6
29

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

SISTEMA DE CONTROL DE CURSOS DE LA DIVISIÓN DE EDUCACIÓN CONTINUA DE LA FACULTAD DE DERECHO

T E S I S

QUE PARA OBTENER EL TITULO DE

INGENIERO EN COMPUTACIÓN

P R E S E N T A N :

AMADOR ALCÁZAR DANIEL
DOMÍNGUEZ ESTRADA ALEJANDRO

DIRECTOR DE TESIS: ING. GABRIEL CASTILLO HERNÁNDEZ

MÉXICO, D.F.. OCTUBRE 1998.

AGRADECIMIENTOS

A LA UNIVERSIDAD NACIONAL AUTONOMA DE MÉXICO COMO TESTIMONIO DE GRATITUD POR TODA LA FORMACIÓN ACADÉMICA Y POR PODER PRONUNCIAR SU NOMBRE CON ORGULLO: **"LA MÁXIMA CASA DE ESTUDIOS"**.

A LA FACULTAD DE INGENERÍA QUIEN CON TODOS SUS PROFESORES E INGENIEROS, EN ESPECIAL EL ING. GABRIEL CASTILLO HERNÁNDEZ CONFIARON EN NOSOTROS COMO FUTUROS INGENIEROS BRINDÁNDONOS EL APOYO, DISCIPLINA Y CONOCIMIENTOS.

A LA FACULTAD DE DERECHO Y A LA DIVISIÓN DE EDUCACIÓN CONTINUA POR LA ATENCIÓN Y EL APOYO RECIBIDO PARA LA ELABORACIÓN DE ESTA TESIS.

POR TODO: GRACIAS

D.A.A.
A.D.E.

INDICE

CAPÍTULO 1

CAPÍTULO 2

INTRODUCCIÓN

Se vislumbra una nueva era en el mundo del manejo de la información. Está en proceso una gran cantidad de cambios ingeniosos en la manera como recibimos información y las fuentes de donde la obtenemos, pero es que el surgimiento de una era siempre es gradual y sutil. Somos muy afortunados de poder atestiguar la increíble multiplicación de información de la que disponemos. Por si fuera poco, cada día surge un aspecto nuevo y se da un paso más hacia una nueva forma de vida.

Sabemos que vivimos en la Era de la Información. A la fecha, hemos producido información al por mayor, generando datos acerca de cualquier cosa imaginable. Todos los días se producen teorías, se hacen descubrimientos y se gestan tendencias de las cuales pronto se informa al público. Aunque algunos de nosotros parecemos sufrir de sobrecarga de datos, el diluvio no da signos de aminorar.

¿Qué hace uno con toda esa información? O lo que es más importante, ¿cómo va uno a obtener datos que tengan significado de una enorme gama de datos en un tiempo razonable?.

Es necesario poner esa información al alcance de las personas que puedan usarla. Cuanto más fácil sea de utilizar, más gente la usará. También es necesario pensar cómo utilizar toda la información para mejorar nuestra forma de trabajo y con ello facilitar la obtención de resultados.

De acuerdo a lo anterior, el propósito de esta tesis ambiciona incluir en la forma de trabajo cotidiano el uso de la computadora como una herramienta que le facilite el manejo de información, de esta manera, se tiene como principal objetivo el desarrollo de un Sistema de Bases de Datos para la División de Educación Continua de la Facultad de Derecho, la cual es la encargada de impartir y controlar los cursos dirigidos a los profesionales del derecho y ramas afines, con el objeto de que constantemente actualicen sus conocimientos conforme al avance de la ciencia jurídica.

Como todo buen sistema que pretende comprender el uso de una computadora, no intentamos automatizar todo (volver a todo el trabajo esclavo de la computadora), más bien conjugar las actividades manuales con el uso de la computadora.

La problemática que se vivía en Educación Continua de la Facultad de Derecho es el hecho que aún con todos los avances tecnológicos se seguía trabajando en la forma antigua, es decir, todo sobre papel y a pulso. Olvidando las faltas de ortografía y la copia de datos equivocados, si algún papel se guardaba en el lugar equivocado de un archivero, éste se podía perder para siempre.

Es entonces cuando toma importancia el siguiente estudio que aspira solucionar en gran medida la pérdida de alguna información útil a la Dependencia y hacer que el modo de trabajo sea más fácil, más controlable, más gestionable y sobre todo amistoso a la gente encargada de sacarlo adelante.

El Sistema de Control de Cursos de la División de Educación Continua de la Facultad de Derecho, como ha sido llamado, hace posible la integración de los datos que se generan durante el proceso de los cursos. La base de datos ha sido desarrollada

utilizando Sybase versión 11.2 como DBMS y como front-end Power Builder versión 5.0.

El trabajo realizado incluye en el primer capítulo, un estudio detallado de los tipos de sistemas para comenzar la delimitación del problema y estructurar de manera ordenada el resto de los pasos a seguir. Posteriormente, es preciso definir el ciclo de vida del proyecto para organizar las actividades de todo el proyecto y con esto asegurar una apropiada secuencia de las actividades, además de tener una visión tanto global como particular del sistema.

Una de las partes más importantes dentro del ciclo de vida del proyecto es la fase del análisis, ya que es el primer paso técnico en el proceso de solución del problema. Siguiendo una metodología estructurada se precisa de la construcción de modelos (Diagrama de Flujo de Datos, Diagrama Entidad-Relación, Diccionario de Datos, etc.) para mostrar las características y funciones principales del sistema.

También en el capítulo 1 se tiene un apartado dedicado a la teoría relativa a las bases de datos, partiendo de sus características y elementos principales. Uno de los aspectos de vital importancia para el caso que nos ocupa se refiere a las ventajas y desventajas de la centralización de los datos dentro de una organización, por tal motivo se hace mención de los aspectos principales, así como de la manera de manejar correctamente esta situación. Siguiendo con la teoría de bases de datos se mencionan conceptos básicos e importantes tanto de clasificación de modelos de bases de datos, como de los elementos principales del modelo Entidad-Relación.

Es necesario también dentro de un estudio serio de bases de datos mencionar términos relativos al álgebra relacional, al igual que presentar un apartado especial para la normalización y conceptos de SQL (Structured Query Languaje).

Como parte final del capítulo 1 se menciona un tema denominado "Normatividad" con el cual se tratan aspectos de suma importancia directamente involucrados con la centralización de los datos, en éste se muestran las normas vigentes de la institución (Facultad de Derecho de la UNAM), las cuales tienen como objetivo principal gestionar los lineamientos a los cuales se deben apegar todos las propuestas de innovación.

En el capítulo 2 (Planteamiento del Problema y Propuesta de Solución), se presenta propiamente dicho el proyecto; planteando de manera detallada el problema, definiendo las restricciones y presentando una propuesta de solución a partir del análisis desarrollado.

Aquí también se incluye de manera detallada y ordenada las fases del ciclo de vida del proyecto totalmente orientadas al problema que nos ocupa. De esta forma, se presenta el estudio de la situación actual, las restricciones y requerimientos de la División de Educación Continua de la Facultad de Derecho, así como el manual de procedimientos hasta entonces utilizado por esta dependencia, y del cual se obtuvo gran cantidad de información importante para los fines de esta tesis.

Uno de los aspectos que más se detalla debido a su trascendencia es la etapa del

análisis, en ella se muestra (de acuerdo a la teoría de Yourdon) el Modelo Esencial y el Modelo de Comportamiento del sistema, incluyendo dentro de ellos algunos de gran importancia como: la Declaración de Propósito, el Diagrama de Contexto, el Diagrama de Flujo de Datos, el Diagrama Entidad-Relación, entre otros.

Dada la importancia de la integridad y consistencia de los datos dentro de una base de datos, es preciso mostrar de manera clara la Normalización de los datos y su influencia sobre el Diagrama Entidad-Relación.

Para la fase de desarrollo se utilizó una herramienta CASE llamada ERwin con la cual se hizo posible conseguir la generación automática de la Base de Datos desde una especificación a nivel de diseño y llevarla directamente al servidor Sybase, con ésta se cubrieron los principales aspectos para permitir la entrada de los datos a la base de datos, tales como: integridad referencial y reglas de validación. También a través de este CASE fue posible la generación del Diccionario de Datos.

Para el usuario la parte más importante del sistema la representa la navegación a través de las pantallas, ya que por medio de éstas tendrán ahora la interacción con los datos que se manejaban de manera manual, por tal motivo, se presenta un apartado donde se detalla la forma en la que se decidió hacer la navegación, en la que además de ser intuitiva representara una interacción fácil y amigable.

Es importante mencionar que se realizaron una serie de pruebas ("Generación de pruebas de Aceptación") para comprobar el correcto funcionamiento, además de constatar la integridad y consistencia de los datos almacenados tanto en el back-end (DBMS) como en el front-end.

En orden de aparición las conclusiones vienen a mostrar los aspectos más importantes que se derivan del estudio realizado.

Finalmente, existen 2 apéndices los cuales muestran la teoría relativa a la normalización (Apéndice A) y el código generado por el CASE Erwin para la base de datos en Sybase (Apéndice B).

CONCEPTOS

1. DEFINICIÓN DE SISTEMA

La palabra "sistema" es posiblemente el término más sobreutilizado y del que más se ha abusado en el léxico técnico. Hablamos de sistemas políticos y educativos, de sistemas de aviación y de fabricación, de sistemas bancarios y de ferrocarril. La palabra nos dice poco, y puede tener cualquier número de acepciones, siempre obedeciendo al ámbito dentro del que se encuentra inmerso. Por tal motivo, es necesario hacer una delimitación de definición de sistema enfocada al campo del conocimiento que la requirió, de lo contrario podríamos tener una definición totalmente general como la que aparece en todos los libros de sistemas.

" Un sistema es un conjunto de elementos interrelacionados ordenadamente entre sí, con un fin común."

La anterior definición nos es útil para empezar a construir una definición de sistema más acorde con nuestro proyecto. Para tal fin, es necesario hacer una clasificación de sistemas.

1.1. TIPOS COMUNES DE SISTEMAS

Como se ha mencionado existen muchos tipos diferentes de sistemas; de hecho, casi todo aquello con lo cual entramos en contacto durante nuestra vida cotidiana es un sistema o bien parte de un sistema (o ambas cosas).

Es útil organizar los diferentes tipos de sistemas en categorías. Dado que nuestro objetivo son los sistemas computacionales, empezaremos por dividir todos los sistemas en dos categorías: sistemas naturales y sistemas hechos por el hombre.

1.1.1. Sistemas naturales

La gran mayoría de los sistemas no están hechos por el hombre: existen en la naturaleza y sirven a sus propios fines. Algunos de estos sistemas son: La rotación de la tierra sobre su propio eje, la fotosíntesis en las plantas, un ecosistema (como un bosque, en donde la fauna, la flora, el clima y muchos otros elementos interactúan entre sí para un bien común), etc.

1.1.2. Sistemas hechos por el hombre

En la actualidad, numerosos de estos sistemas incluyen las computadoras; inclusive, muchos no podrían actuar sin ellas. Sin embargo, es igualmente importante señalar que dichos sistemas existían antes de que hubiera computadoras; de hecho, algunos sistemas continúan por completo sin computarizar y podrían permanecer así durante muchas décadas más. Otros

contienen a la computadora como componente, pero también incluyen uno o más componentes no computarizados (o manuales). Por lo tanto, el hablar de un sistema que incluya a las computadoras no necesariamente implica que éste sea completamente computarizado, pudiendo tener procedimientos internos — muy importantes— que sigan siendo manuales.

El que un sistema de manufactura humana deba o no ser computarizado es una cuestión que debe analizarse; no es algo que se da por hecho, por tal motivo, una de las labores primarias es analizar o estudiar el sistema para determinar su esencia: su comportamiento independientemente de la tecnología utilizada para implantar el sistema. En la mayoría de los casos, podremos determinar si tiene sentido utilizar una computadora para llevar a cabo las funciones del sistema, después de haber hecho un análisis.

1.1.2.1. Sistemas automatizados

Automatizado, no es necesariamente computarizado en su totalidad, hay humanos que realizan tareas automáticamente (mecánicas). Por tanto estos sistemas hechos por el hombre interactúan con, o son controlados por una o más computadoras, no por ello realizan todo el trabajo, más bien trabajan conjuntamente con el hombre. Son precisamente este tipo de sistemas los que nos interesan, y diremos que:

Un sistema automatizado es "un conjunto u ordenación de elementos organizados para llevar a cabo algún método, procedimiento o control mediante el procesamiento, manejo y manipulación de datos." [1]

Aunque hay diferentes tipos de sistemas automatizados, todos tienen por lo regular los siguientes componentes en común:

- ***El hardware de las computadoras***: los procesadores, los discos, terminales, impresoras, unidades de cinta magnética, etcétera.
- ***El software de la computadora***: los programas de sistemas, tales como: sistemas operativos, sistemas de bases de datos, programas de control, además de programas de aplicación que llevan a cabo funciones deseadas por el usuario.
- ***Los usuarios***: los que operan el sistema, los que proveen su material de entrada y consumen su material de salida, y los que proveen actividades de procesamiento manual en un sistema.
- ***El área de sistemas***: encargada de coordinar y llevar a cabo los pasos de diseño, implantación y mantenimiento de este tipo de sistemas.

[1] PRESSMAN, Roger S. Ingeniería del Software. (Un enfoque práctico). 3a edición. Ed. McGraw-Hill. Pag. 140

- ***Los datos***: la información que el sistema almacena durante un período.
- ***Los procedimientos***: las políticas formales e instrucciones de operación del sistema.

Para que el sistema nuevo tenga éxito, el analista debe entender a los demás sistemas con los que va a interactuar, sabiendo cómo se comunica con ellos, además de comprender con razonable detalle cómo se comporta el sistema total, todo con la finalidad de visualizar una interfaz hacia los otros sistemas. Para lograr lo anterior, es conveniente conocer la teoría relativa a sistemas, tanto en un contexto general como particular.

Una división en categorías más útil de los sistemas automatizados es la siguiente:

1.2.1.1.1 Sistemas en línea

Una característica común de los sistemas en línea es que responden a un diálogo con el usuario, interactuando entre ambos.

Dado que un sistema en línea interactúa directamente con personas, es importante que se planee cuidadosamente la interfaz humano-computadora, es decir, se deben crear modelos de acuerdo a las peticiones válidas que el usuario humano pueda solicitar al sistema desde su terminal (sitio de acceso a la información), y de todas las respuestas que el sistema pudiera dar a dichos estímulos, además de las acciones a tomar por el humano ante las respuestas de la computadora, etc.

Dado que los sistemas en línea por lo común requieren recuperar datos con rapidez, suele ser muy importante diseñar los archivos y las bases de datos de la manera más eficiente posible, pues a menudo las operaciones de computación llevadas a cabo por un sistema en línea pueden ser relativamente triviales, mientras que los datos suelen ser bastante complejos.

1.2.1.1.2. Sistemas de tiempo real

Un sistema de tiempo real puede definirse como aquel que controla un ambiente recibiendo datos, procesándolos y devolviéndolos con la suficiente rapidez como para influir en dicho ambiente en ese momento.

Además de la velocidad, existe otra característica que diferencia a los sistemas de tiempo real de los sistemas en línea: estos últimos suelen interactuar con las personas, mientras que los sistemas de tiempo real usualmente interactúan tanto con personas como con un ambiente —del cual recibe datos— que en general es autónomo y a menudo hostil. Inclusive, la principal preocupación del analista de sistemas en tiempo real, es que, si la computadora no responde con la suficiente rapidez, el sistema puede quedar fuera de control. Un ejemplo de estos sistemas es el monitoreo de la contaminación ambiental

1.2.1.1.3. Sistemas de apoyo a decisiones y sistemas de planeación estratégica

Como lo indica el término, estos sistemas computacionales no toman decisiones por sí mismos, sino ayudan a los administradores y a otros profesionistas ("trabajadores del conocimiento") de una organización a tomar decisiones e informarlos acerca de los diversos aspectos de la operación. Típicamente, los sistemas de apoyo a decisiones son pasivos en el sentido de que no operan en forma regular; más bien, se utilizan cuando se les necesita.

Una característica común de los sistemas de apoyo a decisiones, es que no sólo recuperan y exhiben los datos, sino que también realizan varios tipos de análisis matemáticos y estadísticos de los mismos. Los sistemas de apoyo a decisiones también tienen la capacidad, en la mayoría de los casos, de representar la información en una variedad de formas gráficas al igual que en forma de reportes convencionales.

Los sistemas de planeación estratégica son utilizados por los gerentes para evaluar y analizar la misión de la organización. En lugar de dar consejos acerca de alguna decisión de negocios aislada, estos sistemas ofrecen consejos más amplios y generales acerca de la naturaleza del mercado, las preferencias de los consumidores, el comportamiento de la competencia, etc.

1.2.1.1.4. Sistemas de conocimiento

Dichos sistemas se asocian con el campo de la inteligencia artificial. Contienen grandes cantidades de diversos conocimientos que emplean en el diseño de una tarea dada.

2. EL CICLO DE VIDA DEL PROYECTO

Como ya se ha mencionado es importante conocer la teoría relativa a los tipos de sistemas, para de alguna manera delimitar con fundamentos el ámbito dentro del cual nos encontramos inmersos, es decir, conocer las características del tipo de sistema que se desea realizar. Resulta también de vital importancia elegir una metodología que nos permita abordar el problema de manera ordenada.

En la profesión de desarrollo de sistemas, los términos "método", "metodología", "ciclo de vida del proyecto" y "ciclo de vida del desarrollo de sistemas" se usan de manera casi indistinta.

Dependiendo de la naturaleza de un proyecto de desarrollo de sistemas, puede haber razones válidas para adoptar un método en lugar de otro, e incluso algunos proyectos pudieran requerir una combinación de algunos de ellos.

Es muy conveniente utilizar una metodología en un proyecto, por tanto, es fundamental

tener una visión general de lo que se involucra dentro del ciclo de vida del proyecto. Existen algunos objetivos principales:

1. Definir las actividades a llevarse a cabo en un proyecto, de manera que tengan secuencia unas con otras en forma adecuada en el desarrollo de sistemas.

2. Proporcionar puntos de control y revisión administrativos para decidir si continuar o no con el proyecto.

La ayuda más valiosas que proporciona el ciclo de vida del proyecto es que puede organizar las actividades del administrador, aumentando la probabilidad de que se aborden los problemas pertinentes en el momento adecuado. Existen diversos ciclos de vida dependiendo de la metodología empleada, pero lo que realmente nos interesa dentro de esta tesis es la metodología estructurada.

2.1. EL CICLO DE VIDA ESTRUCTURADO DEL PROYECTO

Existen nueve actividades y tres terminadores del ciclo de vida del proyecto. Los terminadores —los rectángulos— son los usuarios, los administradores y el personal de operaciones (operadores). Se trata de individuos o grupos que proporcionan las entradas al sistema, y son los beneficiarios finales del sistema. Ellos interactúan con las nueve actividades que abordaremos a continuación (la figura se encuentra en la siguiente página).

1) La encuesta

Comienza cuando el usuario solicita que una o más partes de su sistema se automaticen. Los principales objetivos son los siguientes:

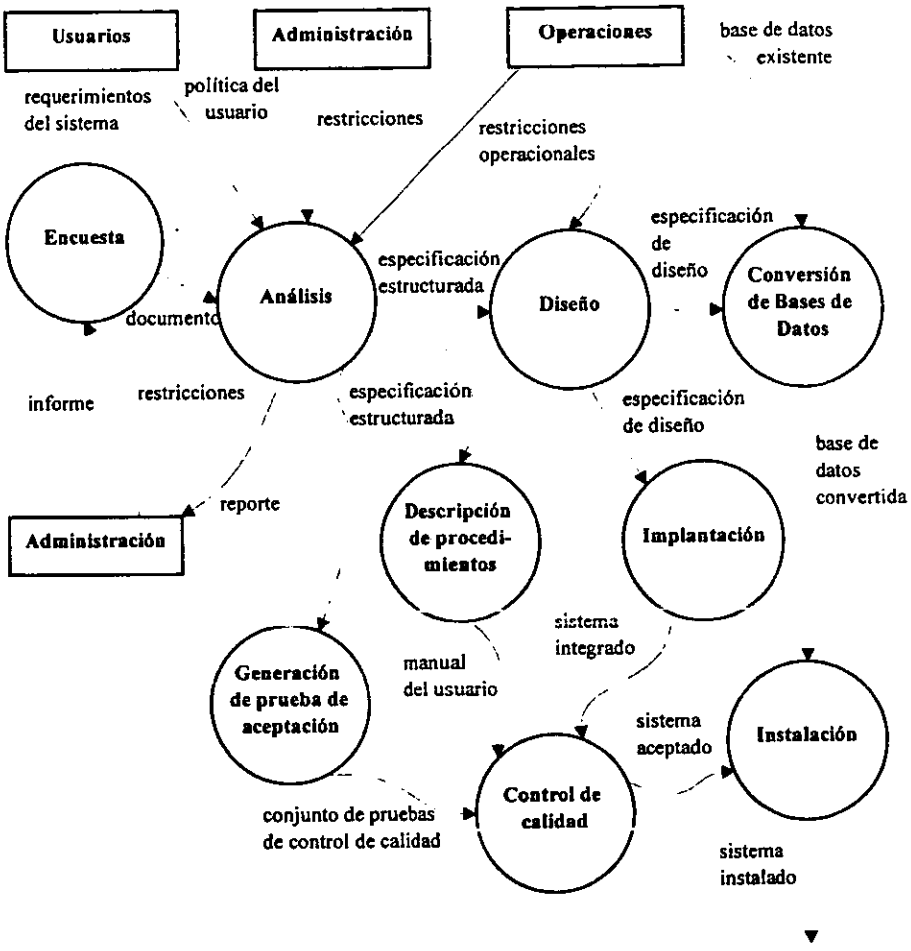
- Identificar a los usuarios responsables y crear un "campo de actividad" inicial del sistema, es decir, qué usuarios estarán comprendidos o serán afectados por el proyecto propuesto.

- Identificar las deficiencias actuales en el ambiente del usuario, numerando todas aquellas tareas que se llevan a cabo insatisfactoriamente y que con la ayuda del sistema nuevo pudieran mejorarse.

- Establecer metas y objetivos para un sistema nuevo.

- Determinar si es factible automatizar el sistema y de ser así, sugerir escenarios aceptables.

- Preparar el esquema que se usará para guiar el resto del proyecto.



2) El análisis de sistemas

Ésta es una de las partes más importantes en todo el ciclo de vida del proyecto. Su propósito principal es modelar toda la información reunida hasta este momento, transformando las políticas y requerimientos del usuario, al igual que el esquema del proyecto, en una especificación estructurada. Esto implica modelar el sistema desarrollando un modelo ambiental y de comportamiento, independientemente de la naturaleza de la tecnología que se

use para cubrir los requerimientos.

3) El diseño

La actividad de diseño se dedica a la creación de una jerarquía apropiada de módulos de programas y de interfaces entre ellos para implantar la especificación creada en la actividad anterior. Además, la actividad de diseño se ocupa de la transformación de modelos de datos de entidad-relación en un diseño de base de datos.

4) Implantación

Esta actividad incluye la codificación e integración de módulos en un esqueleto del sistema final. Incluye tanto programación estructurada como la implantación descendente (el desarrollo de módulos desde el principal hasta el menos significativo).

El analista típicamente no se involucra en esto, pero existen proyectos donde el análisis, el diseño y la implantación (también llamado implementación) lo hacen la misma persona.

5) Generación de pruebas de aceptación

Una vez que se ha definido el sistema y se ha hecho la especificación estructurada (parte esencial de la implantación), puede comenzar la actividad de producir un conjunto de casos de prueba de aceptación. Este desarrollo de pruebas puede realizarse al mismo tiempo que las actividades de diseño e implantación, pero no antes de que acabe la implantación.

6) Garantía de calidad

Es la actividad en donde se verifica que el sistema tenga un nivel apropiado de calidad, llamada también prueba final o la prueba de aceptación. Requiere como entradas los datos de la prueba de aceptación generada en la actividad 5 y el sistema integrado producido en la actividad 4.

7) Descripción del procedimiento

Para nuestro caso el proyecto incluye la creación de un sistema completo, no sólo la porción automatizada (que involucre la computadora), sino también de la parte que llevarán a cabo las personas. Por ello, una de las actividades importantes es la generación de una descripción

formal de las partes del sistema que se harán en forma manual, lo mismo que la descripción de cómo interactuarán los usuarios con la parte automatizada del nuevo sistema. El resultado de esta actividad es un manual para el usuario, un documento comúnmente llamado manual de procedimientos.

8) Conversión de bases de datos

Esta actividad requiere como entrada la base de datos actual del usuario —si existiera—, al igual que la especificación del diseño producida por medio del diseño (actividad 3) para su conversión a una base de datos y realizar pruebas con los datos en la actividad 5.

9) Instalación

La actividad final es la instalación, donde se entrega el sistema completamente aceptado producido por la actividad 6, el manual del usuario producido en la actividad 7 y la base de datos convertida que se creó con la actividad 8.

Con el ciclo de vida del proyecto se siguen los pasos anteriores, pero una vez terminado uno de ellos no quiere decir que jamás lo abordaremos. Siempre regresaremos a ellos aun cuando estemos por terminar nuestro proyecto, sobretodo en los pasos intermedios de pruebas del nuevo sistema.

Existen dos metodologías para la etapa 2 (análisis de sistemas) del ciclo de vida del proyecto. Sea cual sea la metodología utilizada en esta etapa del proyecto su propósito principal es modelar toda la información reunida hasta ese momento.

3. METODOLOGÍAS DE ANÁLISIS

El análisis es el primer paso técnico del proceso de solución de un problema. Es en este punto donde se refinan los requisitos del nuevo proyecto en una especificación concreta que se convierte en la base de todas las actividades subsiguientes.

El análisis debe concentrarse en los ámbitos de información, funcional y de comportamiento del problema. Para comprender mejor lo que se requiere: se crean modelos, se parte el problema y se desarrollan representaciones que muestran la esencia de los requisitos, y posteriormente los detalles de implementación.

En esta fase es donde se identifican las necesidades del usuario, se determina la viabilidad del proyecto, se asignan las funciones al software, al hardware , a la gente y a las bases de datos (los elementos clave del sistema).

A continuación se enuncian los métodos de análisis más trascendentes.

3.1. ANÁLISIS ESTRUCTURADO

Probablemente no exista otro método que haya despertado tanto interés, que haya sido probado por tanta gente, que sea punto de tantas críticas y tanta controversia, lo que lo ha llevado a prosperar y ser ampliamente utilizado alrededor de todo el mundo, en donde el modelado de sistemas resulta muy importante.

El análisis estructurado es una actividad de construcción de modelos. Mediante una notación característica del método de análisis estructurado se crean modelos que reflejan el flujo y el contenido de la información (datos y control); se parte el sistema funcionalmente y según los distintos comportamientos, se establece la esencia de lo que se debe construir. Y es precisamente este tipo de análisis el que se empleará en esta tesis.

3.2. ANÁLISIS ORIENTADO A LOS OBJETOS

Desde principios de los años 80, el análisis orientado a los objetos ha madurado como un enfoque de desarrollo de software, progresando poco a poco como método de análisis de requisitos por derecho propio y como complemento de otros métodos de análisis.

En lugar de examinar un problema mediante el modelo clásico (estructurado) de entrada-proceso-salida (flujo de información), el análisis orientado a los objetos introduce conceptos propios, que aunque parecen ser inusuales son bastante naturales.

Los métodos orientados a los objetos para el análisis de requisitos permiten obtener el modelo del problema representando clases, objetos, atributos y operaciones como componentes principales de modelado. Los puntos de vista orientados a los objetos combinan la clasificación de objetos, la herencia de atributos y los mensajes de comunicación dentro del contexto de la notación de modelado.

Los objetos modelan casi cualquier aspecto identificable del ámbito del problema: entidades externas (individuos o grupos que proporcionan las entradas al sistema, y son los beneficiados finales del sistema), cosas, sucesos, unidades organizativas, lugares, estructuras, etc. Como punto importante, los objetos encapsulan datos y procesos (toda la información está empaquetada bajo un sólo nombre). Las operaciones de procesamiento son parte del objeto y son iniciadas pasando un mensaje al objeto (estímulo).

Los métodos de análisis proporcionan un enfoque sistemático para el análisis de problemas. Aunque cada método tiene un conjunto único de procedimientos y su propia simbología, ambos proporcionan mecanismos para evaluar y representar el campo de la información, para partir el ámbito funcional y para modelar los procedimientos tanto para el mundo físico como

para el mundo lógico (de las computadoras).

De esta manera, como lo menciona Pressman [PRE93], el que se aplique correctamente un método de análisis conlleva a que la ruta sea directa y el viaje llegue a buen término en el proceso de entender, analizar, diseñar e implementar un nuevo sistema.

Dado que el análisis en una de las actividades más importantes del ciclo de vida del proyecto, el analista debe ser un orquestador, un comunicador y un facilitador de toda la información concerniente al sistema de estudio. Es evidente que lo anterior implica una gran cantidad de trabajo, por lo cual, resulta significativo realizarlo en armonía con los demás participantes de los sistemas (los usuarios).

Para todo lo anterior, en el estudio y análisis riguroso de todos los detalles de los sistemas, existen gran cantidad de herramientas que nos ayudan a comprender los sistemas completos, como veremos a continuación.

4. HERRAMIENTAS DEL ANÁLISIS ESTRUCTURADO

Gran parte de la labor que se desempeña involucra el modelado del sistema que desea el usuario. Los modelos de análisis de sistemas son en su mayoría, modelos en papel del futuro sistema, es decir, representaciones abstractas de lo que al final será una combinación de hardware y software de computadora.

Se pueden construir modelos de manera tal que enfaticen ciertas propiedades críticas del sistema, mientras que simultáneamente se desatiendan otras. Esto permite la comunicación con el usuario de manera enfocada, sin distraerse con asuntos y características ajenas al sistema. Y si la comprensión de los requerimientos del usuario no fue correcta (o que el usuario cambió de parecer acerca de sus requerimientos), se pueden hacer cambios en el modelo o desecharlo y hacer uno nuevo, de ser necesario. La alternativa es tener algunas reuniones preliminares con el usuario y luego construir todo el sistema.

Por esta razón, se hace uso de herramientas de modelado para:

- Concentrarse en las propiedades importantes del sistema y al mismo tiempo restar atención a otras menos importantes.
- Discutir cambios y correcciones de los requerimientos del usuario, a bajo costo y con el riesgo mínimo.
- Verificar que el analista comprenda correctamente el ambiente del usuario y que lo haya respaldado con información documental para que los diseñadores de sistemas y los programadores puedan construir el sistema.

5. CRITERIOS PARA SELECCIONAR MODELOS

Para la selección de la herramienta de modelado se deben considerar los siguientes aspectos:

- Debe ser gráfica, con detalles textuales de apoyo apropiados. En general se utilizan los gráficos para identificar los componentes de un sistema y su interfaz.
- Debe permitir que el sistema sea visto en segmentos, en forma descendente. Dado que los problemas reales pocas veces son pequeños, será imposible que alguien, sea usuario, analista o programador, se enfoque a todo el sistema al mismo tiempo. Por eso, la herramienta debe permitir mostrar partes individuales del sistema de manera independiente, además de un mecanismo conveniente para pasar directamente de un nivel alto a un bajo.
- Debe tener redundancia mínima, es decir información repetida, sobre todo en los distintos segmentos del modelo.
- Debe ayudar al lector a predecir el comportamiento del sistema. Un buen modelo debe ser tan fácil de leer que el lector no tenga que detenerse a pensar siquiera que se trata de la representación de un sistema, sino del sistema mismo.
- Debe ser transparente para el lector.

6. MODELADO DEL SISTEMA

Una de las limitaciones más importantes de implantación es la determinación de la frontera de automatización, es decir, la especificación de cuáles funciones del nuevo sistema se automatizarán y cuáles se harán manualmente. Cualquier sistema que se desarrolle, no importa lo ambicioso ni grandioso, será parte de un sistema aún mayor.

Es recomendable saber qué herramientas de modelado son útiles para comenzar, tan pronto como sea posible, a desarrollar un modelo del nuevo sistema que el usuario desea. Este nuevo sistema, conocido en los libros clásicos de análisis estructurado como el nuevo sistema lógico, ***nos referiremos a él, en esta tesis, como el modelo esencial del sistema.***

Es críticamente importante desarrollar el modelo esencial de un sistema, pues muchos sistemas grandes de información tienen una vida media de unos 10 a 20 años. Durante ese período se puede esperar que el hardware mejore por lo menos en un factor de mil, y probablemente se acerque más a un millón o más. Una computadora un millón de veces más rápida, pequeña y barata que las actuales, es en verdad algo cercano a la tecnología perfecta; debe empezarse hoy a modelar sistemas como si se tuviera esa tecnología a la disposición.

6.1. EL MODELO ESENCIAL

El modelo esencial del sistema es un modelo de lo que el sistema debe hacer para satisfacer los requerimientos del usuario, diciendo lo mínimo posible —de preferencia nada — acerca de cómo se implantará.

Específicamente, esto significa que cuando el analista habla con el usuario acerca de los requerimientos del sistema, debe evitar describir o mostrar las funciones del sistema que están siendo realizadas por humanos o equipos de cómputo existentes.

El modelo esencial consiste en dos componentes principales:

1. Modelo ambiental
2. Modelo de comportamiento

El modelo ambiental define la frontera entre el sistema y el resto del mundo (el ambiente en el cual existe el sistema).

El modelo de comportamiento describe precisamente el comportamiento que del sistema se requiere para que interactúe de manea exitosa con el ambiente.

6.1.1. El Modelo ambiental

El primer modelo importante que se debe desarrollar es uno que defina las interfaces entre el sistema y el resto del universo, es decir, el ambiente. Por tal motivo se conoce como el modelo ambiental (modela el exterior del sistema).

Además de determinar qué está en el interior del sistema y qué en el exterior (lo que se logra definiendo la frontera entre el sistema y el ambiente), también es críticamente importante definir las interfaces entre el sistema y el ambiente. Se necesita saber qué información entra al sistema desde el ambiente exterior, y qué información produce como salida al ambiente externo.

Otro aspecto crítico del modelo ambiental consiste en identificar los acontecimientos que ocurren en el ambiente al cual debe responder el sistema. No para todos los acontecimientos el ambiente en su totalidad genera un número infinito de acontecimientos. Sólo son importantes aquellos que: (1) ocurren en el ambiente exterior y (2) requieren de una respuesta del sistema.

El modelo del ambiente consta de tres componentes:

1. Declaración de propósito
2. Diagrama de contexto

3. Lista de acontecimientos

Cada uno se discute a continuación.

6.1.1.1. La declaración de propósito

El primer componente del modelo ambiental es una declaración textual breve y concisa del propósito del sistema.

La declaración de propósito puede constar de una, dos o varias frases. Sin embargo, jamás debe llegar a más de un párrafo, ya que la intención no es proporcionar una descripción completa y detallada del sistema. Tal esfuerzo iría en contra del objetivo, ya que el propósito del resto del modelo ambiental y del modelo de comportamiento es dar todos los detalles pertinentes.

Como resultado, la declaración de propósito será deliberadamente vaga en cuanto a muchos detalles.

6.1.1.2. El diagrama de contexto

La siguiente parte del modelo ambiental empieza a contestar algunas de las preguntas que surgen a raíz de la declaración de propósito. El diagrama de contexto es un caso especial del diagrama de flujo de datos (que se analizará más adelante), en donde de una manera general se representa todo el sistema. El diagrama de contexto enfatiza varias características importantes del sistema:

- Las personas, las organizaciones y sistemas con los que se comunica el sistema se conocen como terminadores.
- Los datos que el sistema recibe del mundo exterior y que debe procesar de alguna manera.
- Los datos que el sistema produce y que se envían al mundo exterior.
- Los almacenes de datos que el sistema comparte con los terminadores. Estos almacenes de datos se crean fuera del sistema para su uso, o bien, son creados en él y usados fuera.
- La frontera entre el sistema y el resto del mundo.

