UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

“DESARROLLO DEL SISTEMA DE ADQUISICIÓN DE DATOS Y REGISTRO DE VARIABLES (SADROV) PARA UN SIMULADOR DE PROCESOS NUCLEARES”

T E S I S

PARA OBTENER EL TÍTULO DE

INGENIERO EN TELECOMUNICACIONES

P R E S E N T A

JULIO ALBERTO GONZÁLEZ MORALES

INGENIERO EN COMPUTACIÓN

P R E S E N T A

JOSÉ SANTIAGO CAMACHO

DIRECTOR DE TESIS:
M.C EDGAR SALAZAR SALAZAR

CODIRECTOR DE TESIS:
DR. CARLOS CHÁVEZ MERCADO

MÉXICO, D.F. 2004

Agradecimientos

Deseamos agradecer a nuestro director de tesis el M.C Edgar Salazar Salazar por su apoyo y guía para la realización del presente trabajo. Así mismo al Dr. Carlos Chávez Mercado por su valiosa colaboración en el desarrollo de esta tesis.

Un agradecimiento especial al M.C Juan Carlos Ramos Pablos por su ayuda a la implementación del SADROV.

Agradecemos al Dr. Juan Luis François Lacouture, a la Dra. Cecilia Martín del Campo Márquez y al Dr. Miguel Moctezuma Flores por sus valiosos comentarios que contribuyeron a la mejora del presente trabajo.

Así también deseamos expresar nuestro agradecimiento al Organismo Internacional de Energía Atómica (OIEA), al Consejo Nacional de Ciencia y Tecnología (CONACYT), al Programa de Apoyo a Proyectos de Investigación e Innovación Tecnológica (PAPIIT) y a la Facultad de Ingeniería de la UNAM, por el apoyo técnico y financiero que hicieron posible la realización del presente trabajo.

Julio Alberto González Morales.
José Santiago Camacho.

Tablas

Figuras

PRÓLOGO

El presente trabajo tiene como objetivo el desarrollo de un sistema de comunicación a distancia, mediante el uso del Internet, para un simulador de procesos nucleares haciendo uso de un Manejador de Bases de Datos. Este sistema constituye un componente de un sistema de mayor envergadura actualmente en desarrollo, denominado Simulador de Aula.

La estructura del presente trabajo sigue el modelo de desarrollo, de un sistema computacional, tipo cascada. Es decir consta de cinco etapas principales: Requerimientos, Análisis, Diseño, Implementación y Pruebas.

Los capítulos uno y dos contienen partes de lo que es el Análisis y Requerimientos del sistema. Así también en ambos capítulos se dan algunos conceptos necesarios para comprender mejor el porqué de la estructura del SADROV y las herramientas utilizadas para implementarlo.

El capítulo tres es básicamente el Diseño del sistema tomando en cuenta los Requerimientos y el Análisis realizado previamente. También se dan algunos conceptos sobre bases de datos relacionales usados para la realización de la base de datos.

En el capítulo cuatro se dan los detalles sobre las funciones implementadas tomando en cuenta el diseño visto en el capítulo tres. De la misma forma se dan las características de las tablas y campos creados en la base de datos.

En el capítulo cinco se analizan las pruebas realizadas al SADROV. Se mencionan las conclusiones obtenidas del presente trabajo. Y finalmente se proponen trabajos a futuro que complementen y/o mejoren el sistema.

CAPÍTULO UNO

INTRODUCCIÓN

En este capítulo se dan algunos antecedentes necesarios para la comprensión del trabajo desarrollado en esta tesis.

1.1 Simulación por Computadora

La simulación es un tipo específico de modelización con el que se trata de representar la realidad de una forma simplificada. Al igual que ocurre con los modelos matemáticos, los modelos de simulación cuentan con una serie de "inputs" o datos de partida que el investigador incluye en el modelo y una serie de "outputs" o resultados que se desprenden de él. [G1]

Las técnicas de simulación en el desarrollo de modelos tienen varias formas de aplicación, la más importante es la de simular modelos matemáticos pre-existentes. Siguiendo este sistema de modelización, es posible usar la simulación por computadora para un propósito puramente instrumental ya que la opción entre el cálculo directo y el cálculo por computadora no es relevante desde el punto de vista de la obtención de la respuesta correcta. Hay, sin embargo, ventajas prácticas en el uso instrumental de computadoras en la realización de los modelos, ya que la deducción automatizada será ciertamente más rápida y, posiblemente, más fiable, que la llevada a cabo por otros medios.

La potencia actual de las PC's (computadoras personales) permite resolver complejas simulaciones a velocidades que hasta hace muy poco tiempo sólo eran posibles en grandes computadoras, lo que ha permitido que este tipo de técnicas sea más fácilmente accesible al conjunto de la comunidad científica y no sea ya de uso exclusivo de los grandes centros de cómputo.

1.2 Aspectos del Simulador de Aula

Se entiende como Simulador de la Sala de Control de una Central Nuclear:

> "...el medio de entrenamiento versátil, capaz de dotar al personal de operación de un grado de formación tal que facilite la comprensión de fenómenos necesarios para el cumplimiento de sus funciones, la utilización de procedimientos, tanto normales como de emergencia y de los principios que rigen el comportamiento de la Central, simulando escenarios de entrenamiento válidos..." [F1]

Una central nuclear, como cualquier industria de síntesis o de segunda generación, está diseñada por sistemas eléctricos, mecánicos o combinados que realizan una función del proceso total. Cada sistema de la Central tiene su propio "modelo", entendido éste como un conjunto de programas informáticos que "simulan" su comportamiento mediante algoritmos de cálculo que determinan los valores de ciertos parámetros a partir de los datos de entrada recibidos por las variables del programa.

Estos datos de entrada pueden tener su origen en “acciones directas” sobre el simulador o en interrelaciones con otros sistemas o con condiciones de contorno definidas por modelos “puentes” de muy diverso alcance y dificultad.

Para desarrollar estos modelos es necesario elaborar una serie de Especificaciones Técnicas que contienen:

- Los componentes a simular en cada sistema.
- La precisión con que debe hacerse el modelado.
- Las anomalías de cada equipo que se pretenden incorporar a la simulación.
- Las acciones remotas necesarias que haya que realizar desde paneles locales.
- Los resultados de los análisis de incidencias, experiencias operativas, manuales y procedimientos.
- Las necesidades de formación del personal de operación de cada instalación concreta.

Así se puede definir la estructura de un simulador en cuatro módulos básicos:

1. Modelos matemáticos. Si éstos cubren todos los sistemas gobernados desde la Sala de Control, excepto situaciones de recarga, estamos ante un simulador de “alcance total”.
2. Sistemas de Cálculo. Su función es ejecutar en tiempo real los modelos matemáticos programados, respondiendo a las señales de control procedentes tanto de las “interfases del operador” como del “puesto de instructor”, generando valores calculados de las variables que actualizan la instrumentación informativa (indicadores, registros y luces de estado) accesibles al operador o al instructor.
3. Interfases de Operador. Es el medio de comunicación del operador con el modelo para el control del proceso. Permite modificar el estado funcional de componentes dinámicos (válvulas, bombas e interruptores principalmente) y recibir la información actualizada sobre la evolución de las variables controladas. Los simuladores denominados “específicos” suelen disponer de una interfaz de operación que es un duplicado de la Sala de Control, tanto en lo que se refiere a la instrumentación como a la arquitectura.
4. Puesto de Instructor. Es el centro neurálgico del sistema y desde ahí, el instructor realiza alteraciones de los procesos para crear anomalías, incidencias y emergencias. También se emplea para evaluar la eficacia del entrenamiento.

Por otra parte el Simulador de Aula es más una herramienta analítica que de entrenamiento en operación, y se le denomina de aula porque no presenta paneles físicos que suelen ser muy grandes sino instrumentación virtual y puede ser confinado a un espacio relativamente pequeño. El Simulador de Aula no sustituye a un Simulador de Entrenamiento de una Central Nuclear, es un complemento y es importante mencionar esta diferencia.

Un simulador para entrenamiento capacita al operador en la operación de la planta en base a procedimientos de operación. El simulador de aula no sustituye esta función básica. Es una herramienta que puede dar al operador una perspectiva sobre la fenomenología que ocurre desde un punto de vista analítico no de operación (el operador analiza, piensa y comprende, el actuar no es particularmente importante porque no es necesario simular en tiempo real cuando el propósito fundamental es el análisis).

1.3 Descripción del Problema

Actualmente el Grupo de Ingeniería Nuclear de la Facultad de Ingeniería desarrolla en el LAIRN[1] un prototipo de Simulador de Aula[2] encaminado a obtener una representación de todos los modelos dinámicos necesarios para simular el balance de planta de una Central NucleoEléctrica. Este prototipo opera mediante los códigos RELAP5[3] como programa de cálculo y los datos que estos códigos generan son mandados a "memoria compartida"[4] en una estación de trabajo con SO[5] UNIX, mediante rutinas en lenguaje C. Mientras que todos los diagramas y mímicos se manejan a través de un software llamado DATAVIEWS.

La ventaja principal que se obtendrá con el SADROV es el aumentar las capacidades del Simulador de Aula incorporando una plataforma más abierta, flexible, confiable, funcional y económica (debido al uso de software libre).

Por lo que el Simulador de Aula prototipo se ha implementado como un sistema multicomputadoras controlado por una interfaz gráfica avanzada, sin embargo se encuentran físicamente en el LAIRN en las instalaciones de la División de Estudios de Posgrado de la Facultad de Ingeniería (DEPFI) en Jiutepec, Morelos, pretendiéndose con el presente trabajo poner al alcance de usuarios que se encuentren geográficamente distantes, el entrenamiento o el análisis de accidentes, utilizando la red de Internet.

[1] Laboratorio de Análisis en Ingeniería de Reactores Nucleares de la Facultad de Ingeniería.
[2] Ver sección 1.2.
[3] Ver sección 1.3.
[4] Junto con los semáforos y las colas de mensajes, son los recursos compartidos que pone UNIX a disposición de los programas para que puedan intercambiarse información.
[5] Acrónimo de Sistema Operativo.

1.3.1 Objetivo

El objetivo de la presente tesis es desarrollar el Sistema de Adquisición de Datos y Registro de Variables (SADROV) para el Simulador de Aula con propósitos de análisis y entrenamiento a distancia, mediante el uso de tecnologías Web.

1.3.2 Justificación

En los últimos años el Organismo Internacional de Energía Atómica (OIEA), el Consejo Nacional de Ciencia y Tecnología (CONACYT), el Programa de Apoyo para Proyectos de Investigación e Innovación Tecnológica (PAPIIT) y la Facultad de Ingeniería de la UNAM, han otorgado recursos técnicos y financieros para el desarrollo del Laboratorio de Análisis en Ingeniería de Reactores Nucleares (LAIRN). Actualmente se cuenta con los recursos necesarios (software, hardware y personal calificado) para el desarrollo del sistema propuesto.

El desarrollo del presente proyecto constituye un componente de un sistema de mayor envergadura actualmente en desarrollo, denominado Simulador de Aula. Adicionalmente, el sistema podrá ser utilizado como herramienta analítica y plataforma experimental para realizar estudios de Ingeniería de Factores humanos tales como:

- Diseño, prueba y evaluación de paneles de control y despliegues gráficos de información.
- Diseño de Interfases Gráficas Avanzadas.
- Estudios de Interacción Hombre-Máquina.

1.4 Solución propuesta

Al analizar propuestas de solución, consideramos que la más factible será implementar el sistema mediante una arquitectura cliente-servidor, usando como "Manejador de Base de datos"[6] MySQL, el servidor Web APACHE, así como también, utilizar los lenguajes de programación PHP y C, que nos permitirán controlar el registro de "variables de despliegue"[7] y "variables de interacción"[8] e inicializar la base de datos, adquirir de manera directa de los datos de los códigos RELAP5 para colocarlos en la BD (Base de Datos), y para la extracción de los datos hacia los despliegues se usará el lenguaje C (específicamente el API C de MySQL).

[6] En el presente trabajo se manejarán como sinónimos gestor y/o manejador de bases de datos.
[7] Sirve para mostrar datos provenientes de los códigos RELAP5, resultado de alguna simulación, en un despliegue gráfico.
[8] Sirven para lograr interactividad con los despliegues gráficos, pues permiten modificar parámetros dentro de los Códigos RELAP5.

1.4.1 Servidor

El servidor que se usará es del tipo LAMP (Linux, Apache, MySQL y PHP). Es decir, se nombra al servidor de esta forma porque se trabajará con Sistema Operativo Linux, con un servidor Web APACHE, con MySQL como Manejador de Base de Datos y con PHP (Preprocesor Hypertext Parser) como lenguaje de programación para páginas Web.

1.4.2 Cliente

Para nuestro caso en especial, se decidió dividirlo en tres partes:

- La primera parte es llamada “Interfaz Gráfica del SADROV”. Esta consiste en una interfaz gráfica colocada en un navegador Web (Internet Explorer, Netscape, Mozilla, etc.) que nos permitirá inicializar la base de datos tomando como entrada un modelo para los códigos RELAP5 “Inputdec”[9], para así registrar los componentes y variables de despliegue en la base de datos. También, a través de esta interfaz gráfica podremos dar de alta “variables de despliegue” y “variables de interacción” en la base de datos. Al termino del proceso anteriormente mencionado, el “Preprocesador de variables” devolverá un nuevo “inputdec” (tratado de tal manera que nos permita facilitar su uso) listo para ser usado como entrada en el RELAP5.

- La segunda parte consiste en la realización de rutinas en lenguaje C, que nos ayudarán a leer la base previamente inicializada por la “Interfaz Gráfica del SADROV” y en cada “iteración de los códigos” estas rutinas leerán los datos arrojados por RELAP5 e introducirán a los mismos en la base de datos, de acuerdo a la variable de la cual provenga. Es importante mencionar que en esta etapa se podrán tener varias máquinas corriendo diferentes simulaciones en “Clientes RELAP5”[10] accediendo a la base de datos, llamando al servidor.

- La tercera parte se refiere a cada despliegue gráfico hecho para el simulador de aula, usando las funciones del API C de MySQL y la arquitectura de la base para obtener las variables de despliegue previamente registradas mediante el sistema.

[9] Archivo de texto que contiene los datos de entrada del modelo para los códigos RELAP5.
[10] Computadora corriendo los códigos RELAP5 con capacidad de comunicación a la base de datos.

La figura siguiente muestra esquemáticamente la solución propuesta:

Diagrama de Implementación del Sistema

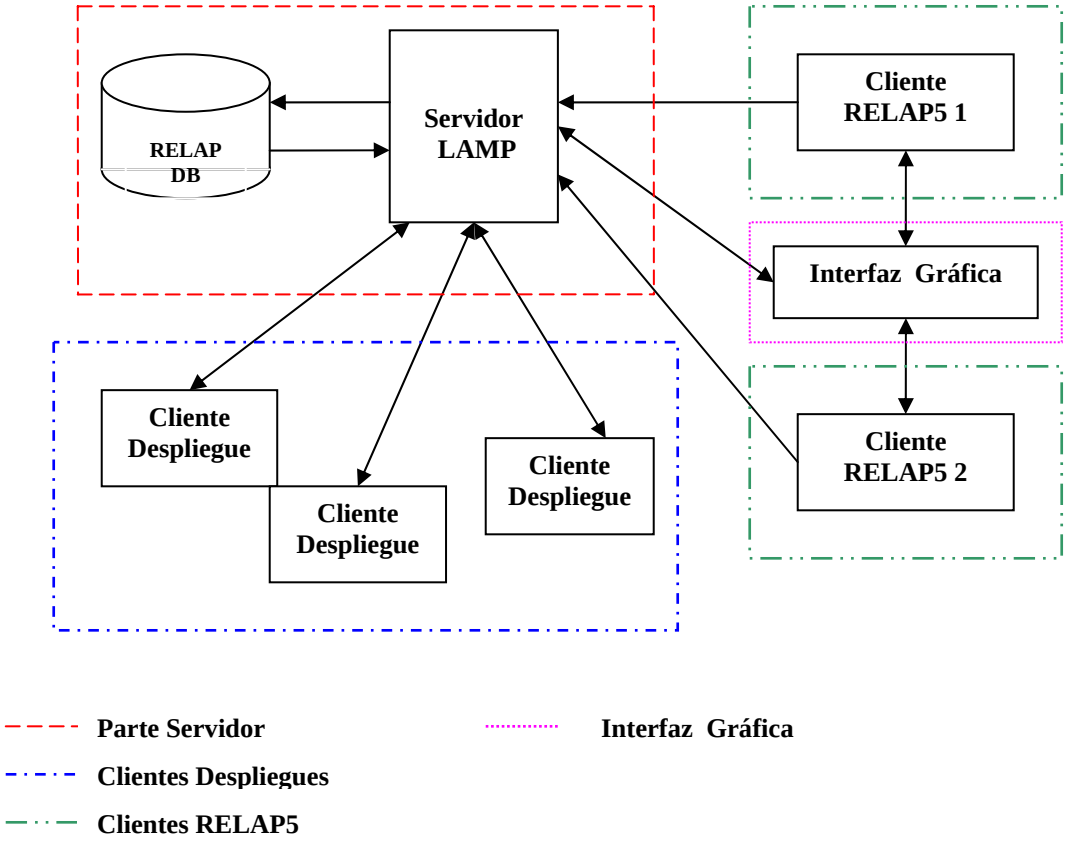

Figura 1.4.1

Es necesario notar que los clientes despliegues se desarrollan conforme a las necesidades del simulador de aula, por lo que no son parte integral del sistema como tal.

1.5 RELAP5

EL código RELAP5 fue desarrollado en el “Idaho National Engineering Laboratory (INEL)” a solicitud de la “U.S Nuclear Regulatory Comisión (NRC)” como un código de análisis de transitorios para un “reactor de agua ligera” (LWR por sus siglas en ingles). Ha sido utilizado ampliamente como base para análisis de plantas nucleares. [I1]

A grandes rasgos RELAP5 es un código genérico que puede ser usado para la simulación de una gran variedad de procesos hidráulicos y térmicos tanto en sistemas nucleares, como en sistemas no nucleares considerando mezclas de vapor, agua y vapores no condensables. El código incluye muchos componentes genéricos, con los cuales se pueden simular sistemas mucho más complejos (Ej. el ciclo de trabajo de una Central Nuclear). Estos modelos de componentes incluyen bombas, válvulas, tubos, estructuras de transferencia de calor, cinética del reactor, calentadores eléctricos, bombas jet, turbinas, separadores de vapor, “acumuladores” y sistemas de control. Además, modelos de procesos especiales están incluidos para lograr diversos efectos como pérdidas de flujo en un cambio abrupto de área, ramificaciones, estrangulación de flujo y transporte de gases no condensables. Los modelos matemáticos del sistema están adaptados dentro de una estructura eficiente en el código. La entrada de los modelos se realiza mediante un formato de texto y los códigos incluyen una gran capacidad para detección de errores en la entrada de los modelos como ayuda a los usuarios.

1.6 El Internet

Internet fue desarrollado originariamente para los militares de Estados Unidos. Después se utilizó para el gobierno, investigación académica y comercial así como para comunicaciones. Se puede decir que es una combinación de “Hardware”[11] (computadoras interconectadas) y “Software”[12] (protocolos y lenguajes que hacen que todo funcione).

Es una infraestructura de “redes a escala mundial”[13] que conecta a la vez a todo tipo de computadoras. Hay alrededor de seis millones de computadoras que utilizan Internet en todo el mundo y que utilizan varios formatos y protocolos Internet:

- Internet Protocol (IP): protocolo que se utiliza para dirigir un paquete de datos desde su fuente a su destino a través de Internet.

[11] Componentes electrónicos, equipos periféricos y otros componentes de un sistema de computadoras.
[12] Colección de códigos de programación y datos de un sistema de computadoras.
[13] Grandes redes principales, tales como MILNET**,** NSFNET, y CREN, así como redes más pequeñas que conectan con ellas.

- Transport Control Protocol (TCP): protocolo de control de transmisión, que se utiliza para administrar accesos.

- User Datagram Protocol (UDP): protocolo del datagrama del usuario, que permite enviar un mensaje desde una computadora a una aplicación que se ejecuta en otra.

Internet tiene varios cuerpos administrativos:

- Internet Architecture Board, que supervisa tecnología y estándares.

- Internet Assigned Numbers Authority, que asigna los números para los accesos, etc.

- InterNIC, que asigna direcciones de Internet.

- También: Internet Engineering and Planning Group, Internet Engineering Steering Group, y la Internet Society.

Visto como una red informática, realmente se trata de un sistema mundial de redes de computadoras. Es un conjunto integrado por las diferentes redes de cada país del mundo, por medio del cual un usuario en cualquier computadora puede, en caso de contar con los permisos apropiados, acceder a la información de otra computadora y poder tener inclusive comunicación directa con otros usuarios en otras computadoras.

El rápido y ascendente crecimiento de Internet ha conseguido que esta red haya pasado a llamarse "La Red" o "La Red de Redes", debido a la existencia de computadoras conectadas a la misma en todo el mundo.

La principal diferencia entre Internet y cualquier otra red informática reside en que ésta no pertenece a ningún país, ni organismo oficial, ni a una empresa determinada, es decir, se trata de una red libre ya que cualquier persona puede acceder a ella desde cualquier punto del planeta, de la misma forma que no existe ningún tipo de restricción para toda la información que circula por la misma.

Solamente existen unos organismos internacionales repartidos por todo el mundo y organizados de forma jerárquica. Estos organismos no tienen ningún afán de lucro, y son los encargados de regular el crecimiento de Internet y garantizar el buen funcionamiento de la Red. Actualmente Internet forma parte de todas las instancias de los negocios y de la vida cotidiana. La implementación de las estrategias actuales de los negocios mediante el uso de la tecnología de

Internet ha recibido el nombre de "Segunda Revolución Industrial". Su definición aceptada consta de dos componentes básicos:

1. Una red de comunicaciones: Que opera sobre medios existentes y las redes telefónicas. Es robusta, global y está basada en un estándar llamado TCP/IP. Si se usan líneas privadas de comunicación se conoce como Intranet, y si se usan líneas públicas se conoce como Internet.

2. Un conjunto de estándares de software: Internet también es un conjunto de estándares de software, incluyendo las funciones básicas de correo electrónico, protocolos de transferencia de archivos, servicios de Web y la búsqueda y obtención de información.

1.6.1 Internet Protocol[14]

El Protocolo IP proporciona un sistema de distribución que es poco fiable incluso en una base sólida. El protocolo IP especifica que la unidad básica de transferencia de datos en el TCP/IP es el "datagrama"[15].

Los datagramas pueden ser retrasados, perdidos, duplicados, enviados en una secuencia incorrecta o fragmentadas intencionadamente para permitir que un nodo con un "buffer"[16] limitado pueda coger todo el datagrama. Es la responsabilidad del protocolo IP reensamblar los fragmentos del datagrama en el orden correcto. En algunas situaciones de error los datagramas son descartados sin mostrar ningún mensaje mientras que en otras situaciones los mensajes de error son recibidos por la maquina origen (esto lo hace el protocolo ICMP).

El protocolo IP también define la ruta inicial por la que serán mandados los datos.

Cuando los datagramas viajan de unos equipos a otros, es posible que atraviesen diferentes tipos de redes. El tamaño máximo de estos paquetes de datos puede variar de una red a otra, dependiendo del medio físico que se emplee para su transmisión. A este tamaño máximo se le denomina MTU (*Maximum Transmission Unit*), y ninguna red puede transmitir un paquete de tamaño mayor a esta MTU. El datagrama consiste en una cabecera y datos. (Ver Figura 1.6.1)

Las características de este datagrama son las siguientes:

- Longitud de la Cabecera. Este campo ocupa 4 bits, y representa el número de octetos de la cabecera dividido por cuatro, lo que hace que éste sea el número de grupos de 4 octetos en la cabecera.

[14] Protocolo de Internet versión 4.
[15] También llamado "Paquete" coloquialmente. Es la unidad de información básica.
[16] Capacidad para la transmisión de información.

- Versión. Ocupa 4 bits. Este campo hace que diferentes versiones del protocolo IP puedan operar en la Internet. En este caso se trata de la versión 4.

- Tipo de servicio. Este campo ocupa un octeto de la cabecera IP, y especifica la precedencia y la prioridad del datagrama IP. Los tres primeros bits del octeto indican la precedencia. Los valores de la precedencia pueden ser de 0 a 7. Cero es la precedencia normal, y 7 está reservado para control de red. Muchos "Gateways" ignoran este campo. Los otros 4 bits definen el campo prioridad, que tiene un rango de 0 a 15. Las cuatro prioridades que están asignadas son: 0, (por defecto, servicio normal), 1 (minimizar el coste monetario), 2 (máxima fiabilidad), 4 (maximizar la transferencia), 8 (El bit +4 igual a 1, define minimizar el retraso). Estos valores son utilizados por los "routers"[17] para direccionar las solicitudes de los usuarios.

- Longitud Total. Se utiliza para identificar el número de octetos en el datagrama total.

- Identificación. El valor del campo identificación es un numero secuencial asignado por el "Host"[18] origen. El campo ocupa dos octetos. Los números oscilan entre 0 y 65,535, que cuando se combinan con la dirección del "Host" forman un número único en Internet. El número se usa para ayudar en el reensamblaje de los fragmentos de datagramas.

- Fragmentos FOCET. Cuando el tamaño de un datagrama excede el MTU, éste se segmenta. El fragmento Offset representa el desplazamiento de este segmento desde el inicio del datagrama entero.

- Flags. El campo flag ocupa 3 bits y contiene dos flags. El bit +5 del campo flags se utiliza para indicar el último datagrama fragmentado cuando toma valor cero. El bit +7 lo utiliza el servidor origen para evitar la fragmentación. Cuando este bit toma valor diferente de cero y la longitud de un datagrama excede el MTU, el datagrama es descartado y un mensaje de error es enviado al Host de origen por medio del protocolo ICMP.

- Tiempo de Vida. El campo tiempo de vida ocupa un octeto. Representa el número máximo de segundos que un datagrama puede existir en Internet, antes de ser descartado. Un Datagrama puede existir un máximo de 255 segundos. El número recomendado para IP es 64. El originador del datagrama manda un mensaje ICMP cuando el datagrama es descartado.

[17] Dispositivo que permite la conexión entre redes encargándose además de que los paquetes en que se divide la información lleguen a su destino. Coloquialmente llamado "enrutador".
[18] Servidor con funciones centralizadas que hace disponibles programas a otras computadoras.

- Protocolo. El campo protocolo se utiliza para identificar la capa de mayor nivel más cercana usando el IP. Éste es un campo de 0 bits, que normalmente identifica tanto la capa TCP (valor 6), como la capa UDP (valor 17) en el nivel de transporte, pero puede identificar hasta 255 protocolos de la capa de transporte.

- *Checksum*. El checksum proporciona la seguridad de que el datagrama no ha sido dañado ni modificado. Este campo tiene una longitud de 16 bits. El checksum incluye todos los campos de todos los campos de la cabecera IP, incluido él mismo, cuyo valor es cero a efectos de cálculo. Un "Gateway" o nodo que efectúe alguna modificación en los campos de la cabecera (por ejemplo en el tiempo de vida), debe recalcular el valor del checksum antes de enviar el datagrama. Los usuarios del IP deben proporcionar su propia integridad en los datos, ya que el checksum es sólo para la cabecera.

- Dirección de Origen. Este campo contiene un identificador de red (Netid) y un identificador de *Host* (Hostid). El campo tiene una longitud de 32 bits. La dirección puede ser de clase A, B, C. (ver Direcciones IP).

- Dirección de Destino. Este campo contiene el Netid y el Hostid del destino. El campo tiene una longitud de 32 bits. La dirección puede ser de clase A, B, C o D (ver Direcciones IP).

- Opciones. La existencia de este campo viene determinada por la longitud de la cabecera. Si ésta es mayor de cinco, por lo menos existe una opción. Aunque un *Host* no está obligado a poner opciones, puede aceptar y procesar opciones recibidas en un datagrama. El campo Opciones es de longitud variable. Cada octeto está formado por los campos Copia, Clase de Opción y Número de Opción.

- Padding. Cuando está presente el campo Pad, consiste en 1 a 3 octetos puestos a cero, si es necesario, para hacer que el número total de octetos en la cabecera sea divisible por cuatro.

- Datos. El campo datos consiste en una cadena de octetos. Cada octeto tiene un valor entre 0 y 255. El tamaño de la cadena puede tener un mínimo y un máximo, dependiendo del medio físico. El tamaño máximo está definido por la longitud total del datagrama. El tamaño del campo Datos (TCD) en octetos se define como:

  TCD = (Longitud Total del Datagrama) - (Longitud de la cabecera)

Formato del Datagrama IP

	msb						lsb	
	7	6	5	4	3	2	1	0	
IP Header	*Version*				*Header Length*				+0
	Type of Service								+1
	Total Length								+2
									+3
	Identification								+4
									+5
	Flags				*Fragment FOCET*				+6
									+7
	Time to Live								+8
	Protocol								+9
	Header Checksum								+10
									+11
	Source Address of Originating Host								+12
									+13
									+14
									+15
	Destination Address of Target Host								+16
									+17
									+18
									+19
	Options								+20
									+21
									+22
	Padding								+23
	IP Data								+0
									+1
MSB									+n

Figura 1.6.1

1.6.2 Transmission Control Protocol[19]

El protocolo TCP proporciona un servicio de comunicación que forma un circuito, es decir, el flujo de datos entre el origen y el destino es continuo. TCP proporciona un circuito virtual el cual es llamado una conexión.

Al contrario de los programas que utilizan UDP[20], los que utilizan el TCP tienen un servicio de conexión entre los programas llamados y los que llaman, chequeo de errores, control de flujo y capacidad de interrupción.

1.6.3 Interfaces TCP

Existen dos tipos de interfaces entre la conexión TCP y los otros programas.

El primero es utilizar la pila de los programas de la capa de red. Como en esta capa sólo está el protocolo IP, la interfaz lo determina este protocolo. El segundo tipo es el interfaz del programa de usuario. Esta interfaz puede variar según el sistema operativo, pero en general tiene las siguientes características.

La interfaz envuelve el programa de usuario llamando a una rutina que introduce entradas en una estructura de datos llamando el bloque de control de transmisión (TCB). Las entradas se realizan inicialmente en la pila de hardware y transferidas al TCB por medio de una rutina de sistema. Estas entradas permiten al TCP asociar un usuario con una conexión particular, de modo que pueda aceptar comandos de un usuario y mandarlos a otro usuario en la otra parte de la conexión. TCP utiliza unos identificadores únicos para cada parte de la conexión. Esto se utiliza para recordar la asociación entre dos usuarios. Al usuario se le asigna un nombre de conexión para utilizarlo en futuras entradas del TCB. Los identificadores para cada extremo de la conexión se llaman “sockets”[21]. El socket local se construye concatenando la dirección IP de origen y el número de puerto de origen. El socket remoto se obtiene concatenando la dirección IP de destino y el número de puerto de destino.

El par de sockets de una conexión forman un único número en Internet. El UDP tiene los mismos sockets, pero no los recuerda. Esta es las diferencia entre un protocolo orientado a conexión y otro a no conexión. A continuación se explican los comandos más usuales:

- Open: Inicia una conexión o comienza a escuchar un socket. El usuario tiene un nombre de conexión local que actúa como un puntero dentro del TCB.

[19] Protocolo de Control de Transmisión (TCP).
[20] Protocolo de Datagrama de Usuario.- Protocolo sencillo que implementa un nivel de transporte orientado a datagramas.
[21] Un socket es un punto final de un enlace de comunicación de dos vías entre dos programas que se ejecutan a través de la red.

- Send: El comando Send manda datos del buffer especificado.

- Receive: El comando Receive es un mensaje de error si el nombre local proporcionado no es utilizado antes con el comando Open.

- Close: El comando Close hace que se cierre una conexión. Se produce un error si la conexión especificada no ha sido abierta, o si no se tiene autorización para cerrar la conexión.

- Status: El comando Status sólo tiene una variable asociada, que es el nombre de la conexión.

- Abort: El comando Abort hace que todos los comandos Send y Receive asociados al nombre de la conexión local se interrumpan. La entrada del usuario del TCB se elimina y se envía un mensaje especial de reinicio a la entidad del otro lado de la conexión.

El TCP recuerda el estado de cada conexión por medio del TCB. Cuando se abre una conexión, se efectúa una entrada única en el TCB. Un nombre de conexión se le asigna al usuario para activar los comandos de la conexión. Cuando se cierra una conexión se elimina su entrada del TCB.

1.6.4 Control de Flujo

El protocolo TCP puede controlar la cantidad de datos que debe enviar mediante el campo *Window*. Este campo indica el número máximo de octetos que pueden ser recibidos. El receptor de un segmento con el campo W*indow* a cero, no puede enviar mensajes al emisor, excepto mensajes de prueba. Un mensaje de prueba es un mensaje de un sólo octeto que se utiliza para detectar redes o "Host" inalcanzables.

1.6.5 Formato del segmento TCP

El segmento TCP consiste en una cabecera y datos. A continuación se describen los campos del segmento TCP. (Ver Figura 1.6.2)

- Número de puerto del Origen/destino (*Source/Destination Port Numbers*): Este campo tiene una longitud de 16 bits.

- Números de Secuencia (*Secuence Numbers*): Existen dos números de secuencia en la cabecera TCP. El primer número de secuencia es el número de secuencia final (*SSN*). El SSN es un número de 32 bits. El otro número de secuencia es el número de secuencia esperado de recepción, También llamado Número de Reconocimiento (*acknowledgement number*).

- Longitud de la cabecera (*Header Length*): Este campo tiene una longitud de 4 bits y contiene un entero igual al número de octetos que forman la cabecera TCP dividido por cuatro.

- Código de Bits (*Code bits*): El motivo y contenido del segmento TCP lo indica este campo. Este campo tiene una longitud de seis bits.

    - *Bit URG (bit +5):* Este bit identifica datos urgentes.

    - *Bit ACK (bit +4):* Cuando este bit se pone a 1, el campo reconocimiento es válido.

    - *Bit PSH (Bit +3):* Aunque el *buffer* no esté lleno, el emisor puede forzar a enviarlo.

    - *Bit RST (Bit +2):* Poniendo este bit, se aborta la conexión. Todos los buffers asociados se vacíen.

    - *Bit SYN (Bit +1):* Este bit sirve para sincronizar los números de secuencia.

    - *Bit FIN (Bit +0):* Este bit se utiliza sólo cuando se está cerrando la conexión.

- Ventana (*Window*): Este campo contiene un entero de 32 bits. Se utiliza para indicar el tamaño de *buffer* disponible que tiene el emisor para recibir datos.

- Opciones (*Options*): Este campo permite que una aplicación negocie durante la configuración de la conexión características como el tamaño máximo del segmento TCP. Si este campo tiene el primer octeto a cero, esto indica que no hay opciones.

- Relleno (*Padding*): Este campo consiste en un número de octetos (de uno a tres), que tienen valor cero y sirven para que la longitud de la cabecera sea divisible por cuatro.

- *Checksum*: Mientras que el protocolo IP no tiene ningún mecanismo para garantizar la integridad de los datos, ya que sólo comprueba la cabecera del mensaje, El TCP dispone de su propio método para garantizar dicha integridad.

Como en el *Checksum* del protocolo TCP también se incluyen campos del protocolo IP, es necesario construir una pseudo-cabecera TCP que se considera únicamente a efectos de cálculo. (Ver Figura 1.6.3).

Formato del mensaje TCP								
	Msb							Lsb
	7	6	5	4	3	2	1	0
TCP Header	*Source Port*							
	Destination Port							
	Sequence Number							
	Acknowledgement Number							
	Header Length				*Reserved*			
	RSV			*Code Bits*				
	Window							
	Checksum							
	Urgent Pointer							
	Options							
	Padding							
	TCP Data							
	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

Figura 1.6.2

Formato del *Checksum* TCP																															
	Octet +0								*Octet +1*								*Octet +2*								*Octet +3*							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Data in Checksum	*Source IP Adress*																															*Pseudo header*
	Destination IP Ardes																															
	Zero								*Protocol Number*								*Number of octets in header and data*															
	TCP Header																															
	TCP Data																															
	TCP Data								*Zero*																							

Figura 1.6.3

1.6.6 Estados del TCP

El inicio, mantenimiento y cierre de una conexión requiere que el TCP recuerde toda la información relativa a cada conexión. Esta información se almacena en una entrada para cada conexión dentro del TCB. Cuando se abre una conexión, la entrada en el TCB se realiza con todas las variables inicializadas con sus respectivos valores. Durante la conexión, la entrada del TCB es actualizada a medida que cambia la información. A continuación se describen algunos de los estados del TCP:

0. CLOSED: No existe, sólo para referencia.

1. LISTEN: Esperando solicitud de conexión de un TCP remoto.

2. SYN-SEN: Esperando un mensaje de solicitud de conexión después de haber enviado una solicitud de conexión.

3. SYN-RECEIVED: Esperando confirmación de reconocimiento de solicitud de conexión, después de haber enviado y recibido una solicitud de conexión.

4. ESTABLISHED: Representa una conexión abierta. Los datos recibidos pueden ser enviados a un protocolo de una capa superior. Este es el estado normal de la fase de transferencia de la conexión.

5. FIN-WAIT-1: Esperando la solicitud de fin de conexión de un TCP remoto, o un reconocimiento de una solicitud de fin de transmisión enviada anteriormente.

6. FIN-WAIT-2: Esperando una solicitud de fin de conexión de un TCP remoto.

7. CLOSE-WAIT: Esperando una solicitud de fin de conexión de un protocolo de una capa superior.

8. CLOSING: Esperando el conocimiento de una solicitud de final de conexión de un TCP remoto.

9. LAST-ACK: Esperando el conocimiento de una solicitud de final de conexión enviada anteriormente al TCP remoto.

10. TIME-WAIT: Esperando el tiempo necesario para que el TCP remoto haya recibido el conocimiento de la solicitud del fin de conexión.

1.6.7 Direcciones IP

Las direcciones IP hacen que el envió de datos entre computadoras se haga de forma eficaz, de un modo similar al que se utilizan los números de teléfono. Las direcciones IP tienen 32 bits, formados por cuatro campos de 8 bits separados por puntos. Cada campo puede tener un valor comprendido entre 0 y 255. Esta compuesta por una dirección de red, seguida de una dirección de subred y de una dirección de host.

Existen cinco clases de subredes:

- La clase A. Contiene 7 bits para direcciones de red, con lo que permite tener hasta 128 redes, con 16,777,216 computadoras cada una. Las direcciones estarán comprendidas entre 0.0.0.0. y 127.255.255.255., y la máscara de subred será 255.0.0.0.

- La clase B. Contiene 14 bits para direcciones de red y 16 bits para direcciones de hosts. El número máximo de redes es 16,536 redes, con 65,536 computadoras por red. Las direcciones estarán comprendidas entre 128.0.0.0. Y 191.255.255.255., y la máscara de subred será 255.255.0.0.

- La clase C. Contiene 21 bits para direcciones de red y 8 para hosts, lo que permite tener un total de 2.097.142 redes, cada una de ellas con 256 computadoras. Las direcciones estarán comprendidas entre 192.0.0.0. Y 223.255.255.255., y la máscara de subred será 255.255.255.0.

- La clase D. Se reserva todas las direcciones para multidestino (multicast), es decir, un ordenador transmite un mensaje a un grupo específico de computadoras de esta clase. Las direcciones estarán comprendidas entre 224.0.0.0. y 239.255.255.255.

- La clase E. Se utiliza exclusivamente para fines experimentales. Las direcciones están comprendidas entre 240.0.0.0 y 247.255.255.255.

1.7 Aplicaciones Web

A grandes rasgos, la Web[22] trabaja con una arquitectura cliente-servidor. Esto significa que el servidor central y la aplicación cliente son responsables por algunos de los procesos generados:

- El cliente: La mayoría de las aplicaciones Web usan un sólo cliente: el “Web browser” o “navegador de Internet”. El lenguaje primario con el que trabaja un navegador es HTML. HTML provee un set de “tags”[23] que describen cómo debe verse una “página Web”.

[22] Web: sinónimo de red, en este caso se refiere al Internet.
[23] Tag: etiqueta.

- El Servidor Web: La mayoría del trabajo para aplicaciones Web toma lugar en el servidor. Una aplicación específica llamada Servidor Web es la responsable de la comunicación con el navegador del cliente.

Por otra parte, servidores de bases de datos relacionales guardan la información que requiera la aplicación. Finalmente se necesita un lenguaje para mandar las peticiones entre el Servidor Web y el Servidor de Base de Datos, esto también puede ser usado para realizar tareas sobre la información, este lenguaje es llamado "middleware". La figura 1.7.1 representa este sistema.

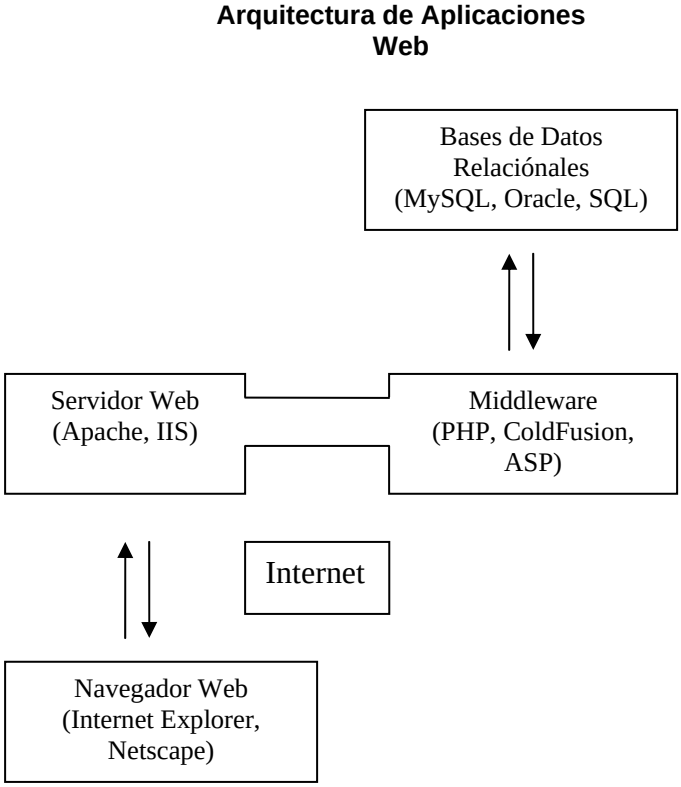

Figura 1.7.1

Por otra parte el HTML no es lenguaje de programación en realidad, más bien, se trata de un lenguaje descriptivo que tiene como objeto dar formato al texto y las imágenes que se pretenden visualizar en el navegador. A partir de este lenguaje es posible introducir enlaces, seleccionar el tamaño de las fonts[24] o intercalar imágenes, todo esto de una manera prefijada y en ningún caso inteligente. En efecto, el HTML no permite el realizar un simple cálculo matemático o crear una página de la nada a partir de una base de datos. Es esta deficiencia del HTML la

[24] Fonts: fuentes o tipo de letra para el texto.

que ha hecho necesario el empleo de otros lenguajes accesorios mucho más versátiles. Estos lenguajes capaces de realizar tareas “inteligentes” a partir de ciertos "scripts"[25] son los llamados “middleware” y permiten la creación de páginas dinámicas o aplicaciones Web.

1.8 Lenguaje de Programación PHP

El navegador es una especie de aplicación capaz de interpretar las órdenes recibidas en forma de código HTML fundamentalmente y convertirlas en las páginas que son el resultado de dicha orden. Sin embargo, si la página que pedimos no es un archivo HTML, el navegador es incapaz de interpretarla y lo único que es capaz de hacer es salvarla en forma de archivo. Es por ello que, si queremos emplear lenguajes accesorios para realizar una aplicación Web, es absolutamente necesario que sea el propio servidor quien los ejecute e interprete para luego enviarlos al cliente (navegador) en forma de archivo HTML totalmente legible por él. Así podemos hablar de dos tipos de lenguajes:

- Lenguajes del lado del Servidor: Son aquellos lenguajes que son reconocidos, ejecutados e interpretados por el propio servidor y que se envían al cliente en un formato comprensible para él.

- Lenguajes del lado Cliente: Son aquellos que pueden ser directamente "digeridos" por el navegador y no necesitan un tratamiento previo (no sólo HTML sino también el Java y el JavaScript los cuales son simplemente incluidos en el código HTML).

Cada uno de estos tipos tiene por supuesto sus ventajas y sus inconvenientes. Así, por ejemplo, un lenguaje del lado cliente es totalmente independiente del servidor, lo cual permite que la página pueda ser albergada en cualquier sitio sin necesidad de pagar más ya que, por regla general, los servidores que aceptan páginas con scripts del lado servidor son en su mayoría de pago o sus prestaciones son muy limitadas. Inversamente, un lenguaje del lado servidor es independiente del cliente por lo que es mucho menos rígido respecto al cambio de un navegador a otro o respecto a las versiones del mismo. Por otra parte, los scripts son almacenados en el servidor quien los ejecuta y traduce a HTML por lo que permanecen ocultos para el cliente.

Así, PHP es un lenguaje del lado del servidor utilizado en plataformas Unix y Linux. Es un lenguaje creado por una gran comunidad de personas. El sistema fue desarrollado originalmente en el año 1994 por Rasmus Lerdorf como un CGI escrito en C que permitía la interpretación de un número limitado de comandos. El sistema fue denominado Personal Home Page Tools y adquirió relativo éxito gracias a que otras personas pidieron a Rasmus que les permitiese utilizar sus programas en sus propias páginas. Dada la aceptación del primer PHP y de

[25] Scripts: Se llama script a todo conjunto de instrucciones que se guardan en un fichero y que son interpretadas tal y como están por un programa.

manera adicional, su creador diseñó un sistema para procesar formularios al que le atribuyó el nombre de FI (Form Interpreter) y el conjunto de estas dos herramientas, sería la primera versión compacta del lenguaje: PHP/FI. [A1]

La siguiente gran contribución al lenguaje se realizó a mediados del 97 cuando se volvió a programar el analizador sintáctico, se incluyeron nuevas funcionalidades como el soporte a nuevos protocolos de Internet y el soporte a la gran mayoría de las bases de datos comerciales. Todas estas mejoras sentaron las bases de PHP versión 3. Actualmente PHP se encuentra en su versión 4, que utiliza el motor Zend, desarrollado con mayor meditación para cubrir las necesidades actuales y solucionar algunos inconvenientes de la anterior versión. Algunas mejoras de esta nueva versión son:

- Rapidez: gracias a que primero se compila y luego se ejecuta.
- Mayor independencia del servidor Web: Creando versiones de PHP nativas para más plataformas.
- API: Ahora cuenta con un API más elaborado y con más funciones.

Para el año 2001, el número de servidores que utilizan PHP se disparó, logrando situarse cerca de los 5 millones de sitios y 800,000 direcciones IP, lo que le ha convertido a PHP en una tecnología popular. Esto es debido, entre otras razones, a que PHP es el complemento ideal para que la dupla Linux-Apache sea compatible con la programación del lado del servidor de sitios Web.

PHP permite realizar una multitud de tareas útiles para el desarrollo de aplicaciones Web:

- Funciones de correo electrónico: Podemos con una facilidad asombrosa enviar un e-mail a una persona o lista parametrizando toda una serie de aspectos tales como el e-mail de procedencia, asunto, persona a responder, entre otras.
- Gestión de bases de datos: El lenguaje PHP ofrece interfases para el acceso a la mayoría de las bases de datos comerciales y por ODBC a todas las bases de datos posibles en sistemas Microsoft.
- Gestión de archivos: Nos permite crear, borrar, mover y modificar archivos a partir de una amplia librería de funciones para la gestión de archivos. También podemos transferir archivos por FTP a partir de sentencias en nuestro código, protocolo para el cual PHP ha previsto también gran cantidad de funciones.
- Tratamiento de imágenes: Nos permite realizar tratamiento de imágenes de forma sencilla. También puede parecer útil el crear botones dinámicos, es decir, botones en los que utilizamos el mismo diseño y sólo cambiamos el

texto. Podremos por ejemplo crear un botón haciendo una única llamada a una función en la que introducimos el estilo del botón y el texto a introducir obteniendo automáticamente el botón deseado. A partir de la librería de funciones gráficas podemos hacer esto y mucho más.

- Funciones para Internet: tratamiento de "cookies"[26], accesos restringidos, comercio electrónico, entre otras.

- Funciones de propósito general: funciones matemáticas, explotación de cadenas, de fechas, corrección ortográfica, compresión de archivos, entre otras, son realizadas por este lenguaje.

A esta inmensa librería cabe ahora añadir todas las funciones personales que los usuarios van creando por necesidades propias y que luego son reutilizadas por otros usuarios.

1.9 Gestor de Bases de Datos

Un Sistema Gestor de Bases de Datos Relaciónales es una herramienta imprescindible en muchos entornos, desde los usos más tradicionales en los contextos de investigación, negocios y educación hasta las aplicaciones más novedosas, como el potenciamiento de los motores de búsqueda en Internet. Sin embargo, a pesar de la importancia de una buena base de datos para gestionar y acceder a los recursos de información, para muchas instituciones, empresas, están fuera del alcance de sus recursos financieros.

En los últimos años, la situación ha cambiado, tanto por el lado del hardware como por el software. Las computadoras personales se han abaratado y ofrecen un buen rendimiento, así ha surgido un movimiento para escribir sistemas operativos de alto rendimiento para las mismas, que están disponibles de forma gratuita en Internet.

También el software de base de datos se ha hecho más accesible. Los sistemas de bases de datos como Postgrest y mSQL han llegado a estar disponibles de forma gratuita o a un bajo costo. Recientemente, los proveedores comerciales como Informix y Oracle han comenzado a ofrecer su software sin costo para sistemas operativos como Linux, sin embargo, estos productos generalmente se entregan en formato binario y sin soporte, lo cual reduce su utilidad.

Actualmente una de las entradas recientes en el terreno de las bases de datos de costo mínimo es MySQL, un sistema gestor de bases de datos relacionales cliente-servidor SQL. MySQL incluye un servidor SQL, programas cliente para acceder al servidor, herramientas administrativas y una interfaz de programación.

[26] Cookies: galletitas, porción de información que es mandada desde el servidor hacia nuestra máquina para ser almacenada durante la gestión de una página Web.

1.10 MySQL

Los orígenes de MySQL se remontan a 1979, con la herramienta de base de datos UNIREG creada por Michael "Monty" Widenius para la empresa Sueca TcX. En 1994, TcX comenzó a buscar un servidor SQL[27] para emplearlo en el desarrollo de aplicaciones Web. De esta forma, comenzaron a probar algunos servidores comerciales, pero se encontraron con que todos eran demasiados lentos para las inmensas tablas de TcX. También observaron mSQL, pero carecía de ciertas características que para TcX eran necesarias. Por lo tanto, Monty comenzó a desarrollar un nuevo servidor. La interfaz de programación fue explícitamente diseñada para que fuera similar a la empleada por mSQL, ya que estaban disponibles varias herramientas gratuitas para mSQL, así que empleando una interfaz similar, esas herramientas se podrían emplear para MySQL, con un esfuerzo mínimo para adaptarlas.

En 1995, David Axmark de Detron HB comenzó a animar a TcX para que hiciera público MySQL en Internet. David también trabajó en la documentación y en la tarea de hacer que MySQL se pudiera construir mediante utilidad GNU configure. MySQL 3.11.1 fue entrega al mundo en 1996 en forma de distribuciones binarias para Linux y Solaris. Hoy en día MySQL funciona en varias plataformas y está disponible tanto en forma binaria como en código fuente.

MySQL no es un proyecto Open Source, ya que bajo ciertas condiciones es necesaria una licencia. No obstante MySQL disfruta de una enorme popularidad entre la comunidad Open Source, ya que los términos de la licencia no son muy restrictivos.

La popularidad de MySQL no está limitada a la comunidad Open Source. Se ejecuta en computadoras personales, es portable y se ejecuta en sistemas operativos comerciales como Solaris, Irix y Windows y en hardware que puede llegar hasta servidores empresariales. Además su rendimiento rivaliza con cualquier sistema de base de datos comercial y puede gestionar grandes bases de datos con millones de registros.

Actualmente empresas que sólo aspiraban a disponer de la potencia de un RDBMS[28] de alto rendimiento para su propio uso, ahora pueden disponer del mismo por un bajo costo. Además el uso de las bases de datos a nivel individual se está incrementando. Personas que no habían considerado el empleo de una base de datos comienzan a pensar en todo tipo de usos para las mismas, una vez que es posible obtenerlas fácilmente, como por ejemplo almacenamiento y acceso de los resultados de una investigación genealógica, seguimiento y mantenimiento de colecciones, ayudar a gestionar el inicio de un negocio o proporcionar capacidades de búsquedas para sitios Web personales.

[27] SQL: Structured Query Language o Lenguaje de Consulta Estructurado.
[28] RDBMS: Relational Data Base Manage System o Gestor de Bases de Datos Relacionales.

1.10.1 Características de MySQL

Existen sistemas de administración de bases de datos libre o gratuito, entre los cuales se puede elegir: MySQL, mSQL, Postgres. Cuando se compare MySQL con otros sistemas de bases de datos es importante considerar los aspectos: rendimiento, mantenimiento, características (conformidad SQL, extensiones, etc.), condiciones y restricciones de la licencia.

Dado que se está usando MySQL como herramienta en el desarrollo del presente trabajo es importante mencionar algunas características del por qué elegir a éste como gestor de base de datos.

1. Velocidad. MySQL es rápido. Es posiblemente la base de datos más rápida que se puede encontrar.

2. Facilidad de uso. MySQL es un sistema de base de datos de alto rendimiento pero relativamente simple y es mucho menos complejo de configurar y administrar sistemas más grandes.

3. Costo. MySQL es gratuito para la mayoría de los usos internos.

4. Capacidad de gestión de lenguajes de consulta. MySQL comprende SQL (Structured Query Lenguaje, Lenguaje de Consulta Estructurado), el lenguaje elegido para todos los sistemas de bases de datos modernos. También es posible acceder a MySQL empleando aplicaciones que admitan ODBC (Open Database Connectivity, Conectividad de Base de Datos), un protocolo de comunicación de bases de datos desarrollado por Microsoft.

5. Capacidad. Pueden conectarse muchos clientes simultáneamente al servidor. Los clientes pueden utilizar varias bases de datos simultáneamente. Se puede acceder de forma interactiva a MySQL empleando diferentes interfases que permiten introducir consultas y visualizar resultados: cliente de línea de comando, navegadores Web o clientes de sistemas X Window. Además, está disponible una amplia variedad de interfases de programación para lenguajes como C, Perl, Java, PHP y Python. Por tanto, existe la posibilidad de elegir entre usar un software cliente pre-empaquetado o escribir las propias aplicaciones.

6. Conectividad y seguridad. MySQL está completamente preparado para el trabajo en red y las bases de datos pueden ser accedidas desde cualquier lugar en Internet, por lo que se puede compartir los datos con cualquiera, en cualquier parte. Pero MySQL dispone de un control de acceso, de forma que aquellos que no deberían ver datos, no los vean.

7. Portabilidad. MySQL se ejecuta en muchas variantes de UNIX, así como en otros sistemas operativos como Windows y OS/2, se ejecuta en hardware que va desde PC hasta servidores de alta capacidad.

La distribución de MySQL incluye las siguientes herramientas:

- Un servidor de SQL. Éste es el motor que impulsa a MySQL y proporciona accesos a sus bases de datos.

- Programas cliente para acceder al servidor. Un programa interactivo que le permite introducir consultas directamente y visualizar los resultados y varios programas administrativos y de utilidad que ayudarán a poner en funcionamiento algún sitio Web. Una utilidad que permite controlar el servidor. Otras permiten importar o exportar datos, comprobar los permisos de acceso.

- Una biblioteca cliente para poder escribir los propios programas. Se pueden escribir clientes en C ya que la biblioteca está en C, también la biblioteca proporciona los fundamentos para enlaces de terceros con otros lenguajes.

1.10.2 Utilidad de MySQL

Esto depende de las necesidades y requerimientos particulares. Los sistemas de administración de base de datos se emplean a menudo para administrar tareas para las que la gente, normalmente, usa archivadores. De hecho, una base de datos es, de alguna manera, como un gran archivador pero con un sistema integrado de archivos. Hay algunas ventajas importantes de los registros mantenidos electrónicamente sobre los manuales. Por ejemplo, si se trabaja en una configuración de oficina en la que se mantienen los registros de los clientes, existen algunas situaciones en las que MySQL puede ayudar:

- Tiempo de grabación reducido. No se tiene que buscar en los cajones para descubrir dónde añadir un nuevo registro. Sólo es cuestión de dejárselo al sistema de archivo, él se encargará de ponerlo en el sitio adecuado.

- Tiempo de recuperación reducido. Cuando se buscan registros, no es necesario mirar cada uno para encontrar la información que se quiere.

- Orden de recuperación flexible. No es necesario recuperar los registros según el orden fijo en que fueron almacenados. Se puede indicar al sistema de archivos que extraiga los registros clasificados en cualquier orden.

- Formato de salida flexible. Después de encontrar los registros que interesan, no es necesario copiar manualmente la información, lo único que se tiene que hacer es imprimir la información.

- El acceso de múltiples usuarios a los registros. Con registros en papel, si dos personas quieren buscar un registro a la vez, la segunda tiene que esperar a que la primera devuelva el registro. MySQL tiene la capacidad de usuarios múltiples, para que ambos puedan acceder simultáneamente al registro.

- Acceso remoto a la transmisión electrónica de registros. Los archivos en papel obligan a estar presentes el lugar de los registros. Los registros electrónicos abren el potencial de acceso remoto a los registros, o a su transmisión electrónica.

1.10.3 API C

En realidad un API, consta de un conjunto de funciones para comunicarse con los servidores MySQL y acceder a las bases de datos y un conjunto de tipo de datos utilizados por dichas funciones.

1.11 Fortran

Fortran es lenguaje de propósito general, principalmente orientado a la computación matemática, por ejemplo en ingeniería. Fortran es un acrónimo de FORmula TRANslator, y originalmente fue escrito con mayúsculas como FORTRAN. Sin embargo la tendencia es poner sólo la primera letra con mayúscula, por lo que se escribe actualmente como Fortran. Fortran fue el primer lenguaje de programación de alto nivel. El desarrollo de Fortran inicio en la década de 1950 en IBM y ha habido muchas versiones desde entonces. Por convención, una versión de Fortran es acompañada con los últimos dos dígitos del año en que se propuso la estandarización [F2]. Por lo que se tiene:

- Fortran 66

- Fortran 77

- Fortran 90 (95)

La versión más común de Fortran actualmente es todavía Fortran 77, sin embargo Fortran 90 está creciendo en popularidad. Fortran 95 es una versión revisada de Fortran 90 la cual fue aprobada por ANSI en 1996. Hay también varias versiones de Fortran para computadoras paralelas. La más importante es HPF (High Performance Fortran), la cual es de hecho el estándar.

Los usuarios deben ser cuidadosos con los compiladores de Fortran 77, ya que pueden manejar un súper conjunto de Fortran 77, por ejemplo contienen extensiones no estandarizadas.

1.11.1 Características de Fortran

Fortran es un lenguaje de programación dominante usado en aplicaciones de ingeniería y matemáticas, por lo que es importante que se tenga bases para poder leer y modificar un código de Fortran. Algunas opiniones de expertos han dicho que Fortran será un lenguaje que pronto decaerá en popularidad y se extinguirá, lo que no ha sucedido todavía. Una de las razones para esta sobrevivencia es “la inercia del software”, ya que una vez que una compañía ha gastado muchos millones de dólares y de años en el desarrollo de software, no le es conveniente traducir el software a un lenguaje diferente, por el costo que implica y por ser una tarea difícil y laboriosa.

CAPÍTULO DOS

DISEÑO DEL SADROV

En este capítulo se tratan los fundamentos de diseño del SADROV.

2.1 Antecedentes

Como el objetivo primario de la tesis es desarrollar un sistema de manejo de datos mediante Internet, lo más sencillo es realizarlo con un Administrador de Bases de Datos Relacionales (MySQL) para la organización e intercambio de los datos como tales[1] y utilizar herramientas nativas del Internet (es decir herramientas Web) como PHP, HTML de elementos de apoyo. Observándolo desde este punto de vista, la piedra angular de este sistema es por supuesto la estructura de la base de datos.

Para el diseño de la base de datos (llamada RELAP_DB) se tomaron en consideración varios puntos:

- Que permita manejar datos de diferentes modelos de entrada corriendo en diferentes lugares (multiusuario).
- Que permita guardar información sobre cada modelo de entrada.
- Que permita guardar información de "despliegue"[2] por cada modelo.
- Que organice de forma clara los datos provenientes de cada modelo de entrada y "despliegue".
- Que sólo puedan darse de alta "despliegues" mediante el propio sistema, para controlar el formato de la misma.
- Que permita dar de alta variables de "despliegue", es decir los datos que requiere un cierto mímico o gráfico para el simulador.
- Que la información guardada del modelo de entrada sea genérica, es decir conforme al formato de organización de los códigos RELAP5, esto con la finalidad de que esta misma base pueda ser usada por cualquier usuario de los códigos y no solamente por el Simulador de Aula.

En las siguientes subsecciones analizaremos la estructura básica de los modelos de entrada de los códigos RELAP5 (sección 2.2), para entender mejor la forma de ingreso de la información a los códigos, con esto como referencia se propondrá el diseño de la base de datos (sección 2.3) y el diseño de la interfaz gráfica (2.4) que servirá a nuestro sistema para registrar modelos en la base de datos.

[1] Ver figura 1.4.1.
[2] Es decir cada modelo de entrada va a requerir diferentes datos para almacenar en la base de datos y aun para modelos iguales podrían requerirse datos diferentes, según las necesidades del desarrollador de éstos.

2.2 Estructura Básica de los Modelos de Entrada de los códigos RELAP5

Tomando en consideración las características deseadas para la base de datos RELAP_DB, se analizó el formato de entrada de los modelos de los códigos RELAP5, en este apartado veremos un poco de este formato y las consideraciones necesarias para el desarrollo del RELAP_DB.

Los modelos de entrada de los códigos de simulación RELAP5 son ingresados mediante un archivo de texto llamado “indta”, como mínimo contiene una línea de titulo que se inicia con un carácter “=” y una línea de terminación de modelo que contiene la cadena “.end”. La organización del “indta”, está basada en líneas de caracteres que llamaremos “tarjetas” con código numérico como primera palabra y en palabras subsecuentes que pueden contener diferentes cosas dependiendo de la primera palabra o “número de tarjeta”, en la figura 2.2.1 se puede apreciar el formato de la entrada:

Formato de Tarjetas

Card code: código numérico de la tarjeta

W1: palabra 1 **W3: palabra 3**

W2: palabra 2 **WN: palabra N**

Figura 2.2.1

El “indta” permite hacer comentarios en el modelo, en este caso los caracteres que indican comentarios son “*” y ”$”. Cada línea del archivo que comience con un espacio en blanco será tomada como comentario así como cualquier cadena de caracteres que siga de un “*” o un “$”.

2.2.1 Componentes Termodinámicos

El modelo de entrada de los códigos RELAP5 contiene componentes genéricos simples con los que se pueden formar sistemas termodinámicos más complejos, cada uno de estos componentes tiene sus variables físicas (por ejemplo velocidad de flujo, potencia, densidad). La lista total de variables de estos componentes se encuentra en el apéndice A del manual del RELAP5 [12]. La lista de componentes termodinámicos del RELAP5 analizados (para efecto de la tesis) se encuentra en la tabla 2.2.1.

Tabla 2.2.1 Componentes Termodinámicos

Componentes Termodinámicos	Descripción
ACCUM	Acumulador, componente que modela el tanque y la salida de un acumulador.
ANNULUS	Volumen de control circular, idéntico al componente PIPE, excepto que éste debe estar en posición vertical.
BRANCH	Componente genérico del cual se derivan componentes especializados como el ECCMIX, JETMIXER, SEPARATOR y TURBINE.
ECCMIX	Mezclador de vapor ECC, componente especializado del BRANCH.
HEAT STRUCTURE	Estructura de calor, componente que modela una estructura de transferencia de calor.
JETMIXER	Mezclador JET, componente que modela un mezclador de vapor.
MTPLJUN	Unión múltiple, componente que modela una juntura múltiple.
PIPE	Volumen de control circular, similar a ANNULUS.
PUMP	Componente que modela una bomba.
SEPARATR	Componente que modela un separador de vapor, componente especializado del BRANCH.
SNGLJUN	Unión simple, modela una juntura de componentes.
SNGLVOL	Volumen de control simple, modela un volumen de control simple.
TMDPJUN	Unión dependiente del tiempo, juntura de componentes que puede cambiar en el tiempo sus características.
TMDPVOL	Volumen de control dependiente del tiempo, componente que puede cambiar sus características en el tiempo.
TURBINE	Componente que modela las características de una turbina.
VALVE	Componente que modela diferentes tipos de válvulas.

2.2.2 Tarjeta CCC0000

Esta tarjeta indica un componente termodinámico en el archivo de modelo, donde CCC es un número de tres cifras (001 - 999), esta tarjeta es la usada para la mayoría de los componentes termodinámicos[3]. Está compuesta por las siguientes palabras [I6]:

- W1[4]. Nombre del componente. Este nombre es descriptivo del componente (depende del "modelador"[5]) con un máximo de 10 caracteres.

- W2. Tipo de componente. Esta palabra distingue cualquiera de los siguientes componentes[6]: SNGLVOL, TMDPVOL, SNGLJUN, TMDPJUN, PIPE, ANNULUS, BRANCH, SEPARATR, JETMIXER, TURBINE, ECCMIX, VALVE, PUMP MTPLJUN, ACCUM.

2.2.3 Tarjeta 1CCCG000

Esta tarjeta me indica una estructura de transferencia de calor, donde CCC es el número de la estructura de calor y G me indica la geometría de la misma. Está compuesta por las siguientes palabras [I10]:

- W1. Número de estructuras de transferencia de calor axiales con esta geometría, nh. Este número debe ser menor a 100 y mayor a cero.

- W2. Número de puntos de malla radial para esta geometría, np. Este número debe ser menor a 100.

- W3. Tipo de geometría. 1 indica rectangular, 2 cilíndrica y 3 para esférica.

En total está compuesta de ocho palabras, sin embargo para nuestros fines es suficiente con la tarjeta en sí, para detectar la presencia del componente en el modelo. En futuros trabajos podría ampliarse la profundidad del análisis de ésta y otras tarjetas para conseguir una información detallada del modelo[7].

[3] Ver sección 2.2.1
[4] W: abreviatura de word, o palabra en español.
[5] Modelador: en este caso se refiere a la persona que define el modelo físico a simular.
[6] Ver tabla 2.2.1
[7] Por ejemplo en la geometría y lograrse una representación gráfica del mismo al consultar la base de datos.

2.2.4 Tarjeta 3XX

Esta tarjeta indica un requerimiento de “minor edit “, es decir una impresión de una cierta variable en el archivo de salida de los códigos RELAP5[8] para su análisis. El número total de variables disponibles para los “minor edits” de los componentes y algunas cantidades extras se encuentran en el apéndice A del manual del RELAP5 [I4].

Debe hacerse notar que para nuestros propósitos (el manejo de variables desde los códigos RELAP5 hacia la base de datos), debemos tener una información de todas ellas, por lo que se puede guardar la información disponible de ellas de manera organizada en la base de datos RELAP_DB[9]. Esta información también podría ser usada como ayuda y referencia rápida para los desarrolladores de nuevos modelos para el simulador de aula.

Donde XX va del 01 al 99. Esta tarjeta está formada por dos palabras:

- W1. Código alfanumérico de la variable[10], dependiendo del componente que haga referencia la palabra W2, este código puede variar.

- W2. Parámetro numérico que indica de que componente debe venir la variable que se pide para el archivo “outdta”, este parámetro es del tipo CCCXXXX00, donde CCC es el número de componente ingresado en la tarjeta CCC0000 y XXXX es diverso, dependiendo de que componente se trate.

2.2.5 Tarjeta 102

Esta tarjeta indica el tipo de unidades de la entrada del modelo y las unidades de la salida. Si la tarjeta es omitida en el modelo las unidades asumidas son las del SI[11] . Está compuesta por las siguientes palabras [I3]:

- W1. Unidades de entrada al modelo. Puede ser SI o BRITISH. Donde SI indica sistema Internacional y BRITISH el Sistema Ingles de unidades.

- W2. Unidades de salida del modelo. Puede ser SI o BRITISH. Donde SI indica sistema Internacional y BRITISH el Sistema Ingles de unidades.

[8] El archivo de salida de los códigos RELAP5 también es llamado “outdta”.
[9] De hecho al diseñarse la DB RELAP_DB se tomó como base esta organización de las variables y componentes.
[10] Por no considerarse de interés para el presente trabajo la lista de variables no fue incluida en este capítulo, sin embargo la base de datos RELAP_DB cuenta con la información completa de éstas, así como su descripción. Tal como se encontró en el Apéndice A del manual de RELAP5.
[11] SI: Sistema Internacional.

2.2.6 Tarjeta 205CCC00 ó 205CCCC0

Esta tarjeta se refiere a componentes de control, es decir variables que funcionarían para lograr un sistema de control en el modelo. Se definen con las variables que proporcionan los códigos a través de los "minor edits"[12], sin embargo pueden ser usadas para realizar cálculos y operaciones sobre las variables antes mencionadas[13]. El número CCC o CCCC se refiere al número de la variable de control y la longitud de este número la define la tarjeta 20500000 [I12]. Las palabras que contiene son las siguientes:

- W1. Código alfanumérico que indica el nombre de la variable dado por el modelador, que puede ser descriptivo de la variable. Hasta de ocho caracteres.

- W2. Tipo de componente, se refiere al tipo de operación que se puede realizar con la(s) variable(s) seleccionadas. Puede ser SUM, MULT, DIV, DIFFRENI, DIFFREND, INTEGRAL, FUNCTION, STDFNCT, DELAY, TRIPUNIT, TRIPDLAY, POWERI, POWERR, POWERX, PROP-INT, LAG, LEAD-LAG, CONSTANT, SHAFT, PUMPCTL, STEAMCTL o FEEDCTL, o el comando DELETE para borrar el componente.

2.2.7 Tarjeta 205CCCNN ó 205CCCCN

Esta tarjeta define los componentes de una variable de control [I12]. EL número CCC o CCCC se refiere al número de la variable de control. El número NN (01-99) o N (1-9) se refiere a que una variable de control puede estar definida en una o más tarjetas. La variable se define con NN=00 o N=0[14] y en las tarjetas sucesivas se definen los componentes de esta variable. El formato de las palabras que la forman se define dependiendo del tipo de operación que se defina en la variable de control (SUM, DIV, DELAY entre otras). Debido a que para el propósito de la base de datos RELAP_DB es suficiente el reconocer el tipo de operación, no se presentarán por ahora las palabras que lo conforman.

2.2.8 Tarjeta 20500000

Esta tarjeta define el número máximo de variables de control, si esta tarjeta es omitida por default se toma el formato 205CCCNN [I12], lo que indica que las variables de control van de 001 a 999. Está formada por las siguientes palabras:

- W1. El valor de esta palabra es 999 si se desea el formato 205CCCNN y 9999 si se desea el formato 205CCCCN.

[12] Ver sección 2.2.4
[13] Por ejemplo un cambio de unidades o una suma de varias variables, entre otras.
[14] Tarjeta 205CCC00 o 205CCCC0.

2.2.9 Tarjetas 401 a la 599

Cada tarjeta define un "trip"[15]. En otras palabras es el disparo, o cambio de valor de una variable mediante una condición, la variable[16] es falsa o tiene un valor hasta que se cumpla la condición (puede ser disparada también por tiempo) especificada en la tarjeta o viceversa. En este caso es de nuestro interés particular esta tarjeta, debido a que mediante estos "trips" se pueden "abrir" o "cerrar" variables de control en el modelo[17] con lo que se puede lograr simular por ejemplo el apagado de una turbina o abrir alguna válvula, etc. En el simulador de aula en desarrollo se obtiene interactividad con los códigos RELAP5 a través de estos "trips" cambiando manualmente la condición y disparándolos a voluntad.

Las palabras que forman esta tarjeta son las siguientes [I5]:

- W1. Código de la variable, puede tomar otros valores pero para nuestros propósitos lo veremos de esta manera.
- W2. Un parámetro.
- W3. Una condición de relación: mayor que, mayor o igual que, menor que, menor o igual que, igual que, diferente que. Estas relaciones están indicadas por las palabras: GT, GE, LT, LE, EQ y NE respectivamente.
- W4. Código de variable.
- W5. Parámetro.
- W6. Constante aditiva.
- W7. Indica si el "trip" es del tipo "latched"[18] (se indica con una "L") o del tipo "unlatched"[19] (se indica con una "N"). Si es de tipo "latched", sólo podrá cambiar su estado una sola vez en la simulación. Si es de tipo "unlatched", cambiará su valor tantas veces como el usuario lo solicite. Se advierte sobre este comportamiento debido a que si el sistema del que el Trip forma parte no está propiamente modelado, los permisivos para su cambio de estado en la lógica real pueden no cumplirse al momento del cambio de estado con consecuencias catastróficas para la simulación.[20]

[15] Trip: En este caso se refiere a un disparo.
[16] El trip en si.
[17] En este caso nos permitiría tener interactividad con el RELAP5 si nosotros disparamos manualmente.
[18] Latched: En este caso es cerrado.
[19] Unlatched: En este caso es abierta.
[20] "Interfaz Interactiva para RELAP/SCADAP y el Simulador de Aula", M.C. Juan Carlos Ramos, 2003, GRINFI UNAM.

- W8. Cantidad TIMEOF, se refiere a un tiempo de ejecución específico en la simulación. Esta palabra es opcional, si no es puesta en la tarjeta el "trip" es inicializado como falso y su tiempo de inicialización es puesto a -1. Si esta cantidad es cero o mayor a cero, el "trip" es inicializado como verdadero.

Debe notarse que el interés primario en este tipo de componente es para lograr una cierta interactividad con los códigos desde la base de datos. Tal como la tiene actualmente el Simulador de Aula.

2.3 Diseño propuesto para la RELAP_DB[21]

Siguiendo los lineamientos propuestos en la sección 2.1 y tomando en cuenta la estructura básica de los modelos del RELAP5 vista en la sección 2.2, la estructura modular propuesta es la que se muestra en la figura 2.3.1:

Estructura Modular Propuesta para la RELAP_DB

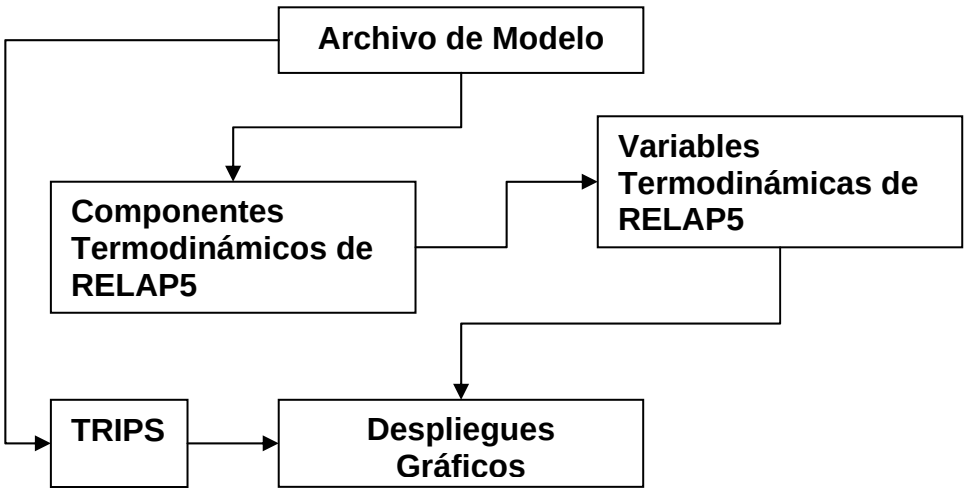

Figura 2.3.1

Cada módulo de la figura 2.3.1 representa una parte a considerar para el diseño de la base de datos, y la forma en la que están conectados representa la interacción entre ellos y el sentido de ésta. Como puede observarse el módulo "Despliegues Gráficos" es el resultado final de los módulos que le preceden y fin último de la base de datos, es decir, dar servicio a los "despliegues gráficos"[22] del simulador.

[21] Base de Datos que guardará la información proveniente de los códigos RELAP5
[22] Clientes finales del sistema.

2.3.1 Archivo de Modelo

Este módulo es básicamente el que permitirá a la base de datos ingresar los diferentes modelos físicos a simular por el RELAP5, tomando en cuenta el formato del archivo "indta"[23]. Este módulo proporcionará la información necesaria a los demás módulos para que puedan realizar sus funciones y almacenar información necesaria para el desarrollo de un "despliegue gráfico".

2.3.2 Componentes Termodinámicos del RELAP5

Este módulo contiene la información necesaria para reconocer los diferentes componentes termodinámicos del RELAP5 y con la información que le proporciona el módulo "Archivo de Modelo" se inicia el reconocimiento y organización de la información del "despliegue gráfico".

2.3.3 Variables Termodinámicas del RELAP5

Este módulo contiene la información necesaria para inicializar las diferentes "variables de despliegue" según el formato del modelo de entrada de los códigos de simulación para variables disponibles en los "minor edits" y variables de control. Usa la información recabada por el módulo de "Componentes Termodinámicos de RELAP5" para reconocer y dar de alta[24] nuevas variables de despliegue. Esto para cada modelo de entrada ingresado en el módulo "Archivo de modelo".

2.3.4 Trips

Este módulo contiene información necesaria para reconocer los "trips"[25] del modelo de entrada según el formato del RELAP5, información que obtiene del "Archivo de Modelo", e inicializarlos en la base de datos para que se puedan dar de alta por los "despliegues gráficos".

2.3.5 Despliegues Gráficos

Este módulo es el que lleva el control de los modelos físicos dados de alta en la base de datos. Aquí es importante destacar que según los lineamientos vistos en la sección 2.1 la RELAP_DB debe ser multiusuario, así este módulo contiene la información necesaria para almacenar la información útil de los diferentes archivos de modelos dados de alta como "despliegues" identificándolos con un número de despliegue único para cada modelo, así mismo contiene la información necesaria sobre las "variables de despliegue" dadas de alta y los "trips" activos para el mismo[26].

[23] Ver sección 2.2.
[24] Se refiere a inicializar el espacio para albergar los datos provenientes del RELAP5 para ser desplegados por los despliegues gráficos.
[25] Ver sección 2.2.9.
[26] No todas las llamadas variables de control van a ser mandadas a la base de datos, solamente las que el modelador desee desplegar. Los "trips" están en el mismo caso.

2.4 Desarrollo de Interfaz Gráfica del SADROV

El objetivo de una interfaz gráfica de usuario es el manejo de información ya sea almacenada tanto en un archivo como en una base de datos relacional, y permitir editar información e incluso agregar elementos nuevos que puedan actualizar la base de datos.

Para nuestros propósitos se han desarrollado interfaces gráficas de usuario, que se emplean como “herramientas” de trabajo ya que a través de ellas se tiene una relación usuario–máquína, por ejemplo, podemos elegir opciones de un menú, procesar y alterar información de archivos, dadas estas peticiones el sistema deberá responder enviándonos la información necesaria para continuar con nuestros procesos.

Para el diseño y programación de las interfaces graficas se han considerado los siguientes puntos:

- El diseño de la interfaz gráfica se ha pensado y adecuado de acuerdo a la estructura de la base de datos.
- Facilitar al usuario final el uso del sistema en sí[27], a través de una presentación sencilla y agradable.
- Que las aplicaciones escritas para cada interfaz sean independientes.
- Interfaz gráfica orientada a la web alojada en el servidor, para multiusuarios.
- Registro de despliegues gráficos en la base de datos.
- Manejo de variables de despliegues[28] y de trips contenidas en la base de datos.
- Adecuación del archivo de modelo de entrada para un despliegue gráfico.
- Limites de formato de la interfaz gráfica.

[27] SADROV. Sistema de Adquisición de Datos y Registro de Variables
[28] Ver sección 2.3.5

2.4.1 Diseño propuesto para la realización de la interfaz y su relación con la Base de Datos

Toda interfaz gráfica se diseña de acuerdo a las necesidades de cada una de las aplicaciones que integran al sistema en sí, es por ello que nuestra interfaz debe ser diseñada de acuerdo a las necesidades de la estructura de la base de datos, es decir, teniendo en cuenta el manejo de variables de despliegue y trips, que son variables importantes para los objetivos del sistema.

El funcionamiento de una interfaz de manera adecuada es todo un reto. Existen muchas soluciones que pueden considerarse y tecnologías atractivas que pueden distraernos de lo que realmente es importante: la construcción del contenido, la flexibilidad y ayudar al usuario a encontrar la información que necesita.

Existen diferentes metodologías o normas que rigen el diseño de una buena interfaz. Para el desarrollo de la interfaz que implementamos tomamos aquellas que consideramos adecuadas para la utilización de información que estaremos manejando y para la audiencia que será principalmente formada por usuarios con conocimientos de códigos RELAP5.

En cuanto a nuestro diseño consideramos los siguientes aspectos[29]: encontrar una presentación que no sea cansada y que no incluya muchas imágenes para evitar una larga espera para que sea cargado en la memoria de la computadora, hacer uso de texto para hacer indicaciones al usuario, el color del fondo de la interfaz debe preferirse de un color suave, que no canse la vista y que permita la visualización de cualquier imagen o texto.

La interfaz será un fracaso si el sistema no hace lo que los usuarios necesitan, en otras palabras, el sistema debe de cumplir con las tareas de los usuarios.

Es por esto, que el proceso del diseño se define como "centrado en las tareas del usuario", y por esta razón, el diseño de la interfaz no puede ser separada del diseño del resto del sistema.

[29] En general Factores Humanos en el diseño de Interfaces Gráficas.

2.4.2 Facilidad de uso de la Interfaz

En sistemas computacionales, es siempre importante contar con una buena interfaz de usuario, para ello es necesario tener clara la misión y visión de la aplicación, balanceando las necesidades de los que ofrecen la información como de los que la consultan. Es importante también, determinar la funcionalidad y organización del sistema en desarrollo.

Teniendo en cuenta las consideraciones anteriores se ofrece al usuario del sistema SADROV, una interfaz gráfica amigable y sencilla de utilizar, con la finalidad de permitir inserciones y búsquedas precisas de variables de despliegue en la base de datos, así como también, manejo de información de archivos, esto gracias al uso de formularios, de manera que el uso adecuado de radio-buttons, check-boxes, es un factor decisivo para el éxito de formularios[30]. Hay que contemplar que la satisfacción del usuario no sólo se logra con buena tecnología y gráficos atractivos.

2.4.3 Independencia entre aplicaciones relacionadas a las interfaces gráficas

Para los autores de aplicaciones, las interfaces gráficas de usuario ofrecen un entorno que se encarga de la comunicación con la computadora. Esto hace que el programador pueda concentrarse en la funcionalidad, ya que no está sujeto a los detalles de la visualización ni a la entrada a través del ratón o del teclado. También permite a los programadores crear programas que realicen de la misma forma las tareas más frecuentes, como tomar o seleccionar un archivo de alguna dirección en especifico, guardar un archivo, porque la interfaz proporciona mecanismos estándar de control como botones de selección, ventanas y cuadros de diálogo[31]. Otra ventaja es que las aplicaciones escritas para cada interfaz gráfica de usuario son independientes, por ejemplo el lector de datos y el manejador de variables de despliegue tienen las siguientes características:

Lector de Datos. Básicamente analiza el contenido del archivo de modelo de los códigos RELAP5 y guarda la información útil[32] de este en la base de datos para inicializar un “despliegue”.

Manejador de Variables de Despliegue. Básicamente utiliza la información del “despliegue” almacenada en la base de datos para registrar variables de despliegue, trips y variables constantes haciendo uso de formularios.

[30] Los formularios interpretan las acciones que una persona realiza sobre nuestra página y obtiene en consecuencia una respuesta determinada.
[31] Cuadro de Diálogo. Es una ventana para un propósito especial que le está pidiendo que ingrese algo. Puede ser que tenga que escribir alguna cosa, que podría ser su nombre o el de un archivo.
[32] Ver sección 2.3

2.4.4 Interfaz Gráfica orientada a la Web

En este punto es importante mencionar que no todas la interfaces gráficas en lo general que se desarrollan para un sistema en especifico son dirigidas a la Web, para nuestro caso, dado que es un sistema en el cual se puede trabajar a distancia mediante el uso del Internet, es necesario el uso de una interfaz de este tipo.

Para visualizar nuestra interfaz no es necesario utilizar o instalar software en especifico basta con utilizar la ventana de Internet Explorer que se encuentra prácticamente en el Sistema Operativo de cualquier computadora, así como también, no es necesario estar conectado al Internet, dado que nuestro sistema funciona a nivel Red LAN (Intranet) y/o Red WAN.

2.4.5 Funciones a realizar por la Interfaz Gráfica

Existen básicamente tres funciones principales que son muy utilizadas en nuestro sistema, estas son: Registro de despliegues gráficos en la base de datos, manejo de variables de despliegue y la adecuación del archivo de modelo de entrada. A continuación describiremos cada una de ellas en detalle:

Registro de despliegues gráficos en la base de datos. Esta etapa consiste básicamente en tomar el archivo de entrada del modelo físico (Indta), analizar al mismo para detectar los componentes termodinámicos, los minor edits y variables de control, posteriormente guardarlas en la tabla “tbl_input” de la base datos Relap_DB. Cabe mencionar que esta tabla contiene los siguientes campos: id de despliegue, nombre de despliegue dado por el usuario y la fecha cuando fue dada de alta dichos componentes, variables y minor edits.

Manejo de variables de despliegue, componentes termodinámicos y trips contenidas en la base de datos. Una vez almacenados los elementos anteriores es posible tener el control de estos mediante el manejador de variables de despliegue. A través del manejador de variables es posible elegir o dar de altas variables de despliegue en la base de datos. Es importante mencionar que sólo se podrán elegir variables de despliegue del archivo de modelo de entrada (Indta) con el cual se está trabajando, esta aplicación está diseñada de tal forma que se guarde en la BD un id de despliegue, con el cual podremos identificar el tipo de modelo que con el cual se esté trabajando, así como también el tipo de variables contenidas en la misma base de datos. Otro aspecto a considerar es que se podrán dar de alta variables de despliegue, creando los campos necesarios en la BD, esto con el objetivo de poderlas utilizar en un despliegue gráfico.

Adecuación del archivo de modelo de entrada para un despliegue gráfico. Consiste básicamente en dar formato al archivo de modelo (Indta), de tal forma que sea adecuado para su utilización en el proceso de ejecución del código RELAP5.

Modularmente, lo anteriormente dicho se muestra en la siguiente figura 2.4.1:

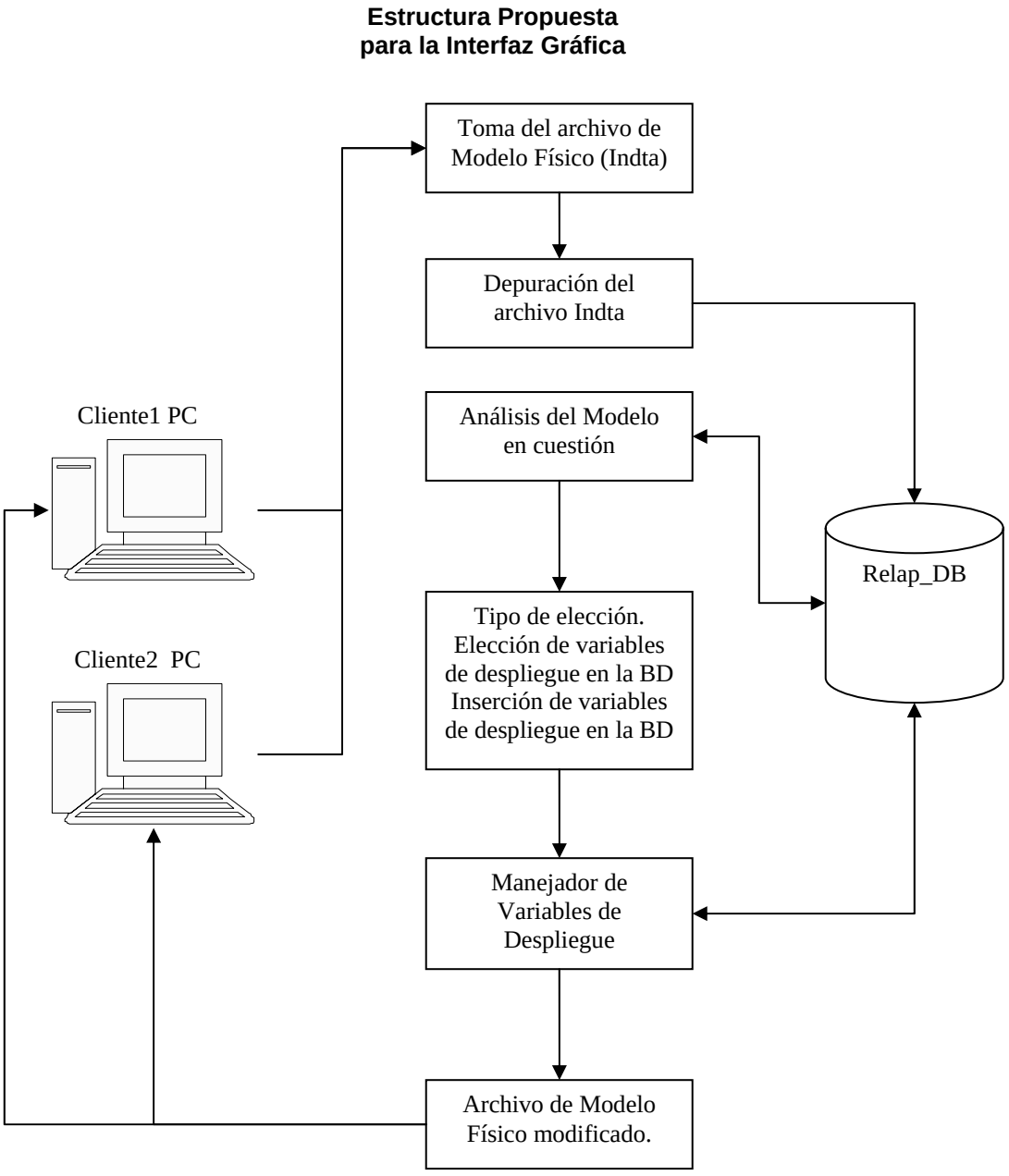

Figura 2.4.1

2.5 Interfaz de Comunicación para los códigos RELAP5

Esta interfaz[33] permitirá a los códigos de simulación comunicarse con la base de datos RELAP_DB de forma continua[34] y remota vía TCP/IP[35]. Para ello esta interfaz debe tener las siguientes características:

- Establecer una comunicación con la RELAP_DB de forma rápida, para que los despliegues reciban esta información.

- Reconocer el despliegue gráfico dado de alta en la base de datos al que le corresponde el modelo de entrada actual.

- Reconocer las "variables de despliegue" y "trips" activos en la base de datos para el modelo de entrada que están simulando los códigos RELAP5.

- Enviar los datos de interés en cada iteración de los códigos. Así como recibir datos de la misma forma para lograr una interactividad despliegue-código.

2.5.1 Antecedentes de Interactividad en el Simulador de Aula

Esta sección tiene por objetivo explicar de manera somera el mecanismo que permite interactuar con el código RELAP5 y demás componentes del Simulador de Aula [S1], debido a que nos basaremos en este trabajo previo para lograr nuestra propia interfaz.

Con la finalidad de interactuar de manera ordenada con los códigos RELAP5 a través de los despliegues gráficos (diagramas de los sistemas, tableros virtuales, gráficas de tendencia, etc.), es necesario definir la forma en la que se obtendrá la información para las salidas, y también la manera en la que se introducirá la información para modificar los parámetros que permitirán operar al simulador.

Información de salida:

- La información de salida que se requiere para los despliegues se proporciona a través de las variables que RELAP define a través de los "minor edits".

- Debido a que la información puede requerirse en distintas unidades, las variables de salida se calculan por medio de variables de control, ya en las unidades finales, de forma que la información que se envía a un canal de memoria compartida va en las unidades necesarias para el despliegue.

[33] El nombre de esta interfaz es ICRELAP.
[34] Para actualizar la información de despliegue del modelo en simulación.
[35] Es la forma más conveniente para comunicar datos (de forma sencilla) a través del Internet.

- En el archivo “indta”[36] se definen las variables de control que se deseen de acuerdo a los requerimientos de información de salida. En el mismo directorio en el que se encuentra el archivo de modelo debe existir un archivo con el nombre “varcdes.dat”, en el que se deben incluir dos columnas. En la primera debe especificarse la variable de control que será enviada a la localidad de memoria compartida que se indica en la segunda columna.

- El usuario es advertido si alguna de las variables de control en la lista no existe, y se detiene la ejecución. De lo contrario, el programa sigue su ejecución y se actualiza la memoria compartida cada iteración.

Información de entrada:

- Los códigos permiten la ejecución de acciones lógicas de disparo e interacción entre componentes por medio de dos tipos de componentes: los “trips” y las variables de control[37]. Con la intención de no modificar este esquema, la interactividad entre los controles del simulador y RELAP5 se define utilizando los mismos medios.

- Interactividad a través de “trips”. Se definen con anticipación aquellos “trips” que serán modificados por el módulo interactivo, estos deberán definirse formalmente en el archivo de entrada del modelo de RELAP5 que se esté utilizando, y se proporciona una lista en un archivo de datos de entrada de nombre “tripint.dat” que es leído por los códigos al iniciar la simulación. El archivo contendrá dos columnas, en la primera se incluye el número de “trip”, y en la segunda la localidad de memoria compartida de la que toma el valor.

- La interactividad a través de variables de control. Se define de tal forma que el valor modificado que se recibe de tableros o mímicos sea utilizado como una constante en construcciones de control más elaboradas dentro del modelo de RELAP5 utilizado. Esta forma se incluye para permitir el intercambio de valores constantes entre el usuario y el programa, y no sólo de condiciones de disparo como lo permiten los Trips. De nuevo, las componentes de control que se utilizan para la interactividad, deberán definirse formalmente en el archivo de tarjetas de entrada del modelo que se emplea, y se proporciona una lista en un archivo de datos de entrada de nombre “contint.dat” que es leído por RELAP5 al iniciar la simulación. El archivo contendrá dos columnas, en la primera se incluye el número de componente de control, y en la segunda la localidad de memoria compartida de la que toma el valor.

[36] Ver sección 2.2
[37] Ver sección 2.2.6

2.5.2 Diseño propuesto de la ICRELAP

Tomando en consideración lo anteriormente expuesto en la sección 2.5.1, la figura 2.5.1 representa la interfaz de comunicación propuesta.

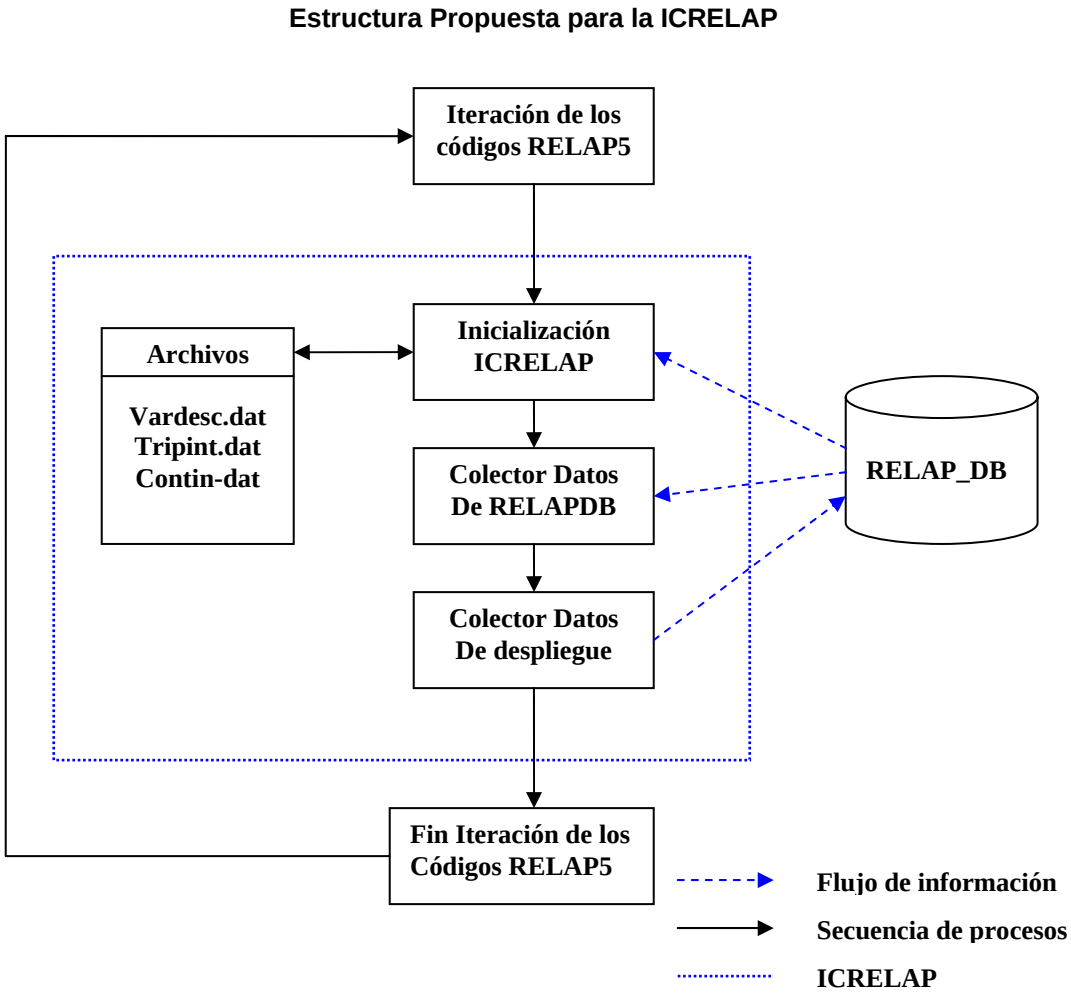

Figura 2.5.1

Como puede observarse en la figura 2.5.1 hay tres módulos que conforman a la ICRELAP (Interfaz de comunicación con RELAP5) y cada uno tiene flujos de información[38] con RELAP_DB.

[38] Flujo de información: Se refiere a intercambio de información con la base de datos.

2.5.3 Inicialización de la ICRELAP

Este módulo tiene como objetivo primordial el recabar la información necesaria de la base de datos para lograr la transferencia de datos de una forma clara. Esto se logra de la siguiente forma:

- Obtener el identificador de despliegue del archivo de modelo.
- Obtener el total de variables de control y trips activos del despliegue.
- Obtener el nombre de las tablas de la base de datos donde se guardará u obtendrá la información del despliegue, según sea el caso.
- Crear, en la primera iteración de los códigos, los archivos varcdes.dat, tripint.dat y contint.dat necesarios para la transferencia de información.

2.5.4 Colector de datos de RELAP_DB

Este módulo tiene como objetivo el recolectar la información contenida en la base de datos para los "trips" y variables constantes[39] y modificar con estos valores la información de los mismos en los códigos RELAP5. Esto se logra de la siguiente forma:

- Obtener, sólo la primera vez, la información de los "trips" y variables constantes de los archivos contint.dat y tripint.dat.
- Obtener los datos requeridos en el punto anterior de la RELAP_DB.
- Cambiar los datos de ser necesario, es decir si los datos de la RELAP_DB y los del RELAP5 son iguales no modificarlos.

2.5.5 Colector de Datos de Despliegue

Este módulo tiene como objetivo recolectar la información de las "variables de despliegue"[40] de la iteración actual en los códigos y enviarlas a la RELAP_DB. Esto se logra de la siguiente forma:

- Obtener, sólo la primera vez, la información de las "variables de despliegue" del archivo varcdes.dat.
- Obtener los datos requeridos en el punto anterior de los códigos de simulación y después enviar los datos a la base de datos. Incrementando un identificador de número de iteración.

[39] Ver sección 2.5.1
[40] Variables de control de RELAP5.

CAPÍTULO TRES

DESARROLLO DEL SADROV

En este capitulo se describe a detalle la estructura del SADROV, tratando cada uno de sus componentes, es decir la RELAP_DB, la ICRELAP y la Interfaz Gráfica. Además se tocan algunos conceptos necesarios para un mejor entendimiento de la estructura del SADROV.

3.1 Estructura de la RELAP_DB

Para comprender mejor la estructura lógica de la base de datos, es necesario revisar brevemente los objetivos básicos de un manejador de bases de datos o SGBD[1] y el modelo de datos usado, que va relacionado con el SGBD utilizado, en este caso el MySQL que es un Manejador de Bases de Datos Relacionales.

3.1.1 Objetivos del SGBD

Si definimos una Base de datos como "un conjunto de archivos relacionados por apuntadores en todos los sentidos, guardando un máximo de coherencia entre ellos organizados de manera que responda eficazmente a una gran variedad de necesidades" [G1]. Podemos definir a un SGDB como un "conjunto de programas y dispositivos que permiten organizar y estructurar datos para que puedan ser manipulados por usuarios y programas".

Los objetivos principales de un SGDB son los siguientes:

- Independencia física. Es decir busca la independencia entre las estructuras de almacenamiento y las estructuras de datos del mundo real. Se trata de poder definir la agrupación de los datos elementales en el sistema informático, independientemente de la agrupación realizada en el mundo real.

- Independencia lógica. Cada aplicación debe poder trabajar conociendo solamente una parte de la semántica de los datos y viendo solamente una parte de ellos. Así cada grupo de trabajo (o aplicación) ve los datos tal y como le interesan.

- Manipulación de los datos por personas sin conocimientos informáticos. Se podrán manipular los datos por medio de lenguajes no sujetos a procedimientos, es decir, describiendo los datos que se desean consultar (o actualizar) sin describir la forma en que hay que consultarlos (o actualizarlos) que es propia de la máquina.

- Eficiencia en el acceso a los datos. Se debe contar con lenguajes de manipulación de datos muy eficaces que permitan acceder rápidamente a los datos a través de los caminos de acceso definidos en la estructura de almacenamiento de la base de datos.

- Administración coherente de los datos. La definición de las estructuras de almacenamiento y las estructuras de los datos así como sus evoluciones es una de las funciones principales del SGBD. Por lo que para contar con un control eficaz de los datos es esencial responsabilizar de estas funciones a personal calificado lo que provoca una administración de los datos centralizada.

[1] SGBD. Sistema Gestor de Bases de Datos

- No redundancia en los datos. Las duplicaciones de datos conducen a un importante desperdicio de recursos computacionales, así como de recursos humanos para capturar y actualizar los datos varias veces.

- Coherencia en los datos. Los datos deben estar sujetos a ciertas reglas, ya sea a nivel dato elemental o a nivel de un conjunto de datos en el que pueden existir determinadas dependencias entre ellos. Así una de las funciones del SGBD es vigilar que las aplicaciones respeten estas reglas durante las modificaciones de los datos asegurando la coherencia de los mismos.

- Compartir los datos. Debe permitir que las aplicaciones compartan los datos simultáneamente. Así una aplicación debe poder acceder a los datos como si fuese la única que los está utilizando sin esperar y también sin necesidad de saber que cualquier otra aplicación puede modificarlos concurrentemente.

- Seguridad de los datos. Los datos deben estar protegidos contra accesos no autorizados o malintencionados. Deben existir mecanismos adecuados para controlar o retirar las autorizaciones de acceso de cualquier usuario para cualquier conjunto de datos.

3.1.2 Arquitectura Cliente-Servidor del SGBD

En esta arquitectura se tienen PC's comunicadas a través de una red[2] con un servidor de bases de datos que contiene bases de datos compartidas. Aquí las funciones del SGBD se dividen en dos:

1. Database Front-end. Que se refiere a las aplicaciones clientes corriendo en las PC'S.

2. Database Back-end. Que se refiere al motor de la base de datos que se encarga de guardar y manejar los datos en el servidor.

Si se considera que más de un cliente esta pidiendo un dato. La petición viaja en la red al servidor. El motor del SGBD en el servidor procesa la petición y busca en la base de datos. Cuando el dato es encontrado el mismo motor manda el dato respuesta a la petición. En la figura 3.1.1 se puede observar esta arquitectura.

[2] Esta red puede ser del tipo LAN o WAN.

Arquitectura Cliente - Servidor

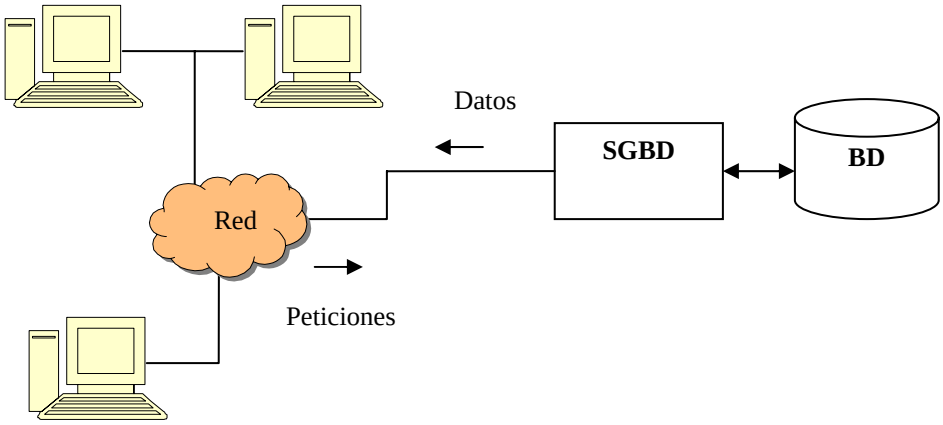

Figura 3.1.1

Esta arquitectura reduce el tráfico de la red y divide la carga de trabajo. Los procesos de usuario como el manejo de los datos y el despliegue de los mismos en pantalla son hechos en la PC del usuario. Mientras que los procesos de datos, como el acceso al disco duro o el procesamiento de las peticiones son realizados en el servidor.

3.1.3 Modelo de datos Relacional

EL SGBD organiza y estructura datos para que estos sean manipulados por los programas clientes. Las estructuras de datos y las técnicas de acceso a estos para un determinado SGBD es llamado “modelo de datos”. Digamos que este modelo de datos determina la “personalidad” del manejador. En este caso MySQL es un SGBD relacional, por lo que es conveniente describir brevemente este modelo.

El modelo relacional nació como un intento de simplificar las estructuras de las bases de datos. Y representa todos los datos contenidos en ellas como tablas de datos simples (organizadas en renglones y columnas). La definición aceptada para una base de datos relacional (BDR) es la siguiente:

> “Una base de datos relacional es tal que, todos los datos visibles para el usuario están organizados estrictamente como tablas de valores de datos, y todas las operaciones permitidas trabajan sobre esas tablas.” [G5]

Las tablas son arreglos rectangulares de datos y puede decirse que forman el principio organizacional en una BDR. Cada tabla en la BDR tiene un nombre único que identifica su contenido.[3] En la figura 3.1.2 se ve claramente la estructura de una tabla. Cada renglón representa una entidad física (en este caso un tipo de dulce). Los dos renglones juntos representan todos los dulces disponibles.

Estructura de Tabla

Tabla de Dulces

Ítem	Color	Nombre	Precio
1	blanco	Chup	10.50
2	rojo	Lol	16.50

Figura 3.1.2

Así podemos definir algunas características de las tablas:

- Cada columna vertical de “Dulces” representa una característica (o tipo de dato) guardada para cada tipo de dulce.
- Cada renglón de la tabla contiene sólo un valor en cada columna.
- Para cada columna de la tabla todos los ítems tienen un mismo tipo de dato. Este set de datos que una columna puede contener es llamado dominio de la columna.
- Cada columna en la tabla tiene un nombre de columna o campo, que es escrito usualmente en la parte superior de la misma. Todas las columnas deben tener nombres distintos en la misma tabla, sin embargo no es prohibición que dos campos tengan nombres iguales en diferentes tablas.
- A diferencia de las columnas, los renglones no tienen un orden en particular.
- Una tabla puede tener cualquier número de renglones. Una tabla con cero renglones es perfectamente legal y es llamada tabla vacia.
- Una tabla vacía conserva su estructura, impuesta por sus columnas, pero no contiene datos.

[3] Debe aclararse que esto depende del diseñador de la base.

Otra característica del modelo relacional son las llaves primarias o “primary keys”. Debido a que los renglones de una tabla están desordenados, no se puede seleccionar un renglón específico por su posición en la tabla. No hay un primer o ultimo renglón. Por tal motivo en una DBR bien diseñada cada tabla tiene una columna o combinación de columnas cuyos valores son únicos para cada renglón de la tabla. Esta columna (s) es llamada la llave primaria de la misma.

La llave primaria tiene un valor único para cada renglón en una tabla. Así una tabla cuyos renglones son diferentes de los otros es una relación[4]. Aquí cabe mencionar algunos conceptos de la teoría de conjuntos matemáticos:

- Dominio. Conjunto de valores.
- Relación. Subconjunto del producto cartesiano de una lista de dominios.

Y con lo mencionado en esta sección podemos incluir un último concepto:

- Atributo. Columna de una relación identificada por un nombre.

Con esto podemos decir que la “relación” entre los datos guardados en diferentes tablas o la “relación” entre las diferentes tablas está dada por los valores comunes guardados en las mismas. En la figura 3.1.3 se pueden apreciar mejor estas relaciones.

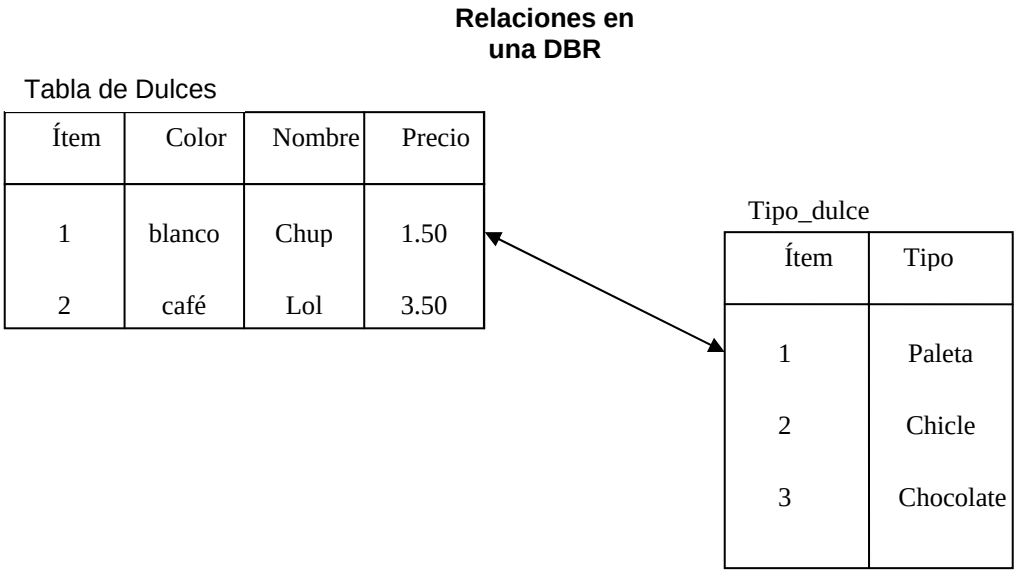

Figura 3.1.3

[4] Relación en términos matemáticos.

3.1.4 El Lenguaje SQL

Los orígenes del modelo relacional se remontan a Junio de 1970 cuando el Dr. Ted Codd, un investigador de IBM, publicó un articulo llamado “A Relational Model of Data for Large Shared Data Bank” en “Communications of the Association for Computing Machinery”. [G5]

Ya en 1974 había un prototipo de un SGBD relacional llamado “System/R” que incluía un lenguaje de búsqueda en bases de datos llamado SEQUEL, un acrónimo de “Estructurated English Query Language”. En 1978 “System/R” fue distribuido y el SEQUEL fue renombrado a SQL o “Structured Query Language”.

En 1982 se empezó a trabajar en el estándar ANSI para el SQL y en 1987 fue adoptado por el gobierno de E. U. como “Federal Information Processing Standard”[5] (FIPS). Aunque se pensaba que este estándar lograría la portabilidad de las BDR esto no se logró. En parte debido a que muchos de los miembros del comité que realizaron el estándar tenían sus propios productos comerciales que implementaban ligeramente diferente dialecto SQL, así este estándar tiene muchos “hoyos” que permiten que muchos “dialectos” SQL cumplan con el mismo.

Las diferencias entre dialectos SQL son significativas por lo que una aplicación escrita para una base de datos con un cierto SGBD debe ser modificada al ser movida a otro. Sin embargo el modelo relacional nos da ciertas herramientas para definir la estructura de una base de datos de una forma generalizada. Esto se logra con el modelo conceptual.

3.1.5 Modelo Entidad Relación

El modelo relacional se presta a la representación de las entidades y las asociaciones:

- Entidad. Está representada por una relación (tabla) cuyo esquema es el nombre de la entidad seguido de la lista de los atributos de la entidad.

- Asociación. Está representada por una relación cuyo esquema es el nombre de la asociación seguido de la lista de los identificadores de las entidades participantes y de los atributos de la asociación.

Para el modelado de la base de datos se representarán mediante rectángulos para las entidades, hexágonos para las asociaciones, los atributos en rectángulos redondeados y círculos como conectores[6]. A continuación en la figura 3.1.4 se muestra el modelo Entidad - Relación de la RELAP_DB.

[5] Estándar Federal para Procesos de Información.
[6] En caso de ser necesario por problemas de espacio.

Modelo Entidad – Relación
RELAP_DB

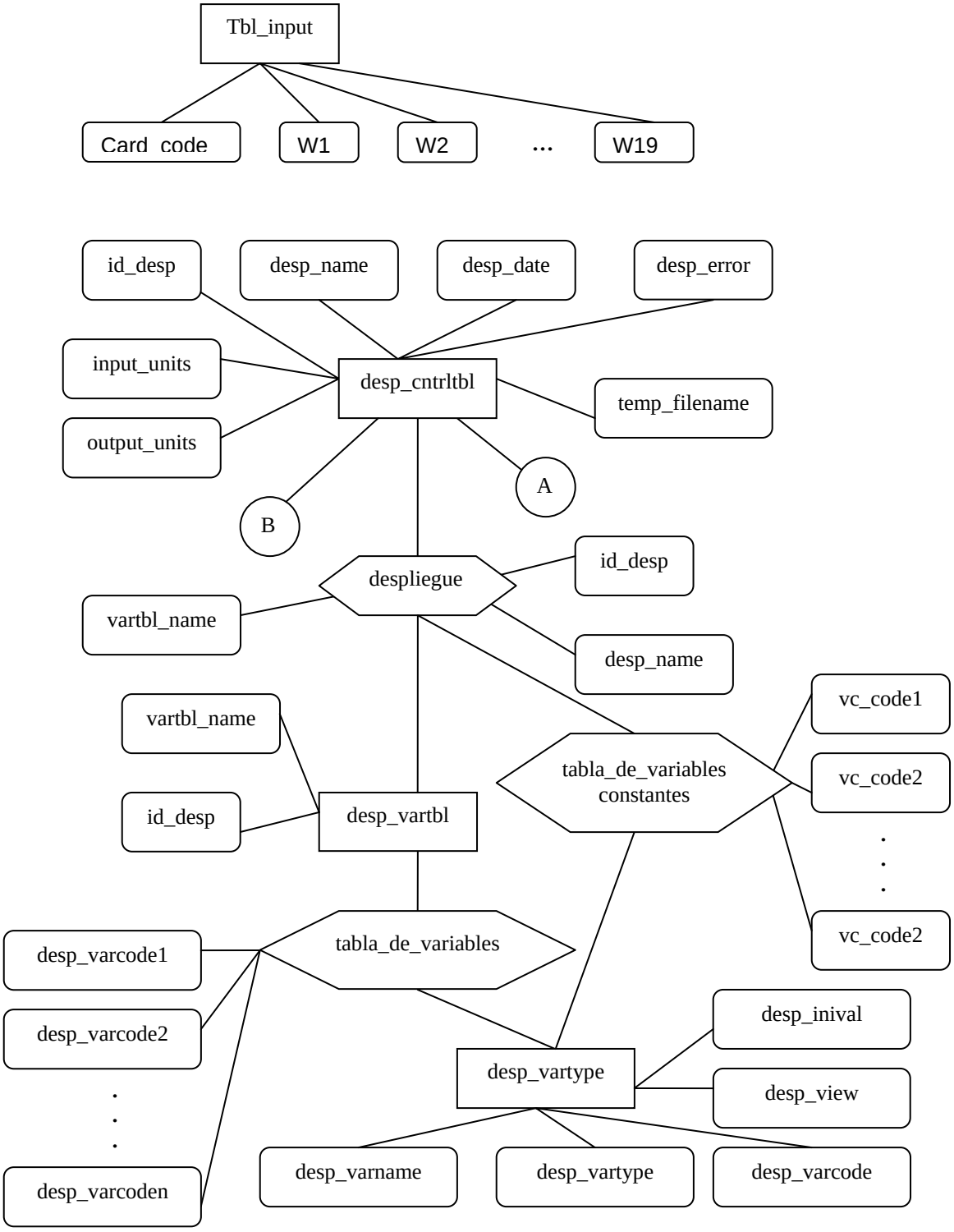

Figura 3.1.4

Modelo Entidad – Relación
RELAP_DB (continuación)

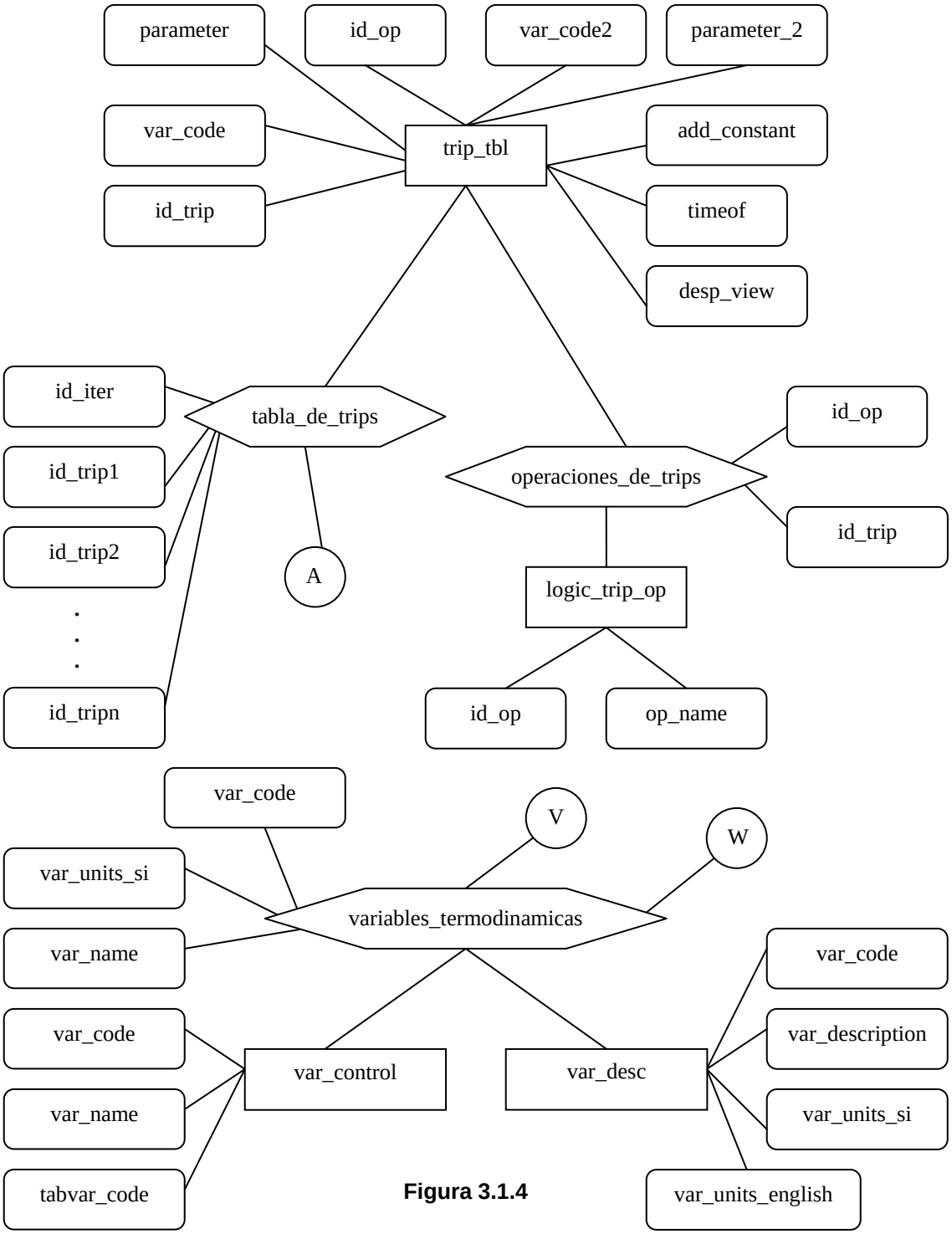

Figura 3.1.4

Modelo Entidad – Relación
RELAP_DB (continuación)

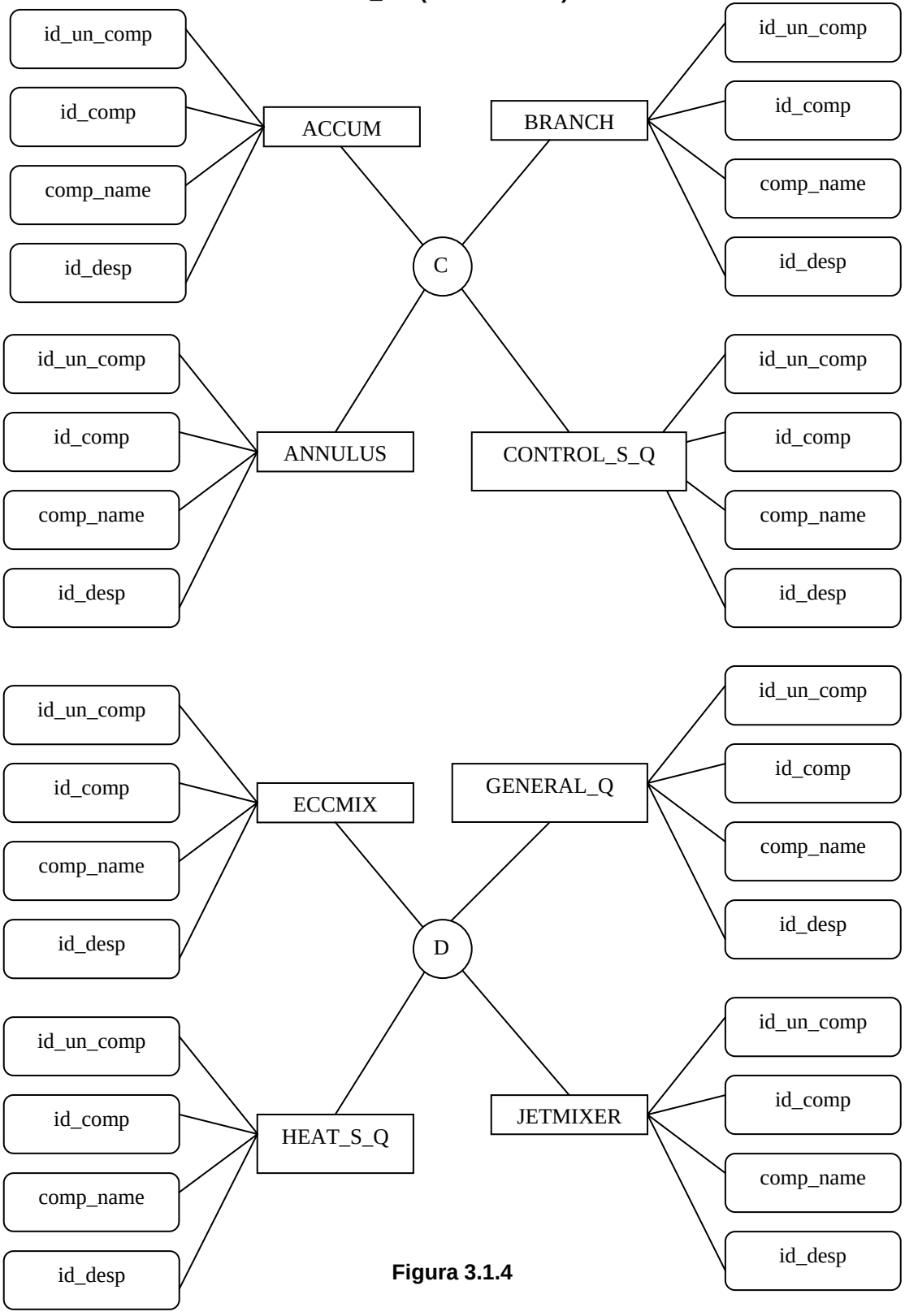

Figura 3.1.4

Modelo Entidad – Relación
RELAP_DB (continuación)

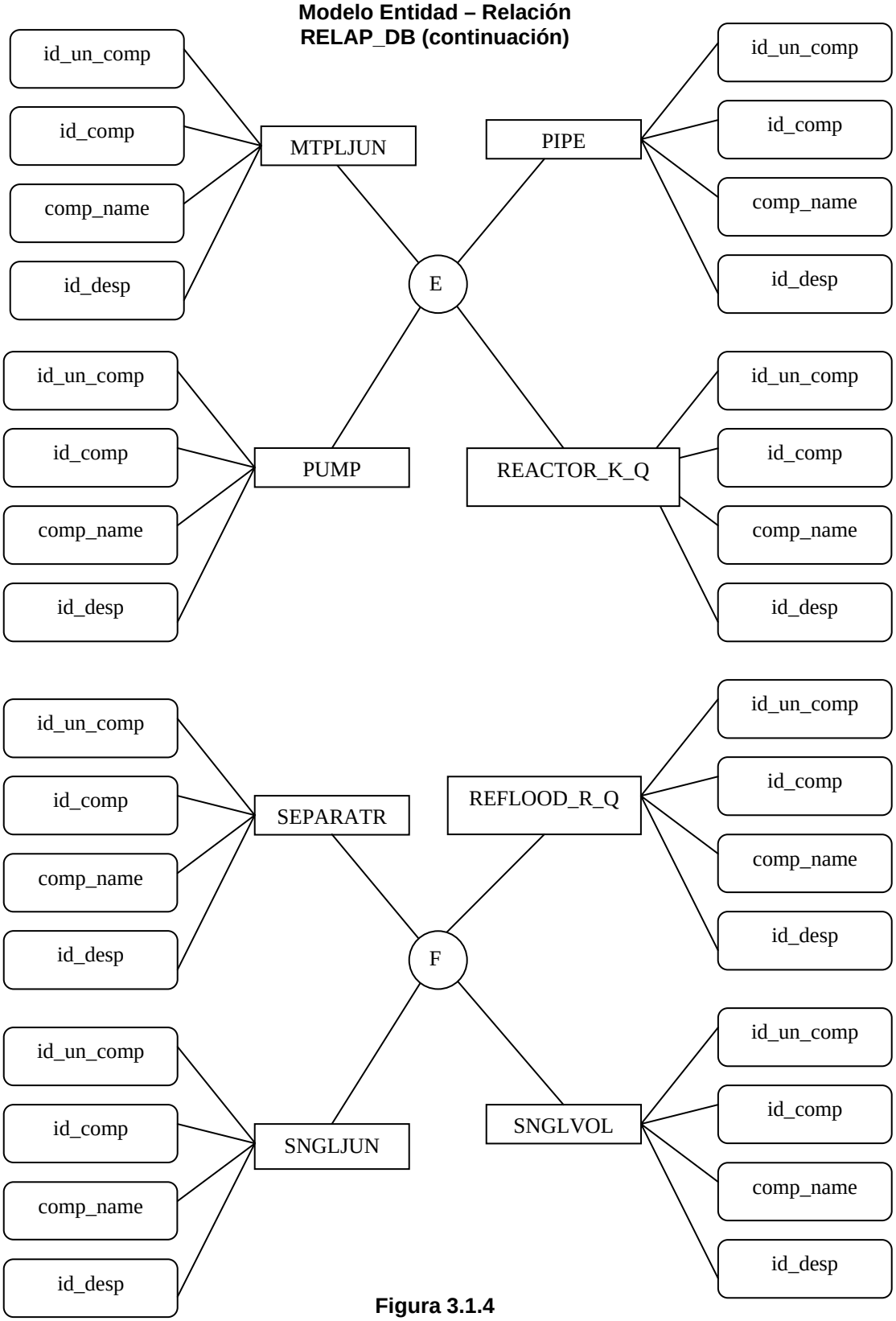

Figura 3.1.4

Modelo Entidad – Relación
RELAP_DB (continuación)

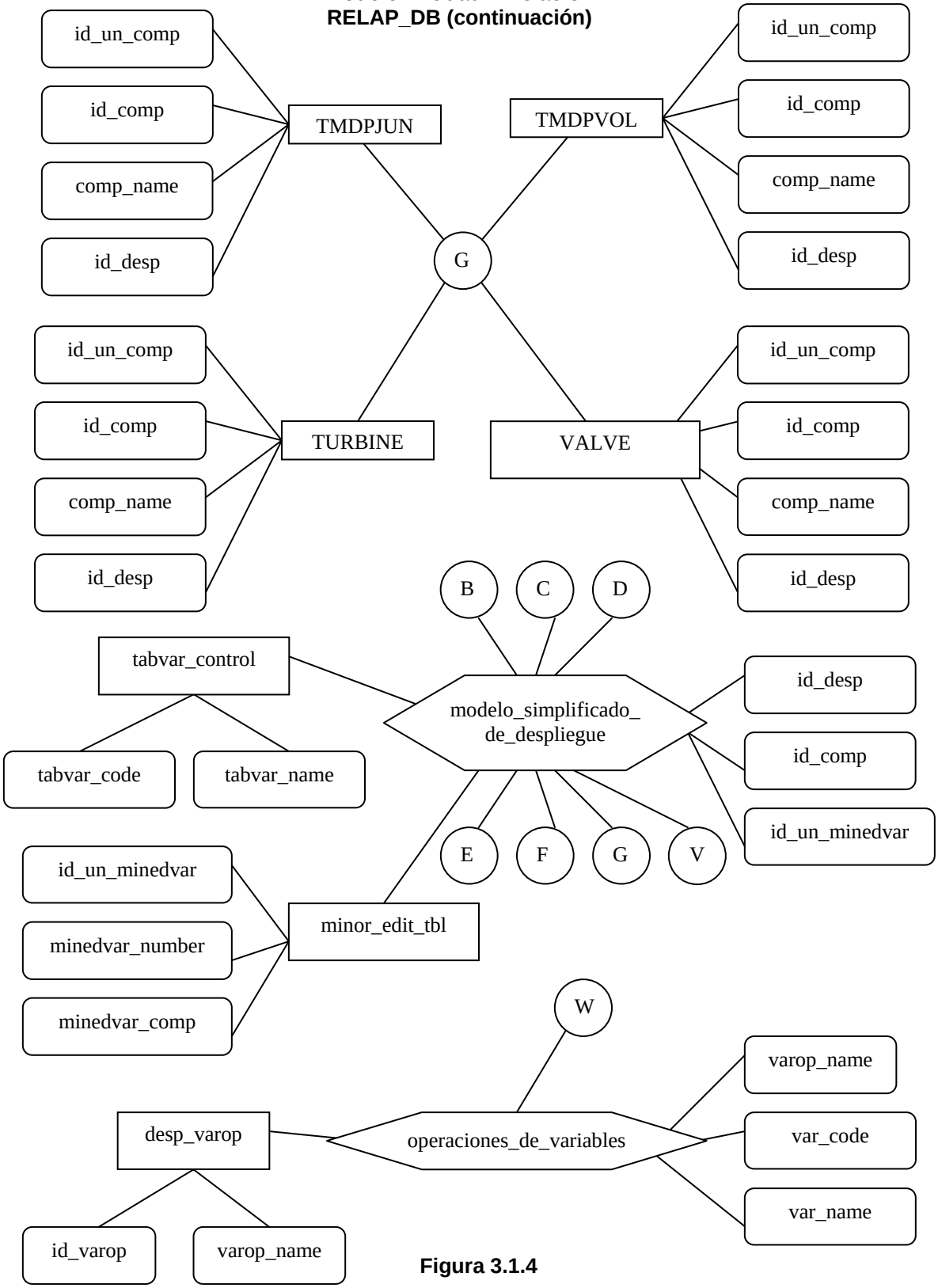

Figura 3.1.4

3.1.6 Comandos SQL [M2]

En esta sección presentaremos la sintaxis de algunas búsquedas usadas en las aplicaciones creadas para el SADROV. En los comandos las partes encerradas entre corchetes "[…]" indican que es opcional, los elemento separados por un "|" indican que es uno u otro. Los elementos entre "{…}" indican que es opcional.

Describir. Este comando llamado "DESCRIBE" regresa las características de una tabla dada.

```
{DESCRIBE | DESC} tbl_name {col_name | wild}
```

Insertar. Este comando llamado "INSERT" es usado para insertar datos en una tabla (normalmente el renglón completo).

```
INSERT [LOW_PRIORITY | DELAYED] [IGNORE]
        [INTO] tbl_name [(col_name,...)]
        VALUES ((expression | DEFAULT),...),(...),...
or  INSERT [LOW_PRIORITY | DELAYED] [IGNORE]
        [INTO] tbl_name [(col_name,...)]
        SELECT ...
or  INSERT [LOW_PRIORITY | DELAYED] [IGNORE]
        [INTO] tbl_name
        SET col_name=(expression | DEFAULT), ...
```

Asegurar. Este comando llamado "LOCK TABLES" es usado para bloquear una tabla o tablas para evitar que sea modificada por otra aplicación mientras nosotros la estamos modificando.

```
LOCK TABLES tbl_name [AS alias] {READ | [READ LOCAL] | [LOW_PRIORITY]
WRITE}
            [, tbl_name {READ | [LOW_PRIORITