiva queda fuera del alcance de esta propuesta de tesis, ya que involucra aspectos altamente dependientes del Departamento de Cómputo, como lo son asignación de recursos o el tiempo en que se decida llevar a cabo dicha integración.

2.4. Resultados esperados

Después de llevar a cabo la metodología descrita, se espera que al liberar la mejora de la base de datos los objetivos para los que fue creada se conserven, que la base de datos cuente con redundancia mínima, exista integridad en los datos y además se obtenga un proceso genérico de ingeniería inversa que permita mejorar una base de datos en operación, esto es, una vez que se pruebe la utilidad de dicho proceso, éste resulte exitoso y así validar a ρDB.

2.5. Mecanismo de validación

Los puntos clave para llevar a cabo la validación del proceso propuesto serán verificar que:

- La consistencia de los datos aumente.
- La inexistencia de datos *basura*.
- La desaparición de tuplas espurias.
- La redundancia innecesaria de datos haya desaparecido.

Lo anterior se verificará mediante valores numéricos, comparando, por ejemplo, cuántas tuplas mal referenciadas había antes de la mejora y cuántas después de ésta.

Entre los principales indicadores que permitirán verificar el proceso están los siguientes:

- A través de entrevistas con los administradores del sistema, esto nos permitirá saber si las quejas que presentaban los usuarios se redujeron.

- Mediante pruebas de usuario, esto es, que los alumnos utilicen el sistema para así verificar que las funcionalidades de éste no se alteraron y comprobar que existen mejoras en el desempeño de dicho sistema.

3. *Proceso de ingeniería inversa*

En este capítulo se detallarán cada una de las fases llevadas a cabo para obtener a ρDB así como el marco de trabajo utilizado para expresarlo.

3.1. KUALI-BEH como marco de trabajo para la definición y descripción del proceso

KUALI-BEH (Morales & Oktaba, 2012) es una propuesta mexicana en respuesta al llamado *A Foundation for the Agile Creation and Enactment of Software Engineering Methods* (FACESEM)[3] lanzado por el Object Management Group en 2011.

KUALI-BEH es un marco de trabajo 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.

De acuerdo con (Morales & Oktaba, 2012) este marco de trabajo está compuesto por dos vistas: la *estática* y la *operacional* y se definen de la siguiente manera:

- *Vista Estática:* en esta vista se definen los conceptos comunes utilizados en proyectos de software por los practicantes de Ingeniería de Software. Basándose en dos conceptos fundamentales: *método* y *práctica*. Un método será un conjunto de prácticas con un objetivo definido, mientras que una práctica representará una *guía,* es decir, la descripción paso a paso mediante *actividades* de cómo se desarrollará la manera de trabajo durante el proyecto, cada actividad puede ser detallada con *tareas*, aunque éstas no son obligatorias. Los elementos que contiene una práctica son: definir las metas u objetivos, elementos de entrada y salida y finalmente conocimientos y habilidades. A continuación en la tabla 1 se define cada uno de éstos:

Tabla 1. Elementos de una práctica.

Elemento	Descripción
Objetivo	Definir qué es lo que se quiere lograr con la práctica.
Entrada	Insumos necesarios para poder llevar a cabo una práctica o actividad.
Resultado	Productos resultantes al finalizar una práctica o actividad.
Actividad	Guía para llevar a cabo una práctica y lograr el objetivo de ésta.
Tarea	Detallado paso a paso de cómo llevar a cabo una actividad, opcional.
Herramientas	Instrumentos necesarios para realizar una actividad.
Conocimientos y Habilidades	Conocimientos y habilidades mínimos con los que se debe contar para realizar una actividad o práctica.
Métricas	Parámetro que se mide al realizar un trabajo, sirve para medir el esfuerzo, tamaño, etc., y así en un futuro sea posible estimar éste valor antes de ejecutar el trabajo.

[3] FACESEM, http://www.omg.org/cgi-bin/doc?ad/2011-6-26

Cada método deberá cumplir con tres características o propiedades para considerarlo bien definido, siendo éstas: coherencia, consistencia y completitud, enseguida se describe cada una de ellas:

- Coherencia: Se refiere a que cada uno de los objetivos definidos en las prácticas aportan algo al progreso de nuestro proyecto, es decir, que al ejecutar cada práctica y en consecuencia cumplir su objetivo, nos aseguramos que estamos avanzando para lograr el propósito del método (Morales & Oktaba, 2012).

- Consistencia: Al generar una salida después de llevar a cabo una práctica, ésta deberá ser consumida por otra práctica. Análogamente con las entradas, éstas deben ser generadas por otra práctica o deben ser provistas como entradas del método (Morales & Oktaba, 2012).

- Completitud: Asegurar que se cumple totalmente el propósito del método y que las prácticas sean suficientes para lograr las necesidades definidas en el proyecto (Morales & Oktaba, 2012).

- *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.

La figura 2 nos muestra el ícono de una práctica, en donde podemos observar que hay Entradas (***I)***, Salidas (***R)*** y un Objetivo (***O)*** definido.

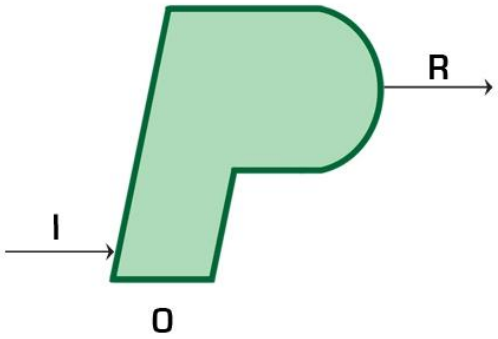


Figura 2. Símbolo para una Práctica utilizado en (Morales & Oktaba, 2012)

Para fines de esta tesis, se utilizará el enfoque de KUALI-BEH para la descripción del proceso de ingeniería inversa, es decir, se construirá un método compuesto por prácticas con el propósito de mejorar el diseño e implementación de una base de datos de un sistema de software ya existente y en operación.

A continuación se describen cada una de las prácticas que componen a ρDB agrupadas por fases.

3.2. Pre-análisis de los sistemas

En esta fase se analizarán cada uno de los sistemas que se encuentran en operación en el Departamento de Cómputo de la División de Estudios de Posgrado de la Facultad de Medicina. El análisis se realizará con la intención de definir las características con las que cuenta cada sistema – como pueden ser el contexto, sus necesidades o sus restricciones – y así discernir cuál de ellos resulta de utilidad para los objetivos de este trabajo.

Será más relevante aquel que cuente con un mayor número de oportunidades de mejora en cuanto a persistencia e integridad de datos que aquellos con oportunidades de mejora en cuanto a usabilidad o interfaz de usuario.

3.3. Selección del sistema

La meta de esta etapa es seleccionar el sistema adecuado para los objetivos de este trabajo después de haber llevado a cabo el pre-análisis de los sistemas y tomando en cuenta criterios como viabilidad e impacto.

3.4. Análisis del sistema

En este punto se identificarán los objetivos para los que fue creado el sistema seleccionado. Además se estudiará la base de datos de éste para así crear una lista de supuestos generales y con ello definir las oportunidades de mejora y deficiencias de la base de datos.

Esta fase se compone de 5 prácticas mismas que se detallan utilizando la tripleta conformada por el Objetivo, Entradas y Salidas propuesto en (Morales & Oktaba, 2012) y que se explicó en la sección 3.1 de esta tesis.

3.4.1 Análisis y búsqueda de supuestos y restricciones

En esta práctica se pretende 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 generará un documento y un diagrama E/R en el que se describirán tanto las tablas como los atributos que conforman a cada una. Ver tabla 2.

Tabla 2. Práctica 1 de ρDB.

1	**Práctica**
Análisis y búsqueda de supuestos y restricciones	
Objetivo	
Conocer los supuestos y restricciones que se deseaban modelar con cada tabla y sus respectivos atributos.	
Entrada	**Resultado**
PT: Respaldo de la base de datos a analizar PT: Consultas predefinidas para análisis	PT: Descripción_tabla_atributo [Borrador] PT: Diagrama E/R [borrador]

Guía			
Actividad	Cargar el respaldo de la base de datos en el SMBD		
Entrada		**Resultado**	
Respaldo de la base de datos a analizar		Base de datos creada en el SMBD	
Tareas (opcional)	**Herramientas (opcional)**	**Conocimientos y Habilidades**	**Métricas**
	SMBD (PostgreSQL)		
Actividad 1	Realizar consultas predefinidas sobre las tablas de la base de datos		
Entrada		**Resultado**	
Base de datos creada en el SMBD		Descripción_tabla_atributo [Borrador]	
Tareas (opcional)	**Herramientas (opcional)**	**Conocimientos y Habilidades**	**Métricas**
Realizar consultas SELECT * FROM tabla SELECT COUNT(*) FROM tabla Donde *tabla* es cada una de las tablas de la base de datos.	Editor de texto (NotePad++) Procesador de textos (Word) SMBD (PostgreSQL)		
Actividad 2	Diseñar el diagrama E/R [borrador]		
Entrada		**Resultado**	
Descripción_tabla_atributo [Borrador] Base de datos creada en el SMBD		Diagrama E/R [borrador]	
Tareas (opcional)	**Herramientas (opcional)**	**Conocimientos y Habilidades**	**Métricas**
	Editor de texto (NotePad++) Herramienta de Diagramación (ER Editor)		

3.4.2 Búsqueda de relaciones

El objetivo de esta práctica será encontrar relaciones existentes entre las entidades, de esta manera se actualizará la descripción de las tablas y los atributos que conforman a la base de datos. Ver tabla 3.

Tabla 3. Práctica 2 de ρDB.

2	**Práctica**		
Búsqueda de relaciones			
Objetivo			
Encontrar las relaciones que existen entre las entidades.			
Entrada		**Resultado**	
C: Base de datos creada en el SMBD PT: Diagrama E/R [borrador] PT: Script con consultas predeterminadas • Script_Busqueda_Relaciones PT: Descripción_tabla_atributo [Borrador]		PT:Descripción_tabla_atributo [Actualizado]	
Guía			
Actividad 1	Realizar consultas sobre cada una de las tablas		
Entrada		**Resultado**	
Base de datos creada en el SMBD. Script con consultas predeterminadas • Script_Busqueda_Relaciones		Reporte de los resultados de las consultas	
Tareas (opcional)	**Herramientas (opcional)**	**Conocimientos y Habilidades**	**Métricas**
	Procesador de Textos (Word) Editor de textos (NotePad++)		
Actividad 2	Analizar los supuestos de las entidades así como sus atributos		
Entrada		**Resultado**	
Diagrama E/R [borrador] Reporte de los resultados de las consultas		Descripción_tabla_atributo [Actualizado]	
Tareas (opcional)	**Herramientas (opcional)**	**Conocimientos y Habilidades**	**Métricas**
	Herramienta de Diagramación (ER Editor)		

3.4.3 Entrevista

Al ejecutar esta práctica se espera obtener información de interés mediante una entrevista con el diseñador de la base de datos y así generar un listado de supuestos y características de la base de datos. Ver tabla 4.

Tabla 4. Práctica 3 de ρDB.

3	**Práctica**		
Entrevista			
Objetivo			
Entrevistarse con el diseñador de la base de datos			
Entrada		**Resultado**	
C: Entrevista programada PT: Preguntas para adquirir información de interés PT: Diagrama E/R [borrador]		PT: Entrevista [grabada] PT: Listado de supuestos [inicial]	
Guía			
Actividad 1	Agendar y preparar la entrevista		
Entrada		**Resultado**	
Diagrama E/R [Borrador]		Solicitud de entrevista otorgada Preguntas para adquirir información de interés	
Tareas (opcional)	**Herramientas (opcional)**	**Conocimientos y Habilidades**	**Métricas**
	Libreta para registrar las preguntas.		
Actividad 2	Presentarse ante el entrevistado y explicar el objetivo de la entrevista		
Entrada		**Resultado**	
Entrevista programada		El Diseñador de la base de datos conoce el objetivo de la entrevista	
Tareas (opcional)	**Herramientas (opcional)**	**Conocimientos y Habilidades**	**Métricas**
Actividad 3	Realizar las preguntas y capturar la información		
Entrada		**Resultado**	
Preguntas para adquirir información de interés Diagrama E/R [borrador]		Entrevista [grabada] Listado de supuestos [inicial]	
Tareas (opcional)	**Herramientas (opcional)**	**Conocimientos y Habilidades**	**Métricas**
	Grabadora de audio Libreta para hacer anotaciones Procesador de textos (Word)		

3.4.4 Validación de hipótesis

Esta práctica tendrá como objetivo precisar detalles respecto a los supuestos de cada entidad así como del modelo relacional anteriormente establecidos. Ver tabla 5.

Tabla 5. Práctica 4 de ρDB.

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)	

3.4.5 Búsqueda de errores de integridad

En esta práctica la meta será analizar cada una de las tablas y generar un archivo con las características específicas de los errores de integridad – referencial, de entidad, de dominio y de nulidad – para cada una de éstas. De igual manera, como resultado se construirán scripts con consultas para llevar a cabo dicho análisis. Ver tabla 6.

Tabla 6. Práctica 5 de ρDB.