El diagrama de contexto utiliza la notación del diagrama de flujo de datos representando las funciones del sistema a través de los terminadores, flujos de datos, procesos y almacenes de datos.

6.1.1.3. La lista de acontecimientos

La lista de acontecimientos es una lista narrativa de los estímulos que ocurren en el mundo exterior a los cuales el sistema debe responder. Los acontecimientos se pueden etiquetar con una F, T o C. Con ello se muestra si son de tipo de flujo, temporal o de control.

No necesariamente existe una correspondencia uno a uno entre los flujos de datos del diagrama de contexto y los acontecimientos de la lista de acontecimientos. En general cada flujo de datos es un acontecimiento (o más precisamente la indicación de que ha ocurrido), o bien, es requerido por el sistema para poder procesar un acontecimiento.

El acontecimiento de flujo es el que se asocia con un flujo de datos, es decir, el sistema se da cuenta de que ha ocurrido el acontecimiento cuando llega un dato (o posiblemente varios).

Como su nombre lo indica, los acontecimientos temporales arrancan con la llegada de un momento dado en el tiempo. Se inician con flujos de datos de entrada; podría imaginarse que el sistema tiene un reloj interno con el cual puede determinar el paso del tiempo.

Los acontecimientos de control deben considerarse un caso especial del acontecimiento temporal: un estímulo externo que ocurre en algún momento impredecible. A diferencia de un acontecimiento temporal normal, al acontecimiento de control no se asocia con el paso regular del tiempo, por lo que el sistema no puede anticiparlo utilizando un reloj interno. Y a diferencia de un acontecimiento de flujo normal, el de control no indica su presencia con el arribo de datos.

El flujo de control puede considerarse como un flujo de datos binario: está activado o desactivado, y puede cambiar de un estado a otro en cualquier momento, señalando así al sistema que se necesita tomar alguna acción inmediata.

6.1.2. El modelo de comportamiento

Es un modelo que el sistema debe tener para manejar con éxito el ambiente. Esto involucra el desarrollo de un diagrama de flujo de datos y un diagrama de entidad-relación, además de la elaboración del diccionario de datos.

Ningún diagrama debe considerarse el dominante, de modo que controle al otro; cada uno es equivalente y puede proporcionar asistencia invaluable al otro.

6.1.2.1. Modelado de las funciones del sistema: El diagrama de flujo de datos

Es una herramienta que permite visualizar un sistema como una red de procesos

funcionales, conectados entre sí por "conductos" y "tanques" de almacenamiento de datos. Muestra con más detalle las funciones del sistema a partir del diagrama de contexto, que es la forma más general del diagrama de flujo. Estos diagramas consisten en procesos, almacenes de datos, flujos y terminadores.

- ***Los procesos*** se representan por medio de círculos o "burbujas" en el diagrama. Representan las diversas funciones individuales que el sistema lleva a cabo. Las funciones transforman entradas en salidas.

- ***Los flujos*** se muestran por medio de flechas curvas. Son las conexiones entre procesos, terminadores y almacenes de datos, representan la información que dichos procesos requieren como entrada o la información que generan como salida.

- ***Los almacenes de datos*** se utilizan para modelar un conjunto de paquetes de datos en reposo, se representan por medio de dos líneas paralelas o mediante una elipse. Muestran conjuntos (o almacenes) de datos que el sistema debe recordar por un período de tiempo. Cuando los diseñadores de sistemas y los programadores terminen de construir el sistema, los almacenes existirán como archivos o bases de datos.

- ***Los terminadores*** muestran las entidades externas con las que el sistema se comunica. Típicamente se trata de individuos o grupos de personas (por ejemplo, otro departamento o división dentro de la organización), sistemas de cómputo externos y organizaciones externas.

Aunque el diagrama de flujo de datos proporciona una visión global bastante conveniente de los componentes funcionales del sistema, no da detalles de éstos. Para mostrar detalles acerca de los datos se ocupa una herramienta textual de modelado adicional: el diccionario de datos.

6.1.2.2. El modelado de datos almacenados: El diagrama entidad-relación

Pese a que el diagrama de flujo de datos es una herramienta muy útil para modelar sistemas, sólo resalta un aspecto principal de un sistema: sus funciones. La notación de los almacenes de datos en los diagramas de flujo de datos muestra la existencia de uno o más grupos de datos almacenados, pero deliberadamente dice muy poco acerca de sus detalles.

Todos los sistemas almacenan y usan información acerca del ambiente en el cual interactúan; a veces la información es mínima, sin embargo, en la mayoría de los sistemas actuales es bastante compleja. No sólo deseamos conocer en detalle qué información hay en cada almacén de datos, sino que también la relación que existe entre almacenes. Este aspecto del sistema no es resaltado por el diagrama de flujo de datos, pero sí lo hace otra herramienta: el diagrama de entidad-relación.

Es un modelo de red que describe con un alto nivel de abstracción la distribución de datos almacenados en el sistema. Es importante modelar los datos de un sistema, pues las estructuras de datos y las relaciones pueden ser complejas, por tal motivo, es importante enfatizarlas y examinarlas independientemente del proceso que se lleva a cabo.

El diagrama de entidad-relación consta de tres componentes principales:

1. ***Tipos de objetos***. Se representa por medio de un rectángulo. Muestra una colección o conjunto de objetos (cosas) del mundo real, cuyos miembros juegan algún papel en el desarrollo del sistema; pueden además ser identificados de manera única y ser descritos por uno o más atributos.

2. ***Relaciones***. Se representan por medio de rombos, y son la serie de conexiones o asociaciones entre los tipos de objetos que están conectados con la relación por medio de flechas.

3. ***Indicadores asociativos de tipo objeto***. Representa algo que funciona como objeto y como relación. Otra manera de ver esto es considerar que el tipo asociativo de objeto representa una relación acerca de la cual se desea mantener alguna información.

6.1.2.3.El diccionario de datos

El diccionario de datos es un listado organizado de todos los datos pertenecientes al sistema, con definiciones precisas y rigurosas, para que tanto el usuario como el analista tengan un entendimiento común de todas las entradas, salidas, componentes de los almacenes de datos y cálculos intermedios. El diccionario de datos define los datos haciendo lo siguiente:

- Describe el significado de los flujos y almacenes que se muestran en los diagramas de flujo de datos.
- Describe la composición de paquetes de datos que se mueven a lo largo de los flujos, es decir, paquetes complejos que pueden descomponerse en unidades más elementales.
- Describen la composición de los paquetes de datos en los almacenes.
- Especifica valores y unidades relevantes de piezas elementales de información en los flujos de datos y en los almacenes de datos.
- Describe los detalles de las relaciones entre almacenes que se enfatizan en un diagrama de entidad-relación.

7. BASES DE DATOS

Dentro de la tecnología informática, las técnicas de bases de datos han aumentado su importancia y sus aplicaciones en los últimos años. La razón para ello ha estado en la demanda, siempre creciente de la utilización de grandes masas de datos, frecuentemente integrados y con requisitos de alta productividad en su manejo, tanto en el desarrollo de aplicaciones por programadores profesionales, como en la parte de los usuarios finales. Las bases de datos han respondido a esta necesidad, permitiendo estructurar éstos con técnicas más formalizadas, de uso más sencillo y con mayor capacidad para reflejar su significado.

En la búsqueda de estos objetivos, las bases de datos relacionales han supuesto un avance fundamental. Sus principios teóricos se establecieron y consolidaron en la década de los setenta, durante los ochenta se pasó del ámbito de la investigación al de las realizaciones prácticas, expandiéndose rápidamente sus aplicaciones. Al mismo tiempo continuó la investigación teórica, pues los conceptos relacionales forman una base sólida para la exploración de nuevas áreas, por ejemplo, el enriquecimiento semántico del modelo de datos, las bases de datos distribuidas, nuevos tipos de datos (gráficos, voz, imágenes, etc.), el uso de técnicas de inteligencia artificial para formar bases de conocimiento, entre otras.

7.1. SISTEMA DE BASE DE DATOS

¿Qué es exactamente un sistema de bases de datos? En esencia, no es más que un sistema de mantenimiento de registros basado en computadoras, es decir, un sistema cuyo propósito general es registrar y mantener información. Tal información puede estar relacionada con cualquier cosa que sea significativa para la organización donde el sistema opera, en otras palabras, cualquier dato necesario para los procesos de toma de decisiones inherentes a la administración de esa organización. Un sistema de bases de datos incluye cuatro componentes principales: *datos, software, hardware* y *usuarios*. A continuación se presenta un análisis breve de cada uno de ellos.

7.1.1. Datos

Aquellos datos que se obtienen de las personas, de lugares o de eventos de la realidad, eventualmente serán almacenados en archivos o bases de datos. Por tanto, la información que en un momento dado resulta ser valiosa para quienes la manejan, son precisamente los datos.

Los datos almacenados en el sistema se dividen en una o más bases de datos. Desde el punto de vista didáctico es más conveniente suponer que sólo hay una base de datos, la cual contiene todos los datos almacenados en el sistema. Una base de datos debe tener una

finalidad. Probablemente ésta sea resolver un problema, poner los datos a disposición de aquellos que los necesitan o presentarlos de manera que posibilite decisiones más informadas. Sea cual fuere el propósito, éste es el que dará forma a la base de datos. En general, es tanto integrada como compartida.

Por "integrada" se entiende que la base de datos puede considerarse como una unificación de varios archivos de datos independientes, donde se elimina parcial o totalmente cualquier redundancia entre los mismos, en otras palabras, se trata de reducir al mínimo el almacenamiento repetido de datos en la misma base de datos.

Por "compartida" se entiende que partes individuales de la base de datos pueden utilizarse por varios usuarios distintos, en el sentido de que cada uno de ellos puede tener acceso a la misma parte de la base de datos con propósitos diferentes. Y como la base de datos es integrada resulta como consecuencia ser compartida.

Se debe mantener la seguridad de la información cuando múltiples usuarios intenten el acceso a los mismos datos simultáneamente. Esto significa que la información debe permanecer completa y precisa todo el tiempo (integridad de la información). Además diferentes usuarios percibirán de modos muy distintos una base de datos específica (aunque dos usuarios compartan la misma información de la base de datos, sus percepciones o vistas de tal información pueden diferir mucho a nivel de detalle).

7.1.2. Hardware

El hardware se compone de los volúmenes de almacenamiento secundario (discos, cintas, unidades ópticas, cartuchos, etc.), donde reside la base de datos, junto con dispositivos asociados como las unidades de control, los canales de comunicación, etc.

7.1.3. Software

Se necesita un sistema que integre los archivos en una base de datos y que pueda proporcionar diversas orientaciones a usuarios diferentes. El software, el hardware y los procedimientos para manejar la base de datos conforman un *sistema de manejo de la base de datos* (DBMS: Data Base Management System). Un sistema de manejo de la base de datos hace posible accesar datos integrados que cruzan límites operacionales, funcionales u organizativos dentro de una empresa. Éste maneja todas las solicitudes de acceso a la base de datos formuladas por los usuarios. Una función general del DBMS, por tanto, es proteger a los usuarios de la base de datos contra los detalles a nivel de hardware. En otras palabras, el DBMS ofrece una vista de la base de datos que está por encima del nivel de hardware y apoya las operaciones del usuario que se expresan en términos de esa vista de nivel superior.

7.1.4. Usuarios

Se consideran tres clases generales de usuarios. La primera la representa el *programador de aplicaciones,* encargado de escribir programas de aplicación que utilicen bases de datos. Estos programas de aplicación operan con los datos de todas las maneras usuales: recuperan información, crean información nueva, suprimen o cambian información existente, etc. (todas estas funciones se realizan formulando las solicitudes adecuadas al DBMS.) Los programas pueden ser aplicaciones convencionales de procesamiento por lotes o programas en línea diseñados para apoyar a un usuario final, que interactúa con el sistema desde una terminal en línea.

La segunda clase de usuario es el *usuario final* que accesa la base de datos desde una terminal. Un usuario final puede emplear un *lenguaje de consulta* proporcionado como parte integral del sistema o recurrir a un programa de aplicación escrito por un usuario programador que acepte órdenes desde la terminal y a su vez formule solicitudes al DBMS en nombre del usuario final. De cualquier manera, el usuario final puede realizar en general, todas las funciones de recuperación, creación, supresión y modificación, aunque tal vez se pueda afirmar que la recuperación es la función más común de esta clase de usuario.

La tercera clase de usuario la representa el *administrador de bases de datos* o DBA (por sus siglas en inglés).

7.1.4.1. Administrador de la base de datos

En el sistema de bases de datos están involucrados muchos usuarios. Es absolutamente necesaria una función que pueda analizar las diferentes necesidades y resolver conflictos de intereses. Se debe establecer una función administrativa permanente para coordinar y llevar a cabo todos los pasos de diseño, implantación y mantenimiento de una base de datos integrada.

Así, la administración de la base de datos, es una función compuesta por gente responsable de proteger un valioso recurso: los datos. En el medio convencional del procesamiento de datos, un programador de aplicaciones "posee" un archivo de datos. Los usuarios "protegen" sus datos para evitar que otros los usen, forzándolos a recopilar los mismos datos. La era de las bases de datos ha eliminado la idea de "propiedad" individual. A la persona encargada de la función de administración de la base de datos se le llama *administrador de la base de datos.* El administrador de la base de datos no es el "propietario" de los datos sino el "protector" de ellos.

En una institución la generación de información y los procesos de decisión se complican a medida que aumenta la gama de funciones. Y si al programador de aplicaciones se le "quita" el control directo sobre los datos, pierde el sentido de contacto personal y responsabilidad por estos. Esta falta de contacto obliga a la empresa a desarrollar procedimientos para asegurar que la integridad de los datos no quede comprometida. Este objetivo deberá estar coordinado

con la función de administración de la base de datos.

La administración de la base de datos es una función que proporciona servicios a los usuarios de la base de datos. Se podría establecer una analogía entre el administrador de la base de datos y el contralor de la empresa. El contralor protege el recurso de la empresa llamado "dinero" y el DBA protege el recurso llamado "datos".

El DBA tiene que coordinar las funciones de recopilación de información acerca de los datos y diseñar, implantar y mantener la base de datos y su seguridad. La misión del DBA no termina con la implantación de la base de datos, ya que la integración de nuevas funciones en la base de datos lo hacen más necesario que nunca. Una de sus principales funciones consiste en poner atención tanto a las futuras como a las presentes necesidades de información de la empresa. Para llevar a cabo esta función, el diseño de la base de datos deberá ser tan flexible, o independiente de los datos, como sea posible.

La función del DBA es claramente significativa para las operaciones de un sistema de base de datos multiusuarios. Sin embargo, no se debe descartar la posibilidad de que algunas de estas funciones algún día se automaticen. Tampoco se debe olvidar la importancia futura de las bases de datos "personales" sobre las cuales el usuario tiene el control total. Incluso con un solo usuario, es posible tener diferentes "enfoques" de los datos o hacer consultas no previstas en el procedimiento.

7.2. VENTAJAS DE TENER UN CONTROL CENTRALIZADO DE LOS DATOS

Un sistema de bases de datos proporciona a la empresa un control centralizado de sus datos que constituyen uno de sus activos más valiosos.

Consideremos ahora algunas de las ventajas de tener un control centralizado de los datos.

- **Puede reducirse la redundancia**. La redundancia debe controlarse, es decir, el sistema debe estar al tanto de ésta y asumir la responsabilidad de propagar las actualizaciones.

- **Puede evitarse la inconsistencia** (los datos almacenados deberán ser correctos y confiables). Desde luego, una base de datos que se halle en estado de inconsistencia puede suministrar información incorrecta o contradictoria.

- **Los datos pueden compartirse**. No sólo significa que las aplicaciones existentes pueden compartir los datos de la base de datos, sino también que es factible desarrollar nuevas aplicaciones que operen con los mismos datos almacenados. En otras palabras, las necesidades de datos de las nuevas aplicaciones pueden atenderse sin tener que crear nuevos archivos almacenados.

- **Pueden hacerse cumplir las normas establecidas**. Con un control central de la base de datos, el DBA puede garantizar que se cumplan todas las formas aplicables a la representación de los datos. Las normas aplicables pueden comprender la totalidad o parte de lo siguiente: de la instalación, departamentales, políticas de la compañía, industriales, nacionales o internacionales. Es muy deseable unificar los formatos de los datos almacenados como ayuda para el intercambio o migración de datos entre sistemas.

- **Pueden aplicarse restricciones de seguridad**. Al tener jurisdicción completa sobre los datos de operación, el DBA puede asegurar que el único medio de accesar la base de datos sea a través de los canales establecidos y por tanto, definir controles de autorización para que se apliquen cada vez que se intente el acceso a datos sensibles.

  Diferentes controles pueden establecerse para cada tipo de acceso (recuperación, modificación, supresión, etc.) a cada parte de la información de la base de datos. Nótese que sin estos controles la seguridad de los datos, corre mayor peligro en un sistema de bases de datos que en un sistema tradicional (de archivos dispersos).

- **Puede conservarse la integridad**. El problema de la integridad es garantizar que los datos de la base de datos sean exactos. La inconsistencia entre dos entradas que representan al mismo "hecho" es un ejemplo de falta de integridad (que por supuesto, sólo ocurre si existe redundancia en los datos almacenados). Por ejemplo: aun cuando la redundancia se elimine la base de datos puede contener aún datos incorrectos.

  El control centralizado de la base de datos ayuda a evitar estas situaciones en la medida de lo posible, pues permite al DBA definir procedimientos de validación que habrán de ejecutarse cada vez que se intente una operación de actualización (el término "actualizar" abarca las operaciones de modificación, creación y supresión).

  Es conveniente señalar que la integridad de los datos es más importante en un sistema de bases de datos que en un sistema de "archivos privados", precisamente porque el primero se comparte y porque sin procedimientos de validación adecuados es posible que un programa con errores genere datos incorrectos que afecten a otros programas que utilicen esa información.

- **Pueden equilibrarse los requerimientos contradictorios**. Cuando se conocen los requerimientos globales de la empresa —en contraste con los requerimientos de cualquier usuario individual—, el DBA puede estructurar el sistema de bases de datos para brindar un servicio que sea mejor para la empresa en términos globales.

La mayoría de las ventajas recién enumeradas son muy claras; sin embargo, otro aspecto que no es tan evidente —aunque se desprende de varios de los anteriores—, debe añadirse a la lista: *la independencia de los datos*.

Independencia de los datos

Toda la información concerniente a las necesidades de los usuarios, así como la información sobre las necesidades futuras, se deberá reflejar en un buen diseño de la nueva base de datos. Por supuesto que no se pueden predeterminar todos los posibles usos de los datos, sin embargo, en la mayoría de los casos los datos básicos son relativamente estables. Lo que suele cambiar son las necesidades de información, es decir, las formas en que los datos se utilizan para producir información.

La capacidad para utilizar la base de datos sin conocer los detalles de cómo se representan o se localizan en la base de datos se conoce como *independencia de los datos.*

Un diseño ideal de la base de datos debe permitir (hasta cierto grado), cambios de esta naturaleza que no afecten los programas de aplicación. La independencia de datos proporciona solución a estos problemas, ya que:

- Permite al DBA hacer cambios de contenido, localización, representación y organización en la base de datos, sin necesidad de volver a escribir los programas de aplicación que utilizan la base de datos.
- Permitir al proveedor de equipo de procesamiento de datos y de software introducir nuevas tecnologías sin que se tenga que reprogramar la aplicación del cliente.
- Facilitar el compartimiento de datos al permitir que parezca que los mismos datos están organizados de manera diferente para los diversos programas de aplicación.
- Simplificar el desarrollo de programas de aplicación y en particular, facilitar el desarrollo de programas para procesamiento interactivo con base de datos.
- Proporcionar la centralización de control que necesita el administrador de la base de datos para garantizar seguridad e integridad de la base de datos.

Es importante tener en cuenta que la independencia de los datos depende tanto del diseño de la base de datos como del sistema de manejo de la base de datos (DBMS).

7.3. DESVENTAJAS DE TENER UN CONTROL CENTRALIZADO

Aunque tiene ventajas importantes el control centralizado de los datos, es fundamental conocer las desventajas que puede acarrear. Como los datos provienen de archivos individuales que se integran en una base de datos, se pierde fácilmente el sentido de propiedad y como resultado, la responsabilidad por los datos. Como consecuencia, los datos inexactos pueden pasar inadvertidos. Esto puede causar serios problemas a menos que se tomen medidas extensivas para la integridad y la validez de los datos.

La base de datos también se puede convertir en blanco de las fallas de seguridad a menos que se mantenga una disciplina estricta. Más aún, una base de datos puede acrecentar conflictos políticos y organizativos dentro de la organización debido a que sirve a las necesidades de múltiples usuarios, algunos de los cuales pueden tener conflictos legítimos de intereses y diferentes necesidades.

Una base de datos integrada también puede amenazar la privacidad. Tradicionalmente la privacidad se ha definido como el derecho de una persona a su individualidad. En el ambiente de una base de datos integrada resulta fácil recopilar información acerca de gente y organizaciones, y después exponer esa información a alguna persona u organización no autorizada. Por tanto, si el sistema de manejo de la base de datos no asegura integridad, seguridad y privacidad adecuadas, y si la base de datos no está diseñada apropiadamente, el uso de la base de datos puede ser causa u origen de nuevos problemas.

Para evitar estos inconvenientes potenciales podemos decir que un sistema de manejo de la base de datos (DBMS) debe satisfacer los siguientes objetivos:

- Que el mismo DBMS atienda de manera efectiva las diferentes funciones de la empresa.
- Minimizar la cantidad de redundancia en los datos almacenados.
- Suministrar información consistente al proceso de toma de decisiones.
- Aplicar controles de seguridad.
- Desarrollar, mantener y mejorar los programas de aplicación de manera más rápida y económica, con menos personal calificado.
- Facilitar la reorganización física de los datos almacenados.
- Posibilitar el control centralizado de la base de datos.

7.4. MODELOS DE DATOS

A la hora de definir una base de datos se debe establecer un proceso partiendo del acotamiento de una parcela del mundo exterior, aquél que nos interesa representar en los datos. En este proceso se debe aprender, comprender y conceptualizar dicho mundo exterior transformándolo en un conjunto de ideas y definiciones que supongan una imagen fiel del comportamiento del mundo real. A esta imagen del mundo exterior se le llama *"modelo conceptual"*.

Una vez definido el modelo conceptual, éste se ha de transformar en una descripción de datos, características y relaciones que se denomina "esquema conceptual de datos".

Por último, este esquema conceptual habrá que traducirlo a estructuras almacenables en soportes físicos.

Resumiendo lo expuesto, podemos por tanto decir que mediante un proceso de abstracción, pasaremos del mundo real al mundo de las ideas estableciendo un modelo conceptual, y a partir de éste, a través de un proceso de organización, pasaremos del mundo de las ideas al de los datos, estableciendo así un modelo convencional de datos.

El proceso en cualquier caso, no necesariamente es éste, pudiéndose pasar directamente del mundo real al mundo de los datos. Ello depende tanto de la complejidad del problema a tratar como de la capacidad y experiencia del sujeto que lo realiza.

El modelo de datos se puede definir como un grupo de herramientas conceptuales para describir los datos, sus relaciones, su semántica y sus limitaciones; de tal forma, que facilita la interpretación de nuestro mundo real y su representación en forma de datos en nuestro sistema informático.

7.4.1. Estructura del modelo de datos

Definido el modelo de datos, es pertinente analizarlo. Para ello, se partirá de las propiedades del mundo real, que se pueden clasificar en dos tipos:

- ***Estáticas:*** Son las propiedades invariantes en el tiempo. Estas son especificadas en el modelo de datos por las *estructuras*.
- ***Dinámicas:*** Son las propiedades que varían con el tiempo. En el modelo de datos son las *operaciones*.

El modelo de datos contiene, por tanto, propiedades estáticas y dinámicas.

A continuación se analizarán con mayor profundidad cada una de las partes.

- ***Parte estática.*** Ésta se define mediante el esquema, con el lenguaje de definición de datos (DDL).

  El esquema, a su vez está constituido por estructura y restricciones.

  La estructura se define por medio de los objetos del modelo y las restricciones inherentes, conformando un conjunto de reglas de definición de dichas estructuras.

  Los objetos y restricciones de la estructura dependen de cada modelo, pero en general son:

  - Entidades.

- Atributos.
- Relaciones.
- Dominios.
- Representación.
- Restricciones inherentes.

Los cuales se explicarán más adelante.

Las restricciones inherentes están impuestas por la propia naturaleza del modelo introduciendo rigideces en el modelado.

Las restricciones opcionales o de usuario, restricciones propiamente dichas en el esquema, son definidas por el usuario, pero el modelo de datos las reconoce y suministra herramientas para manejarlas.

Las restricciones libres de usuario, por último, son responsabilidad del usuario y el modelo de datos ni las reconoce, ni las maneja.

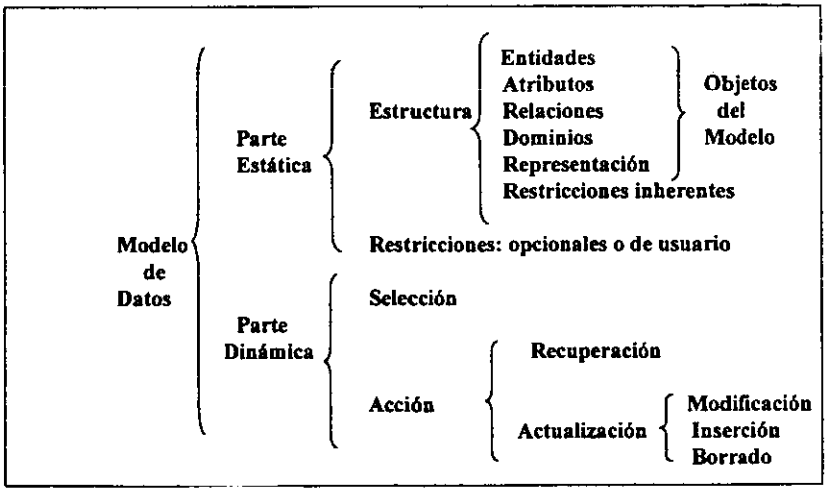
- ***Parte dinámica.*** Se define como un conjunto de operaciones con el lenguaje de manipulación de datos (DML).

Las operaciones sobre un modelo de datos pueden ser de:

- Selección. Localización de los datos deseados.
- Acción. Realización de una acción sobre los datos seleccionados. Dicha acción puede ser:
- Recuperación (obtención de los datos seleccionados) y
- Actualización, que a su vez puede ser:
  - Modificación
  - Inserción
  - Borrado

Generalmente, toda operación de actualización va precedida de una de recuperación, aunque no necesariamente.

En el cuadro siguiente se resume en forma concreta lo expuesto anteriormente.



7.4.2. Clasificación de los modelos

Los modelos de datos se aplican a tres niveles: externo, conceptual e interno. Los correspondientes a los dos primeros son "modelos de datos lógicos" y los correspondientes al último "modelo de datos físicos".

Dentro del grupo de los modelos lógicos, podemos diferenciar:

- *Los modelos lógicos basados en objetos* en cuyo caso el elemento de referencia es el objeto, entendido como tal a aquél que existe y puede distinguirse de otros.
- *Los modelos lógicos basados en registros.* En este caso, el elemento básico es una ocurrencia o conjunto de datos relacionados de algún modo.

7.4.2.1. Modelos lógicos basados en objetos

Los modelos lógicos basados en objetos se utilizan para describir los datos en los niveles conceptual y externo.

Se caracterizan por ser muy flexibles y hacer posible la especificación de las limitaciones de los datos.

Algunos de estos modelos son:

- Modelo Entidad-Relación.
- Modelo Binario.
- Modelo Semántico de datos.
- Modelo Infológico.

7.4.2.2. Modelos lógicos basados en registros

Se utilizan para describir los datos a nivel conceptual y externo al igual que el modelo anterior.

Estos modelos sirven para especificar, tanto la estructura lógica general de la base de datos, como una descripción en un nivel más alto de la implementación, pero no permiten especificar de forma clara las limitaciones de los datos.

Modelos de este tipo son:

- Modelo Jerárquico.
- Modelo en Red.
- Modelo Relacional.

En esta tesis utilizaremos el análisis estructurado, en donde el modelo relacional es parte esencial. A continuación se describe este tipo de modelo.

7.4.3. Modelo Relacional

El modelo de base de datos relacional ha recibido considerable atención en los últimos años, pues ofrece muchas ventajas sobre los modelos jerárquico y de red, tales como:

- La representación usada en el esquema relacional es fácil de comprender por los usuarios y fácil de implementar en el sistema físico de datos.
- El control de acceso a datos sensibles es de implementación inmediata.
- Las búsquedas pueden ser mucho más rápidas que en los sistemas que deben seguir una cadena.
- Las estructuras relacionales son mucho más fáciles de modificar que las estructuras jerárquicas o de red.

- Es más fácil buscar datos tabulados que desenrrollar posibles interconexiones arbitrarias y complejas de elementos de datos.

El modelo relacional fue definido en 1970 por E. F. Codd. En este modelo, tanto las entidades como las relaciones se representan mediante tablas.

Parte Estática

La parte estática del modelo —como se apuntó anteriormente— está constituida por los objetos y las restricciones.

Entre los objetos se encuentran:

7.4.3.1. Entidades

Una entidad es un objeto concreto o abstracto que va a ser representado en un sistema de base de datos. Cuando se crea el modelo entidad-relación se definen las entidades que forman el mundo de la situación real que se tiene que implementar en una base de datos. Por ejemplo: el objeto almacén, fruta y factura son entidades.

Una entidad puede ser de dos tipos:

- **Entidad fuerte o propia:** es aquélla cuyas propiedades que la identifican sólo hacen referencia a la propia entidad. Por ejemplo, coche, alumno, matrícula, almacén y fruta son entidades fuertes puesto que los atributos que las identifican (nombre-alumno, número-matrícula, código-almacén, descripción-fruta, etc.) sólo hacen referencia a la propia entidad.

- **Entidad débil o regular:** es aquélla que sólo tiene sentido gracias a las propiedades que identifican a otras entidades (fuertes o a su vez débiles). Por ejemplo, factura se identifica gracias a las propiedades de un almacén que emite la factura

7.4.3.2. Atributos

Un atributo es una característica de una entidad. Es el concepto de campo en la nomenclatura tradicional de archivos. Por ejemplo, la dirección del almacén, el nombre de la fruta y la fecha de la factura son atributos.

Un atributo se identifica con un nombre (que puede encontrarse en varias entidades) y todos los posibles valores que puede tener (dominio).

7.4.3.3. Dominios

El dominio es el conjunto de todos los posibles valores para uno o más atributos. Por tanto, los valores contenidos en un atributo pertenecen a un dominio que previamente se define.

Pueden distinguirse dos tipos de dominios diferentes:

- **Dominios generales o continuos:** aquéllos que contienen todos los posibles valores entre un máximo y un mínimo predefinido. Por ejemplo:
  - Número de cuenta: todos los números enteros y positivos de ocho dígitos.
  - Peso del material: todos los números reales y positivos.
  - Fecha de nacimiento: todas las fechas desde principio de siglo hasta hoy.
- **Dominios restringidos o discretos:** aquéllos que contienen ciertos valores específicos entre un máximo y un mínimo predefinido. Por ejemplo:
  - Estado civil: casado, viudo, divorciado.
  - Ciudad: una de las ciudades mexicanas.

7.4.3.4. Tablas

Una forma sencilla de estructurar los datos es mediante tablas, es decir, organizados en filas y columnas o lo que es equivalente tuplas y atributos para incidir en la terminología relacional. Es un método muy popular que se utiliza por su simplicidad.

Para que una tabla forme parte de una estructura relacional, la tabla debe cumplir las siguientes condiciones:

- Debe tener un sólo tipo de tupla cuyo formato queda definido por el esquema de la tabla o la relación. Por tanto, todas las tuplas tienen los mismos atributos.
- Cada tupla debe ser única y no pueden existir tuplas duplicadas.
- Cada atributo debe ser único y no pueden existir atributos duplicados.
- Cada atributo debe estar identificado por un nombre específico.
- El valor de un atributo para una tupla debe ser único. Por tal motivo, no pueden existir múltiples valores en la posición de un atributo.

- Los valores de un atributo deben pertenecer al dominio que representa, siendo posible que un mismo dominio se utilice para definir los valores de varios atributos.

Una tabla que cumpla estas condiciones se denomina **tabla relacional** o **relación.** El concepto relación se utiliza generalmente para indicar que en la tabla relacional se mantiene la asociación de otras tablas y por tanto representa a una entidad asociativa. El concepto tabla relacional se utiliza para expresar que los atributos que contiene no relacionan objetos fuertes y representativos, siendo la tabla por tanto, la que representa un objeto o entidad propia.

Una tabla que cumpla las condiciones anteriores tiene asociadas las siguientes propiedades:

- Las tuplas pueden estar en cualquier orden.
- A una tupla se le hace referencia mediante todos los valores que la forman.
- Los atributos pueden estar en cualquier orden.
- Se hace referencia a un atributo mediante el nombre que lo identifica.

Se denomina **Grado** de una tabla relacional al número de atributos que lo forman. De ese modo, si en una tabla se tienen tres atributos: G(tabla) = 3.

Se denomina **Cardinalidad** de una tabla relacional al número de tuplas que contiene. Por tanto, en una tabla con 5 tuplas: C(tabla) – 5.

7.4.3.5. Llaves

En una tabla relacional es necesario poder determinar una tupla concreta, lo cual es posible mediante la llave. Llave es un atributo o conjunto de atributos cuyos valores distinguen unívocamente una tupla específica de una tabla.

Como una de las condiciones de una tabla relacional es que no pueden existir tuplas duplicadas, ello implica que siempre tiene que existir al menos una llave, que en el peor de los casos, estará formada por todos los atributos.

Para buscar la llave de una tabla relacional se consideran los dominios de los atributos junto con los enlaces conceptuales existentes entre ellos, de tal modo, que tomando valores determinados se identifique una única tupla de la tabla. No se debe buscar la llave entre los valores de tuplas concretas, sino a través de todos los posibles valores (dominio) de los atributos.

Es posible que en una tabla, más de un atributo (o combinación de ellos) puedan servir de llave.

Si una tabla dispone de varias llaves, a éstas se las denomina **llaves candidatas,** puesto que

posteriormente entre todas ellas se tendrá que elegir una que será la que identifique la tupla. A esta llave se la denomina **llave principal o primaria** y al resto **llaves alternativas o secundarias**. Los atributos que pertenecen a la llave primaria se denominan atributos primarios y el resto atributos no primarios o secundarios.

Para elegir la llave principal se considera fundamentalmente el dominio, puesto que en la futura administración de la base de datos la llave principal se utilizará para acceder y relacionar otras tablas. Cuando se analice la llave principal se tendrán en cuenta sobre su dominio las siguientes consideraciones:

- Sus valores siempre deben ser conocidos (diferentes de nulos).
- La memoria que ocupen debe ser mínima.
- Su codificación debe ser sencilla.
- El contenido de sus valores no debe variar.
- Que se utilice en otras tablas para crear una interrelación (mediante llaves foráneas).

Las llaves se utilizan para en un futuro definir índices sobre ellas. Los índices son las formas de acceso. El *índice primario* (generalmente obligatorio) estará formado por la llave primaria (que puede ser **simple**: formada por un solo atributo o **compuesta**: formada por varios atributos) y los *índices secundarios* —si los hay— lo formarán las llaves secundarias u otros atributos que nos interesen. Estos últimos índices sólo se usan si en transacciones *on-line* (con tiempo de respuesta limitado) se precisa acceder a las tuplas por otros atributos. Esto generalmente sólo se aplica para realizar consultas diversas.

