UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES ARAGÓN

"BASE DE DATOS PARA ADMINISTRACIÓN DE REPORTE DE FALLAS"

TRABAJO ESCRITO

EN LA MODALIDAD DE SEMINARIOS Y CURSOS DE ACTUALIZACIÓN Y CAPACITACIÓN PROFESIONAL QUE PARA OBTENER EL TÍTULO DE:

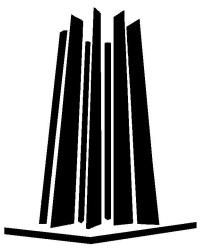
INGENIERO EN COMPUTACIÓN

P R E S E N T A :

A L F R E D O T O R R E S

B E L T R Á N

ASESOR: M. EN C. MARCELO PÉREZ MEDEL



MÉXICO, 2009

AGRADECIMIENTOS

Deseo agradecer a mi familia que me apoyo durante mi vida universitaria, a mi Padre por su confianza y ejemplo de fortaleza, constancia, honradez, y amor hacia sus hijos.

En especial a Juanita, gracias por tu apoyo y escucharme siempre, por tus consejos y por cuidar siempre de mi padre y por mí.

A Dios por las bendiciones que me ha brindado. A Esther, la mayor de todas, por ser mi amiga, por tu amor y confianza; a mis hijos Miguel y Uriel, fuente de inspiración para ser siempre mejor.

A mi Universidad que gracias a su instrucción, aprendizaje y formación obtenida, me ha permitido representarla con orgullo y ser mi cimiento de crecimiento y éxito profesional.

ÍNDICE

INTRODUCCIÓN

El diplomado de Diseño de Sistemas de Información Orientado a Negocios con SQL Server y Oracle, nos ha preparado con los conocimientos que nos permitirán analizar, diseñar, planificar y construir sistemas soportados en lo que defino como la parte medular: la base de datos.

El resultado de un sistema que no está bien analizado, diseñado y soportado bajo una base de datos eficiente y modular, implica proyectos con mayores costos por rediseños, reconstrucción además del costo de mantenimiento que implica mantenerla sin incidencias en producción (desbordamiento de segmentos, alto consumo de recursos por falta de índices, procesos de mantenimiento, respaldo y recuperación, entre otros).

El presente, considera la aplicación de las habilidades y conocimientos adquiridos al analizar, planificar, diseñar y construir una base de datos para la administración de reportes de fallas.

El control de reportes de fallas o incidencias de las áreas operativas de usuarios finales de una empresa, permite obtener métricas para identificar problemas recurrentes y actuar en su recuperación.

Nuestro modelo desarrollado, considera la recepción de reportes de fallas a través llamadas telefónicas que son atendidas a través de un Centro de Atención de Usuarios.

Al momento de recibir la llamada, un operador captura en la base de datos la incidencia reportadas por un usuario(s), y con base al componente afectado, se identificará al responsable de su recuperación.

La base de datos posee los datos de localización del responsable, de tal forma que el operador del Centro de Atención de Usuarios, escale el reporte con base al nivel de severidad o impacto que representa la incidencia.

El reporte posee un ciclo de vida de atención, considerando los estados de ABIERTO, RECUPERADO y CERRADO.

El reporte puede ser cerrado sólo al contar con un diagnóstico de lo ocurrido y especificando la causa origen que lo provocó.

Finalmente con el registro de fallas, se facilitará la extracción de métricas a través de reportes, de los problemas recurrentes, a fin de analizar y orientar las actividades de atención para su solución.

El entregable final de este proyecto, es el código fuente de construcción de la base de datos, así como los procedimientos almacenados que permiten cubrir las reglas de negocio, altas, bajas y cambios de los registros de información con sus respectivas validaciones para mantenimiento de integridad de los datos y consultas de extracción de información.

CAPÍTULO I – BASE DE DATOS

Orígenes del Hardware y Administración de Datos

El manejo de la información puede ser considerado como una de las fuentes de poder de nuestro mundo actual, el que la posee en forma precisa y a tiempo, podrá realizar en base a su experiencia, la más acertada toma de decisión.

La administración y uso de la información, ha estado presente en la historia del hombre, asimismo ha contribuido a su propio crecimiento. El avance tecnológico alcanzado en el presente siglo para administrar la información, supera en mucho los antiguos métodos. Pero, el haber alcanzado este logro tecnológico, no fue algo sencillo. Aún más, si se toma en cuenta la capacidad de almacenamiento y comunicación de las computadoras de hoy. Haciendo una perspectiva de las principales contribuciones tecnológicas en la historia del hombre, se encuentra la cultura egipcia, la cual asentaba su información mediante jeroglíficos[1], una de las primeras formas de comunicación escrita, inventada 3000

[1] Se caracteriza por el uso de signos, cuyo significado se conoce gracias al descifrado de los textos contenidos en la Piedra de Rosetta, que fue encontrada en 1799. Conseguir descifrar este documento se lo debemos a los estudios realizados por Thomas Young y, fundamentalmente, a Jean-François Champollion quien logró descifrar el método de su lectura en 1822.

años antes de nuestra era. La invención del papel por los chinos en el año 105[2]. La imprenta en 1426 por Gutenberg[3]. El teléfono de Alexander Graham Bell[4] en 1876, etcétera. Pero fue hasta el siglo XVII en Francia, que se inventó uno de los primeros dispositivos para cálculos por Blaise Pascal[5] (1640). Consistía en una pequeña caja de engranes unidos capaz de hacer sumas y restas. En 1694 Gottfried Wilhelm von Leibniz[6] construye otro dispositivo más sofisticado, que consistía en cilindros y ruedas unidas por engranes, la cual era capaz de multiplicar y dividir.

Creada en 1805 por Joseph Marie Jacquard[7], el dispositivo que sirvió como cimiento a muchos diseño posteriores, incluyendo las primeras computadoras del presente siglo. Tal dispositivo era capaz de crear automáticamente diseños textiles

[2] El proceso para hacer papel de fibra vegetal había sido inventado primero por Cai-lun en China durante la Dinastía Han en 105. La invención de papel facilitó grandemente el desarrollo del idioma, las artes gráficas y cultura, primero en China, luego en el mundo árabe, y después en el Oeste.

[3] La invención de la imprenta con caracteres móviles, obra del alemán Johannes Gutenberg, permitió la posibilidad de realizar tiradas de múltiples ejemplares de libros, facilitó el acceso de un mayor número de personas en todo el mundo al saber escrito y conllevó radicales transformaciones en la política, la religión y las artes.

[4] Históricamente la invención del teléfono se le ha atribuido al escocés-norteamericano Alexander Grahan Bell; no obstante, en junio de 2002, el Congreso de Estados Unidos reconoció que el teléfono fue concebido por un desconocido inmigrante italiano llamado Antonio Meucci.

[5] Blaise Pascal (Clermont-Ferrand, Auvernia, Francia, 19 de junio de 1623 - París, 19 de agosto de 1662) fue un matemático, físico, filósofo y teólogo francés, considerado el padre de las computadoras junto con Charles Babbage.

[6] Nacido el 1 julio de 1646, en Leipzig, Saxony (ahora Alemania), Fallecido el 14 de noviembre de 1716, en Hannover, Hanover (ahora Alemania).

[7] Joseph Marie Jacquard (n.Lyon, 7 de julio de 1752 m. Oullins, 7 de agosto 1834) Fue un inventor francés conocido por automatizar, mediante el uso de tarjetas perforadas, el llamado telar de Jacquard utilizaba las tarjetas perforadas para conseguir tejer patrones en la tela.

mediante una serie de tarjetas perforadas y constituyó uno de los primeros medios de almacenamiento de datos. Utilizando únicamente elementos mecánicos, las primitivas calculadoras desarrolladas por Charles Babbage[8] ("Máquinas de Diferencias" y la "Máquina Analítica"), eran capaces de realizar cálculos numéricos de complicados polinomios para construir tablas para navegación y astronomía. Sus diseños incluyeron muchos de los principios de las computadoras modernas.

El diseño de estas máquinas de cálculo estaba limitado únicamente por la dificultad de fabricación de sus componentes. Para principios del siglo XX, Herman Hollerith[9] diseña la máquina que construía tarjetas con perforaciones y las ordenaba para su uso. Estas tarjetas constituyeron una de las primeras fuentes masivas de almacenamiento de información, y con éstas se contabilizó el censo de 1890 de los Estados Unidos, el cual fue concluido en tan sólo un mes.

[8] (Teignmouth, 1792 - Londres, 1871) Matemático e ingeniero británico, inventor de las máquinas calculadoras programables. En 1833 completó su "máquina diferencial", capaz de calcular los logaritmos e imprimirlos de 1 a 108.000 con notable precisión, y formuló los fundamentos teóricos de cualquier autómata de cálculo.

[9] Herman Hollerith (Buffalo, Nueva York, 29 de febrero de 1860 — 17 de noviembre de 1929) fue un estadista que inventó la máquina tabuladora. Es considerado como el primer informático, es decir, el primero que logra el tratamiento automático de la información.

En 1896 Hollerith crea una empresa denominada Tabulating Machine Company, la cual eventualmente formó lo que hoy es IBM (International Business Machines Corporation).

Estos avances permitieron a James Powers[10] en 1911, desarrollar dispositivos capaces de alimentar tarjetas automáticamente e imprimir resultados. Forma la empresa Powers Tabulating Machine Company, la cual se convertiría en la división UNIVAC de UNISYS.

La necesidad de realizar cálculos de forma rápida y confiable, durante la primera guerra mundial, propicia la investigación y desarrollo de las primeras computadoras analógicas diseñadas para cálculos balísticos por las Universidades de Massachusetts y Pennsylvania.

Con la segunda guerra mundial se incrementa la investigación y desarrollo de nuevos dispositivos de cálculo. En 1944 los laboratorios Bell concluyen la primera computadora digital electromecánica, la Mark I[11](figura I.1), la cual almacenaba 72 cifras numéricas. Superada posteriormente en 1946, por la

[10] Poco se sabe de él, excepto que nació en Rusia, que era ingeniero eléctrico, y que fue capaz de acabar con el monopolio de Hollerith.

[11] El proyecto entre IBM y Howard Aiken para construir una computadora se inició en 1939. La Mark I se terminó en 1943, presentándose oficialmente en 1944. En un principio la MARK I se llamaba ASCC (Calculadora Automática de Secuencias Controladas). Era una máquina automática eléctrica, aunque tenía componentes electromecánicos.

ENIAC[12] (Electronic Numerical Integrator and Calculator) cuyo diseño era completamente electrónico (poseía alrededor de 18000 bulbos) y era capaz de realizar 5000 sumas por segundo (alrededor de 1000 veces más rápida que Mark I) Figura I.2

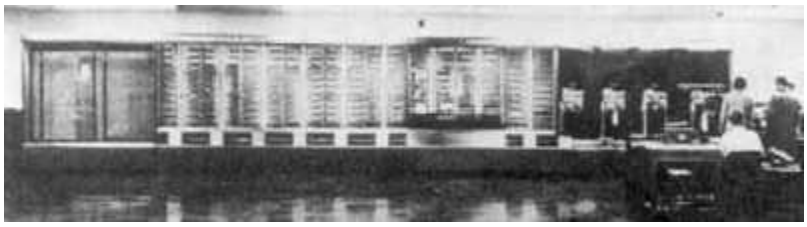


Figura I.1 Computadora MARK I

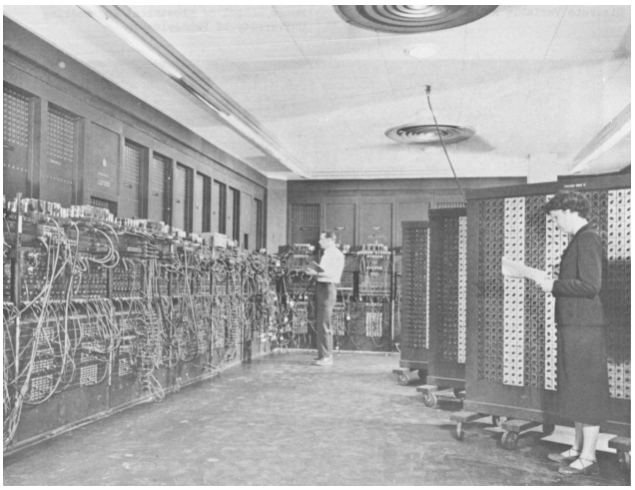


Figura I.2 Computadora ENIAC.

[12] La aparición de la computadora ENIAC estructurada por John Mauchly y John Eckert, y puesta en funcionamiento en 1945, marca el inicio de nuestra era computacional. Se trataba de una máquina programable y universal a la que se podía llamar electrónica.

A partir de estas nuevas computadoras, se inició la clasificación por generaciones, las cuales se dividen en base a las mejoras en diseño y funcionalidad. La primera generación (1950 - 1960) incluye a las primeras computadoras como la UNIVAC I (1951), la IBM 701 (1953), IBM 705 (1955 la cual tenía memoria principal). El pasar del uso de bulbos a transistores, marcó el comienzo de la segunda generación de computadoras, cuyo tamaño era menor y cuya velocidad era mayor. A mediados de los 60's, se inicio la tercera generación de computadoras, las cuales incluyeron los primeros circuitos integrados, y en esta década nace el concepto: Base de Datos.