5	**Práctica**
Búsqueda de errores de integridad	
Objetivo	
Analizar cada una de las tablas para determinar cuáles cuentan con errores de integridad.	
Entrada	**Resultado**
C: Base de datos creada en el SMBD PT: Diagrama E/R [esbozado] PT: Listado de supuestos [Revisado]	Características específicas de la integridad de cada una de las tablas. • Referencial • Entidad • Dominio • Nulidad • Forma Normal Script con consultas predeterminadas • Queries_Info_Entidad • Queries_Info_Nulidad • Queries_Info_Referencial • Queries_Info_Dominio • Queries_Info_Forma_Normal
Guía	
Actividad 1	Verificar la Integridad de Entidad
Entrada	**Resultado**
Base de datos creada en el SMBD Diagrama E/R [esbozado] Listado de supuestos [Revisado]	Documento con resultados finales de las fallas de integridad específicas para cada tabla. • Info_entidad_nombreTabla Script con consultas predeterminadas para cada tabla • Queries_Info_Entidad

Tareas (opcional)	**Herramientas (opcional)**	**Conocimientos y Habilidades**	**Métricas**
Realizar SELECT * FROM tabla; SELECT * FROM tabla WHERE cond1, cond2…; SELECT COUNT(*) FROM tabla; Donde tabla es cada una de las tablas de la base de datos.	SMBD		

Actividad 2	Verificar la Integridad de Dominio
Entrada	**Resultado**
Base de datos creada en el SMBD Diagrama E/R [esbozado] Listado de supuestos [Revisado]	Documento con resultados finales de las fallas de integridad específicas para cada tabla. • Info_Dominio_nombreTabla Script con consultas predeterminadas para cada tabla • Queries_Info_Dominio

Tareas (opcional)	Herramientas (opcional)	Conocimientos y Habilidades	Métricas
Realizar SELECT COUNT (DISTINCT atributo) FROM tabla WHERE cond1, cond 2…; Donde tabla es cada una de las tablas de la base de datos y atributo es algún atributo de la tabla en cuestión.	SMBD		

Actividad 3	Verificar la Integridad de Nulidad

Entrada	Resultado
Base de datos creada en el SMBD Diagrama E/R [esbozado] Listado de supuestos [Revisado]	Documento con resultados finales de las fallas de integridad específicas para cada tabla • Info_Nulidad_nombreTabla Script con consultas predeterminadas para cada tabla • Queries_Info_Nulidad

Tareas (opcional)	Herramientas (opcional)	Conocimientos y Habilidades	Métricas
Realizar SELECT COUNT (DISTINCT atributo) FROM tabla WHERE cond1, cond 2…; Donde tabla es cada una de las tablas de la base de datos y atributo es algún atributo de la tabla en cuestión.	SMBD		

Actividad 4	Verificar la Integridad Referencial

Entrada	Resultado
Base de datos creada en el SMBD Diagrama E/R [esbozado] Listado de supuestos [Revisado]	Documento con resultados finales de las fallas de integridad específicas para cada tabla • Info_Referencial_nombreTabla Script con consultas predeterminadas para cada tabla • Queries_Info_Referencial

Tareas (opcional)	Herramientas (opcional)	Conocimientos y Habilidades	Métricas
Realizar SELECT atributo, COUNT(atributo) AS f1 FROM tabla GROUP BY atributo ORDER BY f1; Donde tabla es cada una de las tablas de la base de datos y atributo es algún atributo de la tabla en cuestión.	SMBD		

Actividad 5	Verificar la Integridad de Usuario.		
Entrada		**Resultado**	
Base de datos creada en el SMBD.		Documentación con los errores de Integridad de Usuario identificados.	
Tareas (opcional)	**Herramientas (opcional)**	**Conocimientos y Habilidades**	**Métricas**
1. Verificar y validar los usuarios existentes dentro de la base de datos, así como los privilegios asignados a éstos. 2. Verificar y validar las reglas del negocio identificadas.	SMBD		

Actividad 6	Verificar en qué Forma Normal se encuentra cada tabla.		
Entrada		**Resultado**	
Base de datos creada en el SMBD.		Documento con resultados finales de la forma normal en que se encuentra cada tabla • Info_Forma_Normal_nombreTabla Script con consultas predeterminadas para cada tabla • Queries_Info_Forma_Normal.	
Tareas (opcional)	**Herramientas (opcional)**	**Conocimientos y Habilidades**	**Métricas**
Realizar SELECT atributo, COUNT(atributo) AS fl FROM tabla GROUP BY atributo ORDER BY fl; Donde tabla es cada una de las tablas de la base de datos y atributo es algún atributo de la tabla en cuestión.	SMBD		

3.5. Diseño de la mejora

Ya que se haya analizado el sistema y se conozca su propósito específico, el Diseño de la Mejora será la fase en donde se propongan las modificaciones necesarias a la base de datos. Lo anterior sólo se realizará después de haber validado lo que realmente se modeló contra lo que se deseaba modelar.

3.5.1 Análisis y propuesta de Integridad

Con esta práctica se pretende analizar el por qué y cómo se generan errores de integridad en cada una de las tablas, de esta manera se hará una propuesta de mejoras de integridad así como verificar el diagrama E/R con el que se contaba. Ver tabla 7.

Tabla 7. Práctica 6 de ρDB.

6	**Práctica**
Análisis y propuesta de Integridad	
Objetivo	
Identificar qué genera o por qué se tienen fallas de integridad en las tablas.	
Entrada	**Resultado**
Listado de tablas y atributos detallados PT: Descripción_tabla_atributo [Detallado] PT: Diagrama E/R [esbozado] C: Base de datos creada en el SMBD PT: Script con consultas predeterminadas • Queries_Info_Entidad • Queries_Info_Nulidad • Queries_Info_Referencial • Queries_Info_Dominio • Queries_Info_Forma_Normal Info_Referencial_nombreTabla Info_entidad_nombreTabla Info_Nulidad_nombreTabla Info_Dominio_nombreTabla Info_ Forma_Normal_nombreTabla	PT: Factores_errores_Integridad PT: Propuesta de Mejoras de Integridad PT: Diagrama E/R [verificado]
Guía	

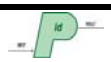
Actividad 1	Análisis de errores de integridad		
Entrada		**Resultado**	
Info_Referencial_nombreTabla Info_entidad_nombreTabla Info_Nulidad_nombreTabla Info_Dominio_nombreTabla Info_Forma_Normal_nombreTabla Script con consultas predeterminadas Base de datos creada en el SMBD		Factores_errores_Integridad Script_Análisis_Integridad	
Tareas (opcional)	**Herramientas (opcional)**	**Conocimientos y Habilidades**	**Métricas**
Realizar SELECT * FROM tabla; SELECT * FROM tabla WHERE cond1, cond2…; SELECT COUNT(*) FROM tabla; Donde tabla es cada una de las tablas de la base de datos	SMBD		

Actividad 2	Análisis y registro de mejoras de integridad		
Entrada		**Resultado**	
Base de datos [creada] Diagrama E/R [esbozado] Script con consultas predeterminadas Script_Análisis_Integridad		Propuesta de Mejoras de Integridad Diagrama E/R [verificado]	
Tareas (opcional)	**Herramientas (opcional)**	**Conocimientos y Habilidades**	**Métricas**
Realizar SELECT * FROM tabla; SELECT * FROM tabla WHERE cond1, cond2…; SELECT COUNT(*) FROM tabla; Donde tabla es cada una de las tablas de la base de datos	SMBD		

3.5.2 Preparación del ambiente de trabajo

El objetivo de esta práctica será preparar el ambiente de trabajo, de esta manera se esperan obtener los datos de acceso necesarios para manipular la base de datos en operación. Ver tabla 8.

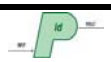
Tabla 8. Práctica 7 de ρDB.

 7	**Práctica**		
Preparación del ambiente de trabajo			
Objetivo			
Preparar las características del ambiente de trabajo			
Entrada		**Resultado**	
PT: Respaldo de la base de datos en operación		C: Ambiente de trabajo validado PT: Datos de acceso	
Guía			
Actividad 1	Solicitar permisos para manipular la base de datos en operación		
Entrada		**Resultado**	
Acceso a la base de datos en operación requerido		Datos de acceso obtenidos: • Base de datos en operación • Sistema • Servidor	
Tareas (opcional)	**Herramientas (opcional)**	**Tareas (opcional)**	**Herramientas (opcional)**
Actividad 2	Verificar datos de acceso		
Entrada		**Resultado**	
Acceder a: • Base de datos en operación • Sistema • Servidor		Ambiente de trabajo validado	
Tareas (opcional)	**Herramientas (opcional)**	**Tareas (opcional)**	**Herramientas (opcional)**
	SMBD		SMBD

3.5.3 Análisis de la base de datos en operación

Esta práctica tendrá como objetivo el interactuar con la base de datos en operación para así observar el comportamiento de ésta e identificar errores específicos en la aplicación, además de actualizar el diagrama E/R. Ver tabla 9.

Tabla 9. Práctica 8 de ρDB.

 8	**Práctica**
Análisis de la base de datos en operación	
Objetivo	
Interactuar con la base de datos en operación para observar el comportamiento e identificar errores específicos	
Entrada	**Resultado**
PT: Datos de Acceso C: Acceso a la base de datos en operación otorgado C: Acceso al sistema otorgado C Acceso al servidor otorgado PT: Script con consultas predeterminadas PT: Diagrama E/R [verificado]	PT: Diagrama E/R [actualizado] PT: Errores en la aplicación C: Conocimiento del comportamiento de la base de datos en operación.

Guía			
Actividad 1	Manipular la base de datos en operación e ir analizando el comportamiento en el SMBD		
Entrada		**Resultado**	
Acceso a la base de datos en operación otorgado		Diagrama E/R [verificado] Errores en la base de datos en operación Script_Análisis_BD[operación]	
Tareas (opcional)	**Herramientas (opcional)**	**Conocimientos y Habilidades**	**Métricas**
SELECT * FROM tabla SELECT COUNT(*) FROM tabla Donde tabla es cada una de las tablas de la base de datos			
Actividad 2	Actualizar el diagrama E/R		
Entrada		**Resultado**	
Diagrama E/R [verificado]		Diagrama E/R [actualizado]	
Tareas (opcional)	**Herramientas (opcional)**	**Conocimientos y Habilidades**	**Métricas**

3.5.4 Análisis de la navegación del sistema y su relación con las tablas de la base de datos

Al realizar esta práctica se pretende analizar la navegación del sistema y así identificar qué tablas son utilizadas dentro de éste y de qué manera. Se obtendrá un diagrama de estados así como scripts de ayuda para el conteo y borrado de tuplas para cada tabla. Ver tabla 10.

Tabla 10. Práctica 9 de ρDB.

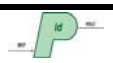
9	**Práctica**		
Análisis de la navegación del sistema y su relación con las tablas de la base de datos			
Objetivo			
Identificar qué tablas son utilizadas dentro del sistema y de qué manera.			
Entrada		**Resultado**	
C: Acceso al Sistema otorgado C: Acceso a la Base de Datos otorgado PT: Datos de Acceso		PT: Diagrama de estados PT: Script de conteo de tuplas PT: Script de borrado de tuplas	
Guía			
Actividad 1	Generar scripts		
Entrada		**Resultado**	
Acceso a la Base de Datos otorgado		PT: Script de conteo de tuplas PT: Script de borrado de tuplas	
Tareas (opcional)	**Herramientas (opcional)**	**Conocimientos y Habilidades**	**Métricas**
SELECT count(*) FROM tabla, now(); DELETE FROM tabla;	Editor de textos (NotePad++)		

Actividad 2	Familiarizarse con el uso del sistema		
Entrada		**Resultado**	
C: Acceso al Sistema otorgado		Sistema navegado	
Tareas (opcional)	**Herramientas (opcional)**	**Conocimientos y Habilidades**	**Métricas**
Navegar en el sistema para conocer su comportamiento Registrar resultados observados			
Actividad 3	Construir los diagramas de estado que reflejen la navegación del sistema		
Entrada		**Resultado**	
		Diagramas de estado	
Tareas (opcional)	**Herramientas (opcional)**	**Conocimientos y Habilidades**	**Métricas**
Realizar SELECT count(*) FROM tabla, now() Registrar los estados (pantallas) Registrar las tablas que se ven afectadas en cada transición Agregar código de colores Verde.- Nombre de la tabla Naranja.- Decisiones Rojo.- No interactúan tablas	Diagramador Procesador de textos (Word)		

3.5.5 Actualización del Modelo de la base de datos

El objetivo de esta práctica será mejorar las hipótesis del diagrama E/R y/o incluir características de las tablas y atributos. Ver tabla 11.

Tabla 11. Práctica 10 de ρDB.

 10	**Práctica**		
Actualización del Modelo de la base de datos			
Objetivo			
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.			
Entrada		**Resultado**	
PT: Diagrama de estados PT: Diagrama E/R [actualizado]		PT: Diagrama E/R [propuesto]	
Guía			
Actividad 1			
Entrada		**Resultado**	
Tareas (opcional)	**Herramientas (opcional)**	**Conocimientos y Habilidades**	**Métricas**

3.6. Implementación y prueba de la mejora

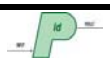
Una vez identificadas las modificaciones a realizar a la base de datos, en esta fase se implementará el modelo mejorado en un ambiente de prueba para así validar su integración con el sistema de software y los datos almacenados sin afectar al sistema en operación.

Si el sistema de software requiriera cambios para su correcta integración con la base de datos, éstos serán reportados y documentados para que el personal del Departamento de Cómputo los considere y planee su realización.

3.6.1 Primera intervención de la base de datos (Tablas inútiles).

Como resultado de esta práctica se espera identificar tablas inútiles de la base de datos así como aquellos archivos PHP que interactúan con la base de datos. Una vez que se verifique el listado de tablas inútiles se eliminará cada una de éstas obteniendo así la primera intervención de la base de datos y su respectivo respaldo. Ver tabla 12.

Tabla 12. Práctica 11 de ρDB.

<table>
<tr><td> 
11</td><td colspan="3">Práctica</td></tr>
<tr><td colspan="4">Primera intervención de la base de datos (Tablas inútiles).</td></tr>
<tr><td colspan="4">Objetivo</td></tr>
<tr><td colspan="4">Identificar y eliminar las tablas inútiles de la base de datos.</td></tr>
<tr><td colspan="2">Entrada</td><td colspan="2">Resultado</td></tr>
<tr><td colspan="2">PT: Diagrama E/R [propuesto]
C: Base de datos creada en el SMBD
PT: Diagramas de estado
PT: Script Borrado de tablas inútiles</td><td colspan="2">PT: Listado de tablas inútiles [verificado]
PT: ContienenNombreTabla
C: Base de datos intervenida
PT: Respaldo primera intervención</td></tr>
<tr><td colspan="4">Guía</td></tr>
<tr><td>Actividad 1</td><td colspan="3">Identificar tablas inútiles</td></tr>
<tr><td colspan="2">Entrada</td><td colspan="2">Resultado</td></tr>
<tr><td colspan="2">Diagramas de estado
Diagrama E/R [propuesto]
Base de datos creada en el SMBD</td><td colspan="2">Listado de tablas inútiles [propuesto]
ContienenNombreTabla</td></tr>
<tr><td>Tareas (opcional)</td><td>Herramientas (opcional)</td><td>Conocimientos y Habilidades</td><td>Métricas</td></tr>
<tr><td>Registrar en el Diagrama de Estados qué archivos PHP son utilizados en el sistema
Registrar qué archivos PHP interactúan con la base de datos</td><td>Terminal de unix</td><td></td><td></td></tr>
<tr><td>Actividad 2</td><td colspan="3">Analizar archivos PHP</td></tr>
<tr><td colspan="2">Entrada</td><td colspan="2">Resultado</td></tr>
<tr><td colspan="2">Listado de archivos php que interactúan con la base de datos</td><td colspan="2">Listado de tablas inútiles [verificado]</td></tr>
<tr><td>Tareas (opcional)</td><td>Herramientas (opcional)</td><td>Conocimientos y Habilidades</td><td>Métricas</td></tr>
<tr><td>Búsqueda en archivos PHP tablas a eliminar</td><td></td><td></td><td></td></tr>
</table>

Actividad 3	Verificar listado de tablas inútiles		
Entrada		**Resultado**	
Listado de tablas inútiles [verificado]		Script Borrado de tablas inútiles	
Tareas (opcional)	**Herramientas (opcional)**	**Conocimientos y Habilidades**	**Métricas**
	Editor de textos (NotePad++)		
Actividad 4	Eliminar tablas inútiles		
Entrada		**Resultado**	
Script Borrado de tablas inútiles		Base de datos intervenida	
Tareas (opcional)	**Herramientas (opcional)**	**Conocimientos y Habilidades**	**Métricas**
Actividad 5	Validar el funcionamiento adecuado del sistema		
Entrada		**Resultado**	
Base de datos intervenida. Navegar el sistema verificando que su funcionamiento no haya sido alterado.		Primera intervención validada.	
Tareas (opcional)	**Herramientas (opcional)**	**Conocimientos y Habilidades**	**Métricas**
Actividad 6	Respaldar la primera intervención (Respaldo [PI])		
Entrada		**Resultado**	
Base de datos intervenida [creada]		Respaldo Primera Intervención	
Tareas (opcional)	**Herramientas (opcional)**	**Conocimientos y Habilidades**	**Métricas**

3.6.2 Segunda intervención a la base de datos (Mejoras de integridad).

Esta práctica tendrá como objetivo realizar la segunda intervención a la base de datos, ello se logrará mediante la implementación de las restricciones de integridad. De igual manera se obtendrá el respaldo de la base de datos intervenida. Ver tabla 13.

Tabla 13. Práctica 12 de ρDB.

12	**Práctica**
Segunda intervención a la base de datos (Mejoras de integridad).	
Objetivo	
Implementar las restricciones de integridad a la base de datos.	
Entrada	**Resultado**
PT: Propuesta de mejoras de Integridad [propuesto] PT: Diagrama E/R [propuesto] C: Base de datos intervenida creada en el SMBD PT: Diagramas de estado PT: Respaldo de la base de datos [PI] PT: Script Borrado_tuplas PT: Script Conteo_tuplas	PT: Propuesta de mejoras de Integridad [verificado] C: Base de datos intervenida [2] PT: Respaldo segunda intervención
Guía	
Actividad 1	Implementar restricciones de Entidad.
Entrada	**Resultado**
Base de datos intervenida creada en el SMBD • Info_Entidad_nombreTabla	Restricciones de Entidad implementadas. • Info_Entidad_nombreTabla [verificado]

Tareas (opcional)	Herramientas (opcional)	Conocimientos y Habilidades	Métricas
1.- Agregar restricciones de Entidad a una tabla. 2.- Insertar tuplas en la tabla. 3.- Analizar los datos insertados y no insertados para proponer modificaciones. • Info_Entidad_nombreTabla[verificado] 4.- Generar respaldo parcial de datos. • Respaldo 0.1	Editor de textos (NotePad++) SMBD		

Actividad 2	Implementar restricciones de Nulidad.

Entrada	Resultado
Base de datos intervenida creada en el SMBD • Info_Nulidad_nombreTabla	Restricciones de Nulidad implementadas. • Info_Nulidad_nombreTabla [verificado]

Tareas (opcional)	Herramientas (opcional)	Conocimientos y Habilidades	Métricas
1.- Agregar restricciones de Nulidad a una tabla. 2.- Insertar tuplas en la tabla. 3.- Analizar los datos insertados y no insertados para proponer modificaciones. • Info_Nulidad_nombreTabla [verificado] 4.- Generar respaldo parcial de datos. • Respaldo 0.2	Editor de textos (NotePad++) SMBD		

Actividad 3	Implementar restricciones de Dominio.

Entrada	Resultado
Base de datos intervenida creada en el SMBD • Info_Dominio_nombreTabla	Restricciones de Dominio implementadas. • Info_Dominio_nombreTabla [verificado]

Tareas (opcional)	Herramientas (opcional)	Conocimientos y Habilidades	Métricas
1.- Agregar restricciones de Dominio a una tabla. 2.- Insertar tuplas en la tabla. 3.- Analizar los datos insertados y no insertados para proponer modificaciones. • Info_Dominio_nombreTabla [verificado] 4.- Generar respaldo parcial de datos. • Respaldo 0.3	Editor de textos (NotePad++) SMBD		

Actividad 4	Implementar restricciones de Referencial.

Entrada	Resultado
Base de datos intervenida creada en el SMBD • Info_Referencial_nombreTabla	Restricciones de Referencial implementadas. • Info_Referencial_nombreTabla [verificado] Propuesta de mejoras de Integridad [verificado]

Tareas (opcional)	Herramientas (opcional)	Conocimientos y Habilidades	Métricas
1.- Agregar restricciones de Referencial a una tabla. 2.- Insertar tuplas en la tabla 3.- Analizar los datos insertados y no insertados para proponer modificaciones. • Info_Referencial_nombreTabla [verificado] 4.- Generar respaldo parcial de datos. • Respaldo 0.4	Editor de textos (NotePad++) SMBD		

Actividad 5	Validar el funcionamiento del sistema		
Entrada		**Resultado**	
Navegar el sistema verificando que su funcionamiento no haya sido alterado		Segunda intervención validada	
Tareas (opcional)	**Herramientas (opcional)**	**Tareas (opcional)**	**Métricas**

Actividad 6	Respaldar la segunda intervención (Respaldo [SI])		
Entrada		**Resultado**	
Base de datos intervenida [2]		Respaldo segunda intervención	
Tareas (opcional)	**Herramientas (opcional)**	**Tareas (opcional)**	**Métricas**

3.7. Liberación de la mejora

En esta etapa se liberará la versión mejorada de la base de datos asegurando que cumple con las restricciones y necesidades para las que fue creada.

3.7.1 Tercera intervención (Prueba final).

Con esta última práctica se pretende liberar la base de datos intervenida para su integración con el sistema en operación. Lo anterior permitirá la implementación de las restricciones de integridad finales para después verificar el funcionamiento del sistema. Ver tabla 14.

Tabla 14. Práctica 13 de ρDB.

13	**Práctica**
Tercera intervención (Prueba final).	
Objetivo	
Liberar la base de datos intervenida para su integración con el sistema en operación.	
Entrada	**Resultado**
Base de datos intervenida [2] creada en el SMBD • Info_Nulidad_nombreTabla [verificado] • Info_Entidad_nombreTabla [verificado] • Info_Dominio_nombreTabla [verificado] • Info_Referencial_nombreTabla [verificado]	Base de datos final [creada]

Guía			
Actividad 1	Analizara las restricciones de Integridad		
Entrada		**Resultado**	
Base de datos intervenida [2] creada en el SMBD • Info_Nulidad_nombreTabla [verificado] • Info_Entidad_nombreTabla [verificado] • Info_Dominio_nombreTabla [verificado] • Info_Referencial_nombreTabla [verificado]		Resultados finales • Restricciones de integridad [finales]	
Tareas (opcional)	**Herramientas (opcional)**	**Conocimientos y Habilidades**	**Métricas**
Decidir cuáles de las restricciones sí son necesarias en la base de datos			
Actividad 2	Implementar resultados finales		
Entrada		**Resultado**	
Restricciones de integridad [finales] Base de datos intervenida [2] creada en el SMBD		Base de datos final [creada]	
Tareas (opcional)	**Herramientas (opcional)**	**Conocimientos y Habilidades**	**Métricas**
Actividad 3	Validar el funcionamiento del sistema		
Entrada		**Resultado**	
Navegar el sistema verificando que su funcionamiento no haya sido alterado		Tercera intervención validada	
Tareas (opcional)	**Herramientas (opcional)**	**Conocimientos y Habilidades**	**Métricas**
Actividad 4	Respaldar la tercera intervención (Respaldo [TI])		
Entrada		**Resultado**	
Base de datos final [creada]		Respaldo generado	
Tareas (opcional)	**Herramientas (opcional)**	**Conocimientos y Habilidades**	**Métricas**
Actividad 5	Liberar la base de datos e integrarla con el sistema en operación		
Entrada		**Resultado**	
Base de datos final [creada] Sistema en operación		Liberación de la base de datos	
Tareas (opcional)	**Herramientas (opcional)**	**Conocimientos y Habilidades**	**Métricas**

A la par de la definición de cada una de las prácticas de ρDB se verificó que éste estuviera bien definido, para ello se creó un diagrama del método – que se muestra en la figura 3 – con el cual se iba comprobando que el proceso propuesto estuviera bien definido, es decir, que cumpliera las propiedades de ser coherente, consistente y completo.

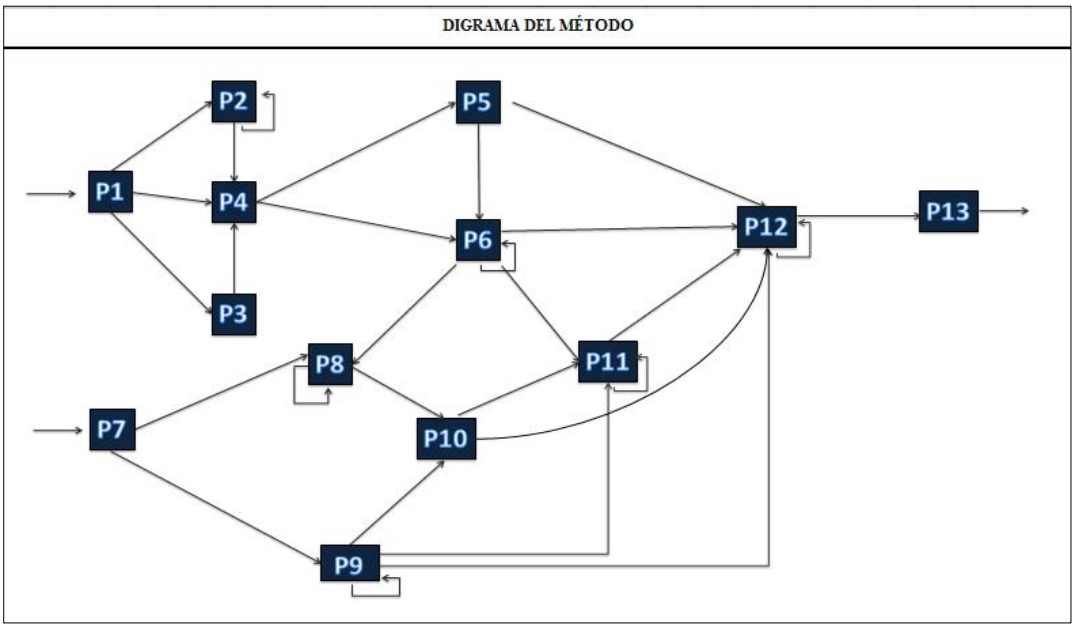


Figura 3. Diagrama del Método.

3.8. Productos de trabajo

En esta sección se describen cada uno de los productos de trabajo generados en las prácticas de ρDB. Además se representa gráficamente el flujo de los status para aquellos productos de trabajo que cuentan con éste.

3.8.1 Diagrama E/R

Descripción: Diagrama Entidad/Relación asociado a la base de datos inmersa en el sistema. Los posibles status de este producto de trabajo se presentan en la figura 4.

Figura 4. Status del Diagrama E/R.

3.8.2 Descripción_tabla_atributo

Descripción: Nociones de lo que puede modelar cada tabla y atributo. Los posibles status de este producto de trabajo se presentan en la figura 5.

Figura 5. Status de Descripción_tabla_atributo.

3.8.3 Entrevista

Descripción: Grabación de la entrevista realizada al diseñador de la base de datos. Los posibles status de este producto de trabajo se presentan en la figura 6.

Figura 6. Status de Entrevista.

3.8.4 Info Integridad

Descripción: 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. Los posibles status de este producto de trabajo se presentan en la figura 7.

- Info_Entidad_nombreTabla
- Info_Dominio_nombreTabla
- Info_Nulidad_nombreTabla
- Info_Referencial_nombreTabla
- Info_Forma_Normal_nombreTabla

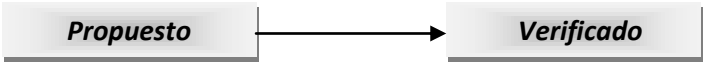


Figura 7. Status de Info Integridad.

3.8.5 Factores_errores_Integridad

Descripción: Posibles factores que dan lugar a los errores de integridad en cada tabla.

3.8.6 Propuesta_mejoras_Integridad

Descripción: Listado de propuestas a modificar en la base de datos. Los posibles status de este producto de trabajo se presentan en la figura 8.

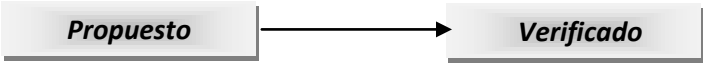


Figura 8. Status de Propuesta_mejoras_Integridad.

3.8.7 Respaldo de la base de datos en operación

Descripción: Respaldo de la base de datos en operación.

3.8.8 Errores en la aplicación

Descripción: Errores encontrados durante el uso de la aplicación.

3.8.9 Diagramas de estado

Descripción: Diagrama que muestra el comportamiento del sistema.

3.8.10 Script Borrado de Tablas_inútiles

Descripción: Script con consultas para el borrado de tablas inútiles.

3.8.11 Listado tablas inútiles

Descripción: Script con consultas para el borrado de tablas inútiles. Los posibles status de este producto de trabajo se presentan en la figura 9.

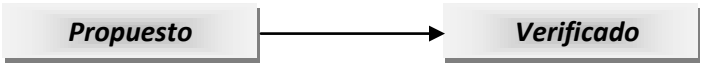


Figura 9. Status de Listado tablas inútiles.

3.8.12 ContieneNombreTabla

Descripción: Listado de archivos php que interactúan con la base de datos.

3.8.13 Script Busqueda_Relaciones

Descripción: Script con consultas a realizar para adquirir información de las relaciones existentes entre las distintas tablas.

3.8.14 Queries Integridad

Descripción: 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.

- Queries_Info_Dominio
- Queries_Info_Nulidad
- Queries_Info_Referencial
- Queries_Info_Entidad
- Queries_Info_Forma_Normal

3.8.15 Script Analisis_Integridad

Descripción: Script con consultas de ayuda para el análisis de errores de integridad.

3.8.16 Script Analisis_BD [operación]

Descripción: Script con consultas para analizar el comportamiento de la base de datos en operación al manipular ésta.

3.8.17 Script Conteo_tuplas

Descripción: Script con consultas para el conteo de tuplas en cada tabla.

3.8.18 Script Borrado_tuplas

Descripción: Script con consultas para el borrado de tuplas de cada tabla.

3.8.19 Respaldo 0.1

Descripción: Respaldo de la base de datos después de aplicar las propuestas de integridad de entidad.

3.8.20 Respaldo 0.2

Descripción: Respaldo de la base de datos después de aplicar las propuestas de integridad de nulidad.

3.8.21 Respaldo 0.3

Descripción: Respaldo de la base de datos después de aplicar las propuestas de integridad de dominio.

3.8.22 Respaldo 0.4

Descripción: Respaldo de la base de datos después de aplicar las propuestas de integridad referencial.

3.8.23 Consultas predefinidas para Análisis

Descripción: Exploración de qué modelan las tablas y atributos de la realidad.

3.8.24 Datos de Acceso

Descripción: Permisos requeridos para manipular la base de datos. Los posibles status de este producto de trabajo se presentan en la figura 10.

Figura 10. Status de Datos de Acceso.

3.8.25 Reporte de los resultados de las consultas

Descripción: Reporte de las consultas realizadas para obtener información de las relaciones existentes entre las tablas de la base de datos.

3.8.26 Preguntas para adquirir información de interés

Descripción: Listado de preguntas a realizar durante la entrevista con el diseñador de la base de datos.

3.8.27 Listado de supuestos

Descripción: Lista de supuestos realizados respecto a la base de datos, como pueden ser las características de tablas y atributos. Los posibles status de este producto de trabajo se presentan en la figura 11.

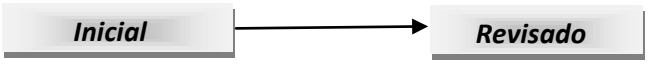


Figura 11. Status de Listado de supuestos.

3.8.28 Respaldo primera intervención

Descripción: Respaldo de la base de datos previo a la primera intervención.

3.8.29 Respaldo segunda intervención

Descripción: Respaldo de la base de datos previo a la segunda intervención.

3.8.30 Respaldo tercera intervención

Descripción: Respaldo de la base de datos previo a la tercera intervención.

3.9. Discusión

Resulta relevante dar una definición concreta para Ingeniería Inversa, definiremos a éste concepto como:

> La *ingeniería inversa* es el proceso de descubrir los principios tecnológicos de un dispositivo, un objeto o sistema, mediante el análisis de su estructura, funcionamiento u operación. El objetivo será construir un dispositivo o programa nuevo que hará lo mismo pero sin copiar todos los aspectos del original (Juárez-Ramírez et. al., 2007).

Por otra parte, como se mencionó en el capítulo 2 sección 1, dentro de la literatura hay varios trabajos referentes a métodos o procesos para la refactorización de bases de datos, entre la información más relevante para este trabajo se consideró la siguiente:

> 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 (Fowler, 1999).

Cabe señalar que será ésta la definición de *refactorización* que se utilizará para esta tesis.

> La refactorización para bases de datos relacionales permite mejorar la estructura del esquema relacional sin alterar la semántica de la información de éste, así el esquema relacional proporcionará la misma información antes y después de realizar la refactorización (Ambler, 2003; Ambler & Sadalage, 2006).

Por tanto, para este trabajo, la ingeniería inversa será el medio para lograr una refactorización, es decir, se mejorará una base de datos en operación sin documentación alguna realizando adecuaciones basados en la extracción de conocimiento e información a nuestro alcance.

Una de las metodologías más utilizadas para la refactorización de bases de datos relacionales es el *"Test-Driven Development (TDD) of Relational Databases"*, en donde, como su nombre lo indica, el objetivo es realizar pruebas para validar el diseño de la base de datos. En TDD se lleva a cabo 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 (Ambler, 2007).

En la *Figura 12* se muestra el proceso de refactorización utilizando el método TDD, como podemos observar decidir cuál es la "*refactorización óptima*" no es algo trivial, es por ello que dados los trabajos ya existentes para la refactorización de bases de datos (Fowler, 1999; Ambler, 2003; Ambler & Sadalage, 2006), la propuesta de este trabajo resulta ser un complemento para éstos pues el objetivo es aportar una metodología que permita realizar modificaciones o mejoras sin que éstas cambien o alteren las necesidades para las que la base de datos y el sistema de software fueron diseñados.

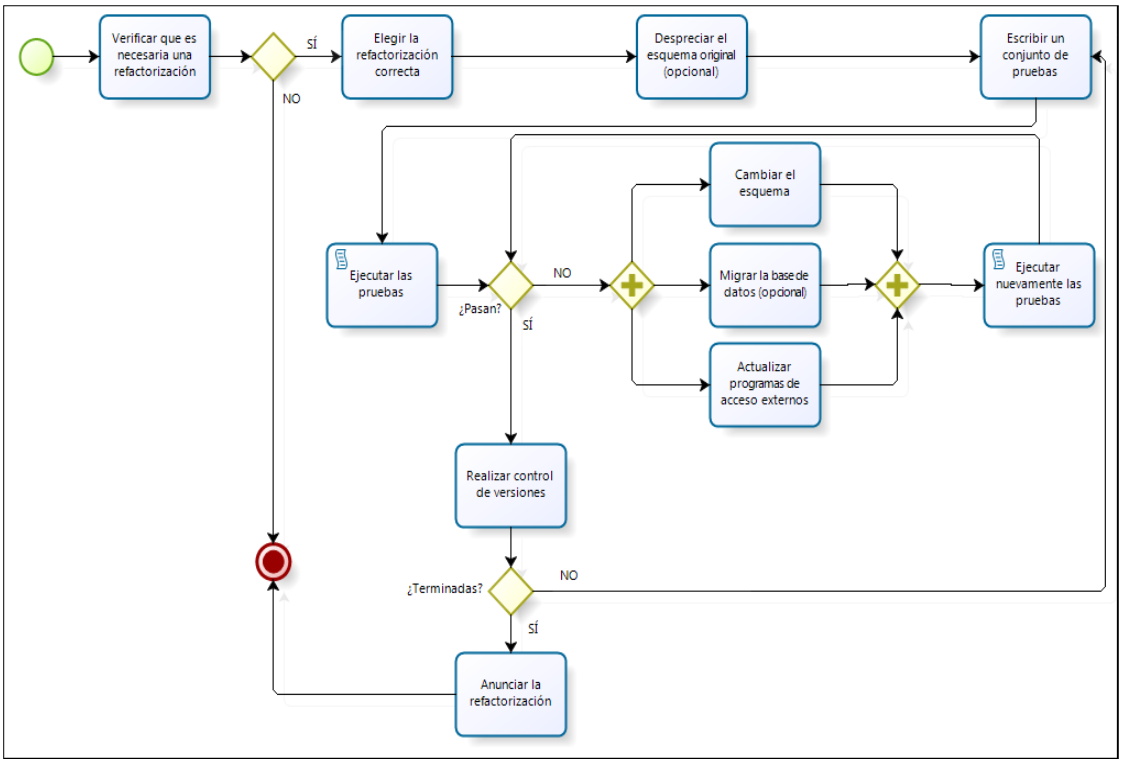


Figura 12. Test Driven Development (Ambler, 2007).

El proceso descrito anteriormente fue utilizado como marco para la ejecución de las prácticas descritas en ρDB. Es importante recalcar que TDD, en particular TDD para bases de datos relacionales, sirvió como ciclo de vida para ρDB.

4. Caso práctico

En este capítulo se detallará la aplicación del método de ingeniería inversa propuesto en un caso práctico. A continuación se describirán cada una de las fases, con sus respectivas prácticas, presentadas en el capítulo 3.

4.1. Pre-análisis de los sistemas

Para esta práctica se llevó a cabo un análisis exploratorio para cada uno de los sistemas que se encuentran en operación en el Departamento de Cómputo de la División de Estudios de Posgrado de la Facultad de Medicina, de esta manera fue posible definir las características con las que contaba cada sistema así como seleccionar aquel que resultara más relevante para ésta tesis. Los sistemas analizados fueron los siguientes:

- *Profesores.2.0:* Éste sistema presentaba fallas en el almacenamiento de datos, principalmente por la falta de PK's en las tablas y redundancia en los registros.
- *Jornadas:* Las oportunidades de mejora para éste sistema se presentaban esencialmente en la cantidad de tablas y atributos con las que contaba la base de datos, de igual manera carecía de PK's.
- *Carteles Jornadas:* Para éste sistema notamos mejoras, por ejemplo, en el nombrado de tablas y atributos, pues éstos no eran descriptivos por lo que resultaba difícil identificar qué se deseaba modelar con ellos.
- *Sistema de Razonamiento Ético en la Clínica*: Este sistema presentaba oportunidades de mejora en cuestiones de integridad, ya que no contaba con mecanismos para asegurar ninguno de sus tipos. Contaba con tuplas espurias y datos basura que afectaban la consistencia y eficiencia de la base de datos.

4.2. Selección del sistema

Para este trabajo será de interés el Sistema de Razonamiento Ético en la Clínica, cuyo contexto se detalla a continuación.

Sistema de Razonamiento Ético en la Clínica

Este sistema fue heredado al Departamento de Cómputo e Informática del Posgrado de Medicina, y está orientado tanto para alumnos de residencia como para profesores. 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. El contenido de este curso, así como sus evaluaciones, está coordinado por la Dr. Irene Duarte, Jefa del Consejo Técnico de la Facultad de Medicina.

Para los alumnos del Posgrado, acreditar el curso de fortalecimiento de la ética médica es un requisito curricular que lleva un promedio de 20 horas abarcar los niveles básicos e intermedio, y 15 horas más para el nivel avanzado.

El curso está organizado en tres niveles de aplicación y para avanzar en cada nivel se requiere obtener una calificación aprobatoria.

- El nivel básico aborda los conceptos teórico-prácticos elementales de la ética y del razonamiento ético y los aplica a casos de tipo general.
- En el nivel intermedio se ensayan los pasos del razonamiento ético y la autoevaluación mediante casos clínicos de los cuatro programas troncales.
- El nivel avanzado permite aplicar el razonamiento ético en casos específicos de cada especialidad.

Con este contexto se continuó con la ejecución de las fases restantes del método de ingeniería inversa.

4.3. Análisis del sistema

En esta fase se describirán los objetivos para los que fue creado el Sistema de Razonamiento Ético en la Clínica, además se presentarán los productos de trabajo generados al realizar las prácticas detalladas en el capítulo anterior.

4.3.1 Análisis y búsqueda de supuestos y restricciones

Al ejecutar esta práctica se tuvo un primer contacto con la base de datos y fue posible describir algunos de los atributos así como las tablas inmersas en la base de datos. A continuación se presentan los resultados obtenidos.

Productos de trabajo

- Descripción_tabla_atributo [Borrador]: Éste producto de trabajo permitió notar que existían tablas y atributos con nombres que no guardaban relación con el negocio, por lo que resultaba difícil saber qué es lo que diseñaban cada uno de estos.

- Diagrama E/R [Borrador]: Las figuras *13 y14* muestran el borrador del diagrama E/R de la base de datos que utiliza el Sistema de Ética. Gracias a éste fue posible notar que había tablas que no se relacionan con alguna otra, además de que resultó una división entre estas por lo que se obtuvieron dos diagramas.

4.3.2 Búsqueda de relaciones

Como resultado de esta fase se logró mejorar los supuestos hechos respecto a qué modelan cada tabla y sus respectivos atributos, además de detectar las posibles relaciones que existen entre las entidades de la base de datos.

Productos de trabajo

- Descripción_tabla_atributo [Actualizado]: Permitió describir algunas de las tablas así como atributos de los cuales se desconocía qué diseñaban de la realidad.

- Script Busqueda_Relaciones: Producto de apoyo para encontrar relaciones entre las tablas existentes dentro del diseño.

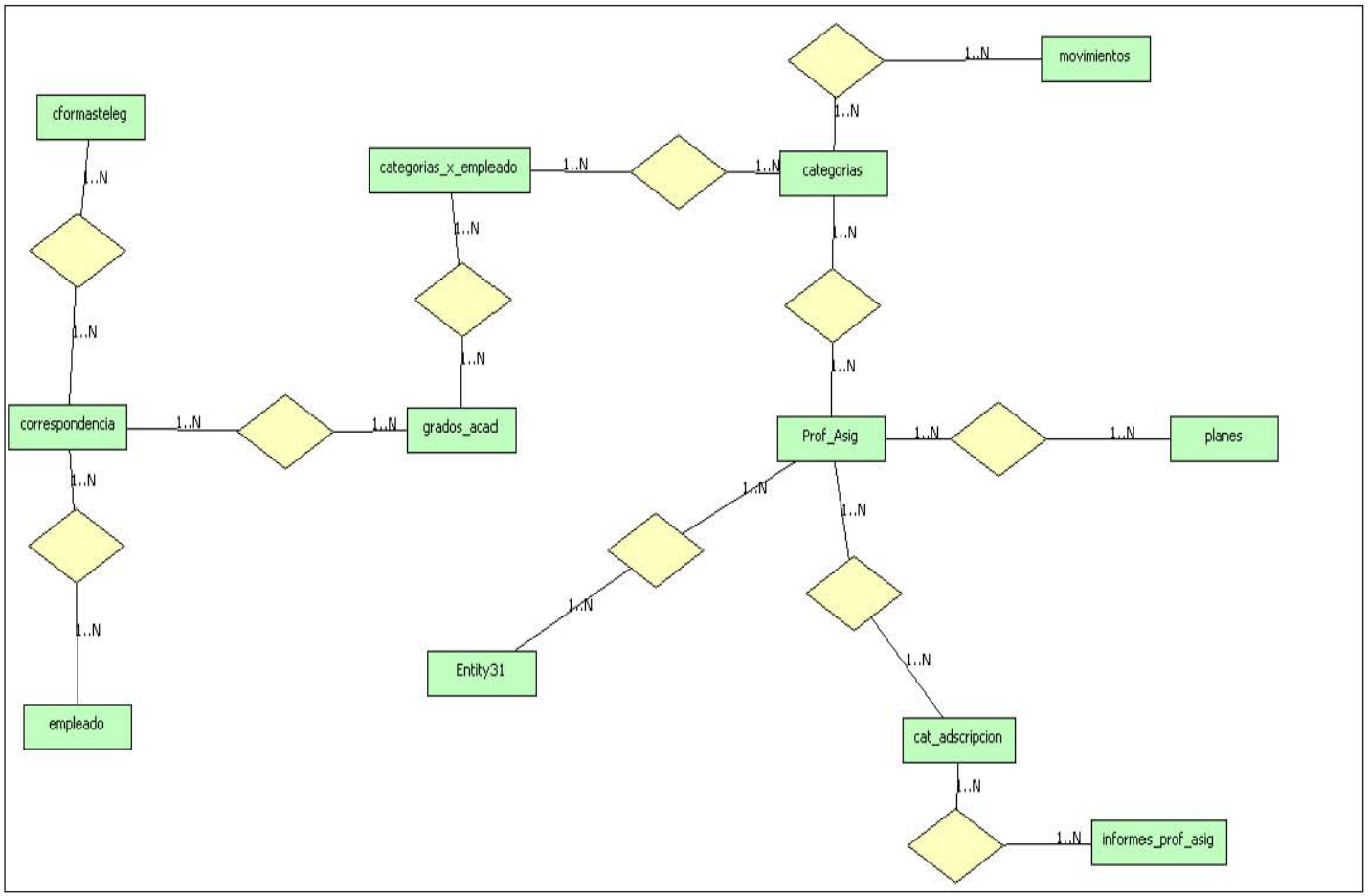


Figura 13. Borrador del Diagrama E/R para la base de datos inmersa en el Sistema de Ética.

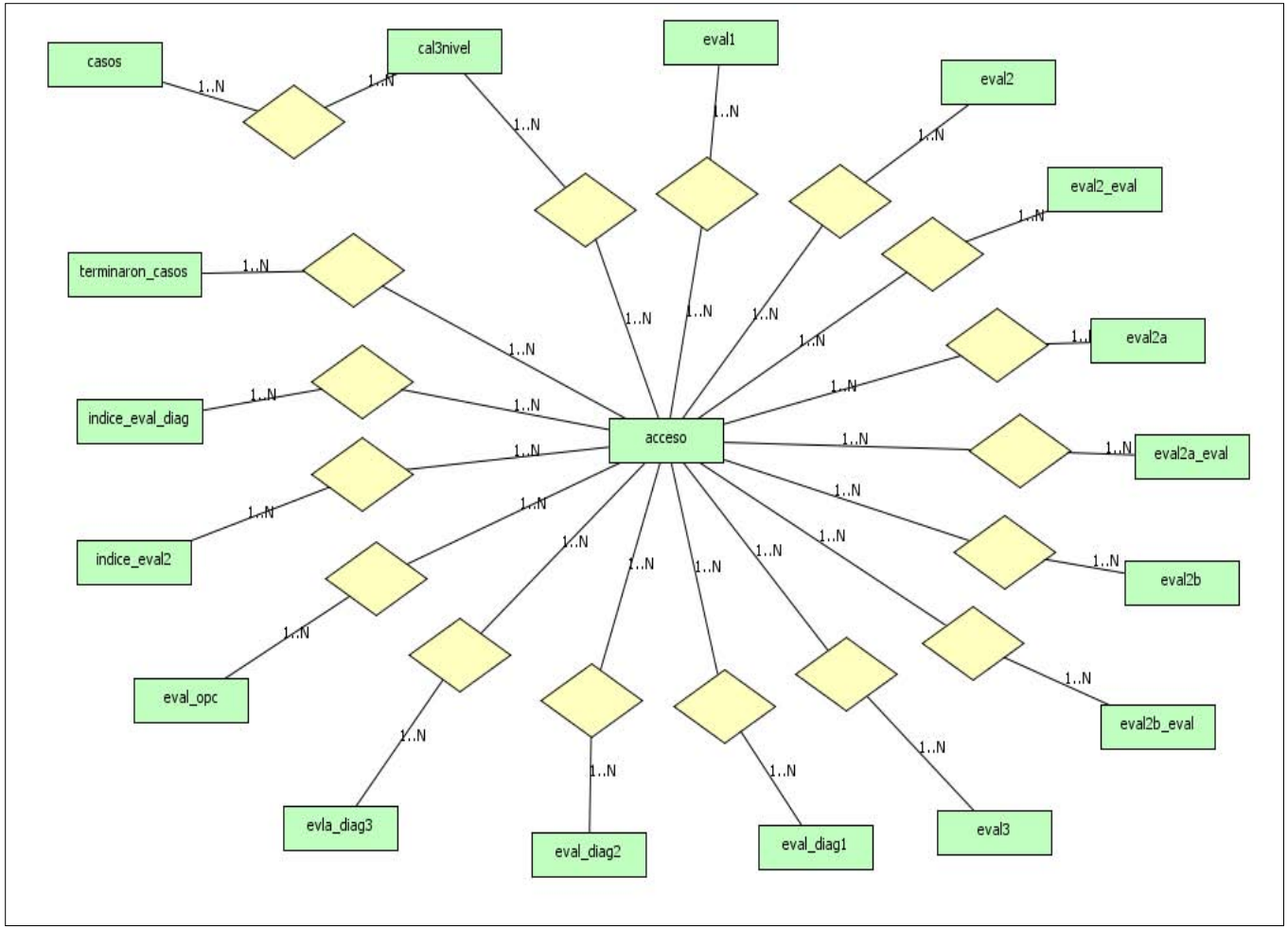


Figura 14. Borrador del Diagrama E/R para la base de datos inmersa en el Sistema de Ética.

Script Busqueda_Relaciones: A continuación se muestra un ejemplo de un Script para la búsqueda de posibles relaciones entre las entidades. Nótese que se aprovecharon funciones de agregación y agrupamientos propios del SMBD para realizar la búsqueda, situación que facilitó dicha tarea.

Queries para ejecución en SQL.

```
SELECT COUNT (*)
FROM acceso;

SELECT *
FROM acceso
WHERE clave IS NULL;

SELECT COUNT (*)
FROM eval1;

SELECT COUNT (*)
FROM eval1
WHERE cve_user LIKE '';

SELECT COUNT (*)
FROM eval1
WHERE id_preg1 IS NULL;


--- Reunión de acceso y eval1
SELECT COUNT (*)
FROM acceso JOIN eval1 ON clave=cve_user;

SELECT COUNT (*)
FROM acceso JOIN eval1 ON clave=cve_user
WHERE clave LIKE '';

SELECT COUNT (*)
FROM eval1
WHERE cve_user NOT IN (SELECT clave AS cve_user FROM acceso);

-- Agrupación por cve_user de eval1 para saber cuántas veces han realizado el examen
SELECT cve_user, COUNT (cve_user) as repeticiones
FROM eval1
WHERE cve_user NOT IN (SELECT clave AS cve_user FROM acceso)
GROUP BY cve_user
ORDER BY repeticiones;


SELECT COUNT (*)
FROM acceso JOIN eval1 ON clave=cve_user
WHERE id_preg1 IS NULL;

SELECT *
FROM acceso JOIN eval1 ON clave=cve_user
WHERE id_preg1 IS NULL AND fecha_ingreso LIKE '%2011%';

SELECT to_date (fecha_ingreso, 'DD MM YYYY'), count (*) AS repeticiones
FROM acceso JOIN eval1 ON clave=cve_user
WHERE id_preg1 IS NULL
GROUP BY to_date (fecha_ingreso, 'DD MM YYYY')
ORDER BY repeticiones;

SELECT *
FROM acceso JOIN eval1 ON clave=cve_user
WHERE id_preg1 IS NOT NULL AND fecha_ingreso LIKE '%2011%';
```

4.3.3 Entrevista

Durante esta fase se acudió al Departamento de Cómputo de la División de Estudios de Posgrado de la Facultad de Medicina, después de haber programado una cita y haber diseñado una serie de preguntas para adquirir información de interés. Dicha reunión se llevó a cabo con el diseñador de la base de datos del Sistema de Ética, quien fue de gran ayuda para comprender varios aspectos del diseño de ésta y resolvió varios detalles al respecto. Lo anterior permitió mejorar los supuestos que se tenían y se detalla en la siguiente sección.

4.3.4 Validación de hipótesis

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, en seguida se muestran los resultados obtenidos.

Productos de trabajo

- Diagrama E/R [esbozado]: Gracias a la información que se obtuvo durante la entrevista con el diseñador de la base de datos, se descartaron varias de las entidades inmersas en el diseño, pues éstas no eran útiles para los objetivos para los que fue creada la base de datos. La figura 15 muestra el modelo que se obtuvo después de las modificaciones.

- Descripción_tabla_atributo [Detallado]

4.3.5 Búsqueda de errores de integridad

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; ésta fue la fase en donde se comenzó a analizar los errores de integridad con los que contaba el diseño de la base de datos.

Productos de trabajo

- Queries Integridad: Después de ejecutar cada uno de los queries fue posible identificar errores de integridad en las entidades. La figura 16 muestra un ejemplo de los queries a ejecutar en SQL utilizando la herramienta pgAdminIII.

- Info Integridad: Archivo de ayuda para resumir los errores de integridad detectados en cada una de las entidades. En la figura 17 se muestra a manera de ejemplo parte de éste producto utilizando Notepad++.

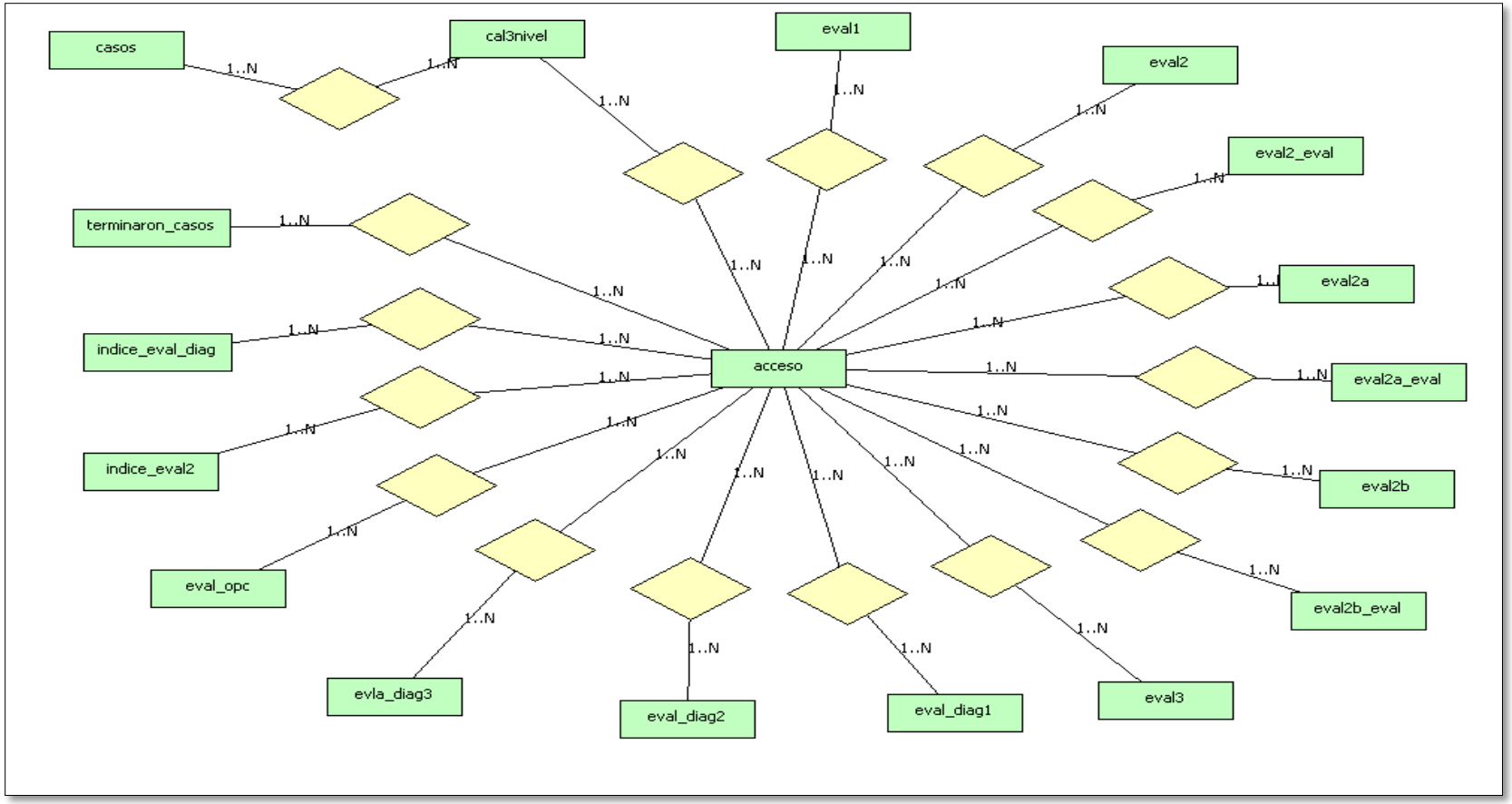


Figura 15. Diagrama E/R [Esbozado]

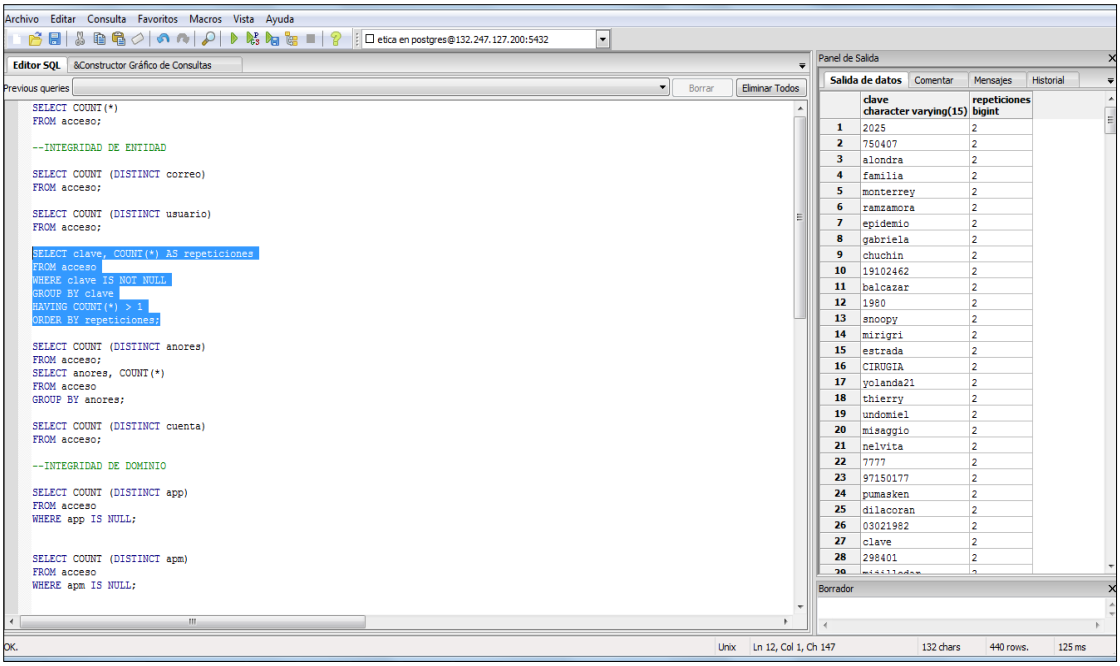


Figura 16. Queries Integridad en pgAdminIII.

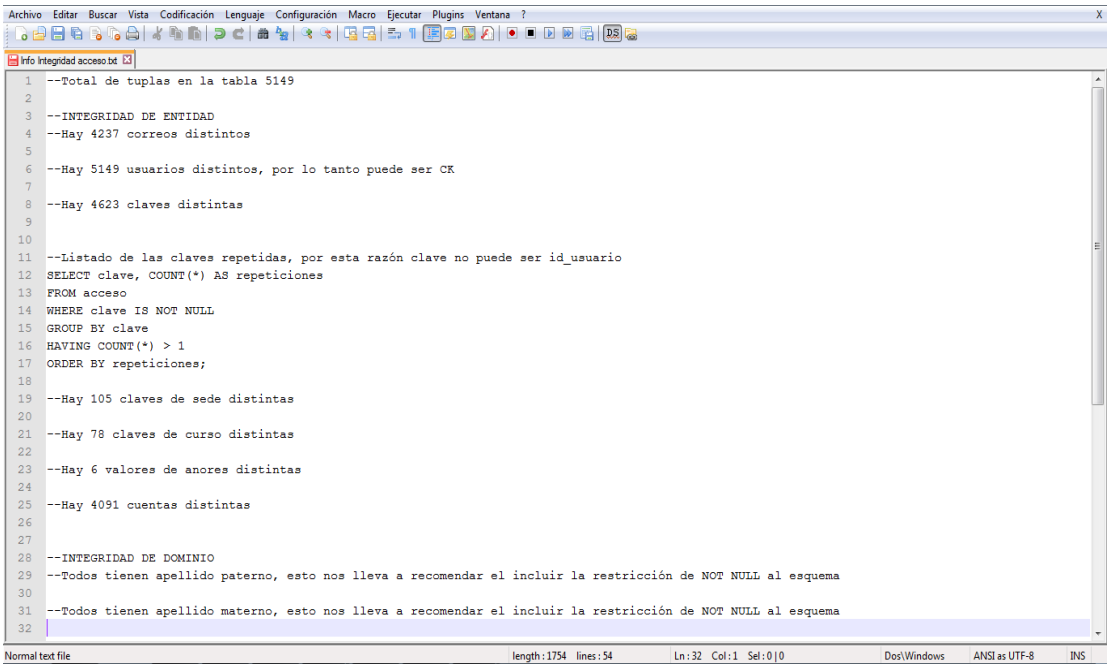


Figura 17. Info Integridad en Notepad++.

4.3.6 Discusión del análisis realizado

En esta sección se discutirán los puntos más importantes del diseño de la base de datos encontrado. Dentro de las desventajas con las que contaba el diseño de la base de datos se tenían las siguientes:

- No se contaban con restricciones para la integridad de entidad, dominio, referencial y de no nulidad. Respecto a la integridad de usuario podemos decir que el sistema atendía el requerimiento no funcional de Seguridad mediante PHP, sin embargo las reglas del negocio eran implementadas mediante restricciones en la interfaz de usuario.
- Fue posible notar que existía nula normalización debido a que ninguna de las tablas contaba con llave primaria (PK), en consecuencia no se encontraba en 1FN.
- Dentro de las dependencias funcionales que se pudieron identificar fue posible notar que ninguna de éstas 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.
- El equipo administrativo no contaba con un diagrama E/R que modelara el diseño de la base de datos.

4.4. Diseño de la mejora

En las etapas anteriores fue posible validar qué es lo que se desea modelar con la base de datos y lo que realmente se estaba modelando, se logró especificar los objetivos del diseño de ésta y además se detectaron errores en la integridad de la base de datos. Basados en lo anterior en esta fase se diseñaron las mejoras a aplicar.

4.4.1 Análisis y propuesta de Integridad

En esta fase se hizo un análisis de las causas que pudieran generar errores de integridad en la base de datos, con ello se mejoró el Diagrama E/R de ésta y se hizo una propuesta de mejoras a aplicar.

Productos de trabajo

- Diagrama E/R [verificado]: Este producto de trabajo permitió encontrar nuevas relaciones entre las entidades y mejorar el diseño anterior. Los resultados se muestran en la figura 18.

4.4.2 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.

4.4.3 Análisis de la base de datos en operación

Una vez que se tuvo validado el ambiente de trabajo el siguiente paso fue analizar la base de datos en operación, ello permitiría detectar errores específicos en el funcionamiento de ésta.

Productos de trabajo

- Diagrama E/R [actualizado]: Gracias al análisis que se realizó de la bases de datos en operación fue posible detectar nuevas relaciones entre las entidades, las cuales se muestran en la figura 19.

- Errores en la aplicación: archivo con errores detectados en la aplicación, el cual fue de ayuda para entender el funcionamiento de la base de datos en operación.

4.4.4 Análisis de la navegación del sistema y su relación con las tablas de la base de datos

En la práctica anterior fue posible detectar los factores que generaban errores de integridad en la base de datos, en esta fase 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 se utilizan y de qué manera.

Productos de trabajo

- Diagrama de estados: Producto de trabajo que fue de ayuda para entender el funcionamiento del Sistema de Ética. En la figura 20 se muestra parte de éste producto de trabajo utilizando la herramienta Bizagi.

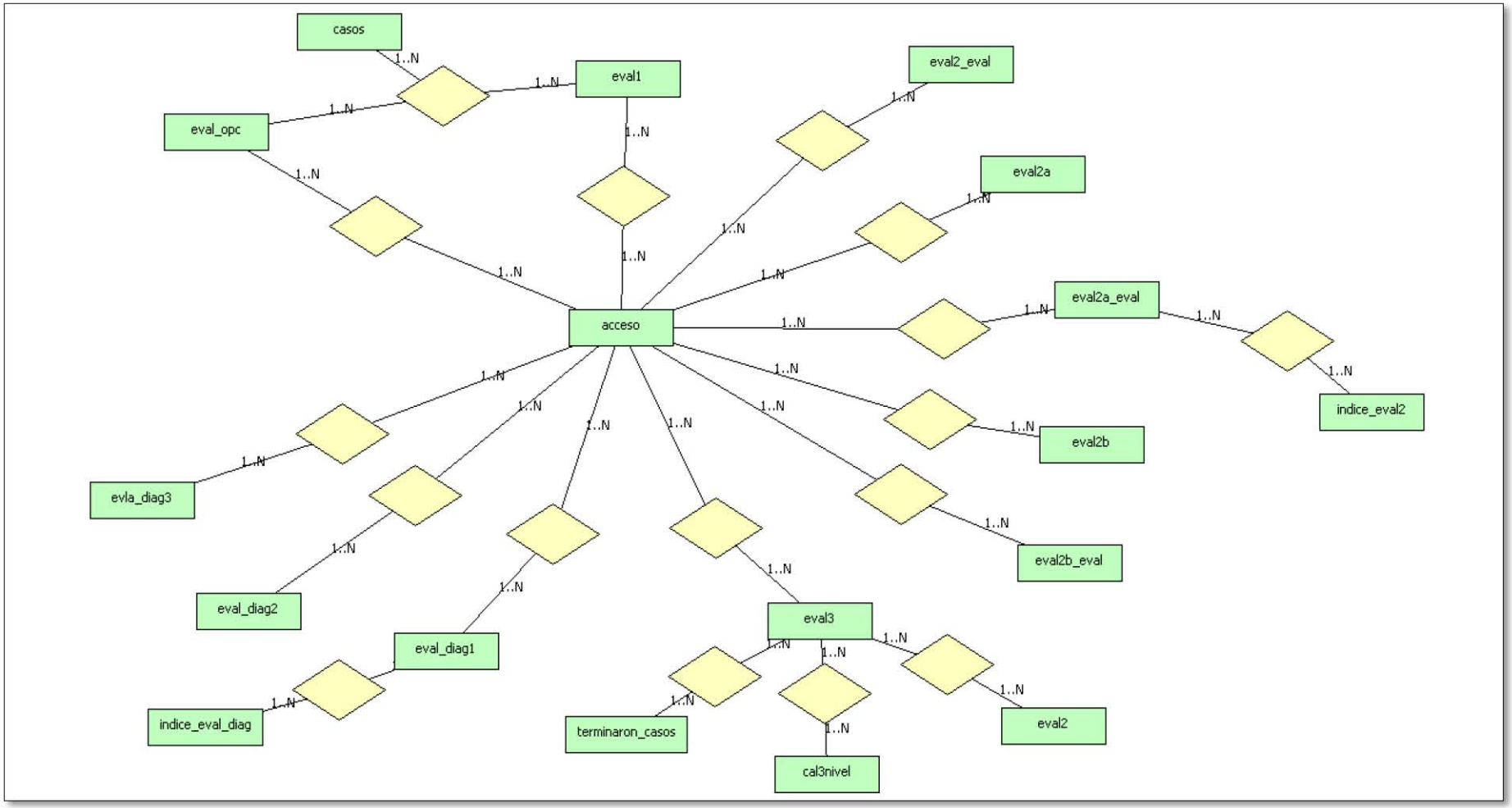


Figura 18. Diagrama E/R [verificado]

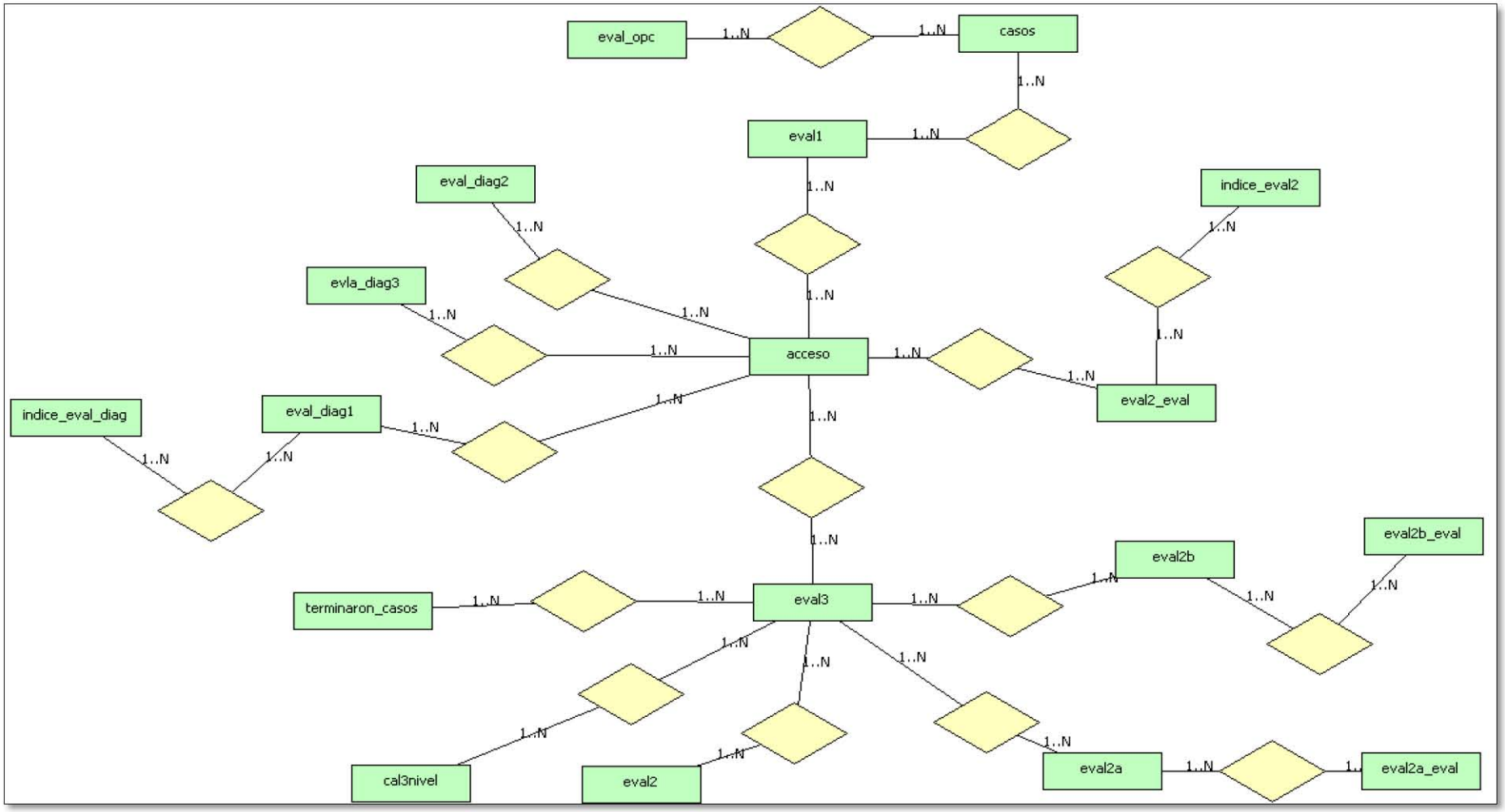


Figura 19. Diagrama E/R [actualizado]

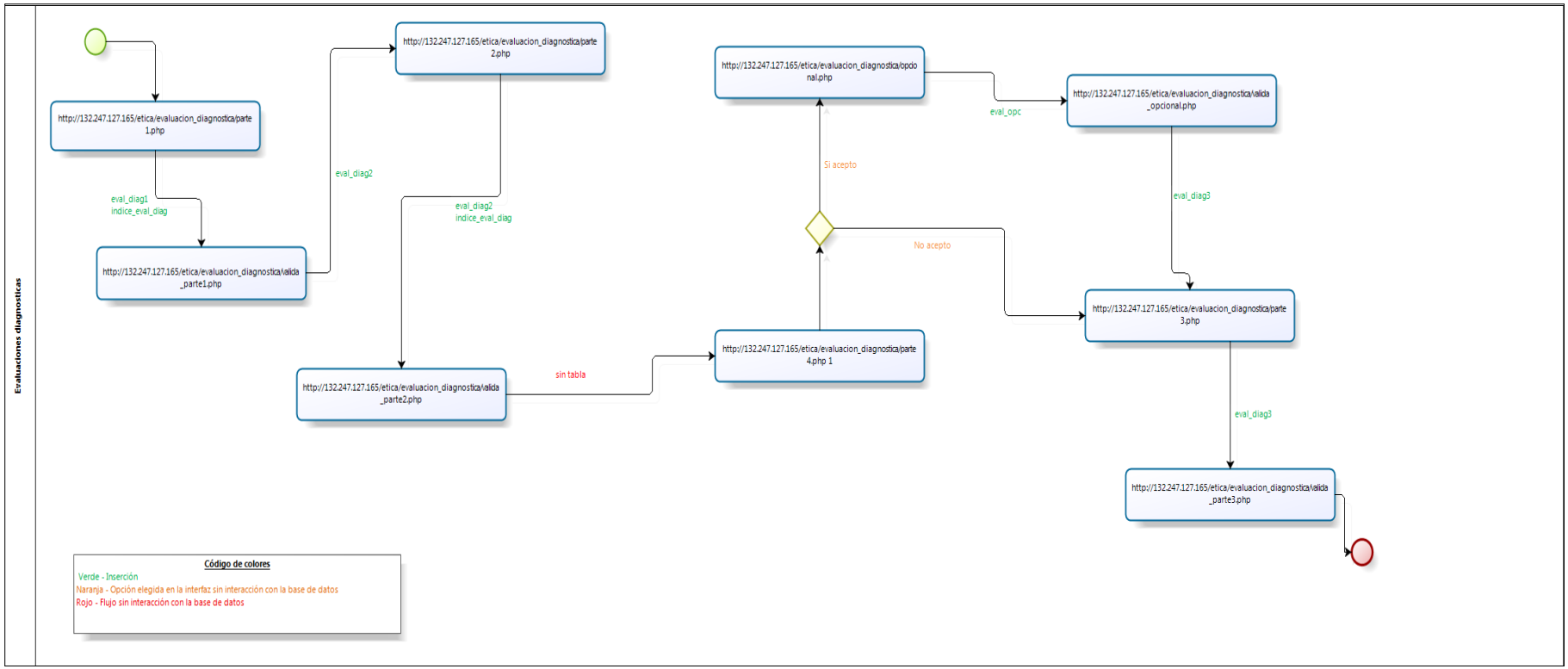


Figura 20. Diagrama de estados en Bizagi.

- Script Conteo_tuplas: Scripts con los cuales fue posible detectar aquellas tablas que se utilizaban y las que no. A continuación se muestra en la figura 21 a manera de ejemplo parte de éste producto utilizando pgAdminIII.

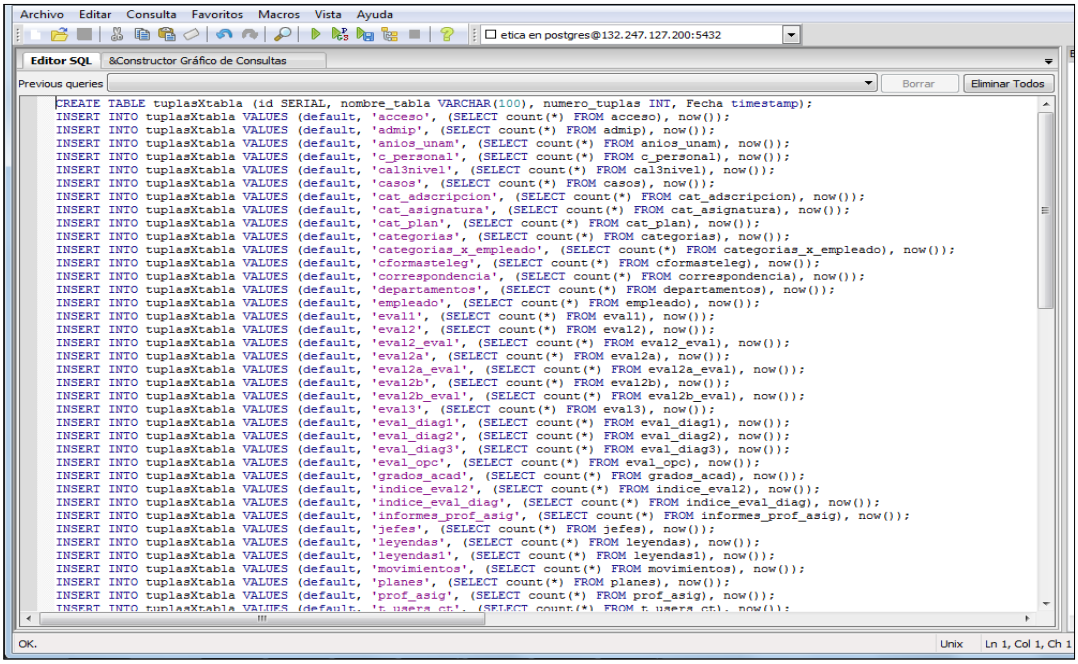


Figura 21. Script Conteo_tuplas en pgAdminIII.

- Script Borrado_tuplas: Script que permitió eliminar las tuplas de cada tabla con el fin de analizar el comportamiento de éstas. A continuación en la figura 22 se muestra un ejemplo de éstos utilizando la herramienta Notepad++.

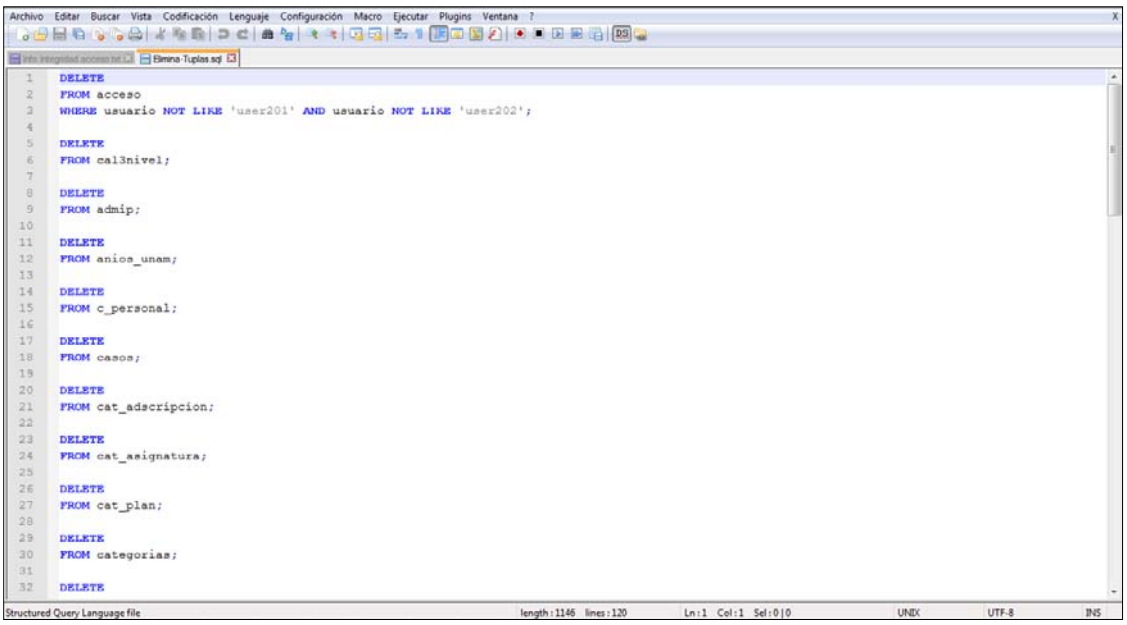


Figura 22. Script Borrado_tuplas en Notepad++.

4.4.5 Actualización del Modelo de la base de datos

Después de llevar a cabo el análisis de la base de datos en operación y navegar en el sistema fue posible mejorar el modelo de la base de datos y será éste Diagrama E/R el propuesto para la base de datos inmersa en el Sistema de Ética.

Productos de trabajo

- Diagrama E/R [propuesto]: Producto de trabajo con el cual fue posible describir el modelo final de la base de datos. En la figura 23 se muestra lo obtenido.

4.5. Implementación y prueba de la mejora

En esta fase se implementó el modelo mejorado en un ambiente de prueba para así validar su integración con el sistema de software y los datos almacenados sin afectar al sistema en operación.

4.5.1 Primera intervención de la base de datos (Tablas inútiles).

Etapa en la que se detectaron y eliminaron las tablas que resultaban inútiles dentro del diseño de la base de datos.

Productos de trabajo

- Listado tablas inútiles [verificado]: este producto de trabajo fue de ayuda para detectar las tablas inútiles dentro del modelo de la base de datos.

- ContienenNombreTabla: Producto de trabajo que permitió detectar aquellos archivos *PHP* del sistema de software que contenían el nombre de cada tabla.

- Base de datos intervenida: Base de datos después de haber eliminado las tablas inútiles detectadas.

- Respaldo primera intervención

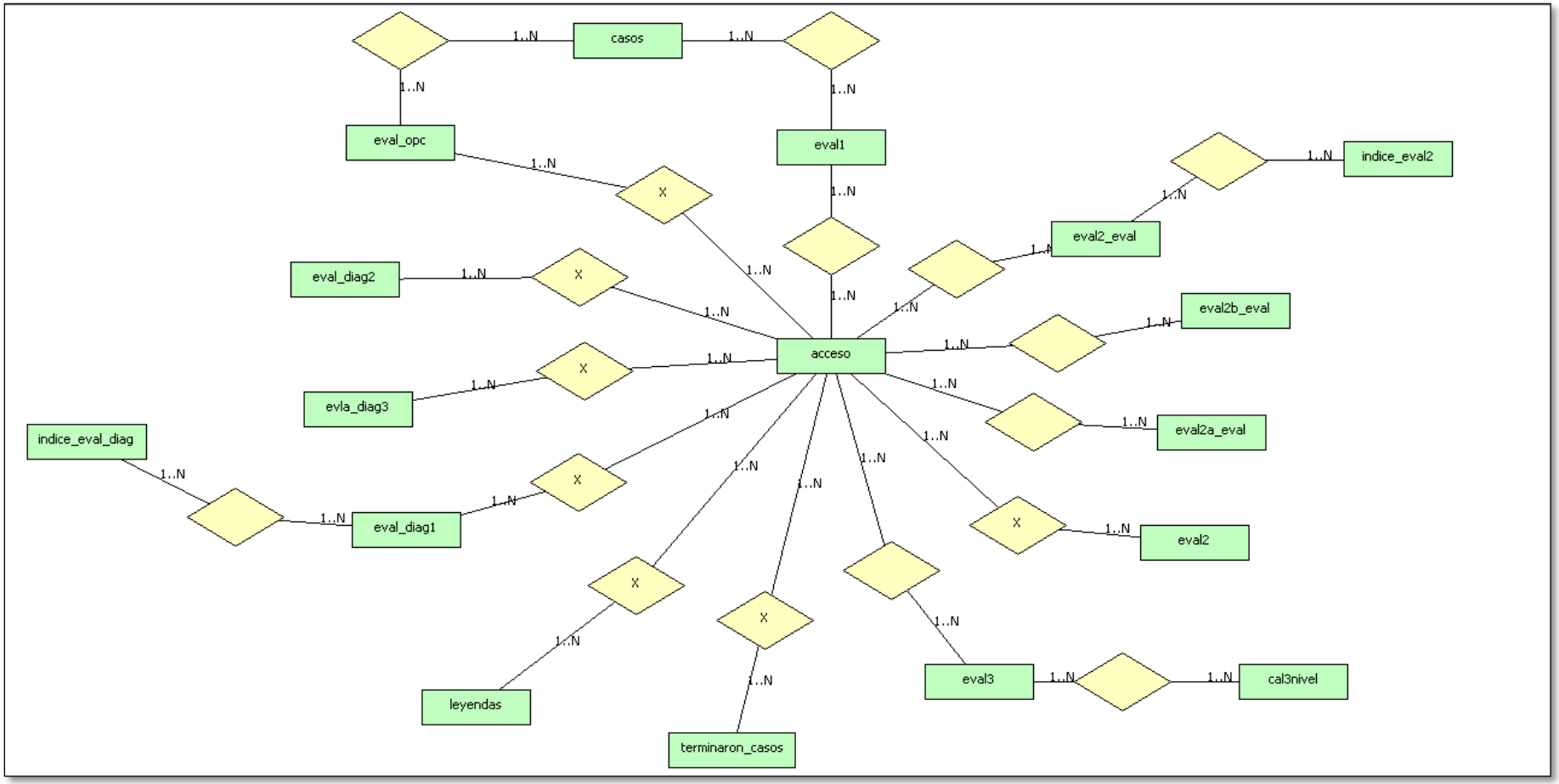


Figura 23. Diagrama E/R [propuesto]

4.5.2 Segunda intervención a la base de datos (Mejoras de integridad).

En esta fase se realizó la segunda intervención cuyo objetivo fue implementar las restricciones de integridad propuestas.

Productos de trabajo

- Propuesta_mejoras_Integridad [verificado]: Listado de restricciones que se aplicaron a la base de datos.
- Base de datos intervenida [2]
- Respaldo segunda intervención

4.6. Liberación de la mejora

En esta etapa 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.

4.6.1 Tercera intervención (Prueba final).

Productos de trabajo

- Base de datos final [creada]

4.7. Mejoras cualitativas y cuantitativas del caso práctico

Las mejoras que se aplicaron al caso práctico fueron las siguientes:

- Existían tablas que no tenían relación con los requerimientos de la base de datos por lo que se eliminaron 21 tablas del modelo relacional. Lo que es más, existían tablas que pertenecían a otros sistemas dentro de la misma base de datos, éstas fueron reubicadas.
- Se agregaron 17 PK's y 17 FK's a las tablas ya que ninguna contaba con ellas.
- Se eliminaron registros *basura,* es decir, se detectaron registros duplicados en cada una de las tablas después de haber agregado las respectivas PK's. Una vez realizado lo anterior se eliminaron dichos registros basura de cada una de las tablas.
- Después de haber agregado las respectivas FK's, se eliminaron 1,638 tuplas espurias de las relaciones. Cabe señalar que estas tuplas fueron el resultado de datos de prueba que el equipo de programadores insertó durante el desarrollo del sistema y mantenimientos posteriores. Dentro de éstas también se tenían datos de usuarios ya eliminados y que probablemente también fueron datos de prueba.
- 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 Entidad-Relación que modela el diseño resultante de la base de datos, ya que los administrativos no contaban con éste.
- Derivado de las mejoras a la base de datos, también se realizaron y reportaron modificaciones a los archivos PHP de la aplicación, resultando en una mejora del sistema en operación. Cabe resaltar que dichas modificaciones no alteraban la lógica del negocio, para aquellas que sí lo hacían o cuya corrección requería de un análisis más profundo éstas fueron informadas al equipo de desarrollo.

4.7.1 Desempeño futuro del nuevo diseño de la base de datos

En esta sección se analiza el diseño final de la base de datos con el objetivo de verificar que ésta no presente errores en el futuro.

En el apartado anterior se discutieron las mejoras aplicadas a la base de datos, éstas en su mayoría están enfocadas a las restricciones de integridad de entidad, no-nulidad, referencial y de dominio; sin embargo, existen ciertos puntos a discutir para conservar el buen desempeño de la base de datos a lo largo del tiempo, entre los más importantes se tienen:

- El equipo de desarrollo así como el diseñador de la base de datos serán los principales responsables de esta tarea. Deberán realizar el trabajo necesario y pertinente para dar mantenimiento apropiado y de forma constante, a la base de datos, poniendo especial atención a posibles cambios en las reglas del negocio u otras restricciones para así conservar a la base de datos "saludable" en el futuro.
- El respetar las dependencias funcionales requiere de más trabajo y esfuerzo, consideramos que para atender adecuadamente este asusto, es necesario que el equipo de programadores realice adecuaciones al sistema.

Considerando al conjunto de aspectos mencionados en esta sección 4.7, permitirá al equipo de desarrollo mantener a la base de datos integra y que a lo largo del tiempo se conserve la integridad de ésta.

5. Resultados

En este capítulo se presentarán los resultados obtenidos al realizar este trabajo a modo de lecciones aprendidas y oportunidades de mejora para ρDB.

5.1. Lecciones aprendidas

Lo más relevante que aprendí al realizar esta tesis fue:

- Es muy importante contar con un adecuado control de versiones en la implementación de un sistema de software así como en una base de datos, pues ello facilitará la toma de decisiones a futuro, las cuales también deben documentarse ya que de igual manera ello resulta ser conocimiento muy valioso.

- Se reafirmó el papel fundamental que tienen las bases de datos dentro de un sistema de software, ya que el buen desempeño de éste dependerá en gran medida del diseño de la base de datos. Si ésta no cumple con los requerimientos necesarios, su explotación no podrá llevarse a cabo de manera eficiente, además la información que el sistema de software arroje al usuario podría no ser confiable lo cual afectará al negocio en donde ésta se utilice.

- Cuando se desean realizar modificaciones o mejoras a una base de datos existente se requiere del apoyo de administrativos, diseñadores, clientes entre otros más, con la intención de adquirir información de interés. Esta situación en ocasiones puede resultar fácil pero claramente no siempre será así y es aquí cuando cobran importancia los productos de trabajo antes mencionados.

  Afortunadamente, durante la realización de esta tesis el equipo de trabajo que colaboró al llevar a cabo el caso práctico estuvo siempre con gran disposición para ayudar y así facilitar la información requerida de la base de datos con la que se trabajó.

  Agradezco al equipo de trabajo del Posgrado de Medicina que participó en el desarrollo del caso práctico presentado en esta tesis, el equipo estuvo conformado por: Sergio Villa, Erick Matla y David Velázquez, quienes me otorgaron todas las facilidades y el apoyo necesario durante la realización de éste trabajo.

- Dados los puntos anteriores, suele ser recomendable intentar ser más específicos al diseñar una base de datos, por ejemplo en el nombrado de tablas y atributos, el asignar un nombre a éstos que describa lo que se desea modelar con ellos será de gran ayuda para el trabajo que se desee realizar a futuro. Ésta para mí fue una de las lecciones más importantes pues fue con la que batallé más al realizar el caso práctico.

- Conocí el funcionamiento de nuevas herramientas como Bizagi Process Modeler, Notepad++ y el lenguaje de programación para Unix. Por otro lado mejoré mis conocimientos en PostgreSQL y pgAdminIII. Además pude aprender nuevos temas como lo son los Sistemas de Software e Ingeniería de Software entre otros más.

- Finalmente, respecto a los resultados obtenidos después de liberar la mejora de la base de datos, fue posible minimizar la redundancia en ésta, se logró obtener integridad en los datos y además se conservaron cada uno de los objetivos para los que fue creada.

5.2. Mejoras al proceso de ingeniería inversa

Al proponer de manera teórica el proceso de ingeniería inversa, ρDB resultó ser un proceso con una lista de productos de trabajo demasiado extensa, lo que auguraba cierta complejidad a la hora 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 a aquellos productos que se les estaba dando menor importancia que otros cuando realmente éstos sí la tenían.

Por otro lado uno de los elementos de la propuesta de KUALI-BEH que no se utilizó fueron los *criterios de verificación*, esto 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 nos 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.

Conclusiones

La motivación para realizar éste 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. El proceso obtenido puede ser mejorado e incluso complementado con algún otro método – por ejemplo el método TDD expuesto en el capítulo 3 – y queda abierto para quien decida hacerlo.

En lo personal, realizar este trabajo me dejo muy satisfecha, me permitió profundizar mis conocimientos en las bases de datos, aplicar lo aprendido a un caso de la vida real además de aprender cosas nuevas. Espero que ρDB sea de utilidad para quien lo consulte y no quedé sólo como un trabajo para obtener un título de Licenciatura. Por otro lado cabe mencionar que dejó en mí una gran curiosidad por aprender más respecto al tema para así poder aplicarlo en la vida real dada la importancia que tienen las bases de datos en mi carrera.

Bibliografía

1. Ambler, S. (2003). *Agile Database Techniques: Effective Strategies for the Agile Software Developer*. New York, NY, USA: John Wiley & Sons, Inc.

2. Ambler, S. (2007). *Test-Driven Development of Relational Databases.* IBM.

3. Ambler, S., Sadalage, P. (2006). *Refactoring Databases: Evolutionary Database Design*. Addison-Wesley Professional.

4. Ángeles M. (2013). Taller anual del área de Bases de Datos. *Sistema Evaluador Universal de Calidad de Datos.* PAEP y el Posgrado en Ciencia e Ingeniería de la Computación, UNAM.

5. Arango, R. (2013). *"Desarrollo y uso de herramientas para la refactorización de bases de datos".* UNAM, Facultad de Ciencias.

6. Avison, D., Fitzgerald, G. (2003). *Where Now for Development Methodologies?* Vol. 46, No. 1, pp. 79-81.

7. Batini, C., Scannapieco, M. (2006). *Data Quality. Concepts, Methodologies and Techniques.* Springer.

8. Chen, P. (1976). *The Entity Relationship Model - Toward a Unified View of Data.*

9. Codd, E. (1969). *A Relational Model of Data for Large Shared Data Banks.* San Jose, California: IBM research Laboratory.

10. Data Integrity, http://technet.microsoft.com/en-us/library/aa933058(v=sql.80).aspx 27/11/2014

11. Data Quality Working Group, *Guidance on S-101 for the Data Quality Working Group*. http://www.iho.int/mtg_docs/com_wg/TSMAD/TSMAD22/TSMAD22_DIPWG3-11.7A_S-101_Data_Quality_FINAL.pdf 27/11/2014

12. Elmasri, R., Navathe, S. (2010). *Fundamentals of Database Systems.* Addison Wesley Iberoamericana.

13. Fowler, M. (1999) *Refactoring: Improving the Design of existing Code*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.

14. García-Molina, H., Ullman, J., Widom, J. (2009). *Database Systems. The Complete Book.* Prentice Hall.

15. Juárez-Ramírez R, Licea G., Cristóbal-Salas, A. (2007). *Ingeniería Inversa y Reingeniería Aplicadas a Proyectos de Software Desarrollados por Alumnos de Nivel*

Licenciatura. Facultad de Ciencias Químicas e Ingeniería, Ingeniería en Computación, Universidad Autónoma de Baja California.

16. Kerr, K., Norris, T., Stockdale, R. (2007). *Data Quality Information and Decision Making a Healthcare Case Study.*

17. Liu, L., Özsu, M. (2009). *Encyclopedia of Database Systems*. Springer Reference, ISBN 978-0-387-49616-0

18. Maydanchik, A. (2012). *Data Quality Assessment*. Technics Publications.

19. Morales, M., Oktaba, H. (2012). *KUALI-BEH Kernel Extension*. Annex B (Informative) in: Essence – Kernel and Language for Software Engineering Methods. Object Management Group.

20. Netronycs, http://www.netronycs.com/modelos_de_base_de_datos.html 27/11/2014

21. Silberschatz, S., Korth, H., Sudarshan, S. (2010). *Database System Concepts*. Mc Graw-Hill.

22. Sommerville, I. (2011). *Software Engineering.* Pearson.

23. Weitzenfeld, A. (2005). *Ingeniería de Software orientada a objetos con UML, Java e Internet.* International Thomson Editors.