Se denomina **llave foránea** a aquel atributo o conjunto de atributos que en la tabla donde se encuentran no son llave, y sus valores se corresponden con la llave principal de la misma u otra tabla. El dominio de la llave foránea y de la llave principal de la tabla a la que hace referencia se tiene que corresponder, es decir, ser compatible.

Las llaves foráneas son un concepto muy importante a tener en cuenta, puesto que la integridad de la base de datos se mantiene o se elimina en gran parte mediante las transacciones que se realizan sobre dichas llaves.

7.4.3.6. Interrelaciones

Una interrelación es una asociación entre tablas mediante atributos que tienen el mismo dominio (o compatible). Estos atributos generalmente hacen referencia a los mismos conceptos y la interrelación se establece entre la llave foránea de una tabla (tabla hija) y la llave principal de la otra tabla (tabla padre).

Por ejemplo, se tienen dos tablas y entre ellas existe una interrelación entre una tabla

llamada "personal" (la cual es la tabla padre, con un identificador "dni" como llave principal) y otra llamada "coches-personal" (tabla hija, suponiendo que el "dni" forma parte de la tabla). Esta interrelación se caracteriza por:

- Interrelación: propiedad-coche
- Tabla padre: personal
- Tabla hija: coches-personal
- Llave foránea: dni

Las interrelaciones se identifican con un nombre (en nuestro caso "propiedad-coche").

7.4.3.7. Vistas

Las tablas que hemos supuesto hasta ahora son tablas reales, cuyo esquema o definición, así como sus tuplas o datos, están almacenados físicamente en memoria, es decir, son tablas que están implementadas directamente dentro de una base de datos. Son las tablas base.

Además de estas tablas, el usuario puede crear y manejar (de forma limitada con algunas restricciones según el DBMS que se use) otro tipo de tablas. Son tablas ficticias cuya definición y tuplas se obtienen a partir de una o más tablas base. Son tablas que obtienen vistas parciales de datos para que usuarios sólo accesen a determinada información.

Las tablas vistas o simplemente vistas, tienen las siguientes características:

- Sus atributos se obtienen a partir de múltiples tablas base, e incluso pueden estar calculadas a partir de valores de las tablas base.
- Pueden estar definidas a partir de otras vistas.
- Sus datos se obtienen como resultado de realizar operaciones de recuperación (lectura) de datos.
- Se puede almacenar su definición (esquema o estructura) para una utilización posterior.

Por tanto, la vista es una tabla virtual que no existe en realidad como una tabla base en memoria auxiliar (disco, por ejemplo), sólo se almacena —si se desea— su definición, donde se muestra cómo se obtiene a partir de tablas base mediante una sucesión de operaciones. Es una forma de ver determinados datos de tablas base. Por tanto, tal y como se manipulen las tablas base con modificaciones, inserciones y borrados, así quedarán afectadas las vistas que se tengan sobre ellas. Es importante resaltar que los datos que se obtienen de una vista no son datos duplicados.

Las operaciones de actualización (el término actualización indica las tres operaciones fundamentales de variación de datos: modificación, inserción y borrado) de datos sobre vistas, debido a las implicaciones que pueden tener sobre las tuplas de las tablas base y el resto de vistas definidas sobre ellas, obligan a que la propia definición de la vista tenga sus propias restricciones para la realización de actualizaciones. Estas restricciones son diferentes según el sistema de administración de la base de datos (DBMS) que se use.

Una vista que admita las operaciones de actualización debe estar limitada por las siguientes restricciones:

- Debe estar derivada de una sola tabla base.
- Cada tupla distinta de la vista se debe corresponder con una única tupla de la tabla base.
- A cada atributo distinto de la vista le debe corresponder un único atributo de la tabla base (no puede tener atributos calculados).

Diversos diseñadores de DBMS estudian y analizan continuamente situaciones donde las normas a pesar de ser válidas, el sistema no debe admitir determinados tipos de actualización.

En otros sistemas se deja libertad al administrador de la base de datos de ser el que se responsabilice de las operaciones que se permitan realizar sobre una vista, y el DBMS no tiene restricciones determinadas.

Las principales ventajas que ofrecen las vistas son:

- La visión de los datos está simplificada de cara al usuario.
- Los mismos datos pueden verse de diferente manera desde distintos usuarios.
- Las vistas no quedan afectadas después de:
  - Aumento de otras tablas.
  - Aumento de la estructura con nuevos atributos.
  - Reestructuración de la posición de los atributos de una tabla.
- Se aumenta la seguridad para aquella información que no se desea mostrar.

Restricciones inherentes

El modelo relacional posee dos:

1. No pueden aparecer dos tuplas iguales en una misma relación (la cual es una de las condiciones más importantes de una tabla relacional).

2. El (los) atributo (s) que forma (n) parte de la llave, no puede tomar valores nulos (por nulos se interpreta valores desconocidos).

Restricciones Opcionales

Están formadas por las dependencias funcionales, dependencias transitivas, dependencias multivaluadas, etc. (que se verán en detalle posteriormente en normalización de bases de datos relacionales).

Parte Dinámica

Con respecto a la parte dinámica, el modelo relacional trabaja por especificación, es decir, se especifica una condición que debe cumplir una serie de tuplas (las que se quieren localizar). Es por tanto el lenguaje utilizado para la manipulación de datos un lenguaje No-Procedimental.

En resumen:

- Una base de datos relacional está formada por un conjunto de datos agrupados en relaciones.
- Estas relaciones se representan mediante tablas y contienen información homogénea.
- Los distintos ítems de información conforman tuplas y cada una de ellas se identifica de forma única mediante una llave.
- Cada tupla está formada por varios campos o atributos cuyos valores no se pueden descomponer.
- Los atributos se asignan a dominios, siendo un dominio un conjunto de valores posibles de un atributo.

A nivel físico, esta información se suele almacenar en forma de listas no secuenciales y más concretamente en cadenas compuestas.

Entre las bases de datos relacionales existentes en el mercado se pueden mencionar:

DB2
INGRES
ORACLE
INFORMIX
SYBASE

y para computadoras personales:

DBASE IV
FOXBASE
ACCESS

7.4.4. Algebra relacional

Puesto que la definición de las tablas parte del producto cartesiano, las operaciones definidas sobre éstas se basan en el álgebra relacional. Cada operación toma como operandos una o varias tablas y como resultado genera otra tabla. Esta tabla resultante puede someterse a nuevas operaciones y se pueden realizar sobre las tablas originales cualquier sucesión de operaciones relacionales.

Todas las operaciones están implementadas internamente dentro del DBMS como rutinas independientes, que se ejecutan cuando el usuario o un programa de una aplicación realiza una llamada a la base de datos para acceder a la información de una tabla.

De todas las operaciones a estudiar, se distinguen aquéllas que son **"básicas"** porque son independientes del resto de operaciones y aquéllas **"derivadas"** que son las que utilizan en su proceso las operaciones básicas.

7.4.4.1. Operaciones básicas

En las operaciones básicas se encuentran aquéllas que utilizan una sola tabla de entrada para obtener el resultado: operaciones unarias y aquéllas que utilizan como entrada dos tablas: operaciones binarias.

7.4.4.2. Operaciones básicas unarias

7.4.4.2.1. Selección

La selección obtiene un subconjunto de tuplas de una tabla con todos sus atributos, creando con este subconjunto una nueva tabla. Lógicamente, para definir el subconjunto de tuplas, debe existir algún criterio de selección que elimine al resto de las tuplas.

7.4.4.2.2. Proyección

La proyección obtiene un subconjunto de atributos de una tabla con todas sus tuplas, creando con este subconjunto una nueva tabla. Del mismo modo que en la operación anterior, para definir el subconjunto de atributos debe existir algún criterio. Aquellas tuplas que estén duplicadas sólo aparecerán una vez.

7.4.4.3. Operaciones básicas binarias

7.4.4.3.1. Unión

La unión de dos tablas sólo se puede realizar si tienen el mismo grado (número de atributos) y los dominios son compatibles (generalmente los mismos atributos). El resultado es una nueva tabla con los atributos de una de ellas y las tuplas de ambas tablas. Si existen tuplas repetidas éstas sólo aparecerán una vez, puesto que es condición a cumplir en una tabla relacional.

7.4.4.3.2. Diferencia

La diferencia entre dos tablas R y T, se representa R-T, y sólo se puede realizar si, al igual que sucede con la unión, tienen el mismo grado y los dominios son compatibles. El resultado es una nueva tabla con los atributos de una de ellas y las tuplas de R que no están en T.

7.4.4.3.3. Producto Cartesiano

También se denomina simplemente producto. Se puede realizar entre dos tablas R y S con grado diferente m y n. El resultado es una nueva tabla cuyo grado es la suma de los grados de las dos tablas operadores (m + n) y con todas las tuplas que resultan de concatenar las dos tablas, es decir, donde los m primeros elementos constituyen tuplas de R y los n últimos tuplas de S; por tanto, la cardinalidad de la nueva tabla será el producto de las cardinalidades de las tablas de entrada.

Si en la tabla resultante aparecen dos atributos nombrados de la misma forma, es decir, que cada una de ellas tengan un atributo con el mismo nombre, resultará una ambigüedad ya que los atributos se identifican por su nombre y no pueden existir dos atributos repetidos en una sola tabla.

7.4.4.4. Operaciones derivadas

7.4.4.4.1. Intersección

La intersección de dos tablas R y T es una operación que se deriva de la operación básica "diferencia". Esto indica que las tablas deben tener el mismo grado y el dominio compatible. Por tanto, la nueva tabla resultante tiene los atributos de una de ellas y las tuplas comunes a

ambas tablas.

7.4.4.4.2. Cociente

El cociente se realiza entre dos tablas Q y S con diferente grado, que cumplen las siguientes condiciones:

- Q debe tener los atributos de S, y por tanto, tener un grado mayor que S, estando el grado de S incluido en el de Q.
- La cardinalidad de S debe ser distinta de cero.

El resultado es una nueva tabla (los atributos de Q que no están en S) y las tuplas de Q, tales que al concatenarlas (hacer el producto cartesiano) con todas las tuplas de S producen tuplas contenidas en Q. De esta manera, la tabla resultante será igual o estará incluida en Q.

7.4.4.4.3. Join, unión natural

La operación de join, también denominada unión natural o yunción por diversos autores, es de las más importantes del álgebra relacional. Se obtiene de realizar el producto cartesiano y aplicar sobre la tabla resultante una selección preestablecida que se denomina selección o condición del join.

Por tanto, como operandos utiliza dos tablas que se confrontan (mediante un operador de igualdad, desigualdad, mayor, etc,) entre sí, tupla a tupla en los atributos que se especifiquen y que generalmente corresponderán a llaves foráneas. El resultado es una tabla con los atributos de las dos tablas y los valores de las tuplas que satisfacen la condición de comparación sobre los atributos definidos.

Los atributos que van a utilizarse para la comparación del join tienen que pertenecer al mismo dominio.

Si la comparación que se usa entre las dos tablas es la de igualdad sobre los atributos comunes, la operación se denomina **equijoin** y es la más utilizada.

Cuando se dice que la tabla resultante contiene los atributos de las dos tablas, no quiere decir que es la suma de los grados de las tablas que intervienen, sólo se pone un atributo, en el caso de que existan atributos similares en ambas. Es así, debido a que la operación al ser de igualdad los dos atributos estarían duplicados, y esto contrastaría con las condiciones que debe cumplir una tabla relacional. Si en la tabla resultado en una operación de equijoin se elimina uno de los atributos que se comparan por ser iguales, la operacion se denomina **join natural** o particularizando **equijoin natural**.

Los atributos del join son aquellos utilizados para la condición. De ese modo, los valores resultantes son coherentes y lógicos y se corresponden unos con otros.

Si la operación se realiza entre atributos que no guardan ninguna relación entre sí y la comparación fuese mayor (>), obtendríamos una tabla absurda en donde los valores no se corresponden y es imposible intentar relacionarlos. En esta operación la tabla resultante obtiene multitud de valores duplicados y redundantes y estos aumentan a medida que existan tuplas con el mismo valor en el atributo del join.

Hay que tener en cuenta que estas operaciones se han analizado por separado estudiando cómo operan y qué valores devuelven, pero son rutinas internas (funciones) que lleva el DBMS y que tras operar con unos argumentos de entrada devuelven un resultado. Estas rutinas se ejecutan cuando se realizan llamadas desde sentencias y el resultado generalmente siempre viene acompañado de una selección y una proyección por lo que los valores no se duplican tanto como parece. Además, la tabla resultante sólo se genera en memoria interna para visualizar sus datos u operar con ella, pero nunca se almacena en memoria auxiliar. Por lo tanto, los valores no están duplicados, sólo se generan de ese modo mientras se trabaja temporalmente con estas operaciones.

Es conveniente resaltar que el algoritmo interno de esta rutina no siempre es el que se ha explicado, puesto que lógicamente en el join de tablas con una cardinalidad muy alta (miles de tuplas) este proceso sería muy lento y costoso.

7.4.4.5. Valores nulos

El permitir que algunos atributos puedan tomar valores nulos tiene efectos importantes en las definiciones anteriores, que deben ser revisadas y adaptadas.

Se suelen emplear los valores nulos para representar información incompleta.

El concepto de dominio nos determina el conjunto de valores de un atributo en particular. Por tanto, el valor nulo de un atributo debe ser indicado en el dominio para que pueda tomarlo. Puesto que el dominio es un conjunto, se considera el valor como un elemento más del conjunto, distinto a todos los demás.

Con el concepto de llave principal se debe tener cuidado. Ahora se tiene que un atributo "A" es una llave principal cuando sus valores no nulos son todos distintos (restricción inherente). Si "A" es compuesto, se interpreta que no es nulo cuando todos sus componentes no lo sean. Nunca la llave principal contendrá valores nulos, debido a que cada valor de la llave representa una instancia única. El valor nulo se representa como un valor desconocido, no significa nada.

Aunque la comparación de un valor cualquiera con otro nulo dé resultado nulo, para poder ordenar los elementos de no dominio es necesario establecer entre ellos un criterio bien

definido de orden, incluyendo al nulo.

Es necesario considerar también un nuevo predicado o condición que permita preguntar si el valor de un atributo es nulo. Lógicamente, sólo podrá dar como resultado V o F (no podrá dar nulo; es decir, no puede ser desconocido). A continuación se presentan tablas de verdad en donde se ve involucrado un valor nulo (representado con el signo de "?"):

Conjunción

$\wedge$	V	?	F
V	V	?	F
?	?	?	F
F	F	F	F

Disyunción

$\vee$	V	?	F
V	V	V	V
?	V	?	?
F	V	?	F

Negación

$\neg$	
V	F
?	?
F	V

7.4.4.6. Codd y las bases de datos relacionales

Codd es el creador de las bases de datos relacionales. Actualmente, multitud de bases de datos se distribuyen con el apellido de relacionales, aprovechando la garantía ofrecida por bases de datos puramente relacionales y de gran prestigio. La mayoría de las bases de datos que se ofrecían como relacionales dejaban mucho que desear, por tal motivo, hubo necesidad de reestructurar algunos conceptos, siendo hasta el año de 1985 en un artículo publicado por la revista Computerworld, donde Codd daba once reglas que debe cumplir cualquier base de datos que se considere relacional:

1. ***Regla de información.*** Toda la información de una base de datos relacional está representada explícitamente a nivel lógico y exactamente de un modo (mediante valores de tablas).

2. ***Regla de acceso garantizado.*** Todos y cada uno de los datos (valor indisoluble, único o atómico) de una base de datos relacional se garantiza que sean lógicamente accesibles recurriendo a una combinación de nombre de tabla, valor de la llave primaria y nombre del atributo.

3. ***Tratamiento sistemático de valores nulos.*** Los valores nulos (distinto de cadena de caracteres vacía o de una cadena de caracteres en blanco y distinta de cero o de cualquier otro número) se soportan en los DBMS completamente relacionales para representar la falta de información y la información inaplicable de un modo sistemático e independiente del tipo de datos.

4. ***Catálogo en línea dinámico basado en el modelo relacional.*** La descripción de la base de datos se representa a nivel lógico, del mismo modo que los datos ordinarios, de modo que los usuarios autorizados pueden aplicar a su interrogación el mismo lenguaje

relacional que aplican a los datos regulares.

5. ***Regla de sublenguaje completo de datos.*** Un sistema relacional puede soportar varios lenguajes y varios modos de uso terminal (por ejemplo, el modo de rellenar con blancos). Sin embargo, debe haber al menos un lenguaje cuyas sentencias sean expresables, mediante alguna sintaxis bien definida, como cadenas de caracteres, y que sea completa en cuanto al soporte de todos los puntos siguientes:

   - Definición de datos.
   - Definición de vista.
   - Manipulación de datos (interactiva y por programa).
   - Restricciones de integridad.
   - Autorización.
   - Fronteras de transacciones (comienzo, cumplimentación y regreso).

6. ***Regla de actualización de vista.*** Todas las vistas que sean teóricamente actualizables son también actualizables por el sistema.

7. ***Inserción, modificación y borrado de alto nivel.*** La capacidad de manejar una relación de base de datos o una relación derivada como un único operando se aplica no solamente a la recuperación de datos, sino también a la inserción, modificación y borrado de los datos.

8. ***Independencia física de los datos.*** Los programas de aplicación y las actividades terminales permanecen lógicamente inalterados cualquiera que sean los cambios efectuados ya sea a las representaciones de almacenamiento o a los métodos de acceso.

9. ***Independencia lógica de los datos.*** Los programas de aplicación y las actividades terminales permanecen lógicamente inalterados cuando se efectúen sobre las tablas de base cambios preservadores de la información de cualquier tipo que teóricamente permita alteraciones.

10. ***Independencia de integridad.*** Las restricciones de integridad específicas para una base de datos relacional particular deben ser definibles en el sublenguaje de datos relacional y almacenables en el catálogo, no en los programas de aplicación.

11. ***Regla de no subversión.*** Si un sistema relacional tiene un lenguaje de bajo nivel (un solo registro cada vez), ese bajo nivel no puede ser utilizado para subvertir o suprimir las reglas de integridad y las restricciones expresadas en el lenguaje relacional de nivel superior (múltiples registros a la vez).

7.4.5. Normalización de bases de datos relacionales

Hace algún tiempo —por los años 60—, los datos que se manejaban dentro de la computadora mediante archivos fueron diseñados y gestionados por programas (generalmente en Cobol). Realmente no quedaba más remedio, puesto que hasta entonces no existían otros medios y no habían nacido herramientas que ganasen la confianza de los responsables del diseño de aplicaciones.

Entonces, se empezaron a realizar estudios del rendimiento de los datos, orientándolos principalmente en dos áreas:

- Las aplicaciones que los gestionaban.
- La evolución de los requerimientos.

Los resultados en la mayoría de los casos coincidían en mostrar las mismas evidencias:

- Para nuevas aplicaciones, la información tal y como estaba diseñada no servía y ello hacía que dichas aplicaciones creasen sus propios datos.
- La información estaba duplicada.
- Era muy difícil y costoso mantener los datos actualizados.
- El rendimiento no era el esperado.

Ante esta situación, se empezaron a generar las primeras bases de datos y junto con ellas se estudió una metodología de análisis como ayuda al diseño de los datos, con objeto de optimizar su tratamiento y futuras necesidades. Apareció entonces el concepto de normalización.

El proceso de normalización se encarga de seguir una serie de pasos o normas que tras aplicar todas ellas, se obtienen los datos agrupados en diferentes tablas, de tal forma, que es la estructura óptima para su implementación, gestión y explotación desde diferentes futuras aplicaciones. Una tabla se dice que está en una forma normal cuando satisface un conjunto de restricciones impuestas por dicha norma.

El proceso de normalización parte de las formas normales definidas por E.F. Codd (1970). Inicialmente, Codd formuló las tres primeras formas normales (1FN, 2FN, 3FN); posteriormente, ciertas anomalías detectadas forzaron a crear una forma normal más completa que la 3FN; es la FNBC (forma normal de Boyce y Codd), después Fagin definió la 4FN y 5FN.

La normalización asume que los datos son independientes de las aplicaciones que los gestionan, y su objetivo es obtener el mayor número de tablas posibles, dejando en cada una de ellas los atributos imprescindibles para representar a la entidad (objeto), o a la relación entre

entidades a la que hace referencia la tabla mediante la conexión de sus llaves.

Las ventajas que se obtiene tras la normalización de datos para su eficaz gestión son:

- ***Facilidad de uso.*** Los datos están agrupados en tablas que identifican claramente un objeto o una relación.
- ***Flexibilidad.*** La información que necesitan los usuarios se puede obtener de las tablas relacionales o relaciones mediante las operaciones del álgebra relacional. Por ejemplo: uniendo tablas, seleccionando sus valores y proyectándolos, etc.
- ***Precisión.*** Las interrelaciones entre las tablas consiguen mantener información diferente relacionada con toda exactitud.
- ***Seguridad.*** Los controles de acceso para consultar o actualizar información (bien a nivel de tablas o a nivel de atributos) son mucho más sencillos de implementar.
- ***Facilidad de implementación.*** Las tablas se almacenan físicamente como archivos planos.
- ***Independencia de datos.*** Los programas no están ligados a las estructuras, con lo que se consigue aumentar la base de datos añadiendo nuevos atributos o nuevas tablas sin que afecten a los programas que las usan.
- ***Claridad.*** La representación de la información es clara y sencilla para el usuario; son simples tablas.
- ***Facilidad de gestión.*** Los lenguajes manipulan la información de forma sencilla al estar los datos basados en el álgebra relacional.
- ***Mínima redundancia.*** La información no estará duplicada innecesariamente dentro de las estructuras.
- ***Máximo rendimiento de las aplicaciones.*** Sólo se utiliza aquella información que sirve de utilidad a cada aplicación.

El proceso de normalización lo realiza el analista tras sucesivas reuniones que mantiene con el usuario y la aplicación de las reglas de normalización.

7.5. SQL (STRUCTURED QUERY LANGUAGE)

El SQL es un lenguaje para acceso a la información almacenada en bases de datos relacionales, cuyas aplicaciones se están expandiendo rápidamente a lo largo de los últimos años. Actualmente hay muchos productos relacionales en el mercado informático que usan este lenguaje, tanto en computadoras personales como medianas o grandes.

El conocimiento del SQL permitirá comprender la extraordinaria flexibilidad y potencia que para el manejo de datos proporcionan los sistemas relacionales.

El SQL puede usarse desde dentro de programas escritos en lenguajes de tipo ensamblador o de tercera o cuarta generación, o también directamente desde una terminal, sin necesidad de incluirlo previamente dentro de un programa. Esto lo hace útil tanto para los programadores como para los usuarios finales que deseen la posibilidad de acceder a los datos mediante consultas abiertas no previamente programadas.

7.5.1. Qué es el SQL

El SQL es un lenguaje que permite expresar operaciones diversas, por ejemplo, aritméticas, combinatorias y lógicas, con datos almacenados en bases de datos relacionales.

La palabra SQL está formada con las iniciales en inglés *Structured Query Language* (Lenguaje Estructurado de Consultas). El nombre no refleja todas las posibilidades del lenguaje, pues éste no sólo permite consultas sino también actualizaciones de datos y otras operaciones.

Como ya se mencionó, suele aceptarse que el concepto de bases de datos relacionales arranca de un artículo publicado en 1970 por Codd, donde se mencionaban los conceptos básicos de un modelo relacional de datos y de un sublenguaje para acceder a ellos basado en el cálculo de predicados. Estas ideas fueron desarrolladas en años posteriores en los laboratorios de IBM dando lugar a un lenguaje llamado SEQUEL y a un prototipo de sistema relacional llamado System R. Hubo más o menos simultáneamente otros ensayos de lenguajes relacionales en diversas universidades, empresas y centros de investigación. Por su parte, el SEQUEL evolucionó y su nombre fue cambiado, dando lugar al SQL.

En los últimos años han aparecido en el mercado múltiples productos de bases de datos basados en el SQL, tanto para micro y minicomputadoras como para grandes sistemas, haciendo de este lenguaje un vehículo común ampliamente utilizado en entornos muy diversos de máquinas y sistemas operativos. Esta tendencia hacia su uso generalizado en la industria se ha visto consolidada al ser adoptada por el Instituto Americano de Normas (ANSI, *American National Standards Institute),* que ha desarrollado y publicado unas especificaciones para este lenguaje, aceptadas también posteriormente (1987) por la organización ISO *(International Standards Organization).*

Esto no quiere decir que los productos concretos disponibles sigan estrictamente estas especificaciones. Hay desviaciones, por una parte porque aquéllas no cubren todas las necesidades y deben ser completadas por los productos, y por otra parte por razones históricas. También hay diferencias entre unos productos y otros. Por ello, se recomienda leer los manuales de referencia del producto concreto que se vaya a usar antes de trabajar con él. En todo caso, estas diferencias suelen referirse a opciones adicionales o detalles menores que pueden fácilmente aprenderse una vez comprendidos los conceptos y sentencias.

Una de las características más importantes del SQL frente a los lenguajes tradicionales no relacionales de bases de datos, es que sus sentencias permiten manejar conjuntos de registros, en vez de un SQL o registro cada vez. Otra es que, al tener una base teórica firme, posee una gran capacidad expresiva aunque su estructura es muy simple. Todo ello dota al lenguaje de una gran potencia que permite expresar con una sentencia SQL consultas complejas, que tradicionalmente podrían requerir uno o más programas para su formulación, lo que redunda en una alta productividad en la codificación y prueba de programas.

Por otra parte, esta flexibilidad y potencia expresiva unidas a la capacidad de usarlo interactivamente para realizar consultas no planificadas, es decir, no previamente incluidas en programas, abre a los usuarios finales la posibilidad de acceder directamente a los datos.

7.5.2. Cómo se usa el SQL

Las peticiones sobre los datos se expresan en SQL mediante sentencias, que deben escribirse de acuerdo con las reglas sintácticas y semánticas de este lenguaje.

Estas sentencias pueden escribirse directamente en la pantalla de una terminal interactiva, en la cual también se recibe el resultado de la petición expresada en ellas.

También pueden escribirse las sentencias de SQL incluidas en programas, incorporándose así su capacidad expresiva a lógica y funciones de éstos. Naturalmente, el autor de los programas deberá conocer, además del SQL, el lenguaje de programación correspondiente, que puede ser cualquiera de los más conocidos, como el COBOL, FORTRAN, C, PL/I, etc. Esta forma de uso requiere por tanto conocimientos informáticos que suelen estar más al alcance de programadores profesionales que de usuarios finales.

Hay dos técnicas para utilizar el SQL en programas. En una de ellas, llamada SQL estático; las sentencias incluidas en el programa no pueden cambiar durante su ejecución. En la otra, llamada SQL dinámico; una sentencia puede ser modificada total o parcialmente por el propio programa durante su ejecución. La mayoría de los programas pueden codificarse empleando SQL estático, más sencillo y eficiente que el dinámico. Éste, en cambio, proporciona una mayor flexibilidad, que puede ser útil en algunos casos especiales, y suele requerir técnicas de programación más avanzadas en el manejo dinámico de memoria y variables.

7.5.3. Para qué sirve el SQL

El SQL permite la realización de consultas y actualizaciones sobre datos almacenados en tablas relacionales. Éste es el principal uso que harán del lenguaje los programadores y usuarios. Pero también hay otras tareas que se pueden realizar mediante sentencias SQL, aunque pertenecen más a las responsabilidades del administrador de bases de datos, las cuales son las siguientes:

1) Definición y destrucción de objetos

Antes de poder usar una tabla para almacenar o consultar datos en ella, hay que describirla al DBMS, dándole su nombre y características. Esto se hace definiéndola. Hay otros muchos objetos que maneja un DBMS, tales como espacios para archivos físicos, índices, etc. Todos ellos se definen con sentencias SQL.

Normalmente, la definición de tablas y vistas corre a cargo del DBA, pero puede haber casos en que se permita a algunos usuarios definir tablas para datos de uso privado. En este caso, el usuario empleará las correspondientes sentencias de SQL.

2) Gestión de las autorizaciones de acceso

Un usuario final no podrá consultar o actualizar datos de una tabla si previamente no ha sido autorizado para ello. Tampoco podrá hacerlo un programa si la persona que ha solicitado su ejecución no ha sido previamente autorizada. Estas autorizaciones se conceden o restringen mediante sentencias SQL.

Normalmente, las autorizaciones las concede el DBA, pero puede haber casos en que lo haga un usuario con respecto a sus datos privados, si se le permite tenerlos.

Hay otras tareas propias del DBA en las que el SQL puede ayudar, pero que necesitan otras herramientas adicionales. Así, por ejemplo, la prevención y corrección de problemas de rendimiento y la toma de copias de seguridad de los datos para el caso de fallos por catástrofes o averías.

En resumen, el SQL permite:

- Definir y destruir objetos de la base de datos.
- Conceder y restringir autorizaciones para usar estos objetos.
- Consultar y actualizar datos.

8. NORMATIVIDAD

Una de las ventajas de tener un control centralizado de datos es que pueden hacerse cumplir las normas establecidas, garantizando con ello el seguimiento de todas las formas aplicables a la representación de los datos, al igual que unificar los formatos de los datos almacenados como ayuda para el intercambio o migración de información entre sistemas. Las normas aplicables para el caso que nos ocupa comprenden las normas de la institución (UNAM, Facultad de Derecho). Por tal motivo, es necesario desarrollar un apartado especial que dé a conocer las normas en materia de cómputo (software y el hardware) pertinentes.

8.1. SOFTWARE

8.1.1. Bases de datos

Actualmente la Universidad Nacional Autónoma de México, ha establecido el uso del manejador de bases de datos Sybase. Se pueden mencionar algunas dependencias como DGAE (Dirección General de Asuntos Escolares), DCAA (Dirección de Cómputo para la Administración Académica), DGP (Dirección General de Personal), la Facultad de Derecho entre otras, que ya lo han incluido. Por tal motivo, cualquier base de datos que se desarrolle debe considerar a Sybase como su manejador, y dado que los sistemas existentes en la Facultad de Derecho utilizan a Power Builder o Delphi como front-end, se debe seguir también esta tendencia.

8.1.2. Programas

El desarrollo de programas recae por completo en el personal propio de esta facultad, debido a los altos costos externos de producción.

Todo el software desarrollado por el personal de esta facultad —de acuerdo a los estatutos de la UNAM—, pasa a ser propiedad de nuestra casa de estudios.

Todo programa debe ser debidamente documentado. Nombres de variables, procedimientos, etc., deben ser mnemónicos. También deben contener un encabezado que indique qué se lleva a cabo, quién lo desarrolló y fecha de creación. En caso de haber sufrido modificaciones mencionar quién las realizó y en qué fecha.

Cada vez que se realice una actualización se debe hacer un respaldo del programa original como prevención a cualquier eventualidad.

El código de los programas no debe ser excesivo. Debe ser modular. Las variables globales deben ser mínimas. Los módulos deben caracterizarse por ser reutilizables. Mientras el código sea óptimo mejor será el programa.

La mayoría de las funciones deben ser introducidas a una biblioteca para la cual existe un control.

Para liberar un sistema debe ser sometido a pruebas tanto por el propio programador como por terceras personas y finalmente el usuario. En caso de tener errores o fallas serán corregidas y se expondrá a nuevas pruebas.

8.1.3. Sistemas operativos

El sistema operativo elegido es el MS DOS de Microsoft para usuarios finales, dada su gran gama de aplicaciones de terceros existentes y su amplia difusión internacional (popularidad).

Por su parte para servidores se ha acordado UNIX como plataforma, debido a su gran robustez y a su calidad de ser abierto.

8.1.4. Interfaces de usuario gráficas (GUI)

Actualmente las interfaces de usuario gráficas han adquirido una gran importancia debido a su apariencia y funcionalidad (amigables), por tal motivo se ha elegido como herramienta gráfica Windows (versión 3.xx y 95) de Microsoft como plataforma para aplicaciones de usuarios finales por su gran uso y difusión a diferencia de OS/2,Next Step, DESQview o el de Macintosh.

Para la administración de servidores los cuales trabajan bajo sistema operativo UNIX (Solaris 2.5.1) el mismo proveedor del equipo incluye una interface de usuario gráfica llamada OPEN LOOK.

En cuanto a navegadores para Internet, la interface gráfica utilizada siguiendo los criterios anteriores es Netscape. El protocolo es el TCP/IP.

8.2. HARDWARE

Los equipos que se han estandarizado para usuarios finales son computadoras personales compatibles con las siguientes características:

- Procesador 486 o superior.

- Velocidad 66 Mhz o superior.
- Memoria 4 Mb o más. Para Windows 95: 8 Mb como mínimo.
- Capacidad de almacenamiento secundario de 500 Mb o mayor.
- Monitor VGA.
- Unidad de diskettes de 3 ½.
- Mouse.
- Impresora de matriz o laser.

8.3. GENERALES

8.3.1. Documentación

Para la documentación de programas se debe contar con información que describa de manera general el funcionamiento de cada módulo y en forma integral al, o los sistemas.

Algunos de los aspectos que debe contener dicha información es:

- Nombre del sistema y/o módulo en particular.
- Objetivo o función.
- Código fuente.
- Notas importantes acerca de su uso o manejo.
- Inclusión de un manual del sistema para los desarrolladores o bien, para su mantenimiento posterior.

8.3.2. Manuales de usuario

Se debe tener un manual sobre el uso y capacidad de los sistemas en un lenguaje sencillo, apoyándose con ilustraciones del propio sistema que definan claramente la forma de trabajar. Así mismo debe contener un apartado con "***Soluciones a los problemas más frecuentes***" sobre el uso del sistema. Además de proporcionarse el nombre y la ubicación de la (s) persona (s) que pueden brindar apoyo ante alguna eventualidad mayor.

8.3.3. Respaldos periódicos

Los respaldos periódicos se realizan de acuerdo a la importancia de los datos generados durante una jornada de trabajo y pueden ser mensuales, semanales o diarios.

8.3.4. Seguridad

En este rubro se incluye la creación de grupos de trabajo, de asignación de privilegios y recursos, control de acceso, cambio periódico de contraseñas, respaldos, etc.

8.3.5. Reportes

La letra del contenido de cualquier reporte debe ser mínimo de 12 puntos. La impresión debe ser en hoja tamaño carta, además de contener siempre un encabezado con las siguientes características:

- Escudos: tanto de la universidad como de la facultad.
- Leyendas: UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

  FACULTAD DE DERECHO
- Título del reporte.
- Fecha y hora de emisión.
- Número de página.
- Posibilidad de salida tanto a pantalla como a impresora.
- Un documento con la siguiente información:
  - Quién lo creó y fecha de creación.
  - Para qué área (s) fue creado.
  - Modificaciones que ha sufrido con su respectiva fecha.

8.3.6. Reportes de fallas

En cuanto a sistemas, se tiene que en caso de presentarse fallas durante el uso del sistema y

una vez leído el manual del usuario y consultar el apartado referente a la "Solución de problemas más frecuentes", se solicitará el soporte necesario para poder reparar la falla. Si se trata de un error en la programación ésta deberá solucionarse a la mayor brevedad posible y así mismo realizar los cambios necesarios en los módulos que se vieran afectados y en los manuales del usuario.

PLANTEAMIENTO DEL PROBLEMA Y PROPUESTA DE SOLUCIÓN

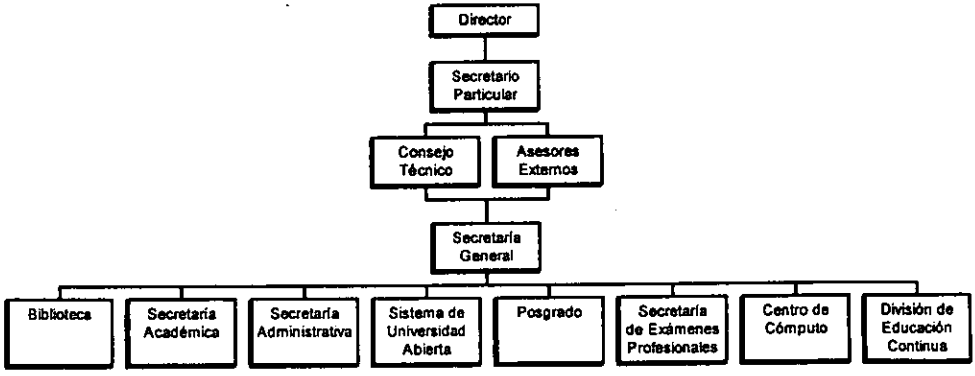
1. SITUACIÓN ACTUAL

La Facultad de Derecho tiene como uno de sus principales objetivos, crear una cultura jurídica en sus distintos contextos, a través de investigaciones y publicaciones de su personal académico, que puedan ser estudiadas y consultadas a nivel nacional e internacional con el fin de abarcar de manera exhaustiva el universo de la ciencia jurídica.

Académicamente tiene la misión de formar profesionales del derecho del más alto nivel, con un dominio profundo y amplio de la disciplina, con los conocimientos y habilidades necesarios para comprender y evaluar el campo y la problemática inherente al derecho, a fin de contribuir al desarrollo del país y la sociedad.

Para lograr estos objetivos cuenta con la siguiente organización.

Organigrama de la Facultad de Derecho



1.1. DIVISIÓN DE EDUCACIÓN CONTINUA

La División de Educación Continua se ubica en las instalaciones que albergaron a la Escuela Nacional de Jurisprudencia (antigua Facultad de Derecho) en el Centro Histórico de la Ciudad de México. Su función permanente es llevar a cabo cursos, seminarios, talleres y diplomados sobre temas de actualidad e interés para los profesionales del derecho y ramas afines, con el objeto de que constantemente actualicen sus conocimientos conforme al avance de la ciencia jurídica.

A través de la Jefatura y la Coordinación de Educación Continua, se lleva a cabo el proceso de planeación, inscripción, coordinación, impartición y control de cursos (diplomados, talleres, seminarios).

En función del tema y de la naturaleza de los cursos, se determina su duración, la cual por lo general es de varios meses, y en promedio de 5 a 6 anuales. El cupo es aproximadamente de 45 personas.

Hemos dividido las actividades mencionadas con respecto a los cursos en 5 rubros para facilitar la descripción y la comprensión de las situaciones que se presentan. Es importante mencionar que la mayoría del trabajo lo realiza la División de Educación Continua, la cual cuenta con 3 personas encargadas de su funcionamiento, apoyadas con dos computadoras personales y herramientas tales como Microsoft Office 4.0.

1.1.1. Publicidad

Una vez que la Jefatura y la Coordinación de Educación Continua han aprobado tema académico y desarrollado el temario para el nuevo curso, se elaboran los formatos de difusión publicitaria para distribuirse a través de periódicos, trípticos o carteles. La distribución depende de la naturaleza del curso y puede ser en periódicos como: Excelsior, Reforma, Financiero, etc. Lo mismo sucede con los demás medios de divulgación, difundiéndose a través de instituciones o escuelas afines con el tema (a excepción de la Facultad de Derecho donde siempre se distribuyen).

La publicidad es de vital importancia, ya que es la única manera en que los aspirantes pueden enterarse de la creación de nuevos cursos, puesto que la apertura es independiente del tema y del tiempo, es decir, no hay manera de predecir el curso que se impartirá, ni la fecha en que se iniciará.

1.1.2. Inscripción

A partir del momento en que comienza la difusión del curso, se cuenta con un período de 15 a 30 días para la inscripción, la cual es una de las actividades que requieren de más esfuerzo y cuidado.

Los alumnos se pueden inscribir de dos maneras:

- Una de ellas es vía telefónica, en la cual los aspirantes se comunican directamente con el personal de la Coordinación de Educación Continua para reservar lugar en el curso deseado o simplemente para pedir información. Ésta queda sujeta al pago respectivo del curso, el cual se debe hacer en alguna institución bancaria a nombre de la Escuela Nacional de Jurisprudencia A. C., y en el último de los casos en las propias instalaciones minutos antes de iniciar el curso.

- La segunda opción es acudiendo personalmente a las instalaciones de la Escuela Nacional de Jurisprudencia y realizar el pago por el curso.

Para ambos casos se precisa el llenado de la una solicitud, acompañada de un Curriculum Vitae.

Existe una variante a las opciones anteriores, prolongándose la posibilidad de inscripción hasta en un 20 % del tiempo de duración del curso, con lo cual se pretende una mayor concurrencia.

Para los alumnos resulta difícil la inscripción, sobre todo para aquellos que vienen del interior de la República Mexicana, pues según los procesos anteriores es necesaria una llamada y en el peor de los casos el traslado a las instalaciones de la División de Educación Continua.

El personal de la Coordinación de Educación Continua, es el encargado de llevar el control de la inscripción; ordenando las solicitudes, pagos y datos del alumno.

Con las personas que reservaron su lugar vía telefónica y las que lo hacen personalmente antes de iniciar el evento (curso), se elabora una lista preliminar, la cual, se modifica en función de las personas que lleguen el día de la inauguración y las que lo hagan durante el transcurso del 20% de la duración del curso. Cabe mencionar, que tanto la primera lista como las subsiguientes se hacen en Word 6.0 o Excel 4.0 para Windows.

1.1.3. Evaluación del curso

Uno de los controles más importantes es el de la evaluación; consiste en un recuento periódico de las asistencias. Cada clase el personal de la Coordinación se encarga de llevar el registro de éstas basándose en la lista más reciente. Al final del curso se hace una cuenta manual y se aprueba a todo aquél que haya acumulado el 80% de asistencias y cubierto la totalidad del monto del curso. Es importante mencionar que en caso de no cubrir el pago total del curso los alumnos son dados de baja, es por esto que también se debe llevar un control de los pagos de los alumnos, actualizándolos constantemente.

El último día de clases se les aplica una encuesta a los alumnos, con el fin de conocer sus impresiones sobre el curso y sugerencias. Esta última parte, es la que tiene más importancia, pues a través de ella es como se convierten en aspirantes a futuros cursos y a su vez definen la posibilidad de que se decida impartir el curso que proponen.

1.1.4. Invitación a nuevos cursos

La invitación a los cursos es una de las actividades más laboriosas, ya que se tiene que hacer primero una clasificación por temas de las sugerencias, para posteriormente comunicarse con cada una de las personas y hacerles directamente la invitación al nuevo curso.

Resulta verdaderamente conflictivo en esta etapa la selección de sugerencias, ya que crecen de manera proporcional a los cursos, multiplicadas por el número de alumnos, esto sin

considerar que algunas veces —sea en período de inscripción o no—, se piden informes por teléfono dejándose datos personales y sugerencias.

Aunado a lo anterior, está la situación bastante común de que la mayoría de las personas que proponen temas —por cualquiera de los medios—, lo hacen tanto de manera general como particular, siendo también muy frecuente, que mencionen más de un tema. De esta forma, la persona encargada de clasificar las sugerencias, puede colocar a una persona como aspirante a varios cursos, lo cual complica la situación de manera significativa.

1.1.5. Planeación de nuevos cursos

Aunque la planeación de los cursos es de incumbencia exclusiva tanto de la Coordinación como de la Jefatura de Educación Continua, de alguna manera tienen influencia los ex alumnos, pues a través de sus sugerencias supeditan la creación de cierto curso. Otra influencia importante la representan las instituciones que se interesan por cierto tema y que cuentan con personal suficiente para que se cree el curso que desean.

Una de las partes trascendentes dentro de este proceso, es la selección de ponentes o expositores, los cuales la mayoría de las veces se eligen en función de su profesión o del curso que previamente hayan impartido. Para esto se cuenta con una agenda, la cual también crece con el paso del tiempo, siendo la selección de manera análoga a la de alumnos.

2. REQUERIMIENTOS DEL USUARIO

Con respecto a los requerimientos de los usuarios (se denomina usuarios, a la gente que necesita y utiliza la información generada por los sistemas) se pueden clasificar en dos partes:

Por una parte el Centro de Cómputo de la Facultad de Derecho está buscando la integración de toda la infraestructura de cómputo disponible en las instalaciones de Ciudad Universitaria.

En primer lugar se ha dado al esfuerzo de conectar a una red todos aquellos equipos que generen información de interés a otras áreas, interconectando de igual forma, las redes que ya existen. También se está organizando y centralizando la información buscando la unificación y difusión para todas las áreas, ya que diversas dependencias de la Facultad exigen más y mejor información de otras, con una fluidez adecuada y con tiempos de respuesta considerablemente rápidos para un mejor desempeño de sus actividades.

Por tal motivo, se desea que la División de Educación Continua se integre también a esta estructura, lo que conllevaría a una difusión considerable de todos los datos pertenecientes a esa dependencia a las demás áreas, reflejándose en un beneficio tanto a División de Educación Continua como a la misma Facultad de Derecho.

Los sistemas que actualmente se encuentran integrados son:

- Reinscripciones a cursos ordinarios y exámenes extraordinarios.
- Control de asistencias del personal docente.
- Directorio de alumnos.
- Directorio de exalumnos de la Facultad de Derecho.
- Registro de tesis.
- Registro de personal académico.
- Asignación de labores.
- Registro de exámenes profesionales.
- Registro de material bibliográfico.
- Sondeo del rendimiento de profesores y alumnos.
- Insaculación.
- Servicios escolares.
- Inventario del equipo de cómputo.

La otra parte de los requerimientos proviene del personal de la División de Educación Continua que labora en el Centro Histórico, la cual debido a los grandes conflictos y la carga de trabajo que representa el proceso de los cursos, requiere de un sistema que automatice las más de las funciones que actualmente se realizan de manera semimanual.

El sistema debe incluir un banco de datos, en el cual se pueda almacenar toda la información de la gente registrada en los diferentes cursos, las sugerencias de los alumnos, los profesores que han impartido cursos, además de que cuente con una interfaz sencilla y amigable al usuario, asegurando la integridad (consistencia) de la información y reduciendo al máximo la redundancia del trabajo, así como de la información.

Se requiere que el sistema también incluya publicidad de los cursos en el Internet, lo cual se reflejaría en un incremento considerable en la matrícula de alumnos y reduciría en gran medida la carga de trabajo en cuanto al proceso de inscripción. Inclusive se pretende que por este medio, gente del interior de la República pueda inscribirse sin la necesidad de comunicarse a la División de Educación Continua o trasladarse a sus instalaciones.

3. RESTRICCIONES

Aunque el seguimiento de las normas de la institución actúa en pro del funcionamiento adecuado, del control de la información y de los resultados esperados, desde otra perspectiva refleja una serie de restricciones que así como las normas, se convierten en inherentes a la institución. Es por ello importante hacer hincapié, primeramente, en las restricciones que se desprenden de las normas de la Facultad de Derecho y de la Universidad misma.

- Sybase (versión 11.2) como manejador de bases de datos.
- Power Builder o Delphi, como interfaz gráfica de usuario para interactuar con la base de datos (front-end).
- Netscape como navegador de Internet.
- Sistema operativo UNIX (Solaris 2.5.1) para los servidores y para computadoras personales MSDOS o Windows 95.

La segunda parte de las restricciones la representa el aprovechamiento de los recursos con los que se cuenta actualmente, los cuales se mencionan a continuación:

- El servidor de bases de datos (Tlatoani) tiene las siguientes características: Sun Ultra 1 Ultra Sparc a 147 Mhz, 64 Mb en memoria RAM y 7 Gb en disco duro.
- El servidor de Internet (Themis) es: Sun Sparc 20 con 2 procesadores Super Sparc a 50 Mhz, 128 Mb en RAM y 1 Gb en disco duro.
- Las computadoras personales con las que se cuenta actualmente en Jurisprudencia son Dell 486 con 4 Mb en memoria RAM a 33 Mhz y disco duro de 240 Mb.
- Evidentemente la ubicación de la información, tanto la base de datos como las páginas del Internet y cualquier programa adicional (CGI), será en las instalaciones del Centro de Cómputo de la Facultad de Derecho en Ciudad Universitaria.
- Los modems para realizar la conexión desde Jurisprudencia con las instalaciones del Centro de Cómputo de Ciudad Universitaria son a 28,800 Mbps, con línea no dedicada (pues la utilización no representa gran cantidad de tiempo).
- Para los procedimientos que continúen sin automatizarse y que requieran de procesadores de palabra u hojas de cálculo, se utilizarán los programas de Microsoft, tal como Word para Windows versión 6.0 o superior y Excel para Windows versión 4.0 o superior.

Cabe mencionar que lo anterior en realidad no representa grandes restricciones, ya que todos los recursos resultan satisfacer inclusive hasta con holgura las demandas del sistema,

tanto en cuestiones operativas, ya que por ejemplo, se cuenta con el personal que mantenga el control de la base de datos (DBA), el control de la información en Internet y los programas pertinentes (personal del Centro de Cómputo), como con algunos recursos extras tales como la implantación de computadoras personales en la División de Educación Continua con las siguientes características:

- Procesador Pentium
- 16 Mb en RAM
- Velocidad de 133 Mhz
- Disco duro de 1 GB

En caso de resultar insuficiente la comunicación con una computadora, se colocaría un modem adicional a la otra computadora disponible.

4. PROPUESTA DE SOLUCIÓN

Para solucionar el problema se adoptó la secuencia del proceso de inscripción, de ahí, que el sistema debe incluir los rubros mencionados, optimizando las actividades lo más posible, es decir, incluir la computadora como uno de los componentes principales en las funciones del sistema y con respecto a ésta redefinir las tareas que se seguirán haciendo de manera manual que desde luego, disminuirán de manera significativa. De esta forma la propuesta de solución tiene las siguientes características:

La creación de un sistema de automatización para agilizar y disminuir el trabajo al personal de la División de Educación Continua, con el cual que aprovechen los recursos disponibles, obedeciendo a la nueva tendencia de centralización de la Facultad de Derecho. El sistema tiene dos parte fundamentales: el uso de Internet y un banco de datos que contenga la información pertinente.

Con respecto a la publicidad, el nuevo sistema incluirá Internet a los medios de difusión existentes, pues además de que es uno de los medios de información y comunicación que hoy en día la gente más consulta, la Facultad de Derecho, a través de la UNAM dispone de este servicio de manera gratuita (no implica gastos evidentes).

La publicidad en Internet realmente tiene otro contexto, es decir, se pretende que además de proporcionar toda la información necesaria sobre los cursos, exista la posibilidad de realizar la inscripción a través de este medio. Será un beneficio sin precedente para los usuarios, pues ya no sería necesaria ni la comunicación directa (vía telefónica), ni la inscripción personal (en las instalaciones de Jurisprudencia). Y por supuesto, se proporcionarían todos los detalles de los cursos, sin un límite de espacio e información tan claro como el que imponen los medios de difusión tradicionales (periódicos, carteles o trípticos).

La publicidad resultaría determinante con respecto al proceso de inscripción, pues se verá considerablemente reducida la carga de trabajo, tanto en el proceso de proporcionar datos (interacción entre la Dependencia y los alumnos), como de elaborar la lista preliminar, pues el sistema proporcionará esta lista en función de las personas inscritas a través de Internet.

Para el caso de los alumnos que aún continúen inscribiéndose por teléfono o personalmente, la simplificación del trabajo para el personal de la División de Educación Continua también será evidente, pues se introducirían los datos directamente a la computadora, los cuales se agregarán automáticamente a la lista preliminar, existiendo la posibilidad permanente de solicitar al sistema las listas de alumnos inscritos, con sus respectivos pagos y datos generales (de ser necesario).

Para la evaluación, las listas que genere el sistema contribuirán a la disminución del trabajo en esta fase de tal forma, la persona que lleve a cabo esta actividad, se limitará a confrontar las asistencias con los respectivos pagos, además, dado el número de cursos y la cantidad de alumnos no representa realmente gran cantidad de trabajo.

Uno de los procesos de más trascendencia y de más esfuerzo es el de invitación a cursos, para solucionar esto el sistema guardará las sugerencias de acuerdo a su naturaleza (tema) y existirá la posibilidad de ordenar a los alumnos de acuerdo al tema que propongan. De tal forma, la persona encargada de esta operación, sólo tendrá que introducir al sistema la petición de acuerdo al tema de su interés, y se visualizarán los posibles candidatos con sus datos respectivos. A partir de las listas generadas sólo se procederá a la invitación vía telefónica.

El sistema contendrá también los datos de los profesores que han impartido clases en esa dependencia, con lo cual el trabajo de selección se verá beneficiado enormemente, y al igual que para el caso anterior, se arrojará una lista con los posibles candidatos y sus respectivos datos.

Es importante mencionar que el sistema tendrá una interfaz amigable, es decir, sencilla, intuitiva y funcional, ya que los principales usuarios serán las personas que laboran en la Escuela Nacional de Jurisprudencia.

5. ESTRATEGIA DE SOLUCIÓN

Una de las actividades de más trascendencia en el proceso de concepción del nuevo sistema es la delimitación de las funciones que estarán dentro del sistema y las que quedarán fuera. Para tal fin el primer paso consistió en la interpretación del Manual de Procedimientos.

5.1. MANUAL DE PROCEDIMIENTOS

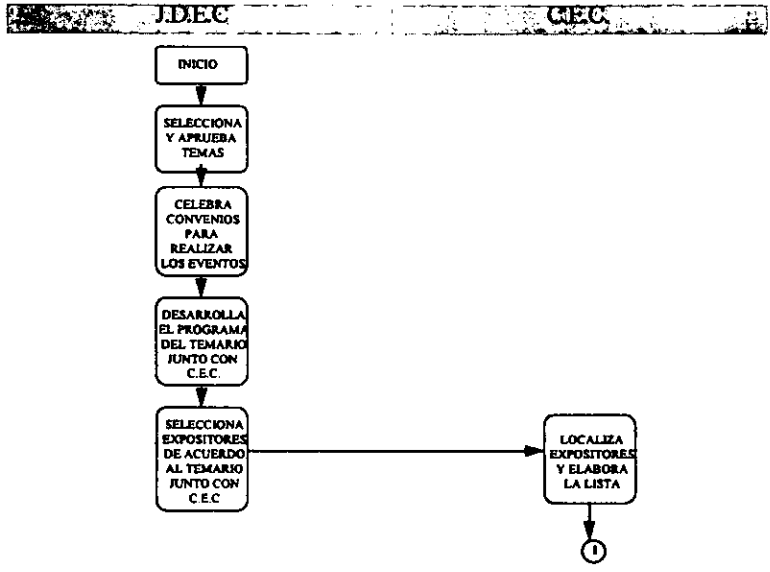
A continuación se muestra la descripción narrativa de las actividades del manual de

procedimientos, así como el diagrama de funcionalidad del proceso de los cursos. Con el fin de tener una visión de forma gráfica de la secuencia que se sigue al realizar las operaciones de un determinado procedimiento, también se muestran las unidades organizativas que intervienen.

Responsable:

Jefatura de la División Educación Continua (J.D.E. C)

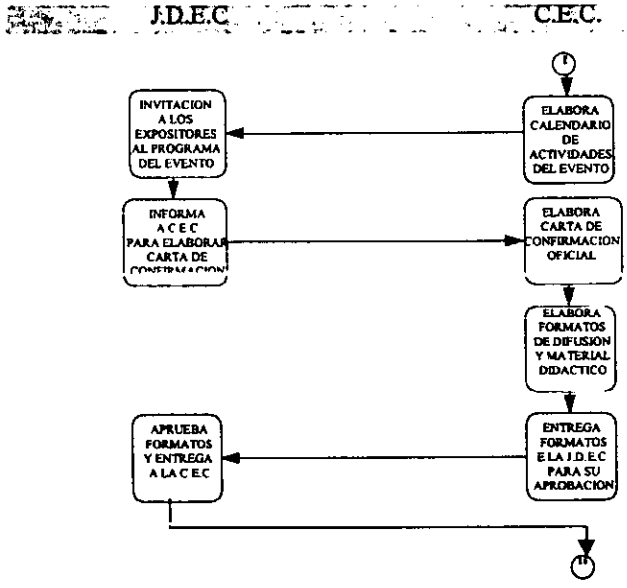
Coordinación de Educación Continua (C. E.C)



Actividad

1. Efectúa reuniones de trabajo para seleccionar y aprobar temas académicos de interés general con funcionarios públicos y privados. 2. Celebra convenios de colaboración con instituciones públicas o privadas, para la realización de diversos eventos.	

3. Desarrolla el programa del temario.	
	4. Selecciona a los expositores o ponentes de acuerdo al temario. 5. Localiza a los expositores y elabora lista.

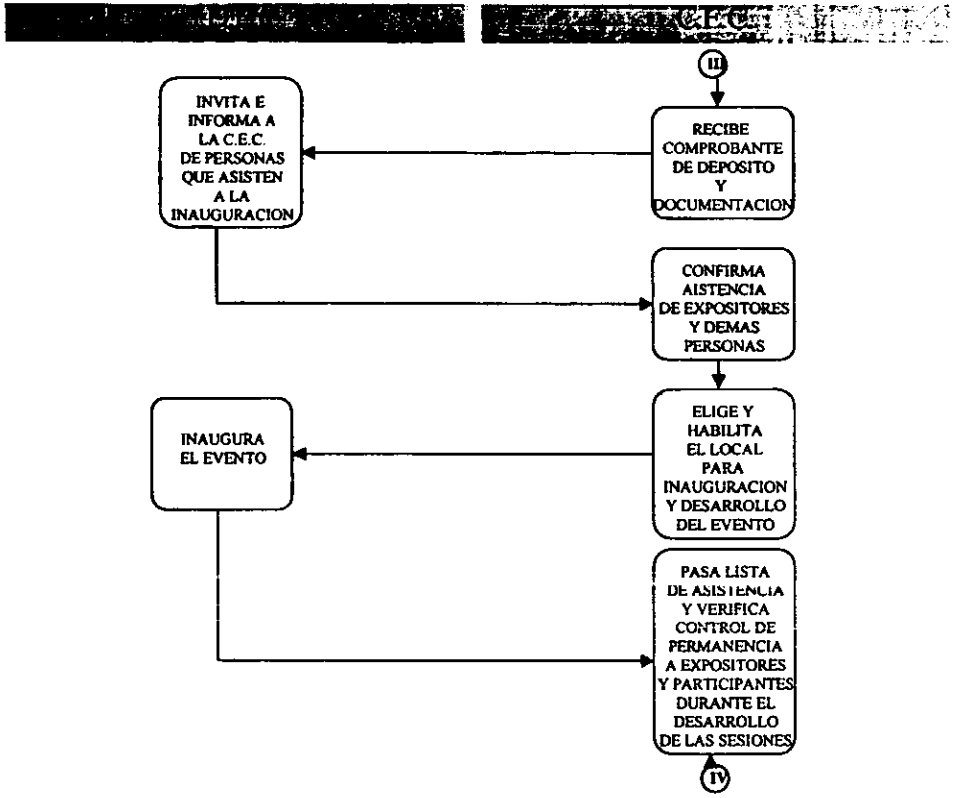


	6. Elabora calendario de actividades del evento que entrega al titular de la jefatura.
7. Procede a la invitación personal de los expositores con base al programa del evento. 8. Informa a la Coordinación de Educación Continua para que	

elabore carta de confirmación oficial del expositor o ponente.	
	9. Elabora carta de confirmación oficial que contiene: tema, lugar, fecha, hora y costo del evento procediendo a su envío. La duración varía dependiendo del tipo de evento: Diplomados: 120 hrs. o más; Talleres y Seminarios 30 a 60 hrs.; Cursos y otros eventos es variable. 10. Elabora Formatos de difusión del evento y material didáctico (cuando es solicitado por el expositor). ♦ Anuncios en los periódicos. ♦ Carteles. ♦ Trípticos. 10.1. Si el expositor no solicita en ese momento elaboración de material didáctico se prepara conforme se desarrolla el evento. ♦ Fotocopias y captura de documentos. 11. Entrega formatos a la Jefatura para su aprobación.
12. Aprueba material de difusión (formatos) y entrega a C.E.C.	

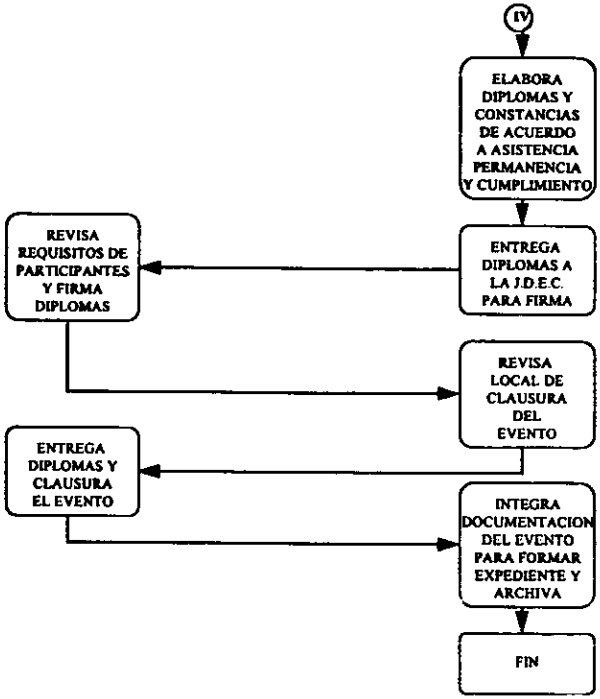
	13. Recibe y envía formatos de difusión a periódicos e imprentas para su reproducción • Imprenta: hace pruebas que presenta para su autorización. • Periódicos: el formato se envía a la Secretaría Administrativa de la Facultad para su publicación. • Entregan directamente el material didáctico a los

	cursantes. UNA VEZ QUE EL MATERIAL HA SIDO IMPRESO: 14. Recibe material de difusión impreso y procede a su distribución con sus posibles usuarios (Directores de Escuelas y diferentes Instituciones). * LOS PROFESIONISTAS INTERESADOS PIDEN INFORMACIÓN VÍA TELEFÓNICA O PERSONAL, PARA INSCRIPCIONES* 15. Informa a los interesados sobre el evento y verifica cumplan con los requisitos para ser inscritos. ◆ Curriculum. ◆ Grado académico. ◆ Experiencia laboral de acuerdo al tema. UNA VEZ QUE VERIFICA QUE CUMPLEN CON LOS REQUISITOS: 16. Indica, realicen pago en banco a nombre de Fundación Escuela Nacional de Jurisprudencia A.C., número de cuenta y cantidad. Informa a participante que realizado el pago, se presente a División de Educación Continua a llenar solicitud y entregar documentación.



	PARTICIPANTES REALIZAN PAGO, SE PRESENTAN EN DIVISIÓN DE EDUCACIÓN CONTINUA 17. Recibe comprobante de depósito bancario y documentación, entrega al participante recibo de inscripción. ♦ Pago de inscripción mediante ficha de depósito bancario.

	♦ 2 fotografías tamaño infantil. ♦ Solicitud de inscripción. ♦ Curriculum.
18. Invita y posteriormente informa a C.E.C. los nombres y cargos de las personas que asisten a inaugurar el evento.	
	19. Confirma la asistencia tanto de las personas que van a inaugurar el evento como de los expositores. 20. Elige y habilita local para la inauguración y desarrollo posterior del evento.
21. Procede a la inauguración del evento al cual sigue la sesión de trabajo.	
	22. Pasa lista de asistencia y verifica control de permanencia a expositores y participantes durante el desarrollo de las sesiones.



	23. Elabora diplomas y constancias cuando va a concluir el evento, con base en la asistencia, permanencia de los participantes y cumplimiento de las requisitos. 24. Entrega diplomas o constancias a la Jefatura de Educación Continua, para firma del Director, del jefe de la División y del titular de la institución coparticipante; anexando listas de asistencia, permanencia y cumplimiento de los requisitos.

25. Verifica la asistencia, permanencia y cumplimiento de los requisitos de los participantes y firma diplomas o constancias.	
	26. Revisa local de clausura del evento.
TERMINA EL EVENTO	
27. Entrega diplomas o constancias y procede a la clausura del evento con otros funcionarios invitados.	
	28. Integra toda la documentación del evento para formar expediente y archiva.
FIN DEL PROCEDIMIENTO	

El paso siguiente después de la comprensión del manual de procedimientos, fue una serie de entrevistas con el personal de la División de Educación Continua de la Facultad de Derecho en las instalaciones de Jurisprudencia.

Fundamentalmente, se entrevistó a la gente de la Coordinación de Educación Continua, ya que es la encargada directa de llevar a cabo la interacción con los alumnos, razón por la cual conciben el proceso general de los cursos —y padecen los conflictos intrínsecos—.

Las entrevistas mostraron de manera clara los procesos que se especifican en el diagrama de funcionalidad. Existen, sin embargo, datos adicionales que son de gran importancia, tal como las formas de inscripción y la de sugerencias.

En primer término tenemos la forma de inscripción, la cual nos proporciona datos trascendentes, tales como:

- Datos personales de los aspirantes.
- Datos relativos a su desempeño profesional.
- Datos de los recibos de pago.

FACULTAD DE DERECHO
DIVISION DE EDUCACION CONTINUA

SOLICITUD DE INSCRIPCION

"CURSO DE ACTUALIZACION EN *NOMBRE DEL CURSO*"

FECHA: _____

DATOS PARTICULARES DEL PARTICIPANTE

NOMBRE COMPLETO: _____
DOMICILIO: _____
CALLE NUMERO
_____
COLONIA C.P. TELEFONO(S)
GRADO ACADEMICO: _____
INSTITUCION: _____

DATOS RELATIVOS A SU DESEMPEÑO PROFESIONAL

EMPRESA: _____
DOMICILIO: _____ _____
CALLE NUMERO
_____
COLONIA C.P. TELEFONO(S)
CARGO QUE OCUPA: _____
MEDIO POR EL CUAL SE ENTERO DEL DIPLOMADO _____
_____

RECIBO OFICIAL: _ _____

La otra forma que manejan y que es de igual importancia para la solución del problema es la de sugerencias, con la cual se conocen los temas de más demanda y con ello los datos de los posibles alumnos. Ésta nos proporciona datos vitales como:

- Sugerencia propuesta.
- Datos de la persona.
- Horario y duración del curso propuesto.

SUGERENCIAS PARA FUTUROS EVENTOS ACADEMICOS:
Nombre: _____
Domicilio: _____
_____
Teléfono (s) de su oficina: _____
Teléfono particular: _____

1.- ¿Qué Diplomado, Seminario o Taller le gustaría que la División de Educación Continua de la Facultad de Derecho UNAM, programará en el futuro próximo._____

NOTA IMPORTANTE:
Duración mínima de los eventos:
Diplomado (teórico-práctico) 120 horas Seminario (teórico) 30 horas
Taller (teórico-práctico)
Curso de actualización (teórico, teórico-práctico)

2.- ¿Qué horario sería de su conveniencia?
_____

6. ANÁLISIS DEL SISTEMA

Después de haber recabado la información necesaria para conocer los requerimientos del nuevo sistema, se procedió a la fase del análisis, para la cual se precisó de la elaboración de modelos que reflejen las características y funciones del nuevo sistema.

6.1. MODELO ESENCIAL

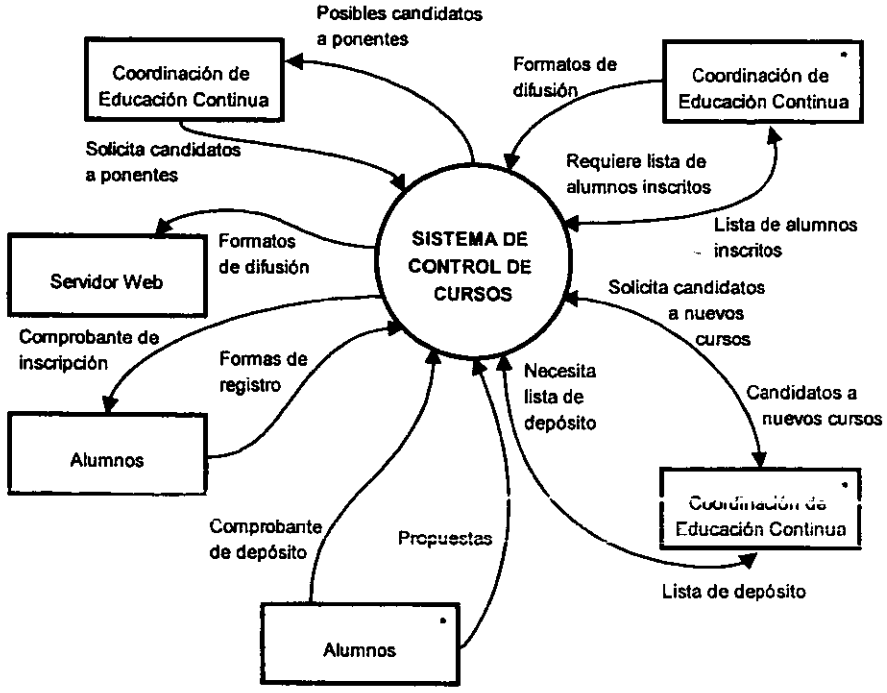
El primer modelo que se desarrolla es el que define la frontera entre el mundo exterior y el sistema. Con él se delimitan las funciones fundamentales del sistema y su interacción con el resto del ambiente.

6.1.1. Modelo ambiental

6.1.1.1. Declaración de propósito

El propósito del Sistema de Control de Cursos es coordinar el proceso de inscripción, así como tener un control de profesores, cursos y alumnos. Esta información debe estar disponible a quienes compartan los recursos de la red interna.

6.1.1.2. Diagrama de contexto



* Terminador repetido.

6.1.1.3. Lista de acontecimientos

1. El coordinador solicita agenda de profesores candidatos a ponentes. (F)
2. El coordinador proporciona formatos de información de los cursos para difusión en Internet.(F)
3. El alumno se inscribe. (F)
4. El coordinador requiere lista de alumnos inscritos. (F)
5. Los alumnos proporcionan comprobante de depósito bancario. (F)

6. El coordinador necesita lista de alumnos en función del depósito bancario. (F)

7. Los alumnos proporcionan propuestas de cursos deseados. (F)

8. El coordinador solicita agenda de alumnos candidatos a nuevos cursos. (F)

Cabe resaltar, que todos los acontecimientos (los estímulos que ocurren en el mundo exterior a los cuales el sistema debe responder), son de tipo flujo, es decir, que para los 8 acontecimientos el sistema se da cuenta que ha ocurrido el acontecimiento cuando llegan los datos.

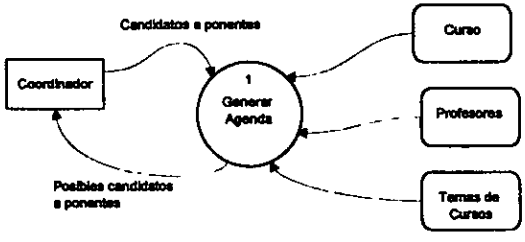
6.1.2. Modelo de comportamiento

El modelo de comportamiento incluye la construcción del diagrama de flujo de datos, el diagrama entidad-relación y el diccionario de datos.

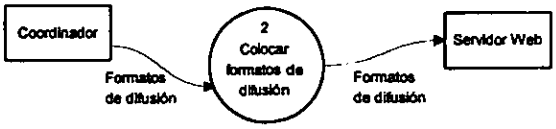
A continuación se muestran los modelos respectivos, empezando por el diagrama de flujo de datos, para lo cual, el paso preliminar para el diagrama de nivel 0 es la construcción de un diagrama de flujo de datos para cada actividad de la lista de acontecimientos como se muestra en seguida.

6.1.2.1. Diagrama de flujo de datos

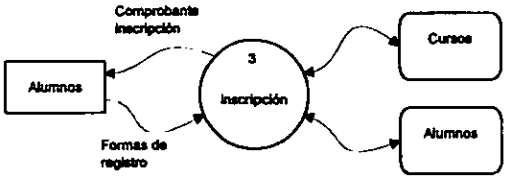
Acontecimiento 1: El coordinador solicita agenda de profesores candidatos a ponentes.



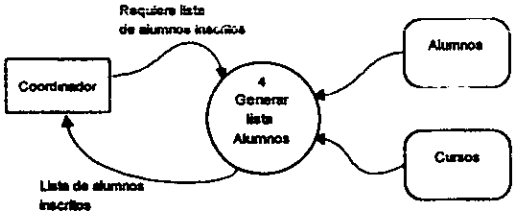
Acontecimiento 2: El coordinador proporciona formatos de información de los cursos para difusión en Internet.



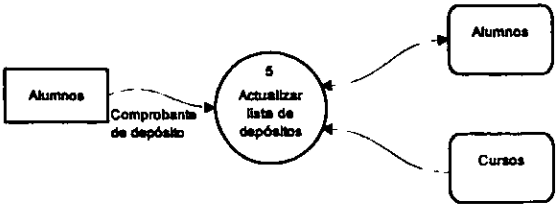
Acontecimiento 3: El alumno se inscribe.



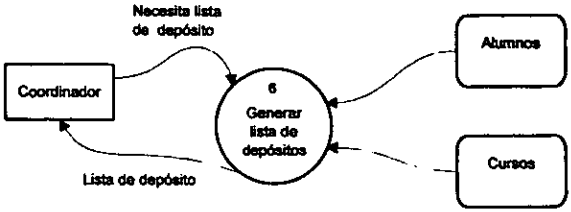
Acontecimiento 4: El coordinador requiere lista de alumnos inscritos.



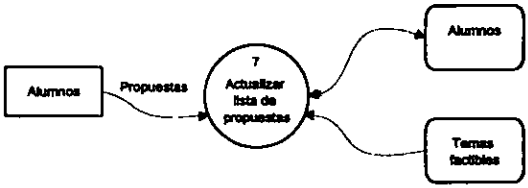
Acontecimiento 5: Los alumnos proporcionan comprobante de depósito bancario.



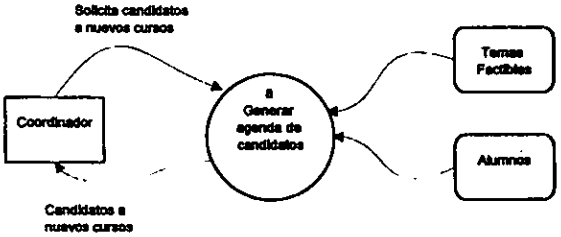
Acontecimiento 6: El coordinador necesita lista de alumnos en función del depósito bancario.



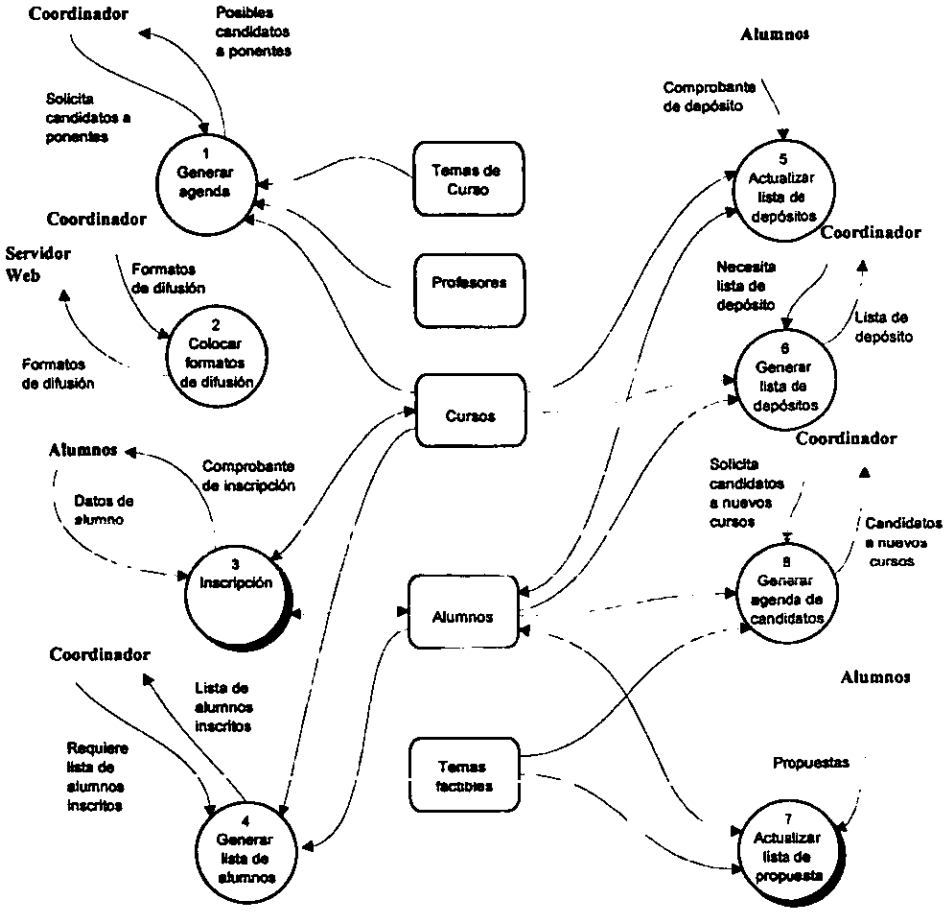
Acontecimiento 7: Los alumnos proporcionan propuestas de cursos deseados.



Acontecimiento 8: El coordinador solicita agenda de alumnos candidatos a nuevos cursos.



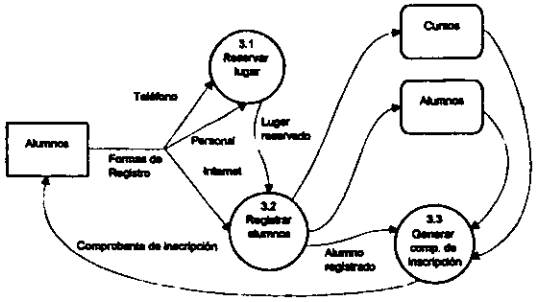
6.1.2.1.1. Diagrama de flujo de datos de nivel 0



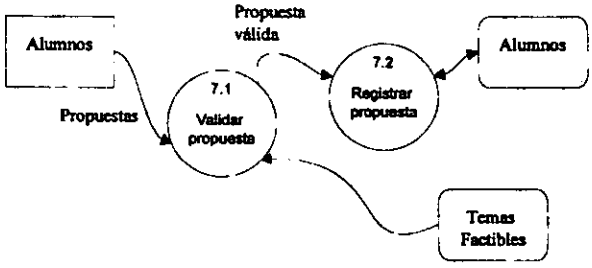
6.1.2.1.2. Diagrama de flujo de datos nivel 1

A partir del diagrama de flujo de datos de nivel 0 se desprende el de nivel 1, pero sólo para los acontecimientos 3 y 7, ya que son los que requieren de un mayor grado de detalle, lo que implica que internamente tienen procesos que deben ser desglosados totalmente.

Proceso 3



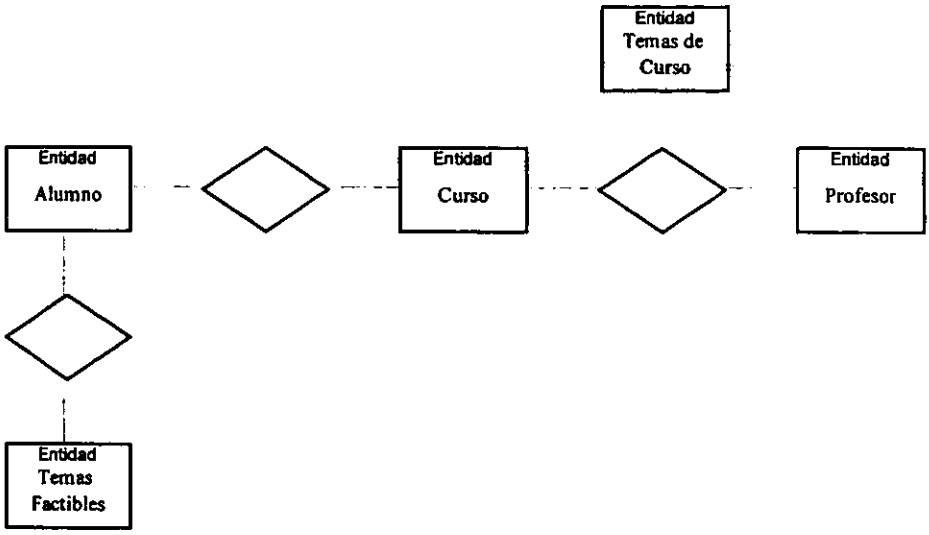
Proceso 7



6.1.2.2. Modelo entidad relación

El modelo que describe las relaciones de los datos almacenados por el sistema es el diagrama entidad-relación.

Una de las primeras aproximaciones al diagrama entidad-relación es la siguiente:

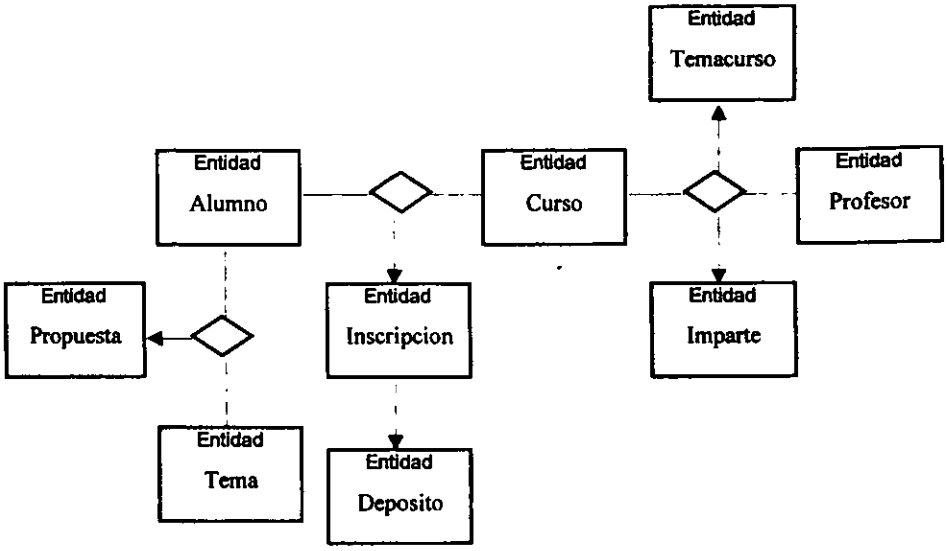


De acuerdo al diagrama resultante se tiene que los tipos de objeto son las entidades Alumno, Curso, Profesor, Temas de Curso y Temas Factibles con sus respectivas relaciones por supuesto.

El modelo anterior presenta algunas anomalías, tales como:

- Registro de propuestas de Alumnos.
- Registro de inscripciones.
- Depósitos de Alumnos.
- Asignación de Profesores a un tema de un curso.

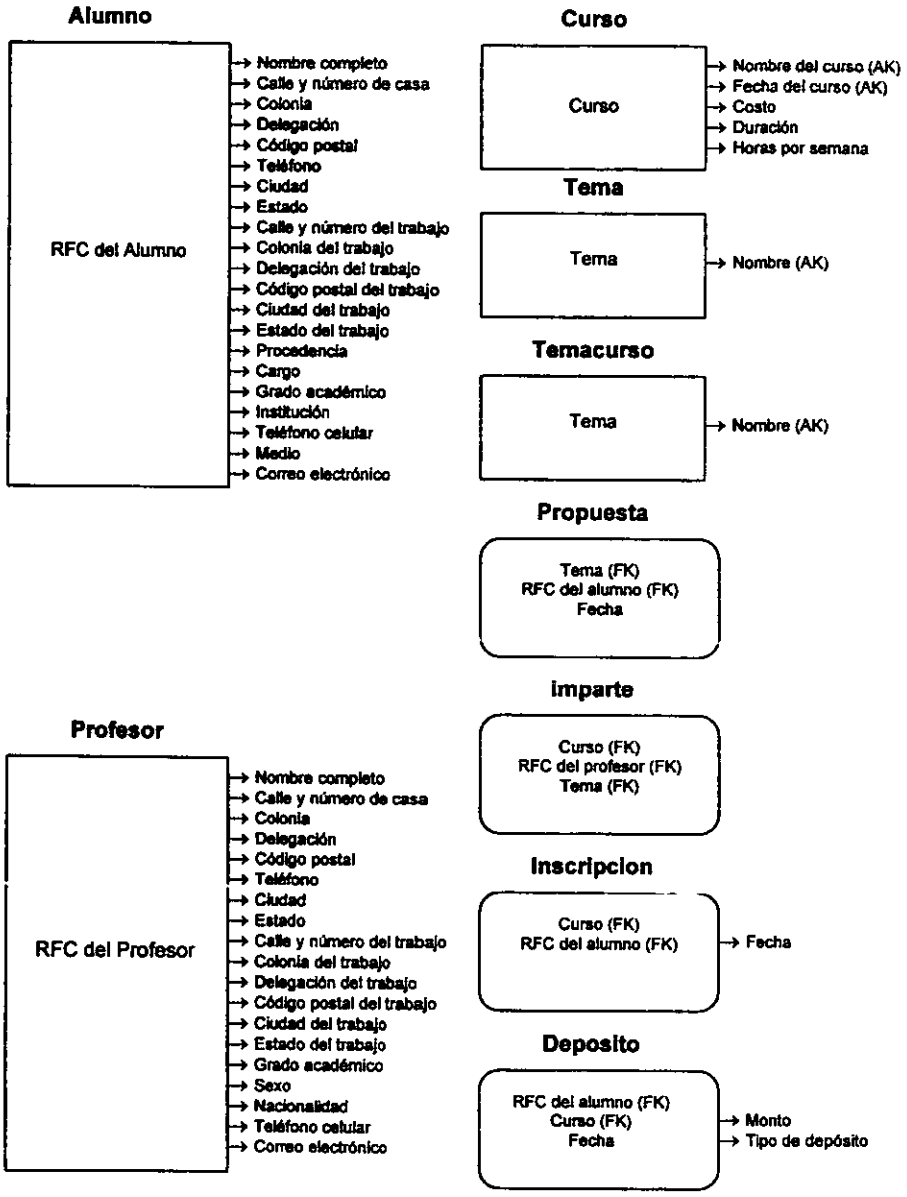
Por tal motivo, fue necesario crear un nuevo diagrama que cubriera los puntos anteriores, obteniéndose el siguiente:



7. NORMALIZACIÓN

A partir del diagrama entidad-relación, es factible pasar al análisis de la base de datos. Para esto, nos apoyaremos en una herramienta que permita mostrar de manera gráfica el proceso de normalización de forma clara, nos referimos al grafo de dependencias funcionales*. Por tal motivo hemos representado las tres primeras formas normales siguiendo este esquema, el cual es el siguiente:

* Para información sobre el Grafo de Dependencias Funcionales consultar el **Apéndice A Formas Normales.**


Alumno
RFC del Alumno
Nombre completo
Calle y número de casa
Colonia
Delegación
Código postal
Teléfono
Ciudad
Estado
Calle y número del trabajo
Colonia del trabajo
Delegación del trabajo
Código postal del trabajo
Ciudad del trabajo
Estado del trabajo
Procedencia
Cargo
Grado académico
Institución
Teléfono celular
Medio
Correo electrónico
Curso
Curso
Nombre del curso (AK)
Fecha del curso (AK)
Costo
Duración
Horas por semana
Tema
Tema
Nombre (AK)
Temacurso
Tema
Nombre (AK)
Propuesta
Tema (FK)
RFC del alumno (FK)
Fecha
Imparte
Curso (FK)
RFC del profesor (FK)
Tema (FK)
Profesor
RFC del Profesor
Nombre completo
Calle y número de casa
Colonia
Delegación
Código postal
Teléfono
Ciudad
Estado
Calle y número del trabajo
Colonia del trabajo
Delegación del trabajo
Código postal del trabajo
Ciudad del trabajo
Estado del trabajo
Grado académico
Sexo
Nacionalidad
Teléfono celular
Correo electrónico
Inscripcion
Curso (FK)
RFC del alumno (FK)
Fecha
Deposito
RFC del alumno (FK)
Curso (FK)
Fecha
Monto
Tipo de depósito

8. DESARROLLO

Para el desarrollo se utilizó primeramente una herramienta CASE llamada ERwin, razón por la cual resulta pertinente mencionar algunos conceptos y definiciones.

CASE (Ingeniería de Software Asistida por Computadora) proporciona al ingeniero la capacidad de automatizar las actividades manuales y de mejorar su enfoque de trabajo.

El objetivo más importante del CASE es conseguir la generación automática de programas desde una especificación a nivel de diseño.

8.1. CASE ERWIN

ERwin ERX/2.5 Release 2.5.01 es una herramienta de diseño de bases de datos para el desarrollo de sistemas cliente-servidor, también permite crear o aplicar un proceso de ingeniería en reversa a la base de datos relacional y obtener el modelo de la base de datos.

ERwin permite crear el modelo de datos, tanto a nivel lógico (entidades y atributos) como a nivel físico (tablas y campos). Se pueden crear diagramas comprensivos que se pueden documentar aun en las situaciones más complejas, con gráficos fáciles de comprender y apoyo total de Windows con respecto a tipos de letras y colores.

Aunque con ERwin se pueden simplemente dibujar diagramas Entidad-Relación, es mucho más que una herramienta de dibujo, ya que permite diseñar el modelo lógico de datos y automáticamente construir la estructura física de los datos. No hay necesidad de escribir definiciones de datos en SQL para crear una nueva base de datos. Cuando el modelo de los datos está listo, simplemente se selecciona el manejador de bases de datos deseado y se escogen las opciones de generación de esquema que se desean crear, de esta manera, se construye automáticamente la base de datos física; inclusive todas las tablas, índices, procedimientos almacenados, triggers de integridad referencial y otros componentes necesarios para un manejo exitoso de los datos.

Erwin tiene una capacidad para la generación del esquema de la base de datos; se puede conectar automáticamente al servidor deseado, generar un modelo de datos lógico, permitiendo agregar, actualizar o borrar definiciones de datos; generar el nuevo esquema y crear la base de datos física para uno u otro DBMS.

Cada entidad en el modelo de transformación llega a ser una tabla relacional, las llaves primarias llegan a ser índices únicos, las llaves alternas y los atributos normales también pueden llegar a ser índices. La cardinalidad puede imponerse y definirse la integridad referencial en función de las capacidades del DBMS seleccionado.

Servidores soportados por ERwin

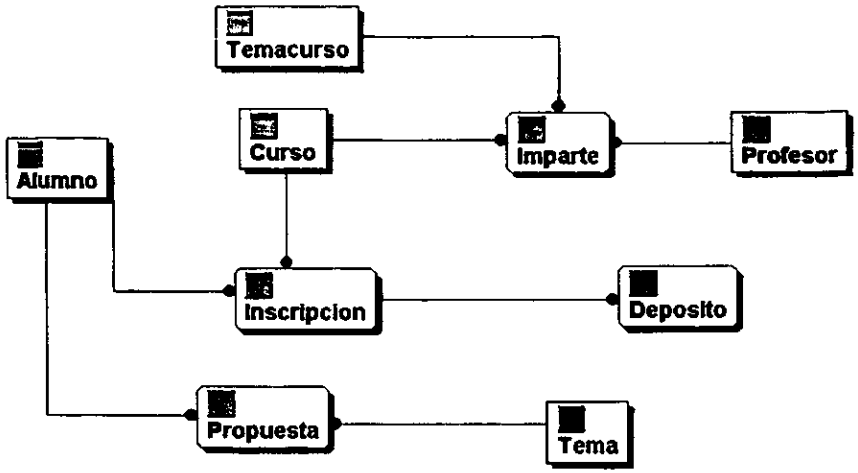
Si quiere emigrar una base de datos existente de una plataforma a otra, ERwin puede utilizar ingeniería en reversa a la base de datos y crear un modelo de datos lógico permitiendo con esto, modificar o agregar nuevos elementos de ser necesario, y construir la base de datos física en cualquiera de los siguientes manejadores:

SYBASE	ORACLE	Microsoft Access
INFORMIX	PROGRESS	Clipper
Ingres	SQL Server	DBASE III
DB2	AS/ 400	DBASE IV
Netware SQL	SQLBase	Paradox
RDB	Teradata	Microsoft FoxPro
WATCOM (SQL Anywhere)		

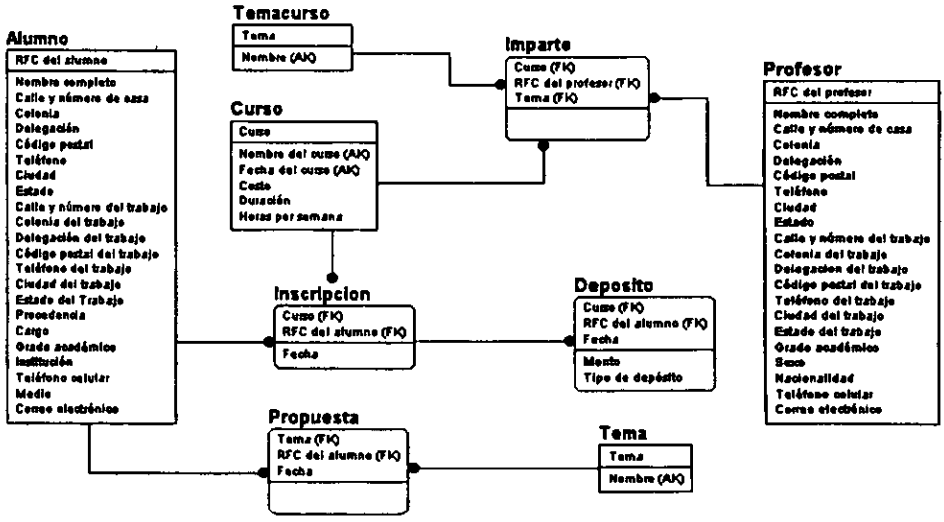
Cuando se crea el esquema físico de la base de datos, se genera un script DDL (lenguaje de definición de datos) usando la sintaxis SQL correcta para el servidor seleccionado. Se puede ver el código generado por ERwin y en caso de ser necesario, modificarlo antes de que la base de datos física sea creada.

Si el servidor seleccionado soporta características avanzadas como procedimientos almacenados y triggers de integridad referencial, ERwin provee editores de plantillas especiales y macros para automatizar el tedioso proceso de crear manualmente esas características en el servidor

Para nuestro caso a partir del diagrama Entidad-Relación, se generó utilizando Erwin el modelo de datos para Sybase, quedando de la siguiente manera:

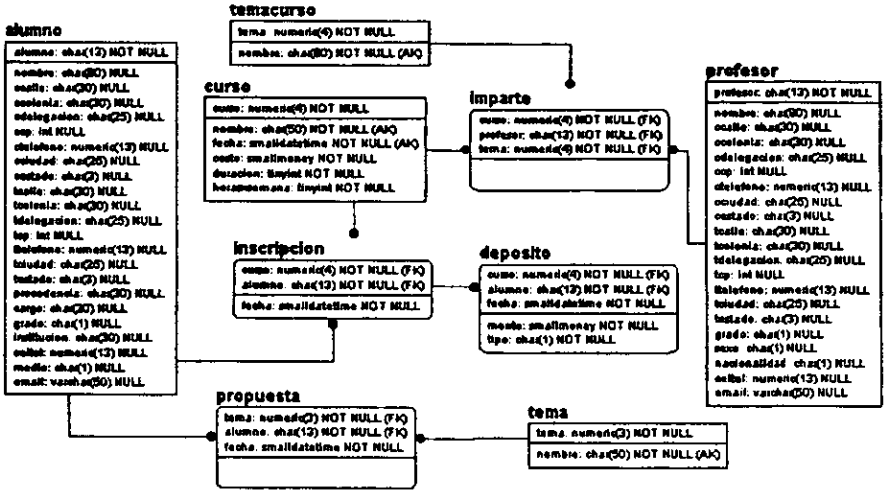


Como se puede observar corresponde totalmente el diagrama Entidad-Relación, pero con ERwin es posible definir con más detalle las características de la base de datos como a continuación se muestra.



El diagrama de la figura anterior nos muestra el modelo lógico de los datos, es decir, el diagrama entidad relación enriquecido por ERwin, donde se pueden apreciar los atributos de cada una de las entidades. También tenemos el modelo de físico el cual nos permite observar

detalles de cómo finalmente quedará la base de datos ya implementada.



En el modelo físico se pueden observar los siguientes elementos:

- Tablas.
- Campos llave y no llave.
  - Llaves primarias (PK), foráneas (FK) y alternas (AK).
- Tipo de datos de cada campo así como longitudes.
- Tipo de relación.

Cabe mencionar que Sybase cuenta con los tipos de datos siguientes:

Numéricos	
Int	Entre -2,147,483,648 y +2,147,483,6487
Smallint	Entre -32,768 y + 32,768
Tinyint	De 0 a 255
Float	Números de 8 bytes de punto flotante.

Shortfloat	Números de 4 bytes de punto flotante.
Money	Las columnas de moneda almacenan valores exactos entre +/- 922,377,203,685,447.5807 dólares con 4 lugares para decimales.
Double precision	Números de 8 bytes de punto flotante.
Carácter	
Char(n)	Columnas de caracteres (letras, números, símbolos), más de 255 caracteres de longitud.
Varchar(n)	Columnas de caracteres (letras, números, símbolos), más de 255 caracteres de longitud variable.
Gran objeto binario	
Text	Columnas de caracteres de longitud variable de más de 2 GB de longitud.
Image	Columna binaria de longitud variable de más de 2 GB de longitud.
Binary	Columna binaria de más de 255 bytes de longitud.
Varbinary(n)	Columna binaria de longitud variable, más de 255 bytes de datos binarios.
Miscelanea	
Bit	Columna de bits, soporta únicamente 0's o 1's.
Datetime	Fecha y hora del día con una precisión de 1/30 de milisegundo.
Shortdatetime	Fecha y hora del día con una precisión de 1 minuto.
Timestamp	Una columna de registro de tiempo se actualiza automáticamente cuando un registro es alterado.
Identity	Número de secuencia mantenido por el sistema.

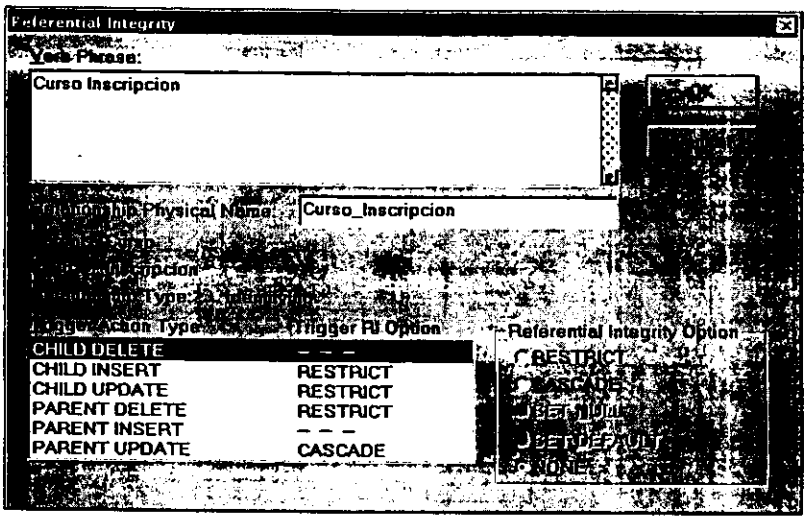
Algunas de las características que no se pueden observar en el diagrama pero que son de suma importancia para conservar la integridad y consistencia de la base de datos, son las reglas de validación, las cuales se pueden definir de manera global para toda la base de datos o de manera particular para algún campo de la tabla.

Para nuestro caso se utilizaron las siguientes reglas de validación:

- ru_clave: utilizada en las llaves de las tablas Curso, Tema y Temacurso para evitar que sean números negativos.
- ru_codigopostal: permite números entre 0 y 99999.
- ru_espacios: el campo debe contener caracteres, es decir, no se permiten sólo espacios (Sybase con el hecho de dar un espacio al campo ya lo interpreta como no nulo).
- ru_estado: dado que la longitud definida para el campo "estado" de la República Mexicana de trabajo o de domicilio particular de alumnos y profesores es de 3, la regla comprueba que la introducción de datos sea de manera abreviada, por ejemplo BCN: Baja California Norte; COL: Colima, etc.
- ru_grado: para "Grado académico" de Profesores, los datos permitidos son L, P, M y D ; Licenciatura, Posgrado, Maestría y Doctorado respectivamente.
- ru_medio: se permite C (Carteles), I (Internet), P (Periódicos) y T (Trípticos) creada para el campo "Medio" de la tabla Alumnos.
- ru_nacionalidad: valida la nacionalidad de los datos de los profesores (campo "nacionalidad"); permite M (Mexicana) o E (Extranjera).
- ru_rfc: comprueba que los primeros 4 caracteres del RFC sean letras y los siguientes 6 corresponden con el año, mes y día, es decir 6 dígitos.
- ru_sexo: permite M (Masculino) o F (Femenino).
- ru_telefono: números entre 9999 y 9999999999999.
- ru_tipo: para el tipo de depósito (Campo "Tipo" de la tabla "Depósito") sólo se permite B o J; Bancario o Jurisprudencia respectivamente.

La siguiente figura muestra datos adicionales de la base de datos como la Integridad Referencial (qué pasará cuando se inserten, borren o actualicen datos de las tablas dependientes (hijas), la cual es una de las partes más importantes para cualquier base de datos).

Desde ERwin se puede controlar la Integridad Referencial, la cual cuenta con las siguientes opciones:



La figura muestra la tabla padre (Curso) y la tabla hija (Inscripcion), las acciones posibles y las opciones respectivas. Así por ejemplo para el caso mostrado tenemos que:

1. Cuando se borra un registro de la tabla hija (CHILD DELETE) no se hace nada, es decir, se permite el borrado (NONE en Referential Integrity Option).

2. Cuando se inserta un registro en Inscripción (CHILD INSERT) se restringe (RESTRICT) a que éste exista en Curso (**I:R** en el dibujo).

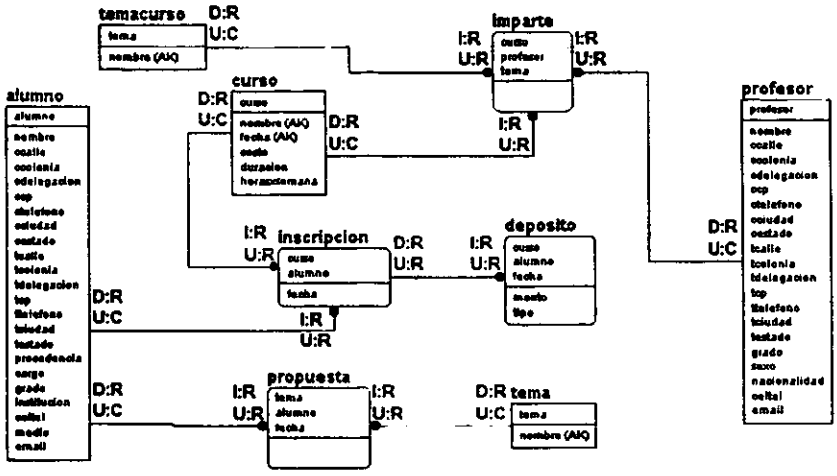
3. Cuando se quiera actualizar un registro en Inscripción (CHILD UPDATE) se restringe, es decir, no se puede hacer (**U:P** en el dibujo).

4. Ahora para la entidad padre (Curso); cuando se desea borrar un registro no se puede hacer a menos de que no tenga hijos (**D:R** en el dibujo).

5. Para cuando se quiere insertar un registro en la tabla padre (Curso), no se hace nada.

6. Finalmente, cuando se desea actualizar un padre (Curso), los hijos se actualizan automáticamente (Inscripción) (**U:C** en el dibujo).

Y así sucesivamente para cada una de las tablas, a través de sus relaciones se define la integridad referencial y con ella se asegura la integridad y consistencia de la base de datos.



Otras de las cualidades de ERwin es que permite introducir la documentación de la base de datos, tanto de entidades como de campos. Aprovechando lo anterior generamos también en ERwin el diccionario de datos, el cual se muestra a continuación de manera completa.

8.2. DICCIONARIO DE DATOS

Alumno **Código postal**	Código Postal del domicilio particular del Alumno.	ccp	int	NULL	ru_codigopostal
Alumno **Código postal del trabajo**	Código Postal del trabajo del Alumno.	tcp	int	NULL	ru_codigopostal
Alumno **Calle y número del trabajo**	Calle y número del trabajo del Alumno.	tcalle	char(30)	NULL	

Alumno **Calle y número de casa**	Calle y número del domicilio particular del Alumno.	ccalle	char(30)	NULL	
Alumno **Cargo**	Cargo que el Alumno desempeña donde trabaja.	cargo	char(20)	NULL	
Alumno **Ciudad**	Ciudad del domicilio particular del Alumno.	cciudad	char(25)	NULL	
Alumno **Ciudad del trabajo**	Ciudad donde trabaja el Alumno.	tciudad	char(25)	NULL	
Alumno **Colonia**	Colonia del domicilio particular del Alumno	ccolonia	char(30)	NULL	
Alumno **Colonia del trabajo**	Colonia del trabajo del Alumno.	tolonia	char(30)	NULL	
Alumno **Correo electrónico**	Dirección de correo electrónico para localizar al Alumno.	email	varchar(50)	NULL	
Alumno **Delegación**	Delegación, Población o Municipio del domicilio particular del Alumno.	cdelegacion	char(25)	NULL	
Alumno **Delegación del trabajo**	Delegación, Población o Municipio del trabajo del Alumno.	tdelegacion	char(25)	NULL	

Alumno **Estado**	Estado de la República Mexicana del domicilio particular del Alumno.	cestado	char(3)	"DF"	ru_estado
Alumno **Estado del Trabajo**	Estado de la República Mexicana donde trabaja el Alumno.	testado	char(3)	"DF"	ru_estado
Alumno **Grado académico**	Grado Académico del Alumno: Licenciatura, Posgrado, Maestría o Doctorado.	grado	char(1)	NULL	ru_grado
Alumno **Institución**	Institución donde el Alumno obtuvo el grado académico.	institucion	char(30)	NULL	
Alumno **Medio**	Medio por el cual se enteró el Alumno del curso.	medio	char(1)	NULL	ru medio
Alumno **Nombre completo**	Nombre completo del Alumno (Apellido Paterno, Apellido Materno y Nombre(s)).	nombre	char(80)	NULL	ru_espacios
Alumno **Procedencia**	Dependencia, Empresa o Institución donde trabaja el Alumno.	procedencia	char(30)	NULL	
Alumno **RFC del alumno**	Registro Federal de Causantes del Alumno.	alumno	char(13)	NOT NULL	ru_rfc
Alumno **Teléfono**	Teléfono del domicilio particular del Alumno.	ctelefono	numeric(13)	NULL	ru_telefono

Alumno **Teléfono celular**	Teléfono celular del Alumno.	celtel	numeric(13)	NULL	ru_telefono
Alumno **Teléfono del trabajo**	Teléfono del trabajo del Alumno.	ttelefono	numeric(13)	NULL	ru_telefono
Curso **Costo**	Costo del Curso.	costo	smallmoney	NOT NULL	
Curso **Curso**	Clave del Curso (Diplomado, Seminario, etc.).	curso	numeric(4)	NOT NULL	ru_clave
Curso **Duración**	Duración del Curso en horas.	duracion	tinyint	NOT NULL	
Curso **Fecha del curso**	Fecha de inicio del Curso.	fecha	smalldatetime	NOT NULL	
Curso **Horas por semana**	Horas por semana para el Curso.	horasxsemana	tinyint	NOT NULL	
Curso **Nombre del curso**	Nombre del Curso.	nombre	char(50)	NOT NULL	ru_espacios
Deposito **Curso**	Clave del Curso (Diplomado, Seminario, etc.).	curso	numeric(4)	NOT NULL	ru_clave
Deposito **Fecha**	Fecha del Depósito para el Curso.	fecha	smalldatetime	NOT NULL	

Deposito **Monto**	Cantidad de dinero del Depósito bancario para el Curso.	monto	smallmoney	NOT NULL	
Deposito **RFC del alumno**	Registro Federal de Causantes del Alumno.	alumno	char(13)	NOT NULL	ru_rfc
Deposito **Tipo de deposito**	Tipo del Depósito: [Bancario o Jurisprudencia].	tipo	char(1)	NOT NULL	ru_tipo
Imparte **Curso**	Clave del Curso (Diplomado, Seminario, etc.).	curso	numeric(4)	NOT NUL	ru_clave
Imparte **RFC del profesor**	Registro Federal de Causantes del Profesor.	profesor	char(13)	NOT NULL	ru_rfc
Imparte **Tema**	Clave del Tema para impartirse en los Cursos.	tema	numeric(4)	NOT NULL	ru_clave
Inscripcion **Curso**	Clave del Curso (Diplomado, Seminario, etc.).	curso	numeric(4)	NOT NULL	ru_clave
Inscripcion **Fecha**	Fecha en que se lleva a cabo la Inscripción.	fecha	smalldatetime	NOT NULL	
Inscripcion **RFC del alumno**	Registro Federal de Causantes del Alumno.	alumno	char(13)	NOT NULL	ru_rfc
Profesor **Código postal**	Código Postal del domicilio particular del Profesor.	ccp	int	NULL	ru_codigopostal

Profesor **Código postal del trabajo**	Código Postal del trabajo del Profesor.	tcp	int	NULL	ru_codigopostal
Profesor **Calle y número de casa**	Calle y número del domicilio particular del Profesor.	ccalle	char(30)	NULL	
Profesor **Calle y número trabajo**	Calle y número donde trabaja el Profesor.	tcalle	char(30)	NULL	
Profesor **Ciudad**	Ciudad donde vive el Profesor.	cciudad	char(25)	NULL	
Profesor **Ciudad trabajo**	Ciudad donde trabaja el Profesor.	tciudad	char(25)	NULL	
Profesor **Colonia**	Colonia del domicilio particular del Profesor.	ccolonia	char(30)	NULL	
Profesor **Colonia trabajo**	Colonia donde trabaja el Profesor.	tcolonia	char(30)	NULL	
Profesor **Correo electrónico**	Dirección de correo electrónico para localizar al Profesor.	email	varchar(50)	NULL	
Profesor **Delegación**	Delegación población o municipio del domicilio particular del Profesor.	cdelegacion	char(25)	NULL	

Profesor **Delegación trabajo**	Delegación, población o municipio donde trabaja el Profesor.	tdelegacion	char(25)	NULL	
Profesor **Estado**	Estado de la República Mexicana donde vive el Profesor.	cestado	char(3)	NULL	ru_estado
Profesor **Estado trabajo**	Estado de la República Mexicana donde trabaja el Profesor.	testado	char(3)	NULL	ru_estado
Profesor **Grado académico**	Grado Académico alcanzado por el Profesor [Licenciatura, Posgrado, Maestría o Doctorado].	grado	char(1)	NULL	ru_grado
Profesor **Nacionalidad**	Nacionalidad del Profesor: [Mexicano o Extranjero].	nacionalidad	char(1)	NULL	ru_nacionalidad
Profesor **Nombre completo**	Nombre completo del Profesor (Apellido Paterno, Apellido Materno y Nombre(s)).	nombre	char(80)	NULL	ru_espacios
Profesor **RFC del profesor**	Registro Federal de Causantes del Profesor.	profesor	char(13)	NOT NULL	ru_rfc
Profesor **Sexo**	Sexo del Profesor (Masculino o Femenino).	sexo	char(1)	NULL	ru_sexo
Profesor **Teléfono**	Teléfono del domicilio particular del Profesor.	ctelefono	numeric(13)	NULL	ru_telefono

Profesor **Teléfono celular**	Teléfono celular del Profesor.	celtel	numeric(13)	NULL	ru_telefono
Profesor **Teléfono del trabajo**	Teléfono del trabajo del Profesor.	ttelefono	numeric(13)	NULL	ru_telefono
Propuesta **Fecha**	Fecha en que el Alumno propone el Tema.	fecha	smalldatetime	NOT NULL	
Propuesta **RFC del alumno**	Registro Federal de Causantes del Alumno.	alumno	char(13)	NOT NULL	ru_rfc
Propuesta **Tema**	Clave del Tema para Propuesta.	tema	numeric(3)	NOT NULL	ru_clave
Tema **Nombre**	Nombre del Tema para Propuesta.	nombre	char(50)	NOT NULL	ru_espacios
Tema **Tema**	Clave del Tema para Propuesta.	tema	numeric(3)	NOT NULL	ru_clave
Temacurso **Nombre**	Nombre del Tema para impartirse en los Curso.	nombre	char(80)	NOT NULL	ru_espacios
Temacurso **Tema**	Clave del Tema para impartirse en los Cursos.	tema	numeric(4)		ru_clave

Una vez terminado el diagrama en ERwin es posible generar el código para el servidor destino, los cual es posible de dos maneras: una es sincronizándose con el servidor y ejecutar el código directamente, y la otra es guardando el código de la base en un archivo y ejecutándolo directamente en el SQL de Sybase.

Cualquiera de los caminos conduce al mismo resultado, para nuestro caso se seleccionó la primera opción, generando de esta manera la base de datos completa en Sybase.

El paso siguiente fue hacerle una serie de pruebas para comprobar si realmente se acoplaba a nuestras necesidades.

8.3. PRUEBAS REALIZADAS A LA BASE DE DATOS

Las pruebas que se realizaron a la base de datos son predominantemente de funcionalidad, es decir, de cómo funciona la base de datos (en Sybase) cuando se actualizan los datos (inserción, actualización y borrado) y fueron las siguientes.

Para las tablas independientes como: Alumno, Profesor, Tema, Tema en el curso y Curso

- Inserción, recuperación, borrado y actualización de datos a las tablas independientes.
- Comprobación de la nulicidad tanto en campos llave como en no llave.
- Inserción de datos coincidentes con el tipo de dato de cada una de las columnas de las tablas independientes.
- Inserción de datos contrastantes con el tipo de dato permitido en cada columna.
- Inserción de datos reales irreales.
- Comprobación de las reglas de validación adjuntas a cada campo

A tablas dependientes como: Deposito, Imparte, Inscripcion y Propuesta.

- Comprobación de la integridad referencial.
  - Inserción de datos cuando existen y no existen en la tabla padre.
  - Borrado de datos cuando existen y no existen en la tabla padre.
  - Actualización de datos.
- Comprobación de la nulicidad tanto en campos llave como en no llave.
- Inserción de datos coincidentes con el tipo de dato de cada una de las columnas de las tablas.
- Inserción de datos contrastantes con el tipo de dato permitido en cada columna.
- Inserción de datos reales irreales.
- Comprobación de las reglas de validación adjuntas a cada campo.

Las pruebas anteriores las realizamos desde el servidor (Sybase) en el prompt de SQL y en

el Cliente (PowerBuilder), ya que para nuestro caso el front-end tiene las herramientas necesarias para acceder a la base de los datos y su contenido sin ningún problema.

Posteriormente se procedió al diseño de pantallas, con lo cual se evidencia de manera clara si la comprensión de las necesidades del usuario son satisfechas con la nueva base.

8.4. DISEÑO DE PANTALLAS

En las aplicaciones modernas con interfaces gráficas, es el usuario el que gestiona la secuencia de eventos que se producen en un programa en ejecución. El usuario determina cuándo comienza la aplicación, cuándo se abre o se cierra una ventana, cuándo se introduce el siguiente campo o cuándo se finaliza la aplicación. En lugar del programa o el programador (como en la programación estructurada), es el usuario el que controla los eventos.

Una interfaz gráfica de usuario es un entorno orientado a objeto, dirigido a eventos. Las ventanas, así como los controles y otros objetos que aparecen en ellas, son objetos.

Los usuarios pueden provocar varios eventos. Los programas realizados con PowerBuilder responden a estos eventos. El comienzo de la respuesta al evento controla la operativa de la aplicación.

PowerBuilder tiene herramientas para la creación de las aplicaciones y sus objetos, para la conexión a las fuentes de datos y para poder responder a los eventos que van ocurriendo mientras la aplicación se está ejecutando. Todo esto permite construir aplicaciones industriales fiables, de manera rápida, fácil y eficiente.

Para poder realizar programas con PowerBuilder, quizá lo más difícil sea conocer los eventos y como gestionarlos, por tal razón, se deben conocer los diferentes eventos de cada objeto y cómo realizar procedimientos y funciones para dichos eventos.

Con PowerBuilder se pueden realizar aplicaciones que pueden ser ejecutadas en entornos diferentes al de Microsoft Windows, tales como HP/UX de Hewlett Packard, AIX de IBM, Solaris de Sun Microsystems, Macintosh System 7, Microsoft Windows NT y OS/2 de IBM. Se puede realizar una aplicación en un entorno y pasarla a cualquier otro, simplemente recompilándola en el nuevo entorno.

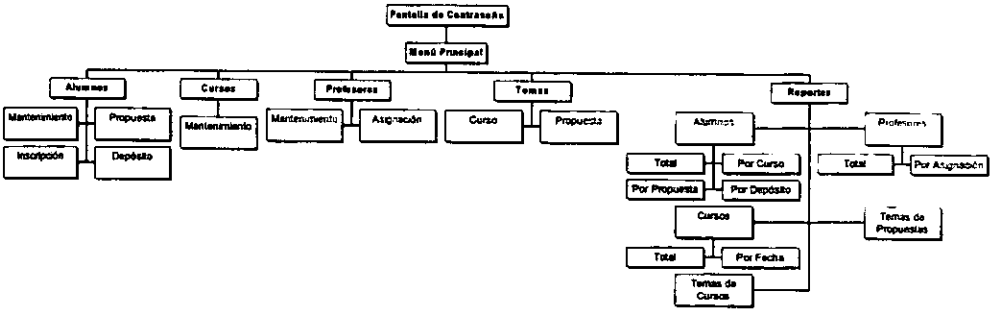
Con PowerBuilder normalmente se realiza poca programación. Lo que se programa son los procedimientos de los eventos asociados a los objetos.

La mayor parte del trabajo consiste en crear objetos para la aplicación, no en programar cómo reaccionarán los objetos. Las herramientas del entorno de desarrollo de PowerBuilder permiten crear fácilmente los objetos y crear los procedimientos.

La mayor parte del trabajo consiste en crear objetos para la aplicación, no en programar

cómo reaccionarán los objetos. Las herramientas del entorno de desarrollo PowerBuilder permiten crear fácilmente los objetos y escribir los procedimientos asociados.

Con respecto a nuestra aplicación, primero se diseñó la secuencia de navegación para el usuario, de tal manera que resultara lo más intuitiva posible quedando de la siguiente manera:



Con el esquema anterior se logra dar mantenimiento a toda la base de datos, así como generar reportes con los datos que requiere la División de Educación Continua de la Facultad de Derecho. En orden de aparición la secuencia es la siguiente:

MENÚ ALUMNOS

1. Mantenimiento: permite la actualización de alumnos (Consulta, Modificación, Borrado y Cambios), lo cual se hace directamente en la tabla ALUMNO.

2. Propuesta: permite introducir las propuestas de los alumnos para futuros cursos, al igual que para la anterior, en ésta se encuentran disponibles los modos de actualización. La tabla principal para este caso es PROPUESTA aunque se requieren datos de ALUMNO y TEMA.

3. Inscripción: con esta pantalla se logra la actualización de las inscripciones. La tabla principal es INSCRIPCION y las tablas de apoyo ALUMNO y CURSO.

4. Depósito: después de la inscripción los alumnos deben pagar por el curso. El control de los depósitos se hace en esta opción, gravándose en la tabla DEPOSITO y recuperando datos de INSCRIPCION, ALUMNO y CURSO.

MENÚ CURSOS

1. Mantenimiento: en está pantalla se actualizan los datos referentes a los cursos, esto se hace directamente de la tabla CURSO.

MENÚ PROFESORES

1. Mantenimiento: al igual que para alumnos esta opción da mantenimiento a profesores directamente de la tabla PROFESOR.

2. Asignación: permite definir los temas que los profesores impartirán en los cursos. La tabla principal es IMPARTE, recuperando datos de PROFESOR, TEMACURSO y CURSO.

MENÚ TEMAS

1. Curso: en esta opción se actualizan los datos de los temas para los cursos los cuales se utilizan para la asignación. La tabla que se utiliza es TEMACURSO.

2. Propuesta: la segunda opción de este menú sirve para actualizar los temas de las propuestas, lo hace directamente de la tabla TEMA.

MENÚ REPORTES

Además de la posibilidad de tener un control de los datos que genera la División de Educación Continua, otro de los requerimientos de gran importancia es el de la generación de reportes que permitan concentrar los datos de acuerdo a cierto criterio definido. La intención del menú de reportes es permitir a los usuarios recuperar los datos de manera rápida, fácil y confiable; por ejemplo, aspirantes a los nuevos cursos, temas que un profesor ha impartido, alumnos que han cubierto la totalidad del curso, entre otros. Las opciones disponibles son las siguientes:

ALUMNOS

1. Total: genera un reporte de todos los alumnos que se encuentran dados de alta en el Sistema de Control de Cursos.

2. Por Curso: concentra los alumnos de acuerdo al curso al cual se inscribieron.

3. Por Propuesta: hace una agrupación de alumnos de acuerdo a los temas que han propuesto para los cursos.

4. Por Depósito: muestra el estatus de los pagos de los alumnos de acuerdo al curso al que se inscribieron.

PROFESORES

1. Total: genera un lista de todos los profesores que existen en la base de datos.

2. Por Asignación: agrupa los profesores de acuerdo al tema que han impartido en el curso.

CURSOS

1. Total: genera una lista de todos los cursos que se han impartido en la División de Educación Continua.

2. Por Fecha: agrupa los datos de los cursos de acuerdo a la fecha.

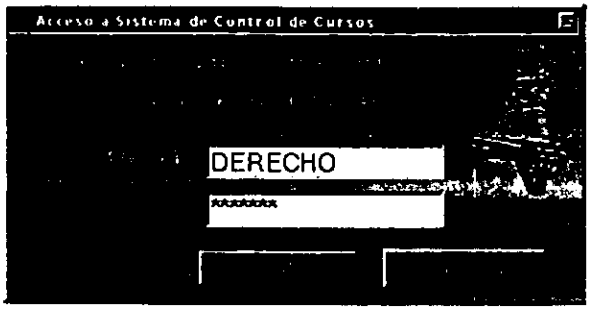
TEMAS DE CURSO

1. Genera la lista de todos los temas disponibles para los cursos.

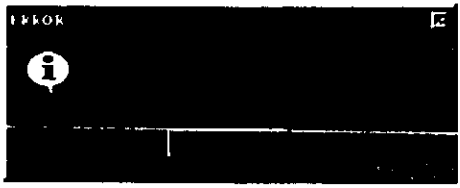
TEMAS DE PROPUESTAS

1. Genera una lista de todos los temas (propuestas) para los cursos que se encuentran disponibles.

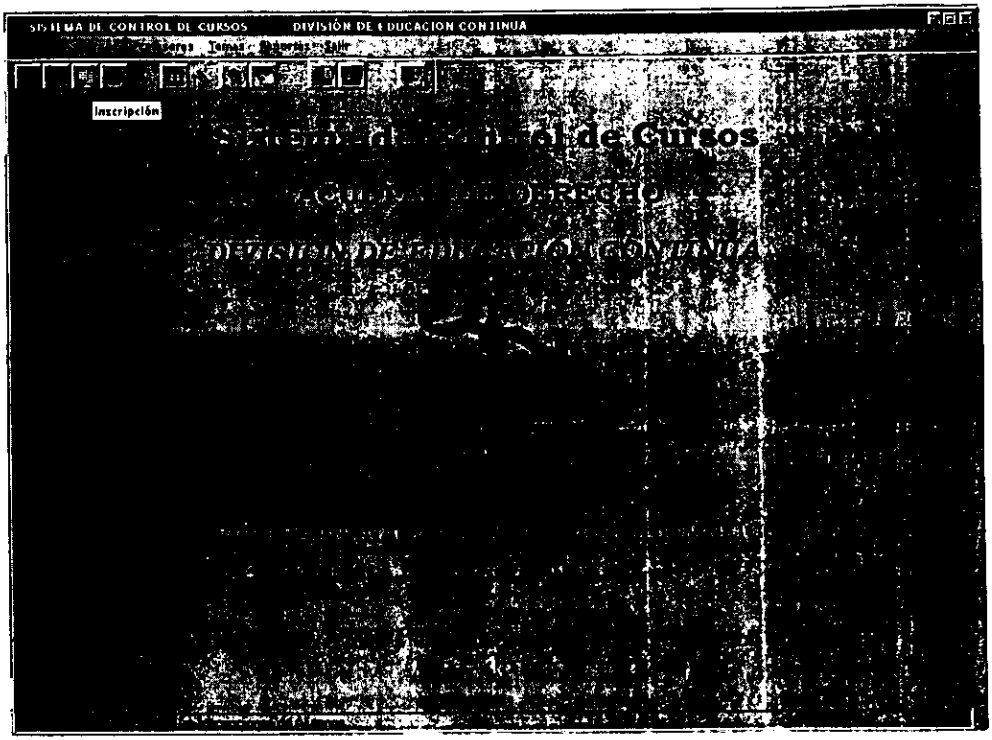
La primer pantalla que aparece al ejecutarse la aplicación es la que a continuación se muestra.



Con ella se restringe el acceso al sistema, de tal manera que sólo el personal autorizado tenga acceso a los datos. En caso de que se introduzca la clave de usuario y la contraseña adecuada aparecerá el menú principal, de lo contrario aparece la siguiente ventana:



MENÚ PRINCIPAL



La pantalla anterior permite accesar a las opciones disponibles del sistema. Para ejecutar una pantalla o ventana en particular se puede hacer a través de los menús seleccionando la opción adecuada o por medio de los botones de la barra de herramientas que aparece en la esquina superior izquierda. A cada uno de los botones se encuentra asociada una de las opciones de los menús (con excepción de los reportes). Cuando se coloca el mouse en el botón aparece la ayuda asociada a éste tanto en la barra de estado como en la parte inferior del botón.

Todas las pantallas permiten la actualización de los datos, es decir, alta, baja y cambios, para lograr esto cada ventana contiene un grupo de botones con las siguientes características:

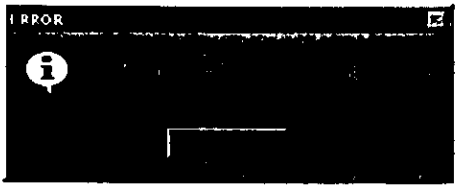
Limpia el contenido de los campos y permite la introducción de un registro nuevo.

Permite cambiar un registro (contiene una ventana asociada).

[redacted] Permite borrar un registro (contiene una ventana asociada).

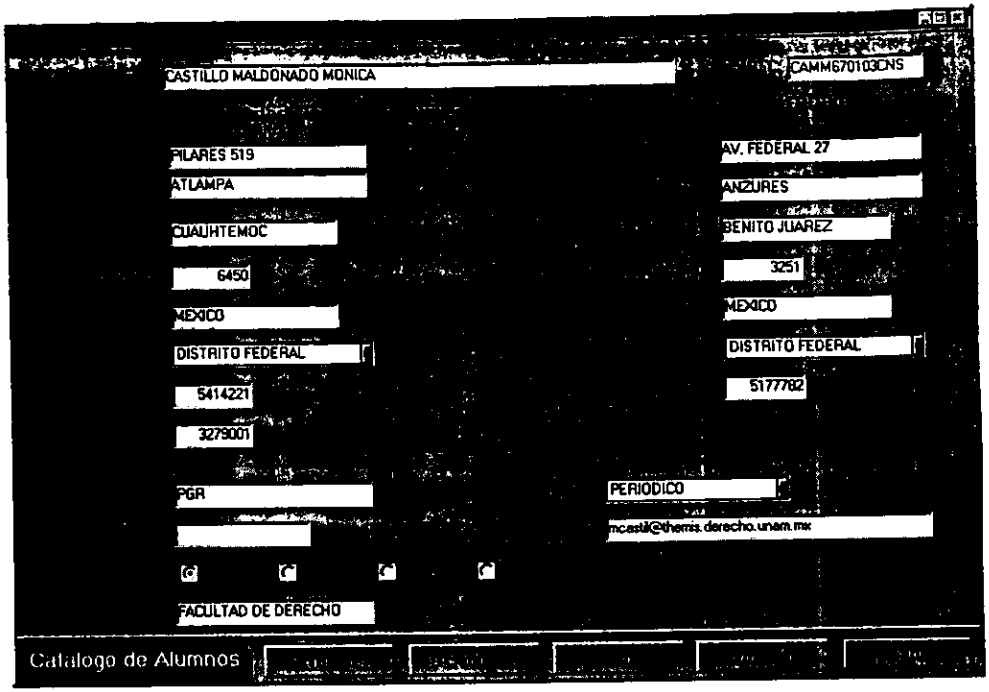
[redacted] Guarda el registro con los datos que se encuentran en la pantalla. Siempre se desplegará el siguiente mensaje:

Cabe mencionar que para todos las pantallas en caso de faltar datos como las llaves aparece un mensaje que informa de la omisión.



[redacted] Cierra la ventana actual.

Tomando como ejemplo la opción de Mantenimiento de Alumnos tenemos los siguiente:

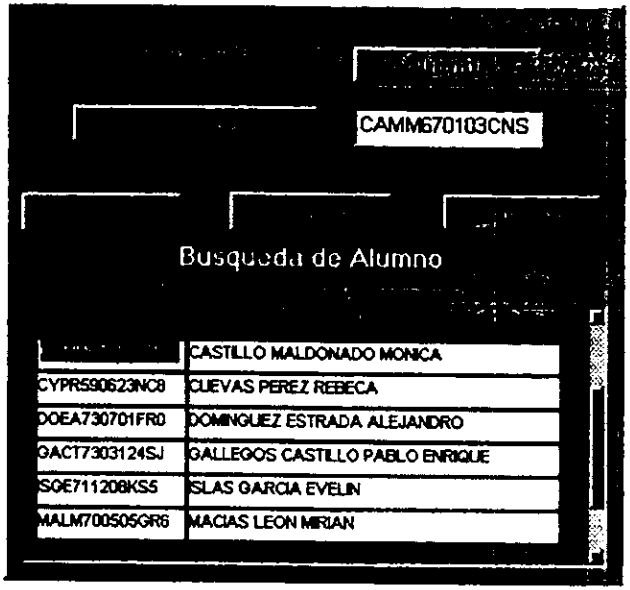


Para las opciones de cambio y baja, existe una ventana asociada a cada uno de estos botones que permite la búsqueda y selección de un registro. En ésta se tiene que introducir primeramente el criterio de búsqueda que servirá de filtro al presonar el botón Buscar; teniendose los siguientes casos:

- Cuando el campo filtro (RFC) se deja en blanco, recupera todos los registros existentes.
- Cuando el campo filtro (RFC) contiene parte de la llave, muestra una lista a partir de dicho criterio de búsqueda.
- Por último, en caso de proporcionar toda la llave aparece una lista que comienza con ese registro.

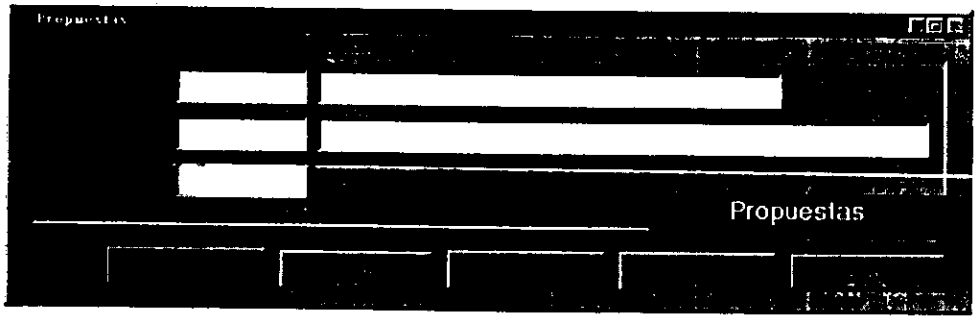
Para que finalmente aparezcan los datos de un registro específico se debe seleccionar la llave (RFC) con el mouse y presionar el botón Aceptar, con esto los datos aparecerán en la pantalla como se muestra en la figura anterior.

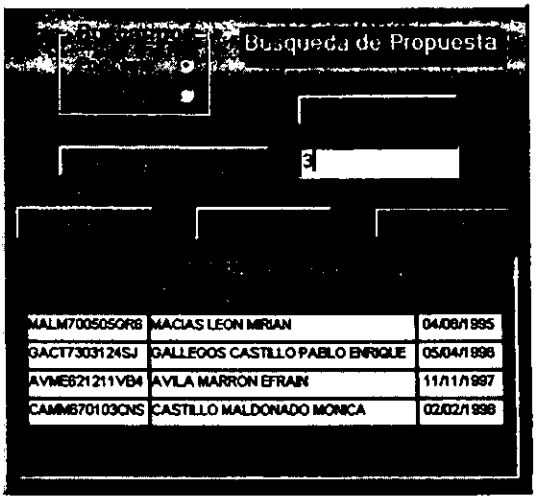
La ventana asociada se muestra a continuación.

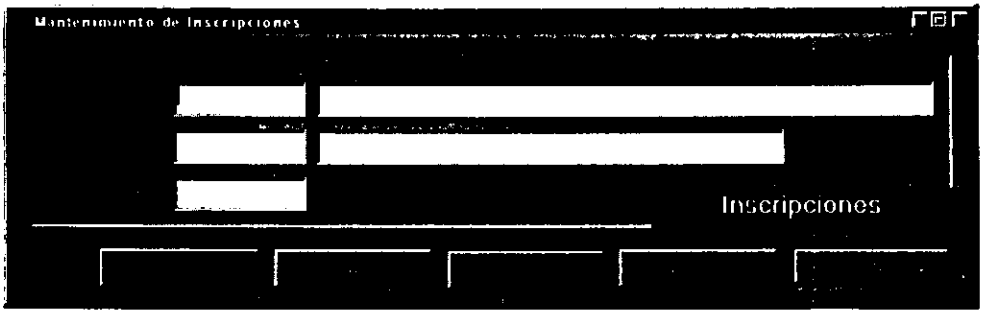


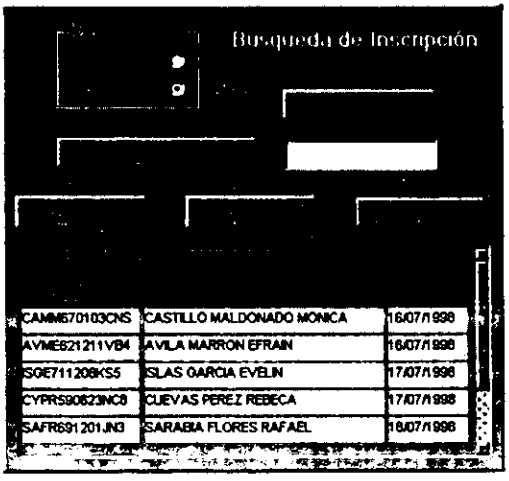
Para abortar cualquier operación no deseada en algún lugar determinado del sistema se puede precionar el botón Cancelar o Salir.

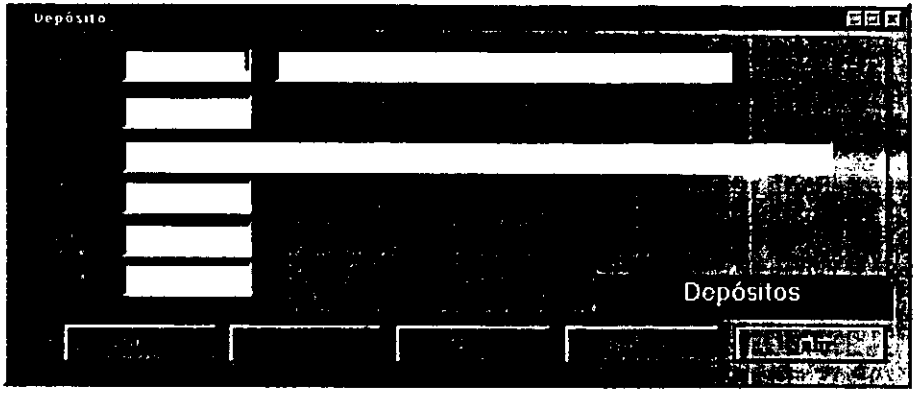
Las pantallas del resto del sistema son similares en cuanto a presentación y funcionalidad.

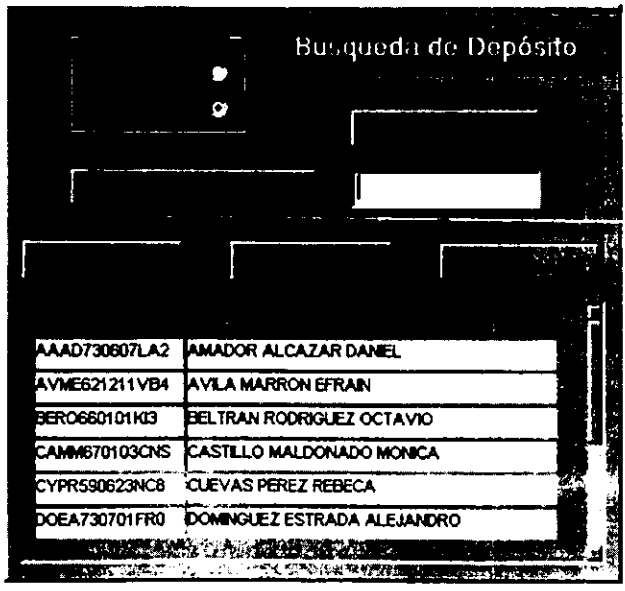



Busqueda de Propuesta
3
MALM700505QR8 MACIAS LEON MIRIAN 04/08/1995
GACT7303124SJ GALLEGOS CASTILLO PABLO ENRIQUE 05/04/1998
AVME621211VB4 AVILA MARRON EFRAIN 11/11/1997
CAMM670103CNS CASTILLO MALDONADO MONICA 02/02/1998

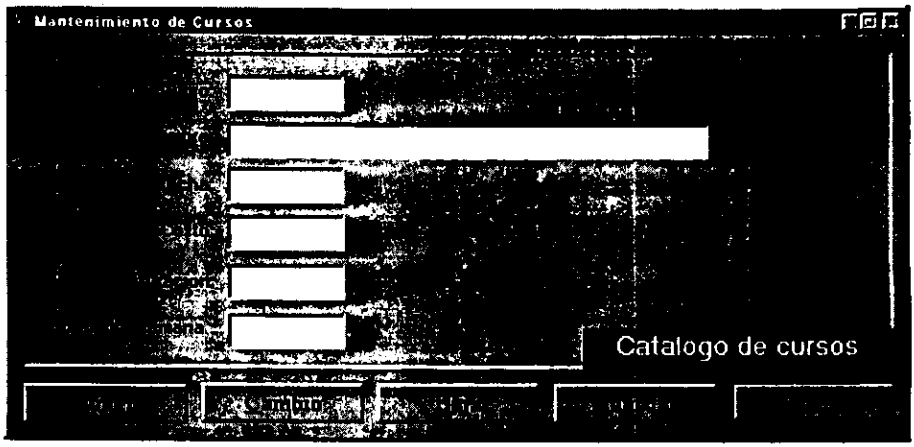

Mantenimiento de Inscripciones
Inscripciones

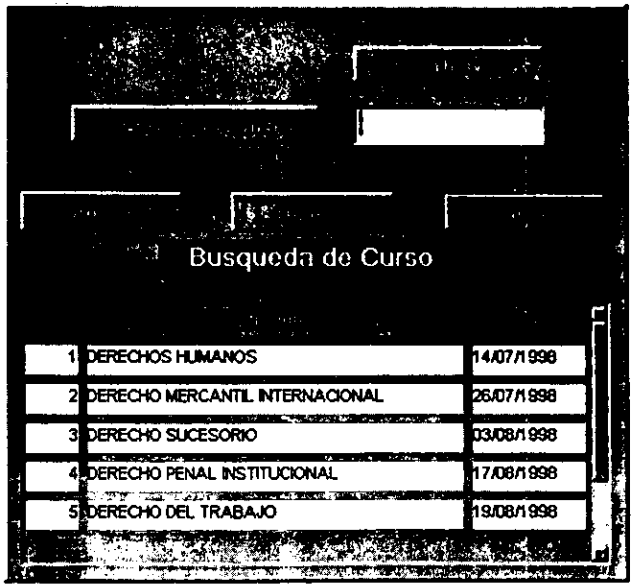

Busqueda de Inscripción
CAMM670103CNS CASTILLO MALDONADO MONICA 16/07/1998
AVME621211VB4 AVILA MARRON EFRAIN 16/07/1998
SGE711208KS5 ISLAS GARCIA EVELIN 17/07/1998
CYPR590623NC8 CUEVAS PEREZ REBECA 17/07/1998
SAFR691201JN3 SARABIA FLORES RAFAEL 16/07/1998


Depósito
Depósitos

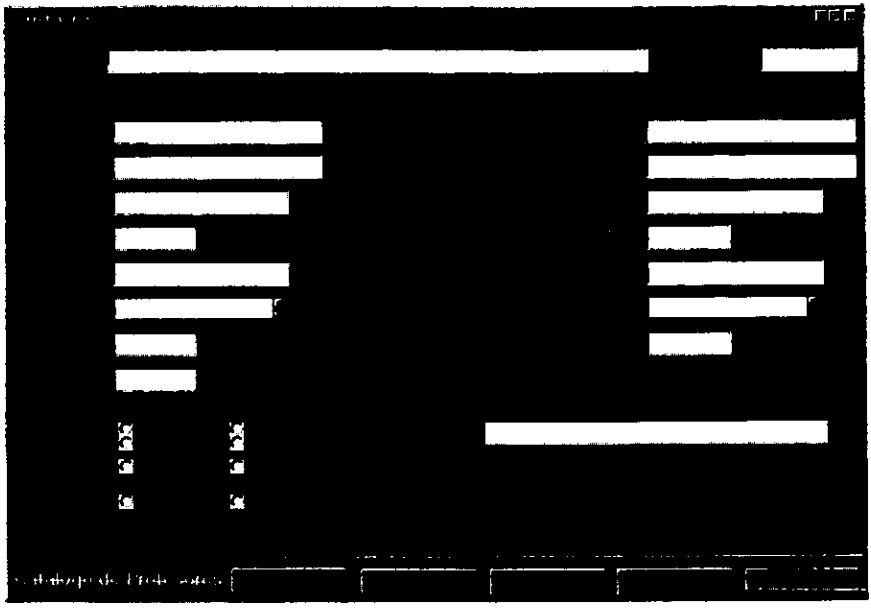

Busqueda de Depósito
AAAD730607LA2 AMADOR ALCAZAR DANIEL
AVME621211VB4 AVILA MARRON EFRAIN
BERO660101KI3 BELTRAN RODRIGUEZ OCTAVIO
CAMM670103CNS CASTILLO MALDONADO MONICA
CYPR590623NC8 CUEVAS PEREZ REBECA
DOEA730701FR0 DOMINGUEZ ESTRADA ALEJANDRO

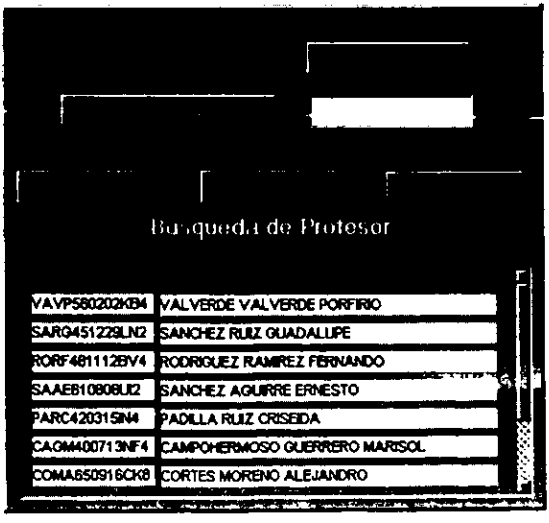
MENÚ CURSOS

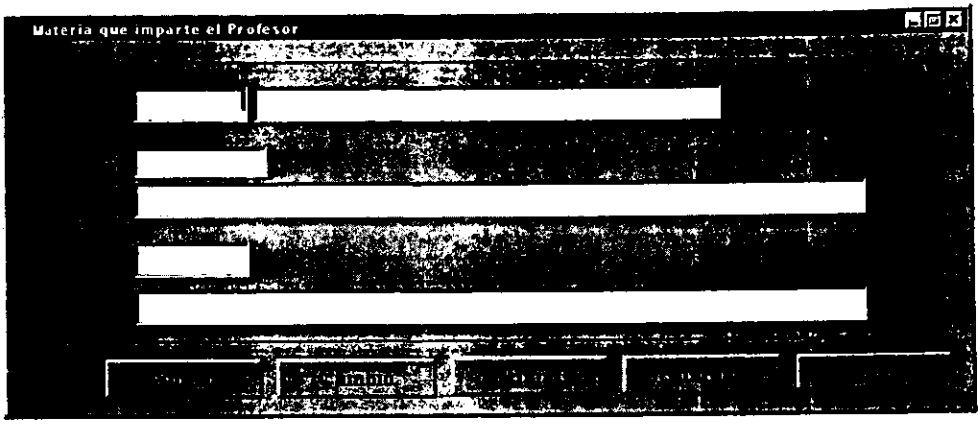

Mantenimiento de Cursos
Catalogo de cursos

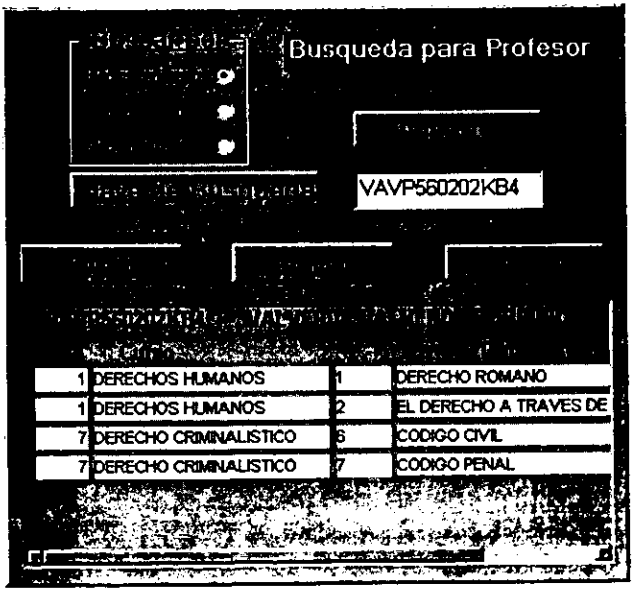

Busqueda de Curso
1 DERECHOS HUMANOS 14/07/1998
2 DERECHO MERCANTIL INTERNACIONAL 26/07/1998
3 DERECHO SUCESORIO 03/08/1998
4 DERECHO PENAL INSTITUCIONAL 17/08/1998
5 DERECHO DEL TRABAJO 19/08/1998

MENÚ PROFESORES

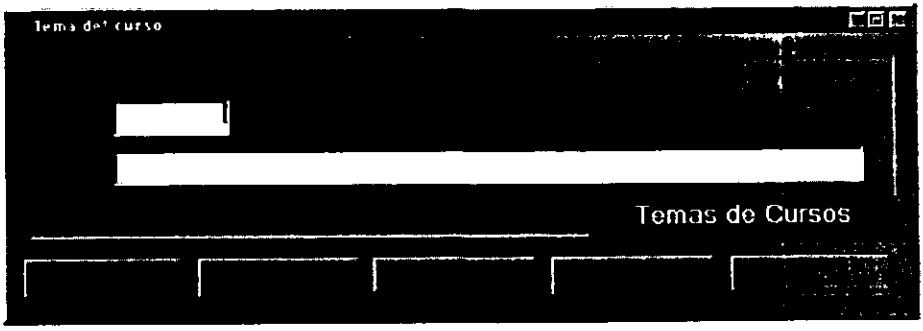


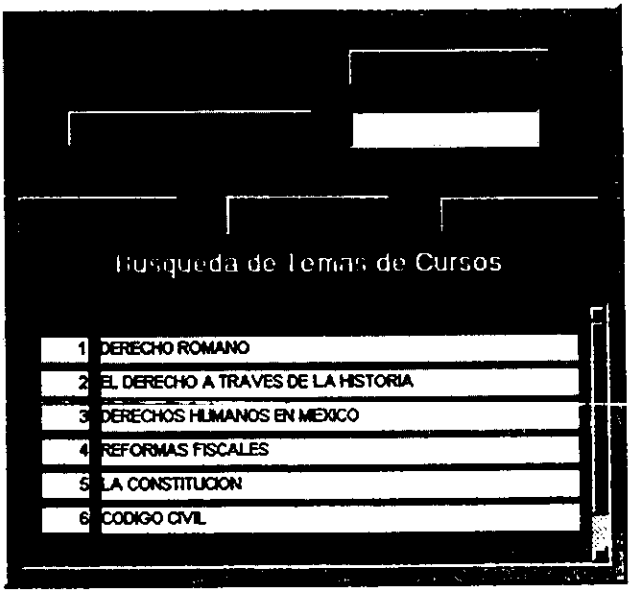


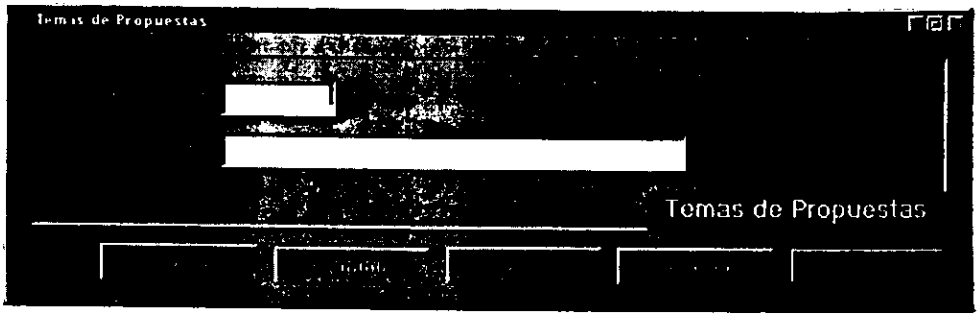

Materia que imparte el Profesor

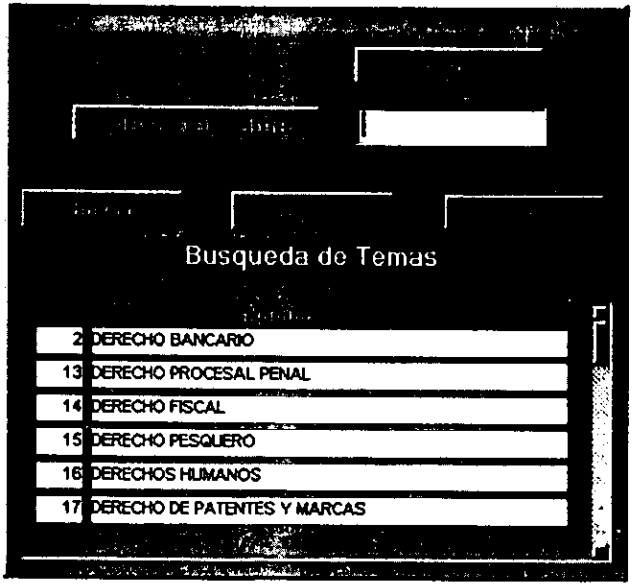

Busqueda para Profesor
VAVP580202KB4
1 DERECHOS HUMANOS
1 DERECHOS HUMANOS
7 DERECHO CRIMINALISTICO
7 DERECHO CRIMINALISTICO
1 DERECHO ROMANO
2 EL DERECHO A TRAVES DE
6 CODIGO CIVIL
7 CODIGO PENAL

MENÚ TEMAS


Temas del curso
Temas de Cursos


Busqueda de Temas de Cursos
1 DERECHO ROMANO
2 EL DERECHO A TRAVES DE LA HISTORIA
3 DERECHOS HUMANOS EN MEXICO
4 REFORMAS FISCALES
5 LA CONSTITUCION
6 CODIGO CIVIL


Temas de Propuestas
Temas de Propuestas

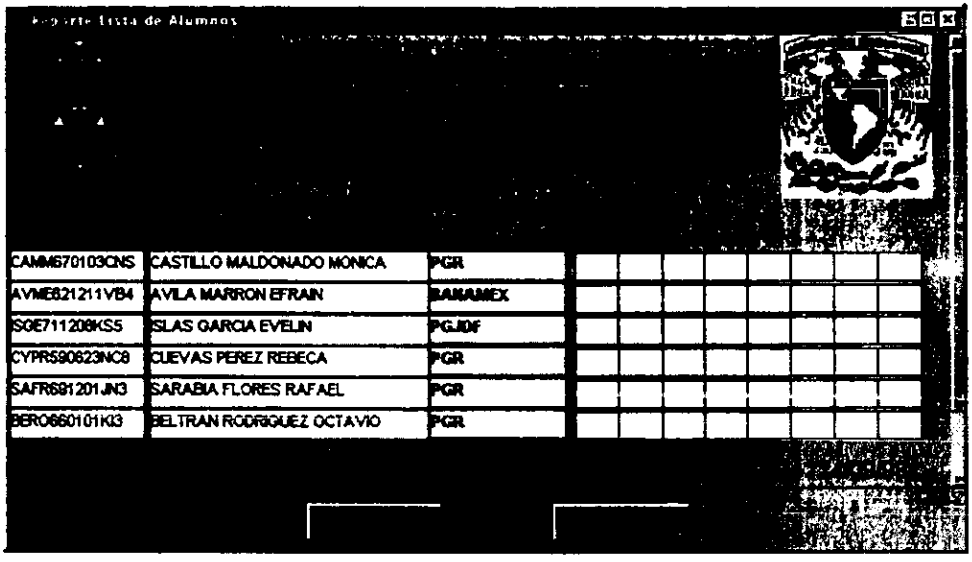

Busqueda de Temas
2 DERECHO BANCARIO
13 DERECHO PROCESAL PENAL
14 DERECHO FISCAL
15 DERECHO PESQUERO
16 DERECHOS HUMANOS
17 DERECHO DE PATENTES Y MARCAS

REPORTES

Finalmente queda el menú de reportes y a continuación se muestra el formato que todos ellos llevan.

La siguiente figura muestra la lista de los Alumnos inscritos al curso: 1 Derechos Humanos.

Es posible imprimir el reporte si así de desea.



Las técnicas en el estudio de sistemas han aumentado su importancia y sus aplicaciones en los últimos años; razón para ello ha estado en la demanda siempre creciente de la utilización de grandes masas de datos, frecuentemente integrados y con requisitos de alta productividad en su manejo, tanto en el desarrollo de aplicaciones por gestores de proyectos (analista, programador y personas encargadas de las pruebas de aceptación y de control de calidad), como en la parte de usuarios finales. De acuerdo a lo anterior decidimos dividir nuestras conclusiones en dos rubros: los usuarios finales y los gestores de proyectos.

LOS USUARIOS

Con el presente trabajo se logró en gran medida, determinar hacia y en donde estaba todo lo necesario para que la División de Educación Continua, como usuario de todo este trabajo, controlara de una manera más sencilla, rápida y segura la gestión del proceso de impartir cursos.

Como se mencionó en una etapa muy temprana de esta tesis, todo buen sistema que pretende el uso de una computadora, no intenta volver esclavo a todo el trabajo de la misma y si además se involucra a gente acostumbrada a realizar su trabajo en la forma que siempre lo ha hecho ("manualmente"), entonces es necesario tener cuidado de no cambiarles completamente su modo de hacer las cosas sino ayudarles a realizarlas más fáciles y gestionables, sin que consideren no cambiar por temor al nuevo estilo de trabajar. Y eso fue el objetivo original y que finalmente alcanzamos al ver cómo el usuario aceptó el sistema y cómo el sistema se integraría al trabajo cotidiano.

Este estudio permitió eliminar vicios que hacían a veces más lento el proceso o inclusive la pérdida de información de vital importancia en la toma de decisiones, como lo era la agenda de profesores y alumnos a la hora de la creación de nuevos cursos. Primero en la planeación de los mismos, determinar qué profesores podrían ser invitados en la cátedra resultó ser más rápido y confiable, además de eliminar por completo el uso de enormes cantidades de papel distribuido en archiveros. De igual forma resultó para la invitación de alumnos a los nuevos curso, la cartera de los mismos en un medio electrónico resultó más fiable y rápida para el usuario.

La gestión de los cursos, con todo lo relacionado a ellos (alumnos, profesores, depósitos de alumnos, alumnos inscritos, propuestas, etc.) se vio extraordinariamente apoyado con el Sistema de Control de Cursos, debido a que se tuvo mayor control sobre el manejo de la información, entendiendo por información los resultados de los datos procesados.

LOS GESTORES DE PROYECTO

En este caso nos referimos a nosotros como responsables de principio a fin de todo el proyecto.

Uno de los principales aspectos que primero se desprende del presente trabajo es haber concebido el tipo de sistema que abordamos. Nos referimos a un "Sistema en línea".

El ciclo de vida del proyecto nos ayudó en gran medida, a seguir una disciplina para el estudio de todo lo que se involucra en un proyecto.

Parte esencial del ciclo de vida lo son las entrevistas las cuales representan el primer contacto con el usuario que requiere el sistema. Es necesario tener una visión tanto de usuario, como de analista y desarrollador, para poder comprender y concebir el nuevo sistema, así como para poder detectar cualquier anomalía en el momento oportuno.

Coincidimos con la gente dedicada al desarrollo de sistemas en el sentido de que la fase de análisis es una de las más importantes, debido a que aquí fue donde delimitamos la frontera entre nuestro sistema y el resto del universo, y la manera en que interactúan ambos. De igual forma nos ayudó a comprender completamente la actividad del sistema al percibir el flujo de datos y la información que estos generaban dentro del mismo.

La propuesta inicial del sistema comprendía el desarrolló del módulo de Internet. Más tarde la Facultad de Derecho decidió que la administración de todo lo relacionado con el Web lo realizarían ellos, sin embargo, existe el análisis respectivo para que un futuro se pueda integrar sin ningún problema al resto del sistema. Con esto se comprueba que para el análisis se debe de considerar una tecnología perfecta, lo cual hace posible la concepción de un sistema con miras al futuro.

Otro de los aspectos de suma importancia es la utilización, en la medida de lo posible, de herramientas que permitan agilizar el ciclo de vida del proyecto, tal es el caso del CASE ERwin, el cual ayuda a conseguir una visión completa de la base de datos en cuestión y además de una mejor comprensión y fundamentación del trabajo que se esté realizando.

Finalmente, un punto a destacar dentro de cualquier base de datos es la integridad y la consistencia de los datos, por tal motivo, es preciso definir de manera clara y correcta la integridad referencial sin olvidar las reglas de validación y realizar las pruebas pertinentes sin escatimar en tiempo.

Las reglas de validación y la integridad referencial las determinamos en el Back-End (Sybase) debido a que éste es el DBMS y el encargado de la manejar los datos; qué mejor para manejar esa serie de reglas. Aunque lo anterior se puede hacer con el Front-End (Power Builder para nuestro caso), es totalmente recomendable hacerlo en el Back-End, ya que el carecer de estos aspectos de suma importancia hace a la base de datos presa fácil de cualquier eventualidad; por el contrario, cuando se definen de manera adecuada, impide cualquier anomalía no prevista por los responsables del sistema y asegura la integridad y la consistencia.

Apéndice A

FORMAS NORMALES

DEPENDENCIA FUNCIONAL

Generalmente, este es el punto de partida real del diseño de una nueva aplicación.

Se dice que el atributo o conjunto de atributos B **depende funcionalmente** del atributo o conjunto de atributos A, y se representa como $A \rightarrow B$ o $A \ _{DF} \ B$, sí y sólo sí, cada valor de A le corresponde (a nivel conceptual) un único valor de B.

Todo el proceso de normalización está basado en distintas variantes de la dependencia funcional. Lo primero que se debe conocer, por tanto, son las DF existentes entre los atributos y sobre todo aquellas relevantes que relacionan atributos de distintas entidades dando lugar a relaciones.

Es importante resaltar que las dependencias funcionales relevantes, que son las que llevan de forma implícita la complejidad del problema, son aquéllas que relacionan las llaves de las entidades que se tratan, y por consiguiente, éstas deben ser cuidadosamente analizadas y estudiadas, así como aprobadas por los usuarios.

DEPENDENCIA FUNCIONAL TOTAL

Se dice que el atributo Y tiene **una dependencia funcional total** con el atributo X, si tiene una dependencia funcional con X y **no** depende funcionalmente de ningún subconjunto de X.

Las dependencias de interés para tratar los problemas son siempre las dependencias funcionales totales, y sólo ellas servirán para encontrar la solución.

A continuación se estudiará el grafo de las dependencias para mostrar de un modo gráfico la relación que existe entre todos los atributos. A partir de él, se realizará la normalización de un modo más sencillo.

Grafo de las dependencias funcionales

Teniendo como base el modelo entidad-relación, se debe tener claramente identificada la información que se quiere incluir en cada entidad y en cada relación. Esa información, que serán atributos, será estudiada más a fondo para obtener las dependencias funcionales que aparecen a simple vista. Por último, se debe identificar la llave de las entidades y relaciones.

El grafo de las DF es una forma clara de tener una visión general de los datos y de la cohesión existente entre ellos. La llave que se obtiene de tratar todas las DF, se representa dentro de una caja de líneas continuas con sus atributos primarios. El resto de los atributos se representa fuera de la caja, y después las dependencias entre todos ellos.

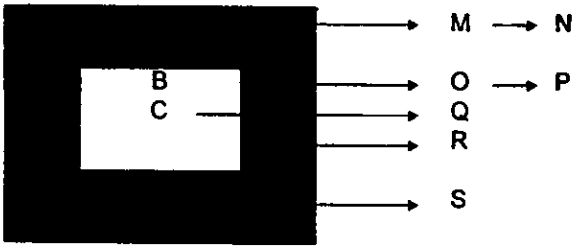
Si existen dependencias de varios atributos, que no son la llave, irán en una caja de líneas

discontinuas para no confundirlo con la propia llave de la tabla.

Por ejemplo, sean las dependencias:

A.B.C → M| N| S
M → N
B.C → O| P| R
O → P
C → Q

El grafo asociado es:



DEPENDENCIA FUNCIONAL TRANSITIVA

La dependencia funcional transitiva se aplica para analizar las tablas en tercera forma normal (3FN). Consiste básicamente en considerar que "un atributo no primario sólo debe conocerse a través de la llave principal o llaves secundarias". En otro caso, se estará produciendo redundancia de información con las anomalías típicas que lleva consigo.

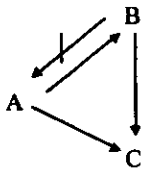
La definición formal de la dependencia funcional transitiva dice así:

Sean tres subconjuntos distintos de atributos A, B y C pertenecientes a una tabla T, de tal modo que se cumplen las condiciones:

$A \xrightarrow{DF} B$ y $B \xrightarrow{no\ DF} A$

Se dice que C tiene una **dependencia funcional transitiva** con A o que es transitivamente dependiente de A si se cumple que B → C.

Gráficamente se tiene:



Por tanto, un atributo (C) es transitivamente dependiente de otro (A) si se conoce por diferentes vías, una directamente y otra a partir de otro atributo intermedio (en nuestro caso, B).

PRIMERA FORMA NORMAL (1 FN)

Una tabla se dice que está en 1FN, sí y sólo sí, los valores que componen el atributo de una tupla son atómicos. Es decir, en un atributo no deben aparecer valores repetitivos y por tanto tienen que ser elementales y únicos.

Este era un defecto típico que se encontraba en los archivos cuando para un valor se podían encontrar un conjunto de valores de otro atributo. El problema se observa en los siguientes aspectos:

- La falta de espacio en el campo para los valores que puedan aparecer o por el contrario, el desaprovechamiento del atributo cuando existen pocos valores.
- La dificultad del tratamiento para actualizaciones, consultas y búsquedas de un valor determinado.

Si para la normalización de algunos datos, éstos no están mecanizados y todos los datos se introducen por primera vez a la computadora, entonces simplemente se debe tener en cuenta la condición formulada por la primera forma normal: "un atributo sólo debe mantener valores elementales o únicos".

Para que una tabla que no se encuentra en 1FN, se pase a 1FN se procede de la siguiente manera:

1. Se localizan los atributos que forman parte de la llave principal.
2. Se descompone la tabla realizando una proyección y se obtiene:
   a) La llave con los atributos que tienen valores únicos.

b) Otra tabla con la llave y los atributos que tienen valores múltiples (teniendo en cuenta que ahora sus valores múltiples se distribuirán cada uno de ellos en una tupla y por tanto en cada tupla existirá un solo valor elemental).

SEGUNDA FORMA NORMAL (2FN)

Una tabla se dice que está en **2FN**, si y sólo si, cumple dos condiciones:

- Se encuentra en 1FN.
- Todo atributo secundario (aquéllos que no pertenecen a la llave principal, los que se encuentran fuera de la caja) depende totalmente (tiene una dependencia funcional total) de la llave completa, y por tanto, no de una parte de ella.

Esta forma normal sólo se considera si la llave principal es compuesta (formada por varios atributos), ya que cuando la llave principal de la tabla está formada por un único atributo, entonces la tabla ya se encuentra en 2FN.

Si una tabla T tiene como atributos A, B, C, D y la llave es A.B cumpliéndose las dependencias:

$A.B \rightarrow C$
$B \rightarrow D$

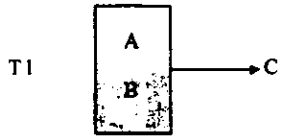
Se observa que la tabla no se encuentra en 2FN, puesto que el atributo D no tiene una dependencia funcional total con la llave entera A.B, sino con una parte de la llave (B).

Veamos la gráfica de las dependencias:

Si existe una flecha que parte del interior de la caja que encierra a la llave, entonces la tabla no está en 2FN.

Para convertir una tabla que no está en segunda forma normal a 2FN, se realiza una proyección y se crea:

1. Una tabla con la llave y todas sus dependencias totales con los atributos secundarios afectados:



2. Otra tabla con la parte de la llave que tiene dependencias, junto con los atributos secundarios implicados:

La llave de la nueva tabla será la antigua parte de la llave.

TERCERA FORMA NORMAL (3FN)

Una tabla se dice que está en **tercera forma normal**, sí y sólo sí, se cumplen dos condiciones:

- Se encuentra en 2FN.
- No existen atributos no primarios que son transitivamente dependientes de cada posible clave de la tabla.

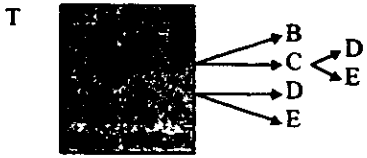
Esto quiere decir, que un atributo secundario sólo se debe conocer a través de la llave principal o llaves secundarias de la tabla y no por medio de otro atributo no primario.

En el grafo de dependencias sólo se deben mostrar las dependencias transitivas y no aquellas dependencias funcionales a partir de las llaves secundarias, puesto que se sabe que por ser llaves ya se conocen todos los atributos.

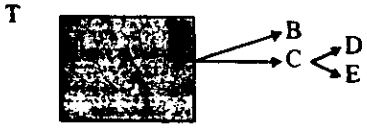
Se mostrará la 3FN por medio de un ejemplo; con los atributos A, B, C, D y E; donde A es la llave principal, B es llave secundaria y existen las siguientes dependencias:

$A \longrightarrow B$	$B \longrightarrow A$	$C \longrightarrow D$
$A \longrightarrow C$	$B \longrightarrow C$	$C \longrightarrow E$
$A \longrightarrow D$	$B \longrightarrow D$	
$A \longrightarrow E$	$B \longrightarrow E$	

La gráfica queda del siguiente modo:



O bien:



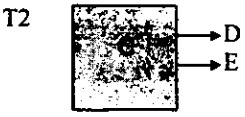
Las flechas que muestran las DF que tiene la llave secundaria no se representan porque son evidentes y no simplifican la visión de la gráfica. Además, para la normalización, no se necesitan; por el contrario, suelen complicar al análisis.

Evidentemente, la tabla T no está en 3FN puesto que los atributos D y E son transitivamente dependientes respecto de la llave A.

De tal forma, para pasar una tabla que no cumple la tercera forma normal a 3FN, se realiza una proyección y se genera:

1. Una tabla con la clave y todos los atributos no primarios que no son transitivos:

2. Otra tabla con los atributos transitivos y el atributo no primario (que será la llave de la nueva tabla) por medio del cual mantienen la transitividad:



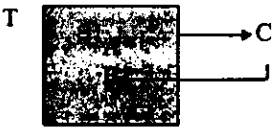
Lógicamente, en la primera tabla T_1, el atributo C es llave foránea con respecto de T_2 y de

ese modo todos los atributos quedan relacionados entre sí. Es lo que se denomina interrelación entre la tabla T_1 y T_2.

FORMA NORMAL DE BOYCE-CODD (FNBC)

Con la creación de la 3FN se observó posteriormente que se encontraban algunas anomalías que no eran consideradas.

Son casos de tablas que estando en 3FN mantienen una dependencia de un atributo secundario con parte de la llave. Para poder manejar esa dependencia en las aplicaciones es imprescindible manejar una gran cantidad de registros innecesarios (aquéllos donde se mantiene fija la parte de la llave que depende y variando el resto de la llave). Gráficamente es el siguiente caso:



Para ello, se pensó en una definición más global que abordase las anomalías observadas.

La nueva definición se debe a Boyce y Codd:

"Una tabla T está en **FNBC**, sí y sólo sí , las únicas DF elementales son aquellas en las que la llave principal (y llaves secundarias) determinan un atributo".

La definición incluye la 3FN puesto que las dependencias transitivas existen por medio de atributos secundarios que no eran llave.

Pero esta definición realmente se creó para evitar los casos anómalos que no se evitaban con la 3FN y que aparecen cuando a partir de un atributo no primario se conoce una parte de la llave.

Si la llave está formada por un solo atributo, la tabla está en FNBC (si ya estaba en 3FN) como sucedía con la 2FN.

Por tanto, para que una tabla que está en 3FN y no cumple la norma Boyce-Codd se encuentre en FNBC, se realiza una proyección procediendo de la siguiente manera:

1. Se crea una tabla con la parte de la llave que es independiente (A) y todos los atributos no primarios (C):

T1 

2. Se crea otra tabla con la parte de la llave restante y el atributo secundario del que depende, y siendo éste último la llave de la nueva tabla:

T2 

Lógicamente, en la primera tabla el atributo C, es llave foránea, a pesar de que pertenece a la llave principal y esto indica que mediante join se puede obtener la tabla original.