Paralelamente al desarrollo de los equipos de cómputo, se incrementó la necesidad de nuevos dispositivos de almacenamiento. El primer medio de almacenamiento fueron las tarjetas perforadas usadas en dispositivos de cálculos de 1920 a 1940, y utilizadas por las primeras computadoras de 1950 hasta principios de los 70's.

Los dispositivos magnéticos se emplearon a principios de los 40's, pero su uso comercial es presentado en cintas magnéticas en 1952 con la computadora UNIVAC[13] (Figura I.3). Ya en la década de los 60's, era el principal medio de

[13] El UNIVAC fue la primera computadora diseñada y construida para un propósito no militar. Fue desarrollada para la Oficina del Censo en 1951 por los ingenieros John Mauchly y John Presper

almacenamiento de datos. Esta tecnología continuó su desarrollo, actualmente todavía se utilizan las cintas, pero de un tamaño mucho menor y con mayor capacidad. Asimismo, se pasó de las cintas a unidades de disco, cuyas capacidades y tamaños actuales difieren en mucho a los primitivos discos usados en 1957. El uso de unidades de disco, poco a poco, se convirtió en el medio más aceptado para el almacenamiento de información y su uso era el más común hasta principios de los 80's. Posteriormente el Disco Compacto[14] y DVD[15] ha surgido como la nueva tecnología para almacenar cantidades enormes de información, no superado ni comparado con los anteriores.

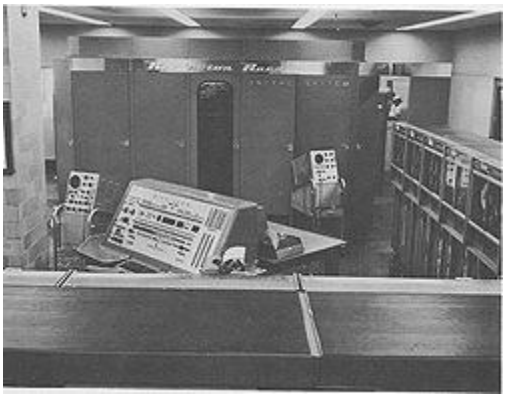


Figura I.3 Computadora UNIVAC

Eckert, que empezaron a diseñarla y construirla en 1946.

[14] Está formado por una base de plástico recubierta de un material que refleja la luz, habitualmente aluminio. La grabación de los datos se realiza creando agujeros microscópicos que dispersan la luz (pits) alternándolos con zonas que la reflejan (lands).

[15] Digital Video Disc o Disco Versátil Digital: Utiliza un láser infrarrojo de mayor capacidad que es el láser rojo utilizado en los CDs, cuya longitud de onda es más corta, lo que ofrece un mayor espacio de almacenaje y contará con una mayor capacidad para evitar los errores por el tiempo, ya que cuenta con una tecnología que evitará hasta seis veces más el error con que cuenta un CD.

Pero el logro tecnológico más significativo en toda la historia del hombre, se inicio con la introducción de las primeras computadoras personales (1981 IBM). Con esta nueva generación de computadoras se da origen al desarrollo de mejor software, el cual debería ser útil y fácil de usar y distribuir. Por primera vez, la administración de datos en forma electrónica se convirtió en una función común para muchas empresas.

El requerimiento de manejar enormes cantidades de datos, almacenarlos, administrarlos o consultarlos, propició que se generaran metodologías para mantener en forma consistente la información. Esta clase de metodologías que permitieron que los datos se convirtieran en fuentes de información coherentes, es conocida actualmente con el término Base de Datos.

Las primeras máquinas de este siglo permitieron guardar datos de una forma muy limitada. En el comienzo de esta época tecnológica, los análisis de las fuentes de datos y su forma de representarlos electrónicamente no variaban mucho uno de otro, los datos almacenados poseían una estructura similar a los expedientes y archivos en papel sobre los cuales se había realizado el análisis. De esta forma, nacieron los denominados Sistemas de Archivos, que pueden ser considerados como las primitivas bases de datos, las cuales fueron ampliamente usadas durante aproximadamente dos décadas.

El software que permitía su manejo era también muy sencillo, y aunque permitía generar reportes, poseía la desventaja de ser poco práctico y flexible al realizar modificaciones. Por ejemplo, el agregar o modificar un campo en un sistema de archivos con este software, impactaba a todo el programa porque los archivos poseían dependencia estructurada, es decir, el acceso a los archivos estaba en función a la forma definida en su estructura porque existía una diferencia en el formato lógico de los datos (lo que ve el usuario) y su formato físico (forma interna de almacenamiento en la computadora).

Los problemas de mantenimiento de los Sistemas de Archivos favorecieron la investigación y desarrollo de sistemas que administraran los datos de forma más eficiente. Se definieron actividades para la administración de datos, las cuales estaban encaminadas a la recopilación, almacenamiento y consulta de datos almacenados en un repositorio central denominado Base de Datos.

El término Base de Datos se asocia a la administración y uso de datos, no obstante, éste se puede definir como "...la estructura integrada y compartida que mantiene una colección de a) Usuarios finales de Datos, b) Metadatos[16] 'datos acerca de los datos' a través de los cuales los datos están integrados" . Los

[16] Los metadatos son datos altamente estructurados que describen información, describen el contenido, la calidad, la condición y otras características de los datos.

Metadatos proveen la descripción de las características de los datos y el conjunto de relaciones que enlazan los datos existentes dentro de la base de datos.

A diferencia de los sistemas de archivos en los cuales existen muchos archivos sin relación, los sistemas de bases de datos permiten almacenar datos relacionados lógicamente dentro una sola entidad central de datos.

Modelo de Base de Datos

El diseño de las estructuras de datos y las relaciones entre ellas dentro de la base de datos es una actividad crucial. Su diseño se realiza de una forma más sencilla cuando se utilizan modelos. Un modelo de base de datos es una colección de construcciones lógicas usadas para representar las estructuras de datos y las relaciones encontradas dentro de la base de datos.

Los modelos de base de datos pueden ser agrupados en dos categorías: modelo conceptual y modelo de implementación. El modelo conceptual se enfoca en la naturaleza lógica de la representación de datos. Por lo cual, el modelo conceptual está concernido con lo que está representado en la base de datos en vez de cómo éste se representa. El modelo conceptual incluye el Modelo Entidad Relación y el Modelo Orientado a Objeto.

El modelo conceptual utiliza tres tipos de relaciones para describir la asociación entre los datos. Estos tres tipos de relaciones son uno a muchos (figura 1.4) , muchos a muchos (figura 1.5) y uno a uno (figura 1.6).

Relación Uno a Muchos.

Figura 1.4

En este tipo de relación, cada elemento de la entidad A pude poseer muchos atributos en la entidad B.

Relación Muchos a Muchos.

Figura 1.5

Cada elemento de la entidad A puede tener varios atributos en la entidad B

Relación Uno a Uno

Figura 1.6

Cada atributo dentro de la entidad A está relacionado con un solo atributo de la entidad B.

Por su parte el modelo de implementación pone especial énfasis en como los datos son representadas en la base de datos o en cómo estas estructuras de datos son implementadas para representar lo que está modelado. Incluye los

modelos de base de datos jerárquico[17] (Figura 1.7), el modelo de base de datos de red[18] (Figura 1.8) y el modelo de base de datos relacional[19], éste último el más común.

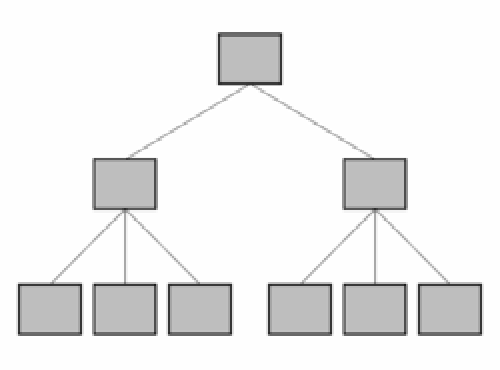


Figura 1.7 Modelo Jerárquico

[17] El modelo jerárquico de bases de datos fue el pionero en los sistemas de bases de datos, allá por comienzos de los años 60. En realidad no hay un modelo teórico detrás sino que surgió a raíz de los trabajos de IBM y de la NAA (North American Aviation) que dieron lugar al IMS (Information Management System) que podemos considerar el primer sistema de base de datos jerárquico.

[18] Una base de datos de red se compone por una colección de registros que se conectan entre si por medio de ligas. Una liga es una relación que se establece solamente entre dos registros; es decir; debe utilizarse una liga para cada relación entre una pareja de registros.

[19] Este modelo considera la base de datos como una colección de relaciones. De manera simple, una relación representa una tabla que no es más que un conjunto de filas, cada fila es un conjunto de campos y cada campo representa un valor que interpretado describe el mundo real. Cada fila también se puede denominar tupla o registro y a cada columna también se le puede llamar campo o atributo.

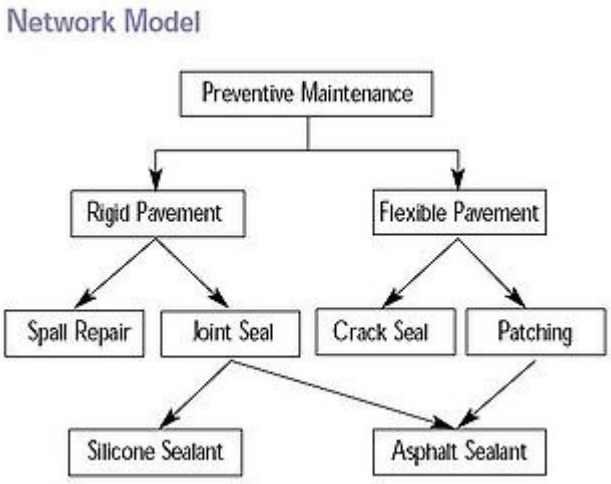


Figura 1.8 Modelo de Red

Modelo de Base de Datos Relacional

El modelo de base de datos relacional (Figura 1.9) es implementado a través de muy sofisticados Sistemas de Administración de Base de Datos Relacional (RDBMS Relational DataBase Management System). El RDBMS permite que el modelo de datos sea más fácil de entender e implantar. En éste, se diseñan las colecciones de tablas en las cuales los datos son almacenados. Cada tabla consiste de una matriz con intersección de renglones y columnas. Las tablas o relaciones se enlazan al compartir un atributo en común.

Atributo 1	Atributo 2	.....	Atributo n	
XXXXXXXX	XXXXXXXX	XXXXXXXX	XXXXXXXX	Tupla 1
XXXXXXXX	XXXXXXXX	XXXXXXXX	XXXXXXXX	Tupla 2
XXXXXXXX	XXXXXXXX	XXXXXXXX	XXXXXXXX	.
XXXXXXXX	XXXXXXXX	XXXXXXXX	XXXXXXXX	.
XXXXXXXX	XXXXXXXX	XXXXXXXX	XXXXXXXX	Tupla n

Figura 1.9 Modelo Relacional

Objetos de la Base de Datos Relacional

Una base de datos relacional se integra por colecciones de objetos, donde la tabla constituye el objeto básico. Una tabla contiene información referente a una entidad. Un reglón o registro, describe una instancia de esa entidad; una columna o campo describe un atributo de esta entidad. Las llaves primarias identifican a un reglón único dentro de la tabla. Las llaves foráneas relacionan los renglones de una tabla en otra. Finalmente, los índices son objetos que mejoran el proceso de localización de datos en una tabla. El índice relaciona cada valor de la columna indexada a la página o posición física sobre el renglón que contiene el dato, mejorando el desempeño.

La ventaja de un sistema RDBMS es la creación de múltiples tablas relacionadas que no contienen datos redundantes, al permitir relaciones de los tipos Uno a Uno, Uno a Muchos y Muchos a Muchos.

Manejador de Base de Datos

Un Data Base Management System o DBMS realiza las funciones y procesos que garantizan la integridad y consistencia de los datos en las base de datos. Muchas de estas funciones son transparentes a los usuarios finales y muchas de estas funciones pueden ser realizadas solo a través del DBMS. Esta funciones incluyen la administración del diccionario de datos, administración de datos almacenados, transformación de datos y presentación, administración de seguridad, control de acceso multi-usuario, administración de respaldo y recuperación, administración de la integridad de datos, leguaje de acceso a la base de datos e interfaces para programación e interfaces de comunicación a la Base de Datos.

1. Administración del Diccionario de Datos

   El DBMS requiere que la definición de los elementos de datos y sus relaciones (metadatos) sean almacenados en un diccionario de datos. El DBMS utiliza el diccionario de datos para buscar las estructuras de datos requeridas y sus relaciones que finalmente facilitan las tareas de consulta y modificación.

2. Administración de Datos Almacenados

   El DBMS crea las complejas estructuras requeridas para el almacenamiento de datos. Además algunos integran reportes de la definición de las estructuras, reglas de validación y procedimientos.

3. Transformación y Presentación de Datos

   Transforma las solicitudes lógicas en comandos que encuentran la ubicación física de los datos, los recupera y presenta en forma coherente al usuario.

4. Administración de Seguridad

   El DBMS crea un sistema de seguridad que asegura la seguridad y control de datos en la base de datos. Las reglas de seguridad determinan cuales usuarios pueden ingresar a la base de datos, cual conjunto de datos puede consultar y cuales operaciones sobre éstos datos puede realizar (lectura, agregar, borrar o modificar).

5. Control de Acceso Multi-Usuario

   Las estructuras generadas por el DBMS permiten acceso multi-usuario a los datos. Este utiliza algoritmos que aseguran que múltiples usuarios pueden ingresar a la base de datos concurrentemente y garantiza la integridad de estos datos.

6. Respaldo y Recuperación

   Proporciona procedimientos para respaldo y recuperación de datos.

7. Integridad de datos

   Utiliza las reglas y relaciones almacenadas dentro del diccionario de datos para asegurar la consistencia de datos, así se minimiza la redundancia de datos.

8. Leguaje de acceso a la base de datos e interfaces para programación

Proporciona acceso a los datos a través un de un lenguaje de búsqueda no procedural. El lenguaje de búsqueda posee dos componentes un Lenguaje de Definición de Datos (DDL Data Definition Language) y Lenguaje de Manipulación de Datos (DML Data Manipulation Language). El DDL define las estructuras en las cuales los datos son almacenados y el DML permite a los usuarios extraer los datos de la base de datos. El DBMS también provee acceso a los datos con lenguajes procedural como COBOL, C, PASCAL, Visual Basic, Power Builder entre otros. También proporciona las utilerías para administración para crear, implantar, monitorear, y mantener la base de datos.

9. Interfaces de Comunicación de la Base de Datos

Los actuales DBMS proporcionan características especiales de comunicación diseñadas para permitir a la base de datos aceptar transacciones dentro de una red de computadoras. Con éstas funciones de acceso es posible accesar la base de datos en Internet.

Generalmente los DBMS están diseñados para producir respuestas inmediatas a una transacción y son conocidos como un DMBS transaccional o DBMS de producción. Sin embargo, las bases de datos utilizadas en un Data Warehouse son las llamadas DB de Soporte de Decisión, en estas los datos sirven como soporte para la toma de decisiones; se enfocan principalmente a la producción de información requerida para hacer decisiones tácticas o estratégicas.

Modelos Data WareHouse

Debido al auge de los sistemas Data Warehouse, existe una gran divergencia para establecer una definición en forma precisa. Se clasifican de acuerdo a su arquitectura, a la forma en que los datos son almacenados o en la forma en que alimenta la información. Pero todas las definiciones convergen a los conceptos establecidos por W. Bill Inmon[20], considerado el padre del Data Warehouse: "Un data warehouse es un conjunto de datos integrados, orientados por componentes, que varía con el tiempo y que no son transitorios, los cuales soportan el proceso de toma de decisiones de una organización" . Cada aspecto comprende los siguientes:

1. Integración. Un Data WareHouse constituye una base de datos centralizada que reúne datos de múltiples y diversas fuentes. Estandariza y organiza los diversos formatos de los datos fuentes para mantener una información uniforme y consistente.
2. Orientado por componente. La información se encuentra resumida e integrada en componentes, para así responder a las necesidades

[20] Inmon defiende una metodología descendente (top-down) a la hora de diseñar un almacén de datos, ya que de esta forma se considerarán mejor todos los datos corporativos. En esta metodología los Data Marts se crearán después de haber terminado el data warehouse completo de la organización.

particulares de cada área funcional de la empresa. Cada componente puede ser visto como un bloque que representa a una unidad de negocio en particular: Productos, Clientes, Departamentos, Regiones, etcétera.

3. Variante en el Tiempo. La información está asociada a una fecha, con lo cual se permite obtener resultados por día, semana, meses e incluso años. Las consultas con variación en el tiempo son realizadas rápidamente a comparación de una base de datos de producción.
4. No Transitorio. El Data WareHouse mantiene datos históricos, que son derivados de los datos de producción. Éstos, pueden ser actualizados pero no eliminados, con lo cual se va incrementando su volumen y el tamaño de la base. Por esto, los requerimientos en equipo de hardware, así como el funcionamiento del DBMS que administra los datos son mayores a los utilizados por una base de datos relacional convencional.

Estos cuatro componentes constituyen los cimientos de una base de datos de soporte de decisión. Básicamente el data warehouse es una aplicación de base de datos que usa su propio sistema de administración. Deriva su información de otros sistemas de base de datos que sustentan las operaciones diarias. Estas bases de datos son conocidas como Bases de Datos con Procesamiento de

Transacciones en Línea (On Line Transaction Processing OLTP[21]) porque la forma de alimentar los datos, actualizarlos o eliminarlos ocurre en forma continua. Para alimentar el Data WareHouse con datos de producción se implantan diversos mecanismos para extraerla, transformarla y agregarla (Figura 1.10).

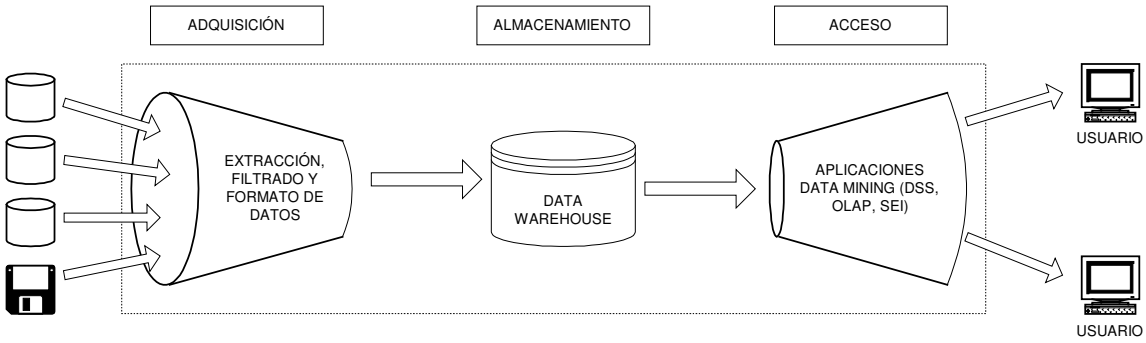


Figura 1.10

Los datos almacenados en un Data WareHouse se convierten en datos que pueden ser analizados y utilizados para realizar tomas de decisiones. Su acceso se realiza mediante “herramientas de consulta” como son los Sistemas Ejecutivos de Información (Executive Information System EIS), Sistemas de Soporte de Decisión (Decision Support System DDS) o Procesamiento Analítico en Línea (On Line Analitycal Proccessing OLAP), conocidos genéricamente como Data Mining[22].

[21] Tipo de sistemas que facilitan y administran aplicaciones transaccionales, usualmente para entrada de datos, recuperación y procesamiento de transacciones.
[22] Consiste en la extracción no trivial de información que reside de manera implícita en los datos. Dicha información era previamente desconocida y podrá resultar útil para algún proceso. En otras palabras, prepara, sondea y explora los datos para sacar la información oculta en ellos.

Las actividades involucradas en el análisis y diseño de un sistema Data WareHouse no son fáciles. El conjunto de actividades que involucra la creación y mantenimiento de un Data WareHouse es conocido como Data WareHousing. Crear un Data WareHouse requiere tiempo, dinero y una correcta planeación. Los problemas de planeación y costos son muy importantes para evitar un incremento no esperado en ambos. Por esta razón, en ocasiones se opta por realizar un análisis de ciertos bloques de datos del negocio, con los cuales se espera obtener la información básica necesaria para la toma de decisiones. Este subconjunto de datos permite conocer las necesidades de pequeños grupos de la empresa. Un Data Mart es un pequeño Data WareHouse creado con un subconjunto de información orientado a una materia, que provee soporte de decisión a un reducido grupo de personas.

Las diferencias entre un Data WareHouse y un Data Mart es sólo en el tamaño y ámbito de información a ser integrada, aunque definitivamente influyen las actividades relacionadas al costo y tiempo de implantación. El Data WareHousing en ambas incluyen actividades como la definición del problema y requerimiento de datos; y de igual manera cumplen las siguientes condiciones, que constituyen los lineamientos base de un Data WareHouse :

1. Los ambientes de producción y el Data WareHouse están separados.
2. Los datos de un Data WareHouse están integrados.

3. Un Data WareHouse contiene datos históricos sobre una medida larga de tiempo.
4. Los datos en un Data WareHouse son resúmenes de datos capturados a una fecha dada.
5. El Data WareHouse contiene datos orientados por materia.
6. Los datos de un Data WareHouse son primordialmente de sólo lectura con actualizaciones periódicas con datos OLTP. Las actualizaciones en línea no son permitidas.
7. El desarrollo de un Data WareHouse difiere a de los sistemas convencionales. El desarrollo del Data WareHouse es guiado por el flujo de los datos; los sistemas convencionales es dirigido por los procesos o funciones.
8. El Data WareHouse contiene datos con diversos niveles de detalle: detalle de datos actuales, detalle de datos antiguos, sin resumir o muy condensados.
9. El ambiente de un Data WareHouse es caracterizado por transacciones de sólo lectura sobre un vasto conjunto de datos. El ambiente de producción es caracterizado por numerosas transacciones update a pocas entidades de datos a un tiempo dado.
10. El ambiente de un Data WareHouse que integra datos fuentes transformándolos y agregándolos.

11. Los metadatos son los componentes críticos del Data WareHouse. Éstos identifican y definen a todos los elementos de datos. Proveen las fuentes, transformaciones, integración almacenamiento, uso, relaciones e historia de cada elemento.
12. El Data WareHouse posee un mecanismo de carga para utilizar los recursos que aseguran un uso óptimo de los datos por los usuarios.

Evolución del Data WareHouse

El Data WareHouse se inicia con los sistemas de reporte que alcanzaron su mayor auge en la década de 1980. Estos sistemas de reportes carecían de flexibilidad, al contar con consultas fijas y asociadas a un formato que ingresaban directamente a base de datos OLTP. Los reportes generados carecían de presentación, al contar únicamente de listados sumarizados.

La siguiente etapa permitió consultas más específicas que condensaban la información OLTP y la almacenaban en una base de datos relacional; los datos se consultaban mediante consultas SQL predefinidas, sin embargo su desventaja consistía en que el usuario tenía que conocer específicamente los datos y su localización para poder conformar sus reportes. En esta etapa, los reportes se complementan con hojas de cálculo y software estadístico que permitían analizar más a detalle el flujo y comportamiento de la información.

Los avances en hardware y software, y en especial la tecnología cliente/servidor a inicios de la década de 1990, propició que los sistemas de reporte se transformaran gradualmente en sistemas de soporte de decisión, los cuales utilizaban un repositorio central de datos conocido como Data WareHouse que condensaba los datos de producción.

La nueva generación de Sistemas de Soporte de Decisión[23] (DSS) permitió que los sistemas Data WareHouse fueran más eficientes, al utilizar diferentes modelos de bases de datos que complementaban los modelos relacionales, estos modelos incluían el concepto Multidimensional. Actualmente, las herramientas DSS que permiten un acceso a las bases de datos realizando consultas multidimensionales de datos, son conocidas como herramientas de Procesamiento Analítico en Línea (OLAP On Line Analytical Processing).

Componentes y Construcción

Las consideraciones previas a la planeación y construcción incluyen la definición y delimitación tareas, detallando las actividades y técnicas para construir cada uno de los tres componentes principales del Data WareHouse:

a) Adquisición

[23] Es un sistema informático utilizado para servir de apoyo, más que automatizar, el proceso de toma de decisiones.

b) Almacenamiento (Data WareHouse)

c) Componentes Front End

Adquisición de Datos

Este componente de adquisición de datos constituye el back end del Data WareHouse. Éste, ejecuta la extracción e importación de datos a las tablas. Está integrado por los siguientes procesos, los cuales deben ser analizados y diseñados:

1. Proceso de extracción y preparación de datos.
2. Esta fase incluye la identificación de los datos que se desean cargar al Data WareHouse. Investigar y establecer qué fuentes de datos alimentarán al sistema, los cuales pueden residir en equipos DB2, bases de datos UNIX, hojas de cálculo o archivos de texto.
3. Extracción de datos.
4. Limpieza de datos.
5. Procesamiento de unión.
6. Procesamiento de claves.
7. Procesamiento de purgas.
8. Almacenamiento

Tipos de Adquisición de Datos

Definir el método de extracción y carga de datos. Existen dos métodos principales para realizar esta actividad:

1. Carga por Volumen, en la cual se actualiza completamente la base de datos a un periodo dado.
2. Carga por Replicación basada en cambios, en donde se realiza una copia o réplica de datos que han cambiado entre los servidores.
3. Las ventajas de la carga por volumen es su fácil instalación y mantenimiento, aunque en ésta se aumenta el tráfico de red por la cantidad datos a transmitir. En la carga por replicación basada en cambios, el tráfico de red es menor, pero se requiere de programación y de un mantenimiento complejo de todos los procedimientos que inician en forma automática la actualización de las diferencias entre las bases de datos.

Almacenamiento

El Data WareHouse es una colección de tablas de una base de datos relacional que cumplen las siguientes características especiales:

1. Almacenan una gran cantidad de datos.
2. Los datos almacenados poseen una alta interdependencia.
3. Permiten consultas ad hoc (no predefinidas).

4. Los datos almacenados son de sólo lectura para el usuario.
5. Actualización periódica de datos de diversas fuentes.
6. Los datos almacenados poseen asociado una variante de tiempo y son históricos.

Los aspectos más importantes a considerar en el almacenamiento de los datos, basándose en las características anteriores son la definición del tipo de hardware donde residirá el Data WareHouse. Éste, deberá permitir la realización de consultas ad hoc sin que se disminuya el desempeño del mismo. La creación de un Meta Datos que mantega la información de las tablas, datos y sus relaciones, con lo cual se facilitará la administración y acceso a los mismos.

Arquitecturas del Data WareHouse

Un sistema Data WareHouse se puede clasificar en arquitecturas de acuerdo al número de componentes físicos que lo integran. Cada arquitectura puede contener los siguientes componentes: “modelo de datos, servidor warehouse, servidor de aplicaciones, capa middleware, utilerías de extracción de datos, utilerías de transporte de datos, programas de replicación, repositorios de meta datos y herramientas de análisis de datos” . Todos estos componentes o capas están unidos en forma coherente por una infraestructura de red, con lo cual se permite alimentar y consultar el Data WareHouse.

La arquitectura básica de un Data WareHouse involucra dos capas homogéneas, (como se muestra en la figura 1.11) en la cual la primer capa une al servidor de datos de producción con la base de datos central Data WareHouse; y una segunda capa que enlaza la aplicación cliente con el servidor Data WareHouse por medio de un front–end DSS y herramientas de análisis

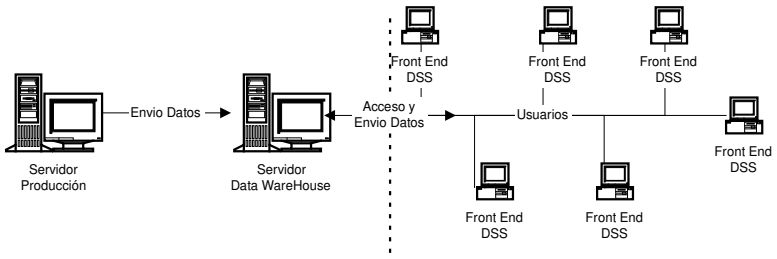


Figura 1.11

Una arquitectura más compleja integra tres capas, (como se muestra en la figura 1.12) en la cual una es usada para el acceso y traslado de datos. La primera capa, corresponde al enlace servidor datos producción con un servidor de datos Data WareHouse. La segunda, corresponde a la herramienta multidimensional OLAP o middleware. Y la tercera capa, corresponde al cliente que maneja soporte de decisión y presentación gráfica de datos.

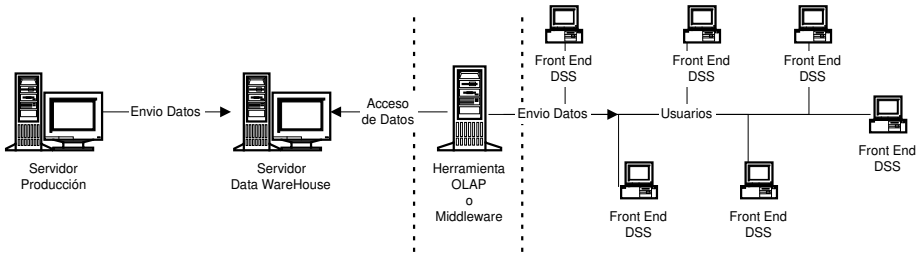

Servidor
Producción
Envio Datos
Servidor
Data WareHouse
Acceso
de Datos
Herramienta
OLAP
o
Middleware
Front End
DSS
Envio Datos
Front End
DSS
Front End
DSS
Usuarios
Front End
DSS
Front End
DSS
Front End
DSS
Front End
DSS

Figura 1.12

CAPÍTULO II

Objetivo

El objetivo principal del proyecto es obtener las métricas de las fallas presentadas a los usuarios, para clasificar e identificar las recurrencias de mayor impacto y exponer esta situación a los responsables para encaminar la mejora de sus áreas de oportunidad.

Para lograr este objetivo principal, el proyecto se divide en el cumplimiento de los siguientes objetivos:

- Analizar las funcionalidades que a cubrir por el sistema del reporte de fallas y sus relaciones para modelarlo bajo un sistema de base de datos.
- Diseñar y modelar una base de datos relacional para administrar el registro de los diferentes reportes de fallas presentadas en hardware, software y periféricos presentados en la operativa diaria de usuarios finales.
- Construir el modelo de base de datos diseñado usando lenguaje de definición de datos permitiendo su portabilidad entre manejadores de base de datos que utilicen SQL.

- Construir bajo lenguaje de manipulación de datos los procesos de registro, modificación y consulta de información, manteniendo como premisa el mantenimiento e integridad de datos.
- Construir los reportes de métricas de los reportes de fallos registrados en la base de datos para su análisis y toma de decisiones.

Alcance del Proyecto

Con base a los conocimientos adquiridos en este seminario, el alcance del proyecto se basa en el análisis, diseño, modelado y construcción de la base de datos para un reporte de fallas. Este proyecto considera las siguientes características de funcionalidad dentro de la base de datos:

- Diseño multi-empresa para ofrecer flexibilidad en la clasificación de los reportes para las diversas áreas de negocio
- La clasificación de los reportes con base al área responsable de atención por medio de buzones
- Clasificar el nivel de severidad e impacto de los reportes de fallas para dirigir las alertas necesarias a los responsables de atención, durante el ciclo de vida del reporte (Abierto, Recuperado, Cerrado)
- Generar reportes resumen de las incidencias registradas por área usuaria, responsable de atención, tipo de causa origen y tiempo total de solución.
- Generar reportes a detalle para revisión de las áreas responsables.

Planeación del Proyecto

Para administrar, controlar e identificar desviaciones en el ciclo de vida del desarrollo del sistema, es condición necesaria la planeación de las actividades, para verificar el desarrollo de las acciones que permitirán completar con éxito el objetivo de nuestro sistema. De esta forma el proyecto se puede definir como el conjunto de actividades interrelacionadas y coordinadas, acotadas a los limites de presupuesto y lapso de tiempo para concluirlas.

A forma ilustrativa, el desglose de actividades que integran el proyecto, se marcan dentro de la figura 2.1.

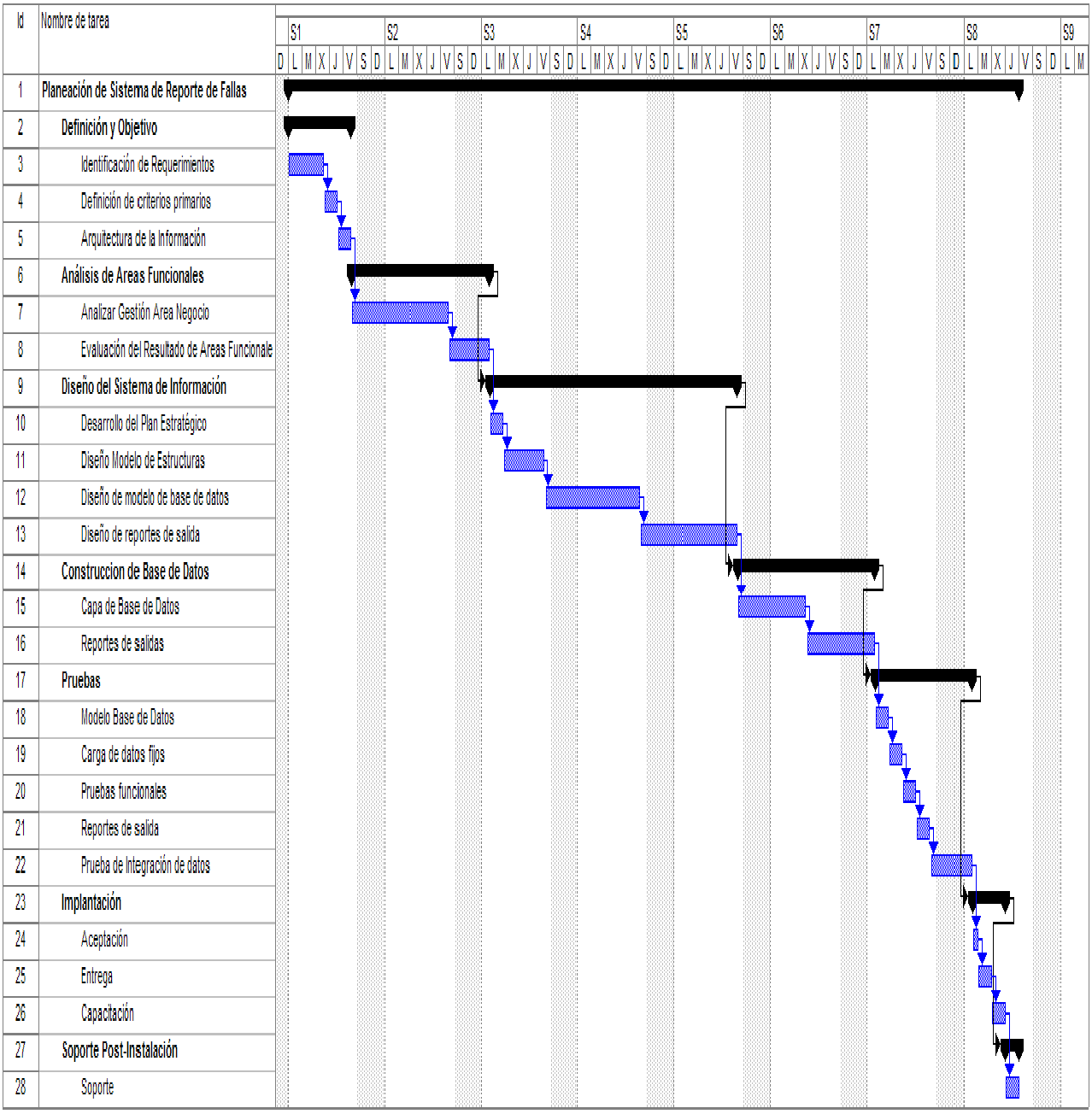

Id
Nombre de tarea
S1
S2
S3
S4
S5
S6
S7
S8
S9
D L M X J V S D L M X J V S D L M X J V S D L M X J V S D L M X J V S D L M X J V S D L M X J V S D L M X J V S D L M X J V S D L M
1 Planeación de Sistema de Reporte de Fallas
2 Definición y Objetivo
3 Identificación de Requerimientos
4 Definición de criterios primarios
5 Arquitectura de la Información
6 Análisis de Areas Funcionales
7 Analizar Gestión Area Negocio
8 Evaluación del Resultado de Areas Funcionale
9 Diseño del Sistema de Información
10 Desarrollo del Plan Estratégico
11 Diseño Modelo de Estructuras
12 Diseño de modelo de base de datos
13 Diseño de reportes de salida
14 Construccion de Base de Datos
15 Capa de Base de Datos
16 Reportes de salidas
17 Pruebas
18 Modelo Base de Datos
19 Carga de datos fijos
20 Pruebas funcionales
21 Reportes de salida
22 Prueba de Integración de datos
23 Implantación
24 Aceptación
25 Entrega
26 Capacitación
27 Soporte Post-Instalación
28 Soporte

Figura 2.1

Donde las actividades se desarrollan conforme su fase de desarrollo.

- Definición y Objetivo

La definición del proyecto, objetivo y alcance es la base que permitirán alinear nuestras actividades a la obtención del desarrollo del proyecto.

En esta fase del proyecto se definieron los objetivos para ser alcanzados bajo la planeación de actividades.

- **Análisis**

En esta fase se completó el estudio de los componentes que conformarán el modelo del sistema de reporte de fallas, se identificaron las funciones las cuales se modelaron a través de un Diagrama de Estructuras.

Los modelos utilizados en este proyecto, nos permitieron plasmar las necesidades y relacionar las funciones que integran el sistema de reporte de fallas; la interpretación de esta necesidad se clarifica en el modelo de base de datos relacional del sistema.

- **Diseño**

El diseño se completó al completar el análisis de los requerimientos que debe cubrir el sistema, se elaboró el modelo entidad-relación del sistema de reporte de fallas que permitió contar con el detalle conceptual de la solución final, es decir, el

modelo fue la especificación para comenzar con el desarrollo (programación) de la base de datos.

- **Desarrollo**

En este fase se desarrollaron los programas para dar cumplimiento a nuestro objetivo, pasando del plano de los modelos conceptuales a una solución informática, cimentada en una modular y compatible capa de persistencia.

El desarrollo involucró la construcción de los procesos de creación de base de datos (entidades, relaciones y reglas de integridad), así como los programas de consulta de datos en procedimientos almacenados.

- **Pruebas**

Para validar que el modelo cubre las necesidades funcionales del sistema, se realizó la carga de datos muestra en la base de datos, y se completaron actividades de alta, modificaciones y cambios en todos los módulos que integran el sistema, para confirmar su funcionalidad. De igual forma, se lanzaron la ejecución de los procedimientos almacenados de extracción de información.

En esta fase, se realizó la corrección de fallos en programación y consulta, de tal forma que el sistema final está depurado y sin fallas.

- **Implantación**

La fase de instalación consiste en la entrega de los códigos fuentes para su ejecución en el servidor DBMS.

- **Soporte post-instalación**

El soporte post.-instalación dentro de este proyecto se considera como las actividades de soporte aplicadas durante la instalación del script de instalación de la base de datos.

CAPÍTULO III

Modelo de la Organización

En la figura 3.1 podemos ver el diagrama del modelo de la organización relacionado con el sistema de este proyecto.

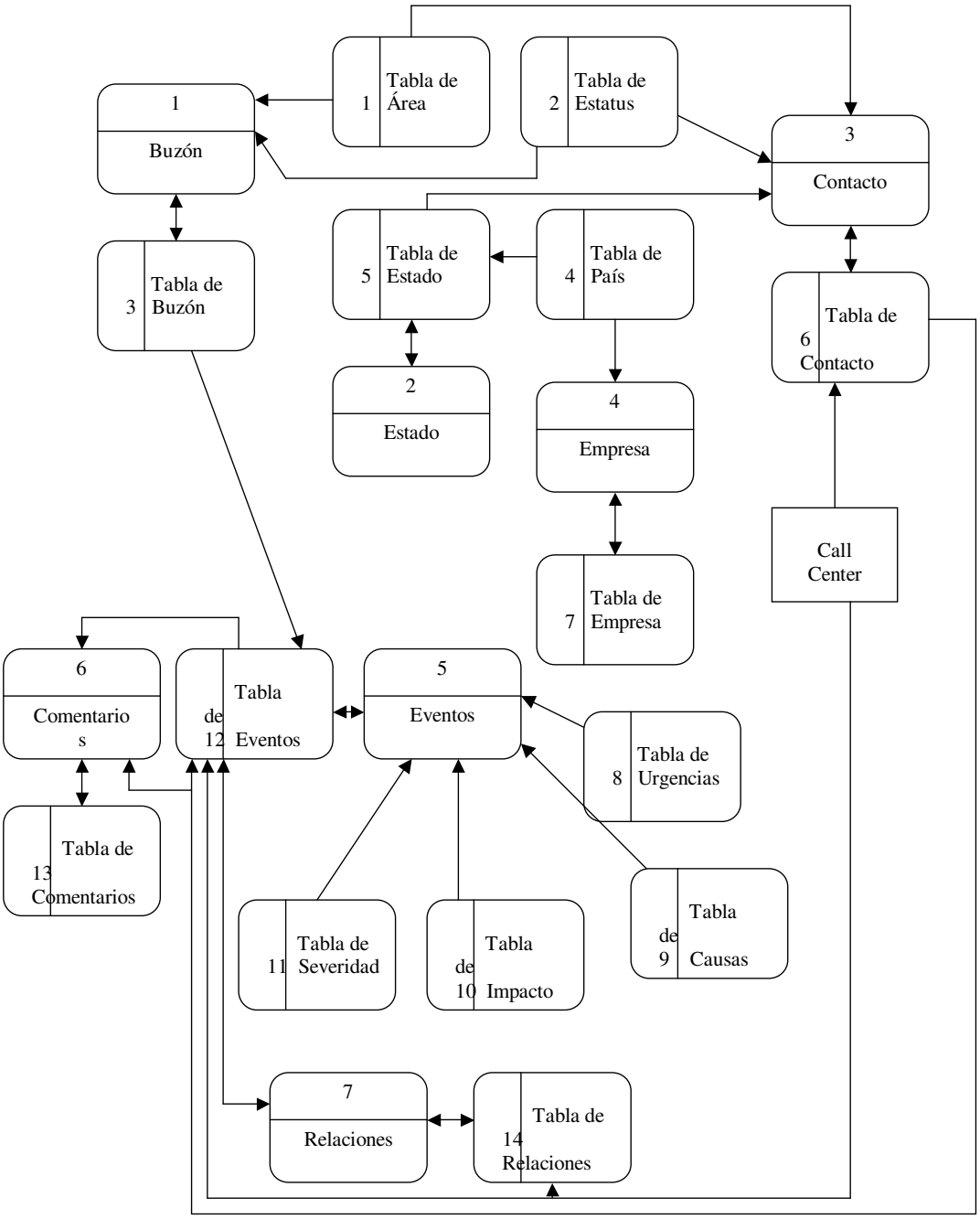


Figura 3.1 Modelo de la Organización

Diagrama Entidad-Relación

A partir del modelo de la organización, podemos obtener el diagrama de entidad relación y los campos que contendrá cada una de ellas (Figura 3.2).

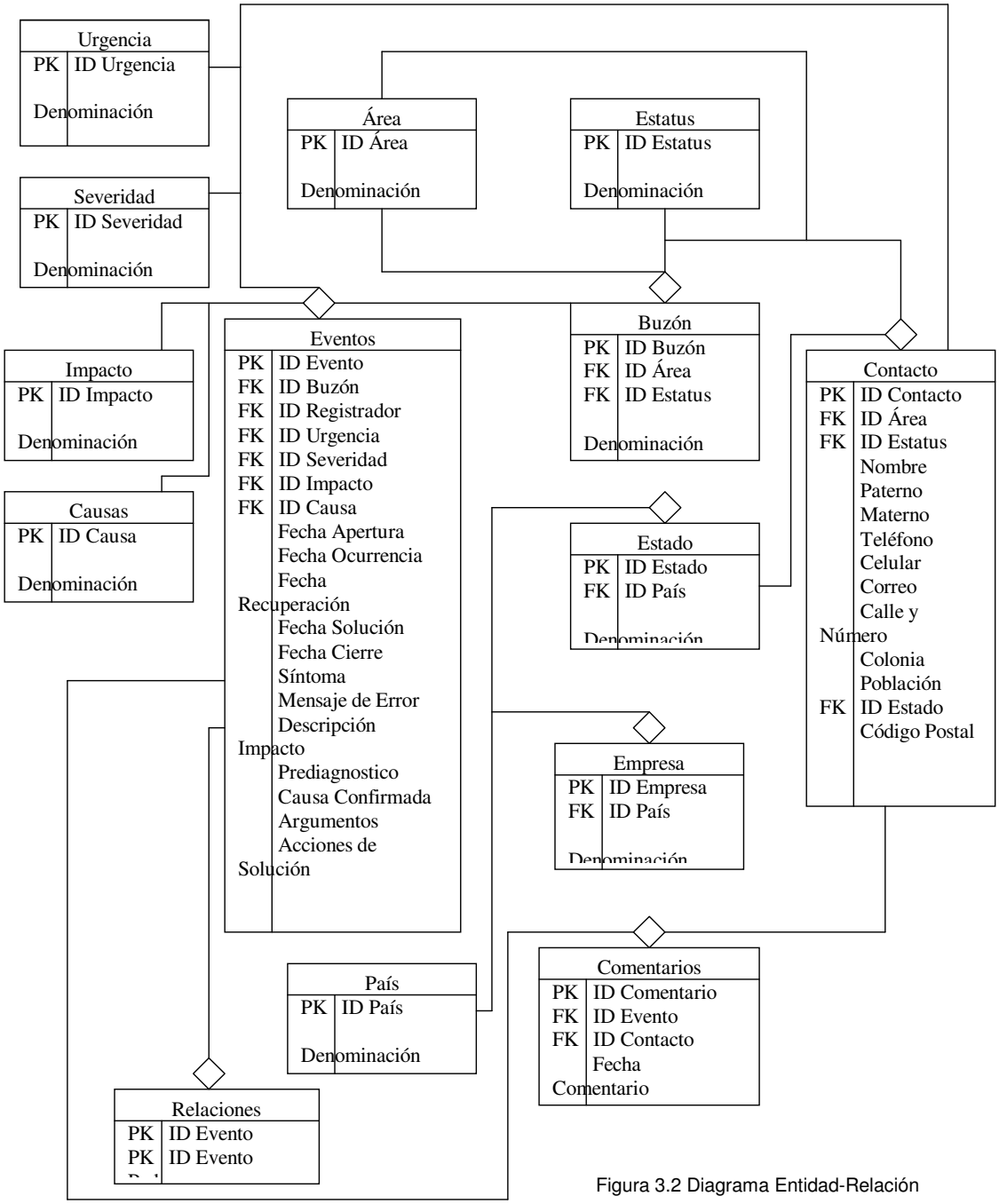


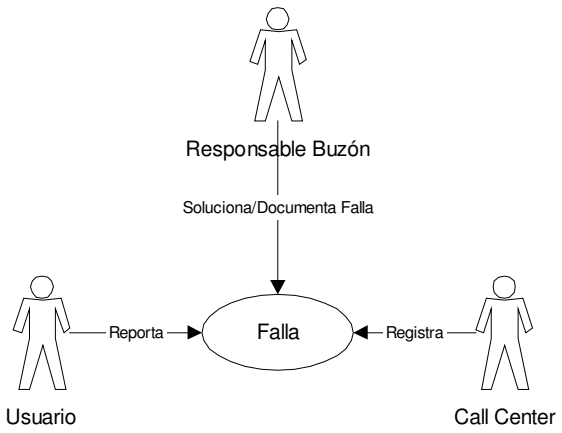
Figura 3.2 Diagrama Entidad-Relación

Casos de uso

Para comprender las acciones y reacciones del flujo operativo del sistema de reportes de fallas, se utilizaron Diagramas de Uso para mostrar su comportamiento.

Este modelo fue seleccionado por considerarse accesible de comprender a nivel de usuarios y permite graficar la determinación de requisitos a ser cubiertos por la base de datos a modelar.

A continuación se muestran los diagramas de uso generados en las fases de análisis y diseño (figura 3.3).



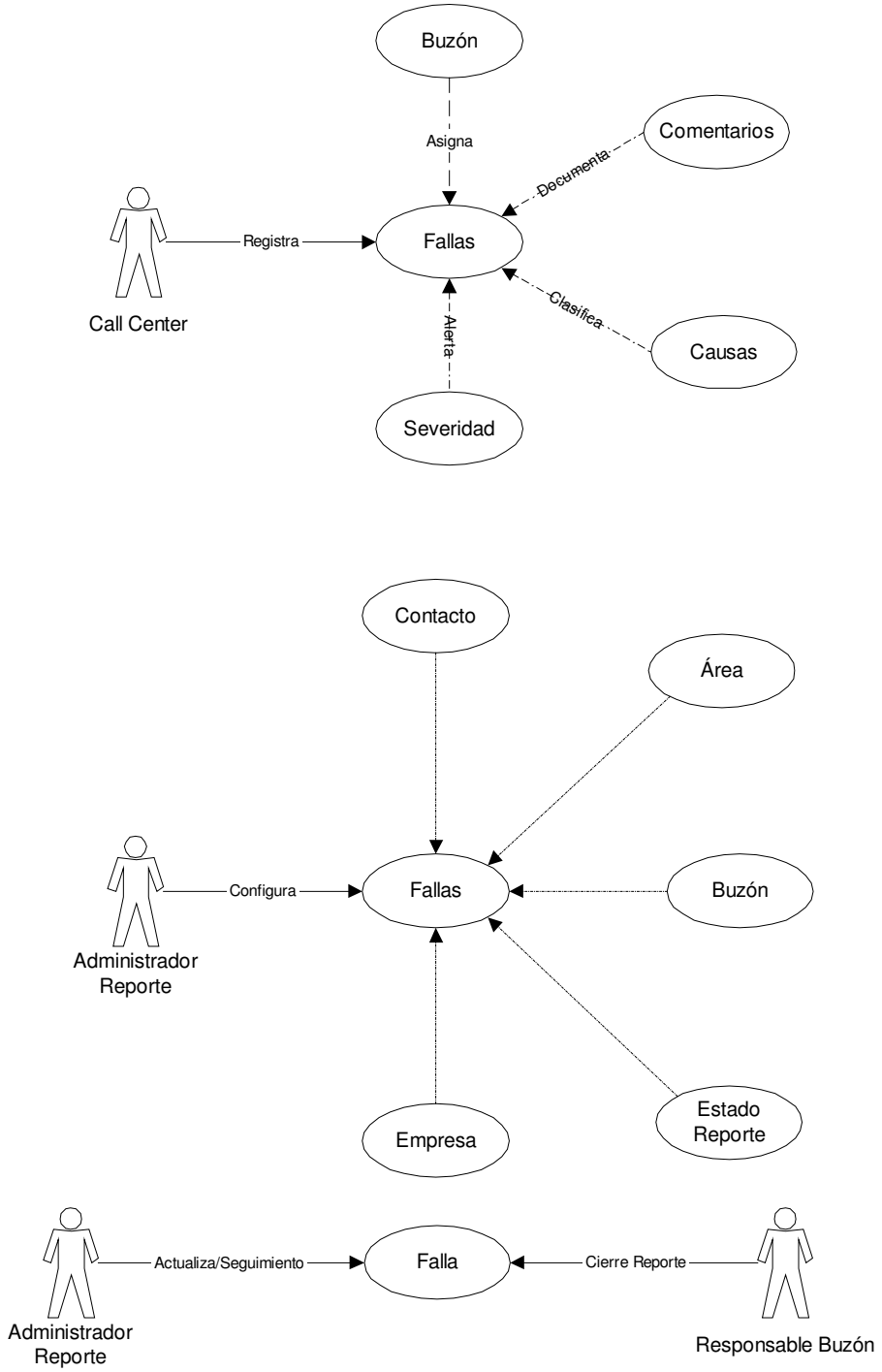


Figura 3.3 Diagrama de Casos de Uso

Los diagramas de caso de uso, las inclusiones señalan que el caso de uso origen incluye también el comportamiento descrito por el caso de uso destino.

Por su parte las extensiones, el caso de uso origen extienden su comportamiento al caso de uso destino.

Para interpretación de las acciones y relaciones a cubrir en el caso de uso Evento, se expone la siguiente tabla:

CÓDIGO- 001	REGISTRO DE FALLA	
Versión y fecha Autor	1.1 Al 16-02-09 Alfredo Torres Beltrán	
Objetivo	Registrar los eventos recibidos por teléfono o correo electrónico, en la base de datos del sistema de reporte de fallas. Identificar y confirmar el nivel de urgencia e impacto, para reportar al responsable de su recuperación solución	
Precondición	Contacto registrado en base de datos Buzón registrado en base de datos Las fechas a registrar durante el ciclo de vida del reporte de falla, serán tomados de la fecha del servidor y no modificables por ningún usuario.	
Secuencia	Paso	Acción
	1	El Call Center recibe reporte de falla, se aplica el caso de uso Eventos
	2	Si no existe Contacto (usuario que reporta) se realiza el caso de uso Alta Contacto

	3	Si no existe Buzón (responsable de solución) se realiza el caso de uso Alta Buzón
Post-condición	Call Center avisará telefónicamente y de forma inmediata a responsable de solución de la incidencia, si impacto es ALTO y severidad 1 Call Center informará telefónicamente a responsable de solución de incidencia Reporte de Falla mantendrá el registro de la incidencia con estado ABIERTO	
Comentarios	El estado inicial de los reportes de falla serán ABIERTO, y solo podrá pasar a estado CERRADO por el responsable de su solución.	

Diccionario de Datos

A partir del diagrama de Entidad-Relación, podemos obtener el diccionario de datos (Figura 3.4)

Nombre	Alias	Tipo	Tamaño	No Nulo	Requerido	Entidad / Relación
Identificador de Área	Id_area	Smallint	2	X	X	Área,Contacto
Descripción del Identificador	Denominacion	Char	40	X	X	Todas
Identificador de Estatus	Id_estatus	Smallint	2	X	X	Estatus,Buzón,Contacto
Identificador de Severidad	Id_severidad	Smallint	2	X	X	Severidad,Eventos
Identificador de Urgencia	Id_urgencia	Smallint	2	X	X	Urgencia,Eventos
Identificador de Impacto	Id_impacto	Smallint	2	X	X	Impacto,Eventos
Identificador de País	Id_pais	Smallint	2	X	X	País,Empresa,Estado
Identificador de Causa	Id_causa	Smallint	2	X	X	Causas,Eventos
Identificador de Buzón	Id_buzon	Smallint	2	X	X	Buzón,Eventos
Identificador de Estado	Id_estado	Smallint	2	X	X	Estado,Contacto

Nombre	Alias	Tipo	Tamaño	No Nulo	Requerido	Entidad / Relación
Identificador de Contacto	Id_contacto	Smallint	2	X	X	Contacto,Comentarios
Nombre del Usuario	Nombre	Char	40	X	X	Contacto
Apellido Paterno del Usuario	Paterno	Char	40	X	X	Contacto
Apellido Materno del Usuario	Materno	Char	40	X	X	Contacto
Teléfono Fijo del Usuario	Telefono	Char	16			Contacto
Teléfono Celular del Usuario	Celular	Char	16			Contacto
Dirección de Correo Electrónico	Correo	Char	25			Contacto
Calle y Número	Calle_numero	Char	40			Contacto
Colonia	Colonia	Char	40			Contacto
Población	Población	Char	40			Contacto
Código Postal	Cp	Char	5			Contacto
Identificador de la Empresa	Id_empresa	Smallint	2	X	X	Empresa
Identificador	Id_evento	Smallint	2	X	X	Eventos,Relacion

Nombre	Alias	Tipo	Tamaño	No Nulo	Requerido	Entidad / Relación
del Evento						es,Comentarios
Fecha de Apertura de Reporte	Fh_apertura	Datetime	8	X	X	Eventos
Fecha del Suceso	Fh_ocurrencia	Datetime	8			Eventos
Fecha de Recuperación	Fh_recuperación	Datetime	8			Eventos
Fecha de Solución	Fh_solucion	Datetime	8			Eventos
Fecha de Cierre de Reporte	Fh_cierre	Datetime	8			Eventos
Descripción del Síntoma	Síntoma	Char	250			Eventos
Mensaje de Error	Mensaje_error	Char	250			Eventos
Descripción del Impacto	Descripción_impacto	Char	250			Eventos
Descripción del Prediagnóstico	Prediagnostico	Char	250			Eventos
Descripción de la Causa del Problema	Causa_confirmada	Char	250			Eventos
Descripción de los	Argumentos	Char	250			Eventos

Nombre	Alias	Tipo	Tamaño	No Nulo	Requerido	Entidad / Relación
Argumentos						
Acciones que se llevaron a cabo para la solución del problema	Acciones_solucion	Char	250			Eventos
Identificador de Comentarios	Id_comentario					Comentarios
Fecha del Comentario	Fh_comentario	Datatime	8			Comentarios

Figura 3.4 Diccionario de Datos

Procedimientos para Entidad-Relación

A partir de los datos obtenidos anteriormente, podemos obtener los procedimientos que serán necesarios para este sistema, los cuales aparecen en la figura 3.5

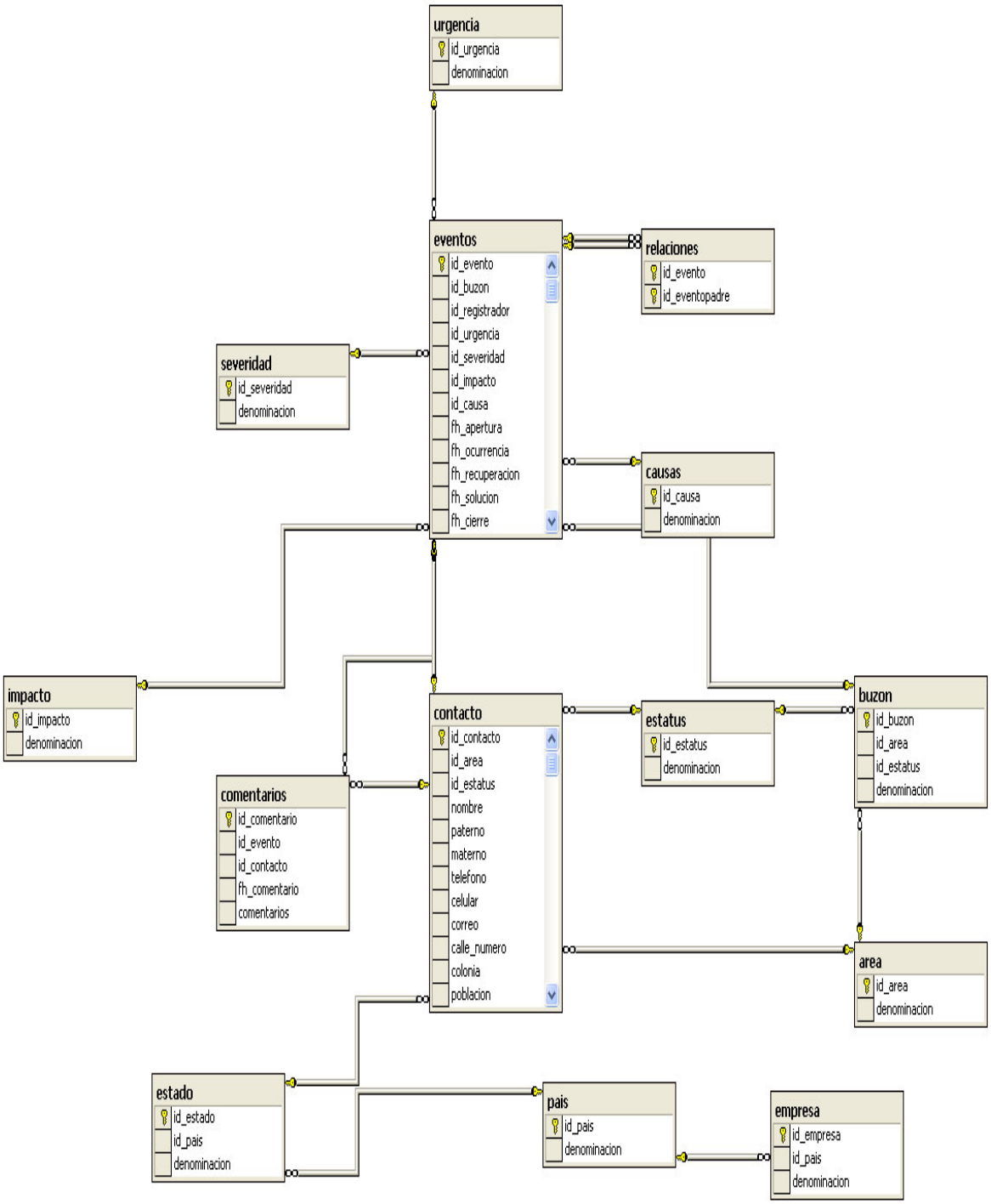
Nombre de la Tabla	Tipo(Entidad / Relación)	Procedimientos Desarrollados
Area	Entidad	Inserta Área Actualiza Área Elimina Área
Estatus	Entidad	Inserta Estatus Actualiza Estatus Elimina Estatus
Severidad	Entidad	Inserta Severidad Actualiza Severidad Elimina Severidad
Urgencia	Entidad	Inserta Urgencia Actualiza Urgencia Elimina Urgencia
Impacto	Entidad	Inserta Impacto Actualiza Impacto Elimina Impacto

Nombre de la Tabla	Tipo(Entidad / Relación)	Procedimientos Desarrollados
Pais	Entidad	Inserta País Actualiza País Elimina País
Causas	Entidad	Inserta Causas Actualiza Causas Elimina Causas
Buzon	Relación	Inserta Buzón Actualiza Buzón
Estado	Relación	Inserta Estado Actualiza Estado Elimina Estado
Contacto	Relación	Inserta Contacto Actualiza Contacto Elimina Contacto
Empresa	Relación	Inserta Empresa Actualiza Empresa Elimina Empresa
Eventos	Relación	Inserta Evento Modifica Evento
Relaciones	Relación	Inserta Relación Modifica Relación Elimina Relación

Nombre de la Tabla	Tipo(Entidad / Relación)	Procedimientos Desarrollados
Comentarios	Relación	Inserta Comentarios

Figura 3.5 Procedimientos para entidad relación

Diagrama de la Base de Datos



Construcción de Base de Datos

A continuación veremos algunas de las instrucciones que tendrá la Base de Datos (generación de la propia Base de Datos y Procedimientos Almacenados), todos estos están realizados con Microsoft SQL Server para que posteriormente puedan ser llamadas por el programa que será la interfaz con el usuario final.

```
/* Creación de Base de Datos */
use master
go
create database GestionFallas on primary
(
  name = GestionFallas,
    filename = 'c:\GestionFallas.mdf',
    size = 10,
    maxsize = 100,
    filegrowth = 10%
)
go
use GestionFallas
go
```

```
/* Creación de Entidades y Relaciones */
drop table comentarios
go
drop table relaciones
go
drop table eventos
go
drop table empresa
go

create table area
(
 id_area smallint not null primary key identity,
 denominacion nvarchar(40) not null
)
go
create table estatus
(
 id_estatus smallint not null primary key identity,
 denominacion nvarchar(40) not null
)
go
```

```
create table severidad
(
 id_severidad smallint not null primary key identity,
 denominacion nvarchar(40) not null
)
go
create table urgencia
(
 id_urgencia smallint not null primary key identity,
 denominacion nvarchar(40) not null
)
go
create table impacto
(
 id_impacto smallint not null primary key identity,
 denominacion nvarchar(40) not null
)
go
```

```
/* Creación de llaves y relaciones */
create table estado
(
 id_estado smallint not null primary key identity,
 id_pais smallint not null foreign key references pais(id_pais),
 denominacion nvarchar(40) not null
)
go
create table contacto
(
 id_contacto smallint not null primary key identity,
 id_area smallint not null foreign key references area(id_area),
 id_estatus smallint not null foreign key references estatus(id_estatus),
 nombre nvarchar(40) not null,
 paterno nvarchar(40) not null,
 materno nvarchar(40) not null,
 telefono nvarchar(16),
 celular nvarchar(16),
 correo nvarchar(25),
 calle_numero nvarchar(40),
 colonia nvarchar(40),
 poblacion nvarchar(40),
```

```
 id_estado smallint not null foreign key references estado(id_estado),
 cp nvarchar(5)
)
go
create table empresa
(
 id_empresa smallint not null primary key identity,
 id_pais smallint not null foreign key references pais(id_pais),
 denominacion nvarchar(40) not null
)
go
create table eventos
(
 id_evento smallint not null primary key identity,
 id_buzon smallint not null foreign key references buzon(id_buzon),
 id_registrador smallint not null foreign key references contacto(id_contacto),
 id_urgencia smallint not null foreign key references urgencia(id_urgencia),
 id_severidad smallint not null foreign key references severidad(id_severidad),
 id_impacto smallint not null foreign key references impacto(id_impacto),
 id_causa smallint not null foreign key references causas(id_causa),
 fh_apertura datetime not null,
 fh_ocurrencia datetime,
```

```
 fh_recuperacion datetime,
 fh_solucion datetime,
fh_cierre datetime,
 sintoma nvarchar(250),
 mensaje_error nvarchar(250),
 descripcion_impacto nvarchar(250),
 prediagnostico nvarchar(250),
 causa_confirmada nvarchar(250),
 argumentos nvarchar(250),
 acciones_solucion nvarchar(250)
)
go

/* Procedimientos Almacenados */
----- Procedimiento Área -------
drop procedure sp_inserta_area
go
create procedure sp_inserta_area
@denominacion nvarchar(40)
as
 declare @mensaje nvarchar(50)
 declare @id_area smallint
```

```
 if exists (select * from area where denominacion = @denominacion)
   begin
     set @mensaje = 'El Área ya Existe'
     print @mensaje
   end
 else
  begin
    insert into area values (@denominacion)
    set @mensaje = 'Área Insertada Exitosamente, con la clave:'
    print @mensaje
    select @id_area = id_area from area where denominacion = @denominacion
    print @id_area
  end


drop procedure sp_actualiza_area
go
create procedure sp_actualiza_area
@id_area smallint,
@denominacion nvarchar(40)
as
 declare @mensaje nvarchar(50)
 if not exists (select * from area where id_area = @id_area)
```

```
  begin
    set @mensaje = 'El Área no Existe'
    print @mensaje
  end
 else
  if exists (select * from buzon where id_area = @id_area)
    begin
      set @mensaje = 'No se puede actualizar el área, porque ya ha sido utilizada'
      print @mensaje
    end
  else
    if exists (select * from contacto where id_area = @id_area)
      begin
        set @mensaje = 'No se puede actualizar el área, porque ya ha sido utilizada'
        print @mensaje
      end
    else
     begin
       update area set denominacion = @denominacion from area
              where id_area = @id_area
       set @mensaje = 'Área Actualizada Exitosamente'
       print @mensaje
```

```
      end
drop procedure sp_elimina_area
go



create procedure sp_elimina_area
@id_area smallint
as
 declare @mensaje nvarchar(50)
 if not exists (select * from area where id_area = @id_area)
   begin
     set @mensaje = 'El Área no Existe'
     print @mensaje
   end
 else
   if exists (select * from buzon where id_area = @id_area)
     begin
       set @mensaje = 'No se puede eliminar el area, porque ya ha sido utilizada'
       print @mensaje
     end
   else
     if exists (select * from contacto where id_area = @id_area)
       bejín
```

```
        set @mensaje = 'No se puede eliminar el área, porque ya ha sido utilizada'
        print @mensaje
       end
    else
     begin
       delete area from area where id_area = @id_area
       set @mensaje = 'Área Eliminada Exitosamente'
       print @mensaje
     end
exec sp_elimina_area 1

----- Procedimiento Estatus -------
drop procedure sp_inserta_estatus
go
create procedure sp_inserta_estatus
@denominacion nvarchar(40)
as
 declare @mensaje nvarchar(50)
 declare @id_estatus smallint
 if exists (select * from estatus where denominacion = @denominacion)
   begin
     set @mensaje = 'El Estatus ya Existe'
```

```
     print @mensaje
   end
 else
  begin
    insert into estatus values (@denominacion)
    set @mensaje = 'Estatus Insertado Exitosamente, con la clave:'
    print @mensaje
    select @id_estatus = id_estatus from estatus where denominacion =
@denominacion
    print @id_estatus
  end


drop procedure sp_actualiza_estatus
go
create procedure sp_actualiza_estatus
@id_estatus smallint,
@denominacion nvarchar(40)
as
 declare @mensaje nvarchar(50)
 if not exists (select * from estatus where id_estatus = @id_estatus)
   begin
     set @mensaje = 'El Estatus no Existe'
```

```
    print @mensaje
  end
 else
  if exists (select * from buzon where id_estatus = @id_estatus)
    begin
      set @mensaje = 'No se puede actualizar el estatus, porque ya ha sido
utilizado'
      print @mensaje
    end
  else
    if exists (select * from contacto where id_estatus = @id_estatus)
      begin
        set @mensaje = 'No se puede actualizar el estatus, porque ya ha sido
utilizado'
        print @mensaje
      end
    else
     begin
       update estatus set denominacion = @denominacion from estatus
            where id_estatus = @id_estatus
       set @mensaje = 'Estatus Actualizado Exitosamente'
       print @mensaje
```

```
      end
exec sp_actualiza_estatus 1,'Inactivo'


drop procedure sp_elimina_estatus
go
create procedure sp_elimina_estatus
@id_estatus smallint
as
declare @mensaje nvarchar(50)
 if not exists (select * from estatus where id_estatus = @id_estatus)
   begin
     set @mensaje = 'El Estatus no Existe'
     print @mensaje
   end
 else
   if exists (select * from buzon where id_estatus = @id_estatus)
     begin
       set @mensaje = 'No se puede eliminar el estatus, porque ya ha sido utilizado'
       print @mensaje
     end
   else
     if exists (select * from contacto where id_estatus = @id_estatus)
```

```
      begin
        set @mensaje = 'No se puede eliminar el estatus, porque ya ha sido
utilizado'
        print @mensaje
      end
    else
     begin
       delete estatus from estatus where id_estatus = @id_estatus
       set @mensaje = 'Estatus Eliminado Exitosamente'
       print @mensaje
     end


----- Procedimiento Severidad -------
drop procedure sp_inserta_severidad
go
create procedure sp_inserta_severidad
@denominacion nvarchar(40)
as
 declare @mensaje nvarchar(50)
 declare @id_severidad smallint
 if exists (select * from severidad where denominacion = @denominacion)
   bejín
```

```
     set @mensaje = 'La Severidad ya Existe'
     print @mensaje
   end
 else
  begin
    insert into severidad values (@denominacion)
    set @mensaje = 'Severidad Insertada Exitosamente, con la clave:'
    print @mensaje
    select @id_severidad = id_severidad from severidad where denominacion =
@denominacion
    print @id_severidad
  end



drop procedure sp_actualiza_severidad
go
create procedure sp_actualiza_severidad
@id_severidad smallint,
@denominacion nvarchar(40)
as
 declare @mensaje nvarchar(50)
 if not exists (select * from severidad where id_severidad = @id_severidad)
```

```
  begin
    set @mensaje = 'La Severidad no Existe'
    print @mensaje
  end
 else
  if exists (select * from eventos where id_severidad = @id_severidad)
    begin
      set @mensaje = 'No se puede actualizar la severidad, porque ya ha sido
utilizada'
      print @mensaje
    end

  else
   begin
     update severidad set denominacion = @denominacion from severidad
           where id_severidad = @id_severidad
     set @mensaje = 'Severidad Actualizada Exitosamente'
     print @mensaje
   end

drop procedure sp_elimina_severidad
go
```

```
create procedure sp_elimina_severidad
@id_severidad smallint
as
 declare @mensaje nvarchar(50)
 if not exists (select * from severidad where id_severidad = @id_severidad)
   begin
     set @mensaje = 'La Severidad no Existe'
     print @mensaje
   end
 else
   if exists (select * from eventos where id_severidad = @id_severidad)
     begin

       set @mensaje = 'No se puede eliminar la severidad, porque ya ha sido
utilizada'
       print @mensaje
     end
   else
    begin
      delete severidad from severidad where id_severidad = @id_severidad
      set @mensaje = 'Severidad Eliminada Exitosamente'
      print @mensaje
    end
```

```
----- Procedimiento Urgencia -------
drop procedure sp_inserta_urgencia
go
create procedure sp_inserta_urgencia
@denominacion nvarchar(40)
as
 declare @mensaje nvarchar(50)
 declare @id_urgencia smallint
 if exists (select * from urgencia where denominacion = @denominacion)
   begin
     set @mensaje = 'La Urgencia ya Existe'
     print @mensaje
   end
 else
  begin
    insert into urgencia values (@denominacion)
    set @mensaje = 'Urgencia Insertada Exitosamente, con la clave:'
    print @mensaje
    select @id_urgencia = id_urgencia from urgencia where denominacion =
@denominacion
    print @id_urgencia
  end
```

```
drop procedure sp_actualiza_urgencia
go
create procedure sp_actualiza_urgencia
@id_urgencia smallint,
@denominacion nvarchar(40)
as
 declare @mensaje nvarchar(50)
 if not exists (select * from urgencia where id_urgencia = @id_urgencia)
   begin
     set @mensaje = 'La Urgencia no Existe'
     print @mensaje
   end
 else
   if exists (select * from eventos where id_urgencia = @id_urgencia)
     begin
       set @mensaje = 'No se puede actualizar la urgencia, porque ya ha sido
utilizada'
       print @mensaje
     end
   else
    begin
      update urgencia set denominacion = @denominacion from urgencia
```

```
            where id_urgencia = @id_urgencia
      set @mensaje = 'Urgencia Actualizada Exitosamente'
      print @mensaje
    end


drop procedure sp_elimina_urgencia
go
create procedure sp_elimina_urgencia
@id_urgencia smallint
as
 declare @mensaje nvarchar(50)
 if not exists (select * from urgencia where id_urgencia = @id_urgencia)
   begin
     set @mensaje = 'La Urgencia no Existe'
     print @mensaje
   end
 else
   if exists (select * from eventos where id_urgencia = @id_urgencia)
     begin
       set @mensaje = 'No se puede eliminar la urgencia, porque ya ha sido
utilizada'
       print @mensaje
```

```
    end
  else
   begin
     delete urgencia from urgencia where id_urgencia = @id_urgencia
     set @mensaje = 'Urgencia Eliminada Exitosamente'
     print @mensaje
   end

----- Procedimiento Impacto -------
drop procedure sp_inserta_impacto
go
create procedure sp_inserta_impacto
@denominacion nvarchar(40)
as
 declare @mensaje nvarchar(50)
 declare @id_impacto smallint
 if exists (select * from impacto where denominacion = @denominacion)
  begin
    set @mensaje = 'El impacto ya Existe'
    print @mensaje
  end
 else
```

```
 begin
   insert into impacto values (@denominacion)
   set @mensaje = 'Impacto Insertado Exitosamente, con la clave:'
   print @mensaje
   select @id_impacto = id_impacto from impacto where denominacion =
@denominacion
   print @id_impacto
 end

drop procedure sp_actualiza_impacto
go
create procedure sp_actualiza_impacto
@id_impacto smallint,
@denominacion nvarchar(40)
as
 declare @mensaje nvarchar(50)
 if not exists (select * from impacto where id_impacto = @id_impacto)
  begin
    set @mensaje = 'El impacto no Existe'
    print @mensaje
  end
 else
```

```
  if exists (select * from eventos where id_impacto = @id_impacto)
    begin
     set @mensaje = 'No se puede actualizar el impacto, porque ya ha sido
utilizado'
      print @mensaje
    end
  else
   begin
     update impacto set denominacion = @denominacion from impacto
          where id_impacto = @id_impacto
     set @mensaje = 'Impacto Actualizado Exitosamente'
     print @mensaje
   end


drop procedure sp_elimina_impacto
go
create procedure sp_elimina_impacto
@id_impacto smallint
as
 declare @mensaje nvarchar(50)
 if not exists (select * from impacto where id_impacto = @id_impacto)
   bejín
```

```
    set @mensaje = 'El Impacto no Existe'
    print @mensaje
  end
else
  if exists (select * from eventos where id_impacto = @id_impacto)
    begin
      set @mensaje = 'No se puede eliminar el impacto, porque ya ha sido utilizado'
      print @mensaje
    end
  else
   begin
     delete impacto from impacto where id_impacto = @id_impacto
     set @mensaje = 'Impacto Eliminado Exitosamente'
     print @mensaje
   end


----- Procedimiento País -------
drop procedure sp_inserta_pais
go
create procedure sp_inserta_pais
@denominacion nvarchar(40)
as
```

```
declare @mensaje nvarchar(50)
declare @id_pais smallint
if exists (select * from pais where denominacion = @denominacion)
  begin
    set @mensaje = 'El País ya Existe'
    print @mensaje
  end
else
 begin
   insert into pais values (@denominacion)
   set @mensaje = 'País Insertado Exitosamente, con la clave:'
   print @mensaje
   select @id_pais = id_pais from pais where denominacion = @denominacion
   print @id_pais
 end


drop procedure sp_actualiza_pais
go
create procedure sp_actualiza_pais
@id_pais smallint,
@denominacion nvarchar(40)
as
```

```
declare @mensaje nvarchar(50)
if not exists (select * from pais where id_pais = @id_pais)
  begin
    set @mensaje = 'El País no Existe'
    print @mensaje
  end
else
  if exists (select * from estado where id_pais = @id_pais)
    begin
      set @mensaje = 'No se puede actualizar el país, porque ya ha sido utilizado'
      print @mensaje
    end
  else
    if exists (select * from empresa where id_pais = @id_pais)
      begin
        set @mensaje = 'No se puede actualizar el país, porque ya ha sido utilizado'
        print @mensaje
      end
     else
       begin
         update pais set denominacion = @denominacion from pais
               where id_pais = @id_pais
```

```
        set @mensaje = 'País Actualizado Exitosamente'
        print @mensaje
      end

drop procedure sp_elimina_pais
go
create procedure sp_elimina_pais
@id_pais smallint
as
 declare @mensaje nvarchar(50)
 if not exists (select * from pais where id_pais = @id_pais)
   begin
     set @mensaje = 'El País no Existe'
     print @mensaje
   end
 else
   if exists (select * from estado where id_pais = @id_pais)
     begin
       set @mensaje = 'No se puede eliminar el País, porque ya ha sido utilizado'
       print @mensaje
     end
   else
```

```
    if exists (select * from empresa where id_pais = @id_pais)
      begin
        set @mensaje = 'No se puede eliminar el País, porque ya ha sido utilizado'
        print @mensaje
      end
     else
      begin
        delete pais from pais where id_pais = @id_pais
        set @mensaje = 'País Eliminado Exitosamente'
        print @mensaje
      end


----- Procedimiento Causas -------
drop procedure sp_inserta_causas
go
create procedure sp_inserta_causas
@denominacion nvarchar(40)
as
 declare @mensaje nvarchar(50)
 declare @id_causa smallint
 if exists (select * from causas where denominacion = @denominacion)
   begín
```

```
     set @mensaje = 'La Causa ya Existe'
     print @mensaje
   end
 else
  begin
    insert into causas values (@denominacion)
    set @mensaje = 'Causa Insertada Exitosamente, con la clave:'
    print @mensaje
    select @id_causa = id_causa from causas where denominacion =
@denominacion
    print @id_causa
  end

drop procedure sp_actualiza_causas
go
create procedure sp_actualiza_causas
@id_causa smallint,
@denominacion nvarchar(40)
as
 declare @mensaje nvarchar(50)
 if not exists (select * from causas where id_causa = @id_causa)
   begin
```

```
set @mensaje = 'La Causa no Existe'
     print @mensaje
   end
 else
   if exists (select * from eventos where id_causa = @id_causa)
     begin
       set @mensaje = 'No se puede actualizar la Causa, porque ya ha sido utilizada'
       print @mensaje
     end
   else
    begin
      update causas set denominacion = @denominacion from causas
            where id_causa = @id_causa
      set @mensaje = 'Causa Actualizada Exitosamente'
      print @mensaje
    end


drop procedure sp_elimina_causas
go
create procedure sp_elimina_causas
@id_causa smallint
as
```

```
declare @mensaje nvarchar(50)
if not exists (select * from causas where id_causa = @id_causa)
  begin
    set @mensaje = 'La Causa no Existe'
    print @mensaje
  end
else
  if exists (select * from eventos where id_causa = @id_causa)
    begin
      set @mensaje = 'No se puede eliminar la Causa, porque ya ha sido utilizada'
      print @mensaje
    end
  else
   begin
     delete causas from causas where id_causa = @id_causa
     set @mensaje = 'Causa Eliminada Exitosamente'
     print @mensaje
   end

----- Procedimiento Buzón -------
drop procedure sp_inserta_buzon
go
```

```
create procedure sp_inserta_buzon
@id_area smallint,
@id_estatus smallint,
@denominacion nvarchar(40)
as
 declare @mensaje nvarchar(50)
 declare @id_buzon smallint
 if exists (select * from buzon where denominacion = @denominacion
                              and id_area = @id_area
                              and id_estatus = @id_estatus)
   begin
     set @mensaje = 'El buzón ya Existe'
     print @mensaje
   end
 else
  if not exists (select * from area where id_area = @id_area)
    begin
      set @mensaje = 'El Área es incorrecta'
      print @mensaje
    end
  else
    if not exists (select * from estatus where id_estatus = @id_estatus)
```

```
    begin
      set @mensaje = 'El Estatus es incorrecto'
      print @mensaje
    end
  else
    begin
      insert into buzon values (@id_area, @id_estatus, @denominacion)
      set @mensaje = 'Buzón Insertado Exitosamente, con la clave:'
      print @mensaje
      select @id_buzon = id_buzon from buzon where denominacion =
@denominacion
                                  and id_area = @id_area
                                  and id_estatus = @id_estatus
      print @id_buzon
    end



drop procedure sp_actualiza_buzon
go
create procedure sp_actualiza_buzon
@id_buzon smallint,
@id_area smallint,
```

```
@id_estatus smallint,
@denominacion nvarchar(40)
as
 declare @mensaje nvarchar(50)
 if not exists (select * from buzon where id_buzon = @id_buzon)
   begin
     set @mensaje = 'El Buzón no Existe'
     print @mensaje
   end
 else
   if not exists (select * from area where id_area = @id_area)
     begin
       set @mensaje = 'El Área es incorrecta'
       print @mensaje
     end
   else
     if not exists (select * from estatus where id_estatus = @id_estatus)
       begin
         set @mensaje = 'El Estatus es incorrecto'
         print @mensaje
       end
     else
```

```
      begin
       update buzon set id_area = @id_area, id_estatus = @id_estatus,
denominacion = @denominacion from buzon
             where id_buzon = @id_buzon
       set @mensaje = 'Buzón Actualizado Exitosamente'
       print @mensaje
      end



----- Procedimiento Estado -------
drop procedure sp_inserta_estado
go
create procedure sp_inserta_estado
@id_pais smallint,
@denominacion nvarchar(40)
as
 declare @mensaje nvarchar(50)
 declare @id_estado smallint
 if exists (select * from estado where denominacion = @denominacion
                          and id_pais = @id_pais)
   begin
     set @mensaje = 'El Estado ya Existe'
```

```
     print @mensaje
   end
 else
  if not exists (select * from pais where id_pais = @id_pais)
    begin
      set @mensaje = 'País erroneo'
      print @mensaje
    end
  else
    begin
      insert into estado values (@id_pais, @denominacion)
      set @mensaje = 'Estado Insertado Exitosamente, con la clave:'
      print @mensaje
      select @id_estado = id_estado from estado where denominacion =
@denominacion
                                    and id_pais = @id_pais
      print @id_pais
    end


drop procedure sp_actualiza_estado
go
create procedure sp_actualiza_estado
```

```
@id_estado smallint,
@id_pais smallint,
@denominacion nvarchar(40)
as
 declare @mensaje nvarchar(50)
 if not exists (select * from estado where id_estado = @id_estado)
   begin
     set @mensaje = 'El Estado no Existe'
     print @mensaje
   end
 else
   if exists (select * from contacto where id_estado = @id_estado)
     begin
       set @mensaje = 'No se puede actualizar el Estado, porque ya ha sido
utilizado'
       print @mensaje
     end
   else
     if not exists (select * from pais where id_pais = @id_pais)
       begin
         set @mensaje = 'País incorrecto'
         print @mensaje
```

```
      end
    else
      begin
        update estado set id_pais = @id_pais, denominacion = @denominacion
from estado
              where id_estado = @id_estado
        set @mensaje = 'Estado Actualizado Exitosamente'
        print @mensaje
       end


drop procedure sp_elimina_estado
go
create procedure sp_elimina_estado
@id_estado smallint
as
 declare @mensaje nvarchar(50)
 if not exists (select * from estado where id_estado = @id_estado)
   begin
     set @mensaje = 'El Estado no Existe'
     print @mensaje
   end
 else
```

```
if exists (select * from contacto where id_estado = @id_estado)
  begin
    set @mensaje = 'No se puede eliminar el Estado, porque ya ha sido utilizado'
    print @mensaje
  end
else
 begin
   delete estado from estado where id_estado = @id_estado
   set @mensaje = 'Estado Eliminado Exitosamente'
   print @mensaje
 end
```

CONCLUSIONES

Como conclusión expongo que las consideraciones esenciales para abordar y concluir un sistema de información (desde su análisis hasta la entrega a producción) facilitan y permiten desarrollos planeados con la menor desviación en costos y tiempo.

El diplomado de Diseño de Sistemas de Información Orientado a Negocios, sin duda es una opción con alto valor académico y profesional que me permitió mejorar y actualizar mis conocimientos respecto a la toma de requerimientos, planeación y diseño de base de datos.

El aspecto de toma de requerimientos, es necesario para identificar las áreas funcionales del negocio y las relaciones de datos y el flujo que tendrán dentro del sistema, y por tanto la consideración de su almacenamiento en una base de datos.

Definir y acotar un alcance durante el diseño de un sistema es obligatorio, para poder realizar planes con mas certidumbre y con menores desviaciones, porque su costo se incrementa al presentarse redefinición o cambios.

El modelo de base de datos, al ser la representación de los datos, y considerar las reglas y su flujo del negocio, se convierte en el esencia y materia primordial que sustenta a cualquier sistema de información.

De esta forma el desarrollo del presente trabajo, se concentro en el análisis de las necesidades a cubrir dentro de un modelo de datos y sus reglas de negocio para un sistema de fallas.

Las técnicas de análisis y modelado empleadas en su elaboración, parten propiamente de los conocimientos y habilidades adquiridas dentro del diplomado. Debido al ámbito de uso y costos de licencia, el modelo se construyó en SQL Server. No obstante, sus instrucciones generadas en lenguaje SQL estándar, son transportables en otro sistema gestor de base de datos.

La base de datos modelada, mantiene la integridad de información y permitirá cubrir las necesidades de seguimiento y alertamiento de reportes de fallas del Call Center. También facilitará la extracción y seguimiento de atención de los reportes problemas.

No obstante, el diseño de la base de datos de este proyecto deberá ser

adaptado en versiones posteriores, debido a que su alcance está cerrado a los lineamientos y necesidades de negocio actuales. Pero es importante señalar que las mejoras no implicarán un rediseño de completo de la base de datos, ya que ésta considera un modelo flexible y cuenta con la mayoría de entidades requeridas por el sistema de fallas.

Este es el mayor valor que podemos ofrecer como desarrolladores de sistemas de información, al modelar base de datos sostenibles, que solo requieren adaptaciones menores y pueden mantenerse con independencia de la capa aplicativa o front end de usuario.

Finalmente si no fuera de esta forma, esto se traduce como una pérdida de tiempo y recursos para el cliente final.

BIBLIOGRAFÍA

DATABASE SYSTEMS
DESIGN, IMPLEMENTATION, AND MANAGEMENT
PETER ROB
CARLOS CORONEL
THIRD EDITION,
UNITED STATED OF AMERICA
EDITORIAL CT / ITP 1997
ISBN 0-7600-4904-1

USING SYBASE SYSTEM XI
PETER HAZLEHURST
EDITORIAL QUE
ISBN 0-7897-0087-5

USING SYBASE SYSTEM XI
PETER HAZLEHURST
EDITORIAL QUE
ISBN 0-7897-0087-5

SYBASE SQL SERVER RELEASE 11.0.X
PERFORMANCE AND TUNING GUIDE
CHAPTER 2 DATABASE DESIGN AND DENORMALIZING FOR PERFORMANCE
DOCUMENT ID: 32645-01-1100-03
FEBRARY 7 1996

DATA WAREHOUSING: LA INTEGRACIÓN DE INFORMACIÓN PARA LA MEJOR TOMA DE DECISIONES
HARJINDER S. GILL
PRAKASH C. RAO
PRIMERA EDICION
MÉXICO, 1996
EDITORIAL PRENTICE-HALL
ISBN 0-7897-0714-4

DATA MANAGEMENT REVIEW: THE 12 RULES OF DATA WAREHOUSE FOR A CLIENT/SERVER WORLD
VOL 4, NUM. 5
MAY 1994

ORACLE MAGAZINE
VOLUME X/NUMBER 2
MARCH/APRIL 1996
DATA WAREHOUSE: BUILDING BLOCKS FOR THE NEXT MILLENIUM
DAVID BAUM
PP 34 -47

DATA MANAGEMENT REVIEW: THE 12 RULES OF DATA WAREHOUSE FOR A CLIENT/SERVER WORLD
VOL 4, NUM. 5
MAY 1994