CODIGO EN SYBASE

```
CREATE RULE ru_clave
    AS @clave > 0
go

CREATE RULE ru_codigopostal
    AS @clave  between 0 and 99999
go

CREATE RULE ru_espacios
    AS rtrim(@texto) != ''
go

CREATE RULE ru_estado
    AS @estado in
("AC","BCN","BCS","CAM","CH",
"CHI","COA","COL","DF",
"DUR","GUA","GUE","HID",
"JAL","MEX","MIC","MOR",
"NAY","NL","OAX","PUE",
"QUE","QR","SLP","SIN",
"SON","TAB","TAM","TLA",
"VER","YUC","ZAC")
go

CREATE RULE ru_grado
    AS @grado in ("L", "D","M","P")
go

CREATE RULE ru_nonegativo
    AS @tel between 0 and 9999999
go

CREATE RULE ru_rfc
    AS @rfc like "[A-Z][A-Z][A-Z][A-Z][0-
9][0-9][0-1][0-9][0-3][0-9][A-Z,0-9,' '][A-Z,0-9,'
'][A-Z,0-9,' ']"
go

CREATE RULE ru_telefono
    AS @tel between 9999 and 9999999999999
go

CREATE DEFAULT estado1
    AS "DF"
go

CREATE TABLE alumno (
    alumno          char(13) NOT NULL,
    nombre          char(80) NULL,
    ccalle          char(30) NULL,
    ccp             int NULL,
    ccolonia        char(30) NULL,
    ctelefono       numeric(13) NULL,
    cdelegacion     char(25) NULL,
    cciudad         char(25) NULL,
    cestado         char(3) NULL,
    tcalle          char(30) NULL,
    tcolonia        char(30) NULL,
    tcp             int NULL,
    tdelegacion     char(25) NULL,
    tciudad         char(25) NULL,
    ttelefono       numeric(13) NULL,
    testado         char(3) NULL,
    procedencia     char(30) NULL,
    cargo           char(20) NULL,
    grado           char(1) NULL,
    institucion     char(30) NULL,
    celtel          numeric(13) NULL,
    medio           char(1) NULL
        CONSTRAINT ru_medio
        CHECK (medio in ("C", "I","P","T")),
    email           varchar(50) NULL,

CONSTRAINT IAlumno      PRIMARY KEY
NONCLUSTERED (alumno)
)
go

exec sp_primarykey alumno,
    alumno
go

exec sp_bindrule ru_rfc, 'alumno.alumno'
exec sp_bindrule ru_espacios, 'alumno.nombre'
exec sp_bindrule ru_codigopostal, 'alumno.ccp'
exec sp_bindrule ru_telefono, 'alumno.ctelefono'
exec sp_bindrule ru_estado, 'alumno.cestado'
exec sp_bindefault estado1, 'alumno.cestado'
exec sp_bindrule ru_codigopostal, 'alumno.tcp'
exec sp_bindrule ru_telefono, 'alumno.ttelefono'
exec sp_bindrule ru_estado, 'alumno.testado'
exec sp_bindefault estado1, 'alumno.testado'
exec sp_bindrule ru_grado, 'alumno.grado'
exec sp_bindrule ru_telefono, 'alumno.celtel'
go

CREATE TABLE curso (
    curso           numeric(4) NOT NULL,
    nombre          char(50) NOT NULL,
    fecha           smalldatetime NOT NULL,
    costo           smallmoney NOT NULL,
    duracion        tinyint NOT NULL,
    horasxsemana    tinyint NOT NULL,
 CONSTRAINT ICurso      PRIMARY KEY
(curso),
    CONSTRAINT XAKcurso
    UNIQUE (
        nombre,
        fecha
    )
)
go
```

```
exec sp_primarykey curso,
    curso
go

exec sp_bindrule ru_clave, 'curso.curso'
exec sp_bindrule ru_espacios, 'curso.nombre'
go

CREATE TABLE deposito (
    curso           numeric(4) NOT NULL,
    alumno          char(13) NOT NULL,
    fecha           smalldatetime NOT NULL,
    monto           smallmoney NOT NULL,
    tipo            char(1) DEFAULT "J" NOT
                    NULL
        CONSTRAINT ru_tipo
        CHECK (tipo in ("B", "J")),
 CONSTRAINT IDeposito     PRIMARY KEY
(curso, alumno, fecha)
)
go

exec sp_primarykey deposito,
    curso,
    alumno,
    fecha
go

exec sp_bindrule ru_clave, 'deposito.curso'
exec sp_bindrule ru_rfc, 'deposito.alumno'
go

CREATE TABLE imparte (
    curso           numeric(4) NOT NULL,
    profesor        char(13) NOT NULL,
    tema            numeric(4) NOT NULL,
 CONSTRAINT IImparte     PRIMARY KEY
(curso, profesor, tema)
)
go

exec sp_primarykey imparte,
    curso,
    profesor,
    tema
go

exec sp_bindrule ru_clave, 'imparte.curso'
exec sp_bindrule ru_rfc, 'imparte.profesor'
exec sp_bindrule ru_clave, 'imparte.tema'
go

CREATE TABLE inscripcion (
    alumno          char(13) NOT NULL,
    curso           numeric(4) NOT NULL,
    fecha           smalldatetime NOT NULL,
CONSTRAINT IInscripcion     PRIMARY KEY
(curso, alumno)
)
go

exec sp_primarykey inscripcion,
    curso,
    alumno
go

exec sp_bindrule ru_clave, 'inscripcion.curso'
exec sp_bindrule ru_rfc, 'inscripcion.alumno'
go

CREATE TABLE profesor (
    profesor        char(13) NOT NULL,
    ccolonia        char(30) NULL,
    nombre          char(80) NULL,
    ccalle          char(30) NULL,
    ccp             int NULL,
    ctelefono       numeric(13) NULL,
    cdelegacion     char(25) NULL,
    cciudad         char(25) NULL,
    cestado         char(3) NULL,
    tcolonia        char(30) NULL,
    tcalle          char(30) NULL,
    tdelegacion     char(25) NULL,
    tcp             int NULL,
    tciudad         char(25) NULL,
    ttelefono       numeric(13) NULL,
    testado         char(3) NULL,
    grado           char(1) NULL,
    sexo            char(1) NULL
        CONSTRAINT ru_sexo
        CHECK (sexo in ("M", "F")),
    nacionalidad    char(1) DEFAULT "M"
                    NULL
        CONSTRAINT ru_nacionalidad
        CHECK (nacionalidad in ("M", "E")),
    celtel          numeric(13) NULL,
    email           varchar(50) NULL,
 CONSTRAINT IIProfesor     PRIMARY KEY
NONCLUSTERED (profesor)
)
go

exec sp_primarykey profesor,
    profesor
go

exec sp_bindrule ru_rfc, 'profesor.profesor'
exec sp_bindrule ru_espacios, 'profesor.nombre'
exec sp_bindrule ru_codigopostal, 'profesor.ccp'
exec sp_bindrule ru_telefono, 'profesor.ctelefono'
exec sp_bindrule ru_estado, 'profesor.cestado'
exec sp_bindefault estado1, 'profesor.cestado'
```

```
exec sp_bindrule ru_codigopostal, 'profesor.tcp'
exec sp_bindrule ru_telefono, 'profesor.ttelefono'
exec sp_bindrule ru_estado, 'profesor.testado'
exec sp_bindefault estado1, 'profesor.testado'
exec sp_bindrule ru_grado, 'profesor.grado'
exec sp_bindrule ru_telefono, 'profesor.celtel'
go

CREATE TABLE propuesta (
    tema            numeric(3) NOT NULL,
    alumno          char(13) NOT NULL,
    fecha           smalldatetime NOT NULL,
 CONSTRAINT IPropuesta     PRIMARY KEY
NONCLUSTERED (tema, alumno, fecha)
)
go

exec sp_primarykey propuesta,
    tema,
    alumno,
    fecha
go

exec sp_bindrule ru_clave, 'propuesta.tema'
exec sp_bindrule ru_rfc, 'propuesta.alumno'
go

CREATE TABLE tema (
    tema            numeric(3) NOT NULL,
    nombre          char(50) NOT NULL,
 CONSTRAINT Itemapropuesta     PRIMARY
KEY (tema),
    CONSTRAINT XAKtema
    UNIQUE (
        nombre
    )
)
go

exec sp_primarykey tema,
    tema
go

exec sp_bindrule ru_clave, 'tema.tema'
exec sp_bindrule ru_espacios, 'tema.nombre'
go

CREATE TABLE temacurso (
    tema            numeric(4) NOT NULL,
    nombre          char(80) NOT NULL,
 CONSTRAINT Iclavecurso     PRIMARY KEY
(tema),
    CONSTRAINT XAKtemacurso
    UNIQUE (
        nombre
    )
)
go

exec sp_primarykey temacurso,
    tema
go

exec sp_bindrule ru_clave, 'temacurso.tema'
exec sp_bindrule ru_espacios, 'temacurso.nombre'
go

ALTER TABLE deposito
    ADD CONSTRAINT Inscripcion_Deposito
FOREIGN KEY (curso, alumno)
        REFERENCES inscripcion
go


exec sp_foreignkey deposito, inscripcion,
    curso,
    alumno
go

ALTER TABLE imparte
    ADD CONSTRAINT Profesor__Imparte
FOREIGN KEY (profesor)
        REFERENCES profesor
go


ALTER TABLE imparte
    ADD CONSTRAINT Temacurso_Imparte
FOREIGN KEY (tema)
        REFERENCES temacurso
go


ALTER TABLE imparte
    ADD CONSTRAINT Curso_Imparte
FOREIGN KEY (curso)
        REFERENCES curso
go


exec sp_foreignkey imparte, profesor,
    profesor
go

exec sp_foreignkey imparte, temacurso,
    tema
go

exec sp_foreignkey imparte, curso,
    curso
go
```

```
ALTER TABLE inscripcion
    ADD CONSTRAINT Alumno_Inscripcion
FOREIGN KEY (alumno)
        REFERENCES alumno
go


ALTER TABLE inscripcion
    ADD CONSTRAINT Curso_Inscripcion
FOREIGN KEY (curso)
        REFERENCES curso
go


exec sp_foreignkey inscripcion, alumno,
     alumno
go

exec sp_foreignkey inscripcion, curso,
     curso
go

ALTER TABLE propuesta
    ADD CONSTRAINT Alumno_propuesta
FOREIGN KEY (alumno)
        REFERENCES alumno
go


ALTER TABLE propuesta
    ADD CONSTRAINT Tema_Propuesta
FOREIGN KEY (tema)
        REFERENCES tema
go


exec sp_foreignkey propuesta, alumno,
     alumno
go

exec sp_foreignkey propuesta, tema,
     tema
go



create trigger tD_alumno on alumno for DELETE
as
/* ERwin Builtin Sun May 03 12:42:06 1998*/
/* DELETE trigger on alumno */
begin
 declare @errno  int,
       @errmsg  varchar(255)
  /* ERwin Builtin Sun May 03 12:42:06 1998*/
  /* alumno Alumno Propuesta propuesta ON
PARENT DELETE RESTRICT */
  if exists (
   select * from deleted,propuesta
   where
    /* %JoinFKPK(propuesta,deleted," = "," 
and") */
     propuesta.alumno = deleted.alumno
  )
  begin
   select @errno  = 30001,
       @errmsg = 'Cannot DELETE "alumno"
                  because "propuesta" exists.'
   goto error
  end

  /* ERwin Builtin Sun May 03 12:42:06 1998*/
  /* alumno Alumno Inscripcion inscripcion ON
PARENT DELETE RESTRICT */
  if exists (
   select * from deleted,inscripcion
   where
    /* %JoinFKPK(inscripcion,deleted," = "," 
     and") */
     inscripcion.alumno = deleted.alumno
  )
  begin
   select @errno  = 30001,
       @errmsg = 'Cannot DELETE "alumno"
                  because "inscripcion" exists.'
   goto error
  end


  /* ERwin Builtin Sun May 03 12:42:06 1998*/
  return
error:
  raiserror @errno @errmsg
  rollback transaction
end
go

create trigger tU_alumno on alumno for UPDATE
as
/* ERwin Builtin Sun May 03 12:42:06 1998*/
/* UPDATE trigger on alumno */
begin
 declare  @numrows int,
          @nullcnt int,
          @validcnt int,
          @insalumno char(13),
          @errno  int,
          @errmsg  varchar(255)

 select @numrows = @@rowcount
 /* ERwin Builtin Sun May 03 12:42:06 1998*/
 /* alumno Alumno Propuesta propuesta ON
    PARENT UPDATE CASCADE */
```

```
if
 /* %ParentPK(" or",update) */
 update(alumno)
begin
 if @numrows = 1
 begin
  select @insalumno = inserted.alumno
   from inserted
  update propuesta
  set
   /* %JoinFKPK(propuesta,@ins," = ",",") */
   propuesta.alumno = @insalumno
  from propuesta,inserted,deleted
  where
   /* %JoinFKPK(propuesta,deleted," = ","
     and") */
   propuesta.alumno = deleted.alumno
 end
 else
 begin
  select @errno = 30006,
       @errmsg = 'Cannot cascade "alumno"
                 UPDATE because more than
                 one row has been affected.'
  raiserror @errno @errmsg
 end
end

/* ERwin Builtin Sun May 03 12:42:06 1998*/
/* alumno Alumno Inscripcion inscripcion ON
   PARENT UPDATE CASCADE */
if
 /* %ParentPK(" or",update) */
 update(alumno)
begin
 if @numrows = 1
 begin
  select @insalumno = inserted.alumno
   from inserted
  update inscripcion
  set
   /* %JoinFKPK(inscripcion,@ins," = ",",") */
   inscripcion.alumno = @insalumno
  from inscripcion,inserted,deleted
  where
   /* %JoinFKPK(inscripcion,deleted," = ","
     and") */
   inscripcion.alumno = deleted.alumno
 end
 else
 begin
  select @errno = 30006,
       @errmsg = 'Cannot cascade "alumno"
                 UPDATE because more than
                 one row has been affected.'
  raiserror @errno @errmsg
 end
end

/* ERwin Builtin Sun May 03 12:42:06 1998*/
return
error:
 raiserror @errno @errmsg
 rollback transaction
end
go

create trigger tD_curso on curso for DELETE as
/* ERwin Builtin Sun May 03 12:42:06 1998*/
/* DELETE trigger on curso */
begin
 declare @errno  int,
       @errmsg  varchar(255)
 /* ERwin Builtin Sun May 03 12:42:06 1998*/
 /* curso Curso Imparte imparte ON PARENT
    DELETE RESTRICT */
 if exists (
  select * from deleted,imparte
  where
   /* %JoinFKPK(imparte,deleted," = "," and")
     */
   imparte.curso = deleted.curso
 )
 begin
  select @errno  = 30001,
       @errmsg = 'Cannot DELETE "curso"
                 because "imparte" exists.'
  goto error
 end

 /* ERwin Builtin Sun May 03 12:42:06 1998*/
 /* curso Curso Inscripcion inscripcion ON
    PARENT DELETE RESTRICT */
 if exists (
  select * from deleted,inscripcion
  where
   /* %JoinFKPK(inscripcion,deleted," = ","
     and") */
   inscripcion.curso = deleted.curso
 )
 begin
  select @errno  = 30001,
       @errmsg = 'Cannot DELETE "curso"
                 because "inscripcion" exists.'
  goto error
 end


 /* ERwin Builtin Sun May 03 12:42:06 1998*/
 return
error:
```

```
  raiserror @errno @errmsg
  rollback transaction
end
go

create trigger tU_curso on curso for UPDATE as
/* ERwin Builtin Sun May 03 12:42:06 1998*/
/* UPDATE trigger on curso */
begin
 declare  @numrows int,
        @nullcnt int,
        @validcnt int,
        @inscurso numeric(4),
        @errno  int,
        @errmsg  varchar(255)

 select @numrows = @@rowcount
 /* ERwin Builtin Sun May 03 12:42:06 1998*/
 /* curso Curso Imparte imparte ON PARENT
   UPDATE CASCADE */
 if
  /* %ParentPK(" or",update) */
  update(curso)
 begin
  if @numrows = 1
  begin
   select @inscurso = inserted.curso
    from inserted
   update imparte
   set
    /* %JoinFKPK(imparte,@ins," = ",",") */
    imparte.curso = @inscurso
   from imparte,inserted,deleted
   where
  /*  %JoinFKPK(imparte,deleted," = "," and") */
    imparte.curso = deleted.curso
  end
  else
  begin
   select @errno = 30006,
        @errmsg = 'Cannot cascade "curso"
                  UPDATE because more than
                  one row has been affected.'
   raiserror @errno @errmsg
  end
 end

 /* ERwin Builtin Sun May 03 12:42:06 1998*/
 /* curso Curso Inscripcion inscripcion ON
   PARENT UPDATE CASCADE */
 if
  /* %ParentPK(" or",update) */
  update(curso)
 begin
  if @numrows = 1
  begin
   select @inscurso = inserted.curso
    from inserted
   update inscripcion
        set
    /*  %JoinFKPK(inscripcion,@ins," = ",",") */
    inscripcion.curso = @inscurso
   from inscripcion,inserted,deleted
   where
    /*  %JoinFKPK(inscripcion,deleted," = ","
      and") */
    inscripcion.curso = deleted.curso
  end
  else
  begin
   select @errno = 30006,
        @errmsg = 'Cannot cascade "curso"
                  UPDATE because more than
                  one row has been affected.'
   raiserror @errno @errmsg
  end
 end


 /* ERwin Builtin Sun May 03 12:42:06 1998*/
 return
error:
  raiserror @errno @errmsg
  rollback transaction
end
go


/* ERwin Builtin Sun May 03 12.42.06 1998*/
/* INSERT trigger on deposito */
begin
 declare  @numrows int,
        @nullcnt int,
        @validcnt int,
        @errno  int,
        @errmsg  varchar(255)

 select @numrows = @@rowcount
 /* ERwin Builtin Sun May 03 12:42:06 1998*/
 /* inscripcion Inscripcion Deposito deposito ON
   CHILD INSERT RESTRICT */
 if
  /* %ChildFK(" or",update) */
  update(curso) or
  update(alumno)
 begin
  select @nullcnt = 0
  select @validcnt = count(*)
   from inserted,inscripcion
    where
     /* %JoinFKPK(inserted,inscripcion) */
     inserted.curso = inscripcion.curso and
```

```
      inserted.alumno = inscripcion.alumno
  /* %NotnullFK(inserted," is null","select
    @nullcnt = count(*) from inserted where","
    and") */

  if @validcnt + @nullcnt != @numrows
  begin
    select @errno  = 30002,
        @errmsg = 'Cannot INSERT "deposito"
                  because "inscripcion" does not
                  exist.'
    goto error
  end
 end


 /* ERwin Builtin Sun May 03 12:42:06 1998*/
 return
error:
   raiserror @errno @errmsg
   rollback transaction
end
go

create trigger tU_deposito on deposito for
UPDATE as
/* ERwin Builtin Sun May 03 12:42:06 1998*/
/* UPDATE trigger on deposito */
begin
 declare  @numrows int,
        @nullcnt int,
        @validcnt int,
        @inscurso numeric(4),
        @insalumno char(13),
        @insfecha smalldatetime,
        @errno  int,
        @errmsg  varchar(255)

 select @numrows = @@rowcount
 /* ERwin Builtin Sun May 03 12:42:06 1998*/
 /* inscripcion Inscripcion Deposito deposito ON
   CHILD UPDATE RESTRICT */
 if
  /* %ChildFK(" or",update) */
  update(curso) or
  update(alumno)
 begin
  select @nullcnt = 0
  select @validcnt = count(*)
   from inserted,inscripcion
    where
     /* %JoinFKPK(inserted,inscripcion) */
     inserted.curso = inscripcion.curso and
     inserted.alumno = inscripcion.alumno
  /* %NotnullFK(inserted," is null","select
    @nullcnt = count(*) from inserted where","
    and") */

  if @validcnt + @nullcnt != @numrows
  begin
   select @errno  = 30007,
       @errmsg = 'Cannot UPDATE "deposito"
                  because "inscripcion" does
                  not exist.'
   goto error
  end
 end


 /* ERwin Builtin Sun May 03 12:42:06 1998*/
 return
error:
   raiserror @errno @errmsg
   rollback transaction
end
go

create trigger tI_imparte on imparte for INSERT
as
/* ERwin Builtin Sun May 03 12:42:06 1998*/
/* INSERT trigger on imparte */
begin
 declare  @numrows int,
        @nullcnt int,
        @validcnt int,
        @errno  int,
        @errmsg  varchar(255)

 select @numrows = @@rowcount
 /* ERwin Builtin Sun May 03 12:42:06 1998*/
 /* profesor Profesor  Imparte imparte ON
   CHILD INSERT RESTRICT */
 if
  /* %ChildFK(" or",update) */
  update(profesor)
 begin
  select @nullcnt = 0
  select @validcnt = count(*)
   from inserted,profesor
    where
     /* %JoinFKPK(inserted,profesor) */
     inserted.profesor = profesor.profesor
  /* %NotnullFK(inserted," is null","select
    @nullcnt = count(*) from inserted where","
    and") */

  if @validcnt + @nullcnt != @numrows
  begin
   select @errno  = 30002,
       @errmsg = 'Cannot INSERT "imparte"
                  because "profesor" does not
                  exist.'
```

```
  goto error
 end
end

/* ERwin Builtin Sun May 03 12:42:06 1998*/
/* temacurso Temacurso Imparte imparte ON
   CHILD INSERT RESTRICT */
if
 /* %ChildFK(" or",update) */
 update(tema)
begin
 select @nullcnt = 0
 select @validcnt = count(*)
  from inserted,temacurso
   where
    /* %JoinFKPK(inserted,temacurso) */
    inserted.tema = temacurso.tema
 /* %NotnullFK(inserted," is null","select
   @nullcnt = count(*) from inserted where","
   and") */

 if @validcnt + @nullcnt != @numrows
 begin
  select @errno  = 30002,
      @errmsg = 'Cannot INSERT "imparte"
                because "temacurso" does not
                exist.'
  goto error
 end
end

/* ERwin Builtin Sun May 03 12:42:06 1998*/
/* curso Curso Imparte imparte ON CHILD
   INSERT RESTRICT */
if
 /* %ChildFK(" or",update) */
 update(curso)
begin
 select @nullcnt = 0
 select @validcnt = count(*)
  from inserted,curso
   where
    /* %JoinFKPK(inserted,curso) */
    inserted.curso = curso.curso
 /* %NotnullFK(inserted," is null","select
   @nullcnt = count(*) from inserted where","
   and") */

 if @validcnt + @nullcnt != @numrows
 begin
  select @errno  = 30002,
      @errmsg = 'Cannot INSERT "imparte"
                because "curso" does not
                exist.'
  goto error
 end
end


/* ERwin Builtin Sun May 03 12:42:06 1998*/
return
error:
  raiserror @errno @errmsg
  rollback transaction
end
go

create trigger tU_imparte on imparte for UPDATE
as
/* ERwin Builtin Sun May 03 12:42:06 1998*/
/* UPDATE trigger on imparte */
begin
 declare  @numrows int,
       @nullcnt int,
       @validcnt int,
       @inscurso numeric(4),
       @insprofesor char(13),
       @instema numeric(4),
       @errno  int,
       @errmsg  varchar(255)

 select @numrows = @@rowcount
 /* ERwin Builtin Sun May 03 12:42:06 1998*/
 /* profesor Profesor  Imparte imparte ON
   CHILD UPDATE RESTRICT */
 if
  /* %ChildFK(" or",update) */
  update(profesor)
 begin
  select @nullcnt = 0
  select @validcnt = count(*)
   from inserted,profesor
    where
     /* %JoinFKPK(inserted,profesor) */
     inserted.profesor = profesor.profesor
  /* %NotnullFK(inserted," is null","select
    @nullcnt = count(*) from inserted where","
    and") */

  if @validcnt + @nullcnt != @numrows
  begin
   select @errno  = 30007,
       @errmsg = 'Cannot UPDATE "imparte"
                 because "profesor" does not
                 exist.'
   goto error
  end
 end

 /* ERwin Builtin Sun May 03 12:42:06 1998*/
 /* temacurso Temacurso Imparte imparte ON
   CHILD UPDATE RESTRICT */
```

```
if
 /* %ChildFK(" or",update) */
 update(tema)
begin
 select @nullcnt = 0
 select @validcnt = count(*)
  from inserted,temacurso
   where
    /* %JoinFKPK(inserted,temacurso) */
    inserted.tema = temacurso.tema
 /* %NotnullFK(inserted," is null","select
  @nullcnt = count(*) from inserted where","
  and") */

 if @validcnt + @nullcnt != @numrows
 begin
  select @errno  = 30007,
      @errmsg = 'Cannot UPDATE "imparte"
                because "temacurso" does not
                exist.'
  goto error
 end
end

/* ERwin Builtin Sun May 03 12:42:06 1998*/
/* curso Curso Imparte imparte ON CHILD
   UPDATE RESTRICT */
if
 /* %ChildFK(" or",update) */
 update(curso)
begin
 select @nullcnt = 0
 select @validcnt = count(*)
  from inserted,curso
   where
    /* %JoinFKPK(inserted,curso) */
    inserted.curso = curso.curso
 /* %NotnullFK(inserted," is null","select
  @nullcnt = count(*) from inserted where","
  and") */

 if @validcnt + @nullcnt != @numrows
 begin
  select @errno  = 30007,
      @errmsg = 'Cannot UPDATE "imparte"
                because "curso" does not exist.'
  goto error
 end
end


/* ERwin Builtin Sun May 03 12:42:06 1998*/
return
error:
  raiserror @errno @errmsg
  rollback transaction
end
go

create trigger tD_inscripcion on inscripcion for
DELETE as
/* ERwin Builtin Sun May 03 12:42:06 1998*/
/* DELETE trigger on inscripcion */
begin
 declare @errno  int,
       @errmsg varchar(255)
  /* ERwin Builtin Sun May 03 12:42:06 1998*/
  /* inscripcion Inscripcion Deposito deposito
     ON PARENT DELETE RESTRICT */
  if exists (
   select * from deleted,deposito
   where
    /* %JoinFKPK(deposito,deleted," = "," 
     and") */
    deposito.curso = deleted.curso and
    deposito.alumno = deleted.alumno
  )
  begin
   select @errno  = 30001,
       @errmsg = 'Cannot DELETE
                 "inscripcion" because
                 "deposito" exists.'
   goto error
  end


  /* ERwin Builtin Sun May 03 12:42:06 1998*/
  return
error:
  raiserror @errno @errmsg
  rollback transaction
end
go

create trigger tI_inscripcion on inscripcion for
INSERT as
/* ERwin Builtin Sun May 03 12:42:06 1998*/
/* INSERT trigger on inscripcion */
begin
 declare @numrows int,
       @nullcnt int,
       @validcnt int,
       @errno  int,
       @errmsg varchar(255)

 select @numrows = @@rowcount
 /* ERwin Builtin Sun May 03 12:42:06 1998*/
 /* alumno Alumno Inscripcion inscripcion ON
    CHILD INSERT RESTRICT */
 if
  /* %ChildFK(" or",update) */
  update(alumno)
```

```
begin
  select @nullcnt = 0
  select @validcnt = count(*)
    from inserted,alumno
      where
        /* %JoinFKPK(inserted,alumno) */
        inserted.alumno = alumno.alumno
  /* %NotnullFK(inserted," is null","select
     @nullcnt = count(*) from inserted where","
     and") */

  if @validcnt + @nullcnt != @numrows
  begin
    select @errno  = 30002,
        @errmsg = 'Cannot INSERT
                   "inscripcion" because
                   "alumno" does not exist.'
    goto error
  end
end

/* ERwin Builtin Sun May 03 12:42:06 1998*/
/* curso Curso Inscripcion inscripcion ON
   CHILD INSERT RESTRICT */
if
  /* %ChildFK(" or",update) */
  update(curso)
begin
  select @nullcnt = 0
  select @validcnt = count(*)
    from inserted,curso
      where
        /* %JoinFKPK(inserted,curso) */
        inserted.curso = curso.curso
  /* %NotnullFK(inserted," is null","select
     @nullcnt = count(*) from inserted where","
     and") */

  if @validcnt + @nullcnt != @numrows
  begin
    select @errno  = 30002,
        @errmsg = 'Cannot INSERT
                   "inscripcion" because "curso"
                   does not exist.'
    goto error
  end
end

/* ERwin Builtin Sun May 03 12:42:06 1998*/
return
error:
   raiserror @errno @errmsg
   rollback transaction
end
go
```

```
create trigger tU_inscripcion on inscripcion for
UPDATE as
/* ERwin Builtin Sun May 03 12:42:06 1998*/
/* UPDATE trigger on inscripcion */
begin
  declare  @numrows int,
         @nullcnt int,
         @validcnt int,
         @inscurso numeric(4),
         @insalumno char(13),
         @errno  int,
         @errmsg  varchar(255)

  select @numrows = @@rowcount
  /* ERwin Builtin Sun May 03 12:42:06 1998*/
  /* inscripcion Inscripcion Deposito deposito ON
     PARENT UPDATE RESTRICT */
  if
    /* %ParentPK(" or",update) */
    update(curso) or
    update(alumno)
  begin
    if exists (
      select * from deleted,deposito
      where
        /*  %JoinFKPK(deposito,deleted," = "," 
            and") */
        deposito.curso = deleted.curso and
        deposito.alumno   deleted.alumno
    )
    begin
      select @errno  = 30005,
          @errmsg = 'Cannot UPDATE
                     "inscripcion" because
                     "deposito" exists.'
      goto error
    end

  /* ERwin Builtin Sun May 03 12:42:06 1998*/
  /* alumno Alumno Inscripcion inscripcion ON
     CHILD UPDATE RESTRICT */
  if
    /* %ChildFK(" or",update) */
    update(alumno)
  begin
    select @nullcnt = 0
    select @validcnt = count(*)
      from inserted,alumno
        where
          /* %JoinFKPK(inserted,alumno) */
          inserted.alumno = alumno.alumno
    /* %NotnullFK(inserted," is null","select
       @nullcnt = count(*) from inserted where","
       and") */
```

```
  if @validcnt + @nullcnt != @numrows
  begin
   select @errno  = 30007,
       @errmsg = 'Cannot UPDATE
                  "inscripcion" because
                  "alumno" does not exist.'
   goto error
  end
 end

 /* ERwin Builtin Sun May 03 12:42:06 1998*/
 /* curso Curso Inscripcion inscripcion ON
CHILD UPDATE RESTRICT */
 if
  /* %ChildFK(" or",update) */
  update(curso)
 begin
  select @nullcnt = 0
  select @validcnt = count(*)
   from inserted,curso
    where
     /* %JoinFKPK(inserted,curso) */
     inserted.curso = curso.curso
  /* %NotnullFK(inserted," is null","select
    @nullcnt = count(*) from inserted where","
    and") */

  if @validcnt + @nullcnt != @numrows
  begin
   select @errno  = 30007,
       @errmsg = 'Cannot UPDATE
                  "inscripcion" because "curso"
                  does not exist.'
   goto error
  end
 end

 /* ERwin Builtin Sun May 03 12:42:06 1998*/
 return
error:
  raiserror @errno @errmsg
  rollback transaction
end
go

create trigger tD_profesor on profesor for
DELETE as
/* ERwin Builtin Sun May 03 12:42:06 1998*/
/* DELETE trigger on profesor */
begin
 declare  @errno  int,
       @errmsg  varchar(255)
  /* ERwin Builtin Sun May 03 12:42:06 1998*/
  /* profesor Profesor  Imparte imparte ON
  PARENT DELETE RESTRICT */
  if exists (
   select * from deleted,imparte
   where
    /*  %JoinFKPK(imparte,deleted," = "," and")
      */
    imparte.profesor = deleted.profesor
  )
  begin
   select @errno  = 30001,
       @errmsg = 'Cannot DELETE "profesor"
                  because "imparte" exists.'
   goto error
  end


  /* ERwin Builtin Sun May 03 12:42:06 1998*/
  return
error:
  raiserror @errno @errmsg
  rollback transaction
end
go

create trigger tU_profesor on profesor for
UPDATE as
/* ERwin Builtin Sun May 03 12:42:06 1998*/
/* UPDATE trigger on profesor */
begin
 declare  @numrows int,
       @nullcnt int,
       @validcnt int,
       @insprofesor char(13),
       @errno  int,
       @errmsg  varchar(255)

 select @numrows = @@rowcount
 /* ERwin Builtin Sun May 03 12:42:06 1998*/
 /* profesor Profesor  Imparte imparte ON
    PARENT UPDATE CASCADE */
 if
  /* %ParentPK(" or",update) */
  update(profesor)
 begin
  if @numrows = 1
  begin
   select @insprofesor = inserted.profesor
    from inserted
   update imparte
   set
    /*  %JoinFKPK(imparte,@ins," = ",",") */
    imparte.profesor = @insprofesor
   from imparte,inserted,deleted
   where
    /*  %JoinFKPK(imparte,deleted," = "," and")
      */
    imparte.profesor = deleted.profesor
  end
```

```
  else
  begin
   select @errno = 30006,
        @errmsg = 'Cannot cascade "profesor"
                  UPDATE because more than
                  one row has been affected.'
   raiserror @errno @errmsg
  end
 end

 /* ERwin Builtin Sun May 03 12:42:06 1998*/
 return
error:
  raiserror @errno @errmsg
  rollback transaction
end
go

create trigger tI_propuesta on propuesta for
INSERT as
/* ERwin Builtin Sun May 03 12:42:06 1998*/
/* INSERT trigger on propuesta */
begin
 declare  @numrows int,
       @nullcnt int,
       @validcnt int,
       @errno   int,
       @errmsg  varchar(255)

 select @numrows = @@rowcount
 /* ERwin Builtin Sun May 03 12:42:06 1998*/
 /* alumno Alumno Propuesta propuesta ON
    CHILD INSERT RESTRICT */
 if
  /* %ChildFK(" or",update) */
  update(alumno)
 begin
  select @nullcnt = 0
  select @validcnt = count(*)
   from inserted,alumno
     where
      /* %JoinFKPK(inserted,alumno) */
      inserted.alumno = alumno.alumno
  /* %NotnullFK(inserted," is null","select
    @nullcnt = count(*) from inserted where","
    and") */

  if @validcnt + @nullcnt != @numrows
  begin
   select @errno  = 30002,
        @errmsg = 'Cannot INSERT "propuesta"
                  because "alumno" does not
                  exist.'
   goto error
  end
 end

 /* ERwin Builtin Sun May 03 12:42:06 1998*/
 /* tema Tema Propuesta propuesta ON CHILD
    INSERT RESTRICT */
 if
  /* %ChildFK(" or",update) */
  update(tema)
 begin
  select @nullcnt = 0
  select @validcnt = count(*)
   from inserted,tema
     where
      /* %JoinFKPK(inserted,tema) */
      inserted.tema = tema.tema
  /* %NotnullFK(inserted," is null","select
    @nullcnt = count(*) from inserted where","
    and") */

  if @validcnt + @nullcnt != @numrows
  begin
   select @errno  = 30002,
        @errmsg = 'Cannot INSERT "propuesta"
                  because "tema" does not exist.'
   goto error
  end
 end


 /* ERwin Builtin Sun May 03 12:42:06 1998*/
 return
error:
  raiserror @errno @errmsg
  rollback transaction
end
go

create trigger tU_propuesta on propuesta for
UPDATE as
/* ERwin Builtin Sun May 03 12:42:06 1998*/
/* UPDATE trigger on propuesta */
begin
 declare  @numrows int,
       @nullcnt int,
       @validcnt int,
       @instema numeric(3),
       @insalumno char(13),
       @insfecha smalldatetime,
       @errno   int,
       @errmsg  varchar(255)

 select @numrows = @@rowcount
 /* ERwin Builtin Sun May 03 12:42:06 1998*/
 /* alumno Alumno Propuesta propuesta ON
  CHILD UPDATE RESTRICT */
  if
   /* %ChildFK(" or",update) */
```

```
 update(alumno)
begin
 select @nullcnt = 0
 select @validcnt = count(*)
  from inserted,alumno
   where
    /* %JoinFKPK(inserted,alumno) */
    inserted.alumno = alumno.alumno
 /* %NotnullFK(inserted," is null","select
   @nullcnt = count(*) from inserted where","
   and") */

 if @validcnt + @nullcnt != @numrows
 begin
  select @errno  = 30007,
      @errmsg = 'Cannot UPDATE "propuesta"
                because "alumno" does not exist.'
  goto error
 end
end

/* ERwin Builtin Sun May 03 12:42:06 1998*/
/* tema Tema Propuesta propuesta ON CHILD
  UPDATE RESTRICT */
if
 /* %ChildFK(" or",update) */
 update(tema)
begin
 select @nullcnt = 0
 select @validcnt = count(*)
  from inserted,tema
   where
    /* %JoinFKPK(inserted,tema) */
    inserted.tema = tema.tema
 /* %NotnullFK(inserted," is null","select
   @nullcnt = count(*) from inserted where","
   and") */

 if @validcnt + @nullcnt != @numrows
 begin
  select @errno  = 30007,
      @errmsg = 'Cannot UPDATE "propuesta"
                because "tema" does not exist.'
  goto error
 end
end

/* ERwin Builtin Sun May 03 12:42:06 1998*/
return
error:
 raiserror @errno @errmsg
 rollback transaction
end
go

create trigger tD_tema on tema for DELETE as
/* ERwin Builtin Sun May 03 12:42:06 1998*/
/* DELETE trigger on tema */
begin
 declare  @errno  int,
       @errmsg  varchar(255)
  /* ERwin Builtin Sun May 03 12:42:06 1998*/
  /* tema Tema Propuesta propuesta ON
    PARENT DELETE RESTRICT */
  if exists (
   select * from deleted,propuesta
   where
    /*  %JoinFKPK(propuesta,deleted," = "," 
      and") */
    propuesta.tema = deleted.tema
  )
  begin
   select @errno  = 30001,
       @errmsg = 'Cannot DELETE "tema"
                  because "propuesta" exists.'
   goto error
  end


  /* ERwin Builtin Sun May 03 12:42:06 1998*/
  return
error:
  raiserror @errno @errmsg
  rollback transaction
end
go

create trigger tU_tema on tema for UPDATE as
/* ERwin Builtin Sun May 03 12:42:06 1998*/
/* UPDATE trigger on tema */
begin
 declare  @numrows int,
       @nullcnt int,
       @validcnt int,
       @instema numeric(3),
       @errno  int,
       @errmsg  varchar(255)

 select @numrows = @@rowcount
 /* ERwin Builtin Sun May 03 12:42:06 1998*/
 /* tema Tema Propuesta propuesta ON PARENT
   UPDATE CASCADE */
 if
  /* %ParentPK(" or",update) */
  update(tema)
 begin
  if @numrows = 1
  begin
   select @instema = inserted.tema
    from inserted
   update propuesta
   set
```

```
    /* %JoinFKPK(propuesta,@ins," = ",",") */
    propuesta.tema = @instema
   from propuesta,inserted,deleted
   where
    /* %JoinFKPK(propuesta,deleted," = "
     and") */
    propuesta.tema = deleted.tema
  end
  else
  begin
   select @errno = 30006,
       @errmsg = 'Cannot cascade "tema"
                 UPDATE because more than
                 one row has been affected.'
   raiserror @errno @errmsg
  end
 end

 /* ERwin Builtin Sun May 03 12:42:06 1998*/
 return
error:
  raiserror @errno @errmsg
  rollback transaction
end
go

create trigger tD_temacurso on temacurso for
DELETE as
/* ERwin Builtin Sun May 03 12:42:06 1998*/
/* DELETE trigger on temacurso */
begin
 declare @errno  int,
      @errmsg  varchar(255)
  /* ERwin Builtin Sun May 03 12:42:06 1998*/
  /* temacurso Temacurso Imparte imparte ON
    PARENT DELETE RESTRICT */
  if exists (
   select * from deleted,imparte
   where
    /* %JoinFKPK(imparte,deleted," = "," and")
      */
    imparte.tema = deleted.tema
  )
  begin
   select @errno  = 30001,
       @errmsg = 'Cannot DELETE "temacurso"
                 because "imparte" exists.'
   goto error
  end


  /* ERwin Builtin Sun May 03 12:42:06 1998*/
  return
error:
  raiserror @errno @errmsg
  rollback transaction
end
go

create trigger tU_temacurso on temacurso for
UPDATE as
/* ERwin Builtin Sun May 03 12:42:06 1998*/
/* UPDATE trigger on temacurso */
begin
 declare @numrows int,
       @nullcnt int,
       @validcnt int,
       @instema numeric(4),
       @errno  int,
       @errmsg  varchar(255)

 select @numrows = @@rowcount
 /* ERwin Builtin Sun May 03 12:42:06 1998*/
 /* temacurso Temacurso Imparte imparte ON
PARENT UPDATE CASCADE */
 if
  /* %ParentPK(" or",update) */
  update(tema)
 begin
  if @numrows = 1
  begin
   select @instema = inserted.tema
    from inserted
   update imparte
   set
    /* %JoinFKPK(imparte,@ins," = ",",") */
    imparte.tema = @instema
   from imparte,inserted,deleted
   where
    /* %JoinFKPK(imparte,deleted," = "," and")
      */
    imparte.tema = deleted.tema
  end
  else
  begin
   select @errno = 30006,
       @errmsg = 'Cannot cascade "temacurso"
                 UPDATE because more than
                 one row has been affected.'
   raiserror @errno @errmsg
  end
 end

 /* ERwin Builtin Sun May 03 12:42:06 1998*/
 return
error:
  raiserror @errno @errmsg
  rollback transaction
end
go
```

BIBLIOGRAFÍA

[ATR91] Atre, Shakuntala. Técnicas de bases de datos. (Estructuración en diseño y administración). Ed. Trillas. 1991.

[BAR94] Barker Richard. El modelo Entidad-Relación (CASE METHOD). Ed. Addison-Wesley. 1994.

[BEN89] Benavides J. SQL para usuarios y programadores. Ed. Paraninfo. España. 1989.

[BIB96] Biberdorf, Glidden Keith. Power Builder 5 How-to. Ed. Waite Group Press. 1996.

[CHR90] Chris Gane. Análisis Estructurado de Sistemas. Ed. Librería el "Ateneo". Argentina. 1990.

[DAT85] Date. Introducción a los sistemas de bases de datos. 3ª. Edición. Ed. Addison-Wesley Iberoamericana. 1985.

[ECK83] Eckols Steve. How to Desing an Develop Business Systems. Ed. Mike Murach & Associates, Inc. USA 1983.

[FAI87] Fairley Richard. Ingeniería de Software. Ed. McGraw-Hill. México.1987.

[GAL96] Gallagher, Herbert Simon. Power Builder 5. Second Edition. Ed. Sams Publishing. 1996.

[GAR95] Garbus, Solomon S. David. Sybase DBA Survival Guide. Ed. Sams Publishing. 1996.

[GOM93] Gomez, Angel Lucas. Diseño y gestión de sistemas de bases de datos. Ed. Paraninfo. 1993.

[GRO86] Groff James R. Using SQL. Ed. Osborne Mc Graw-Hill. 1986.

[GRU92] Gruff, James. Aplique SQL. Ed. Mc-Graw Hill. 1992.

[HAZ96] Hazlehurts Peter. Sybase System XI. Ed. QUE Corporation. 1996.

[KEN96] Kendall, E. Julie y Kenneth. Análisis y diseño de sistemas. Ed. Prentice-Hall Hispanoamericana, S.A. 1996.

[LOR91] Lorie Raymond A. SQL & its applications. Ed. Prentice-Hall. 1991.

[MAR89] Martin James. Strategic Information Planning Methodologies. 2ª. Edition. Prentice-Hall. 1989.

[PRE93] Pressman, Roger S. Ingeniería del software. (Un enfoque práctico). McGraw-Hill. 1993.

[RAN96] Rankins, Garbus Jeffrey R. Sybase SQL Server 11. Ed. Sams Publishing. 1996.

[RIV88] Rivero , Corenelio, E. Bases de datos relacionales. Ed. Paraninfo. 1988.

[SMI94] Swith Patrick N. Client / Server Computing. Second Edition. Ed. Sams Publishing. 1994.

[SOM88] Sommerville Ian. Ingeniería de Software. Ed. Addison-Wesley Iberoamericana. México. 1988.

[WOO95] Wood A. Charles. Using Power Builder 4. Ed. QUE. 1995.

[YOU89] Yourdon Edward. Análisis Estructurado Moderno. Ed. Prentice-Hall. México.1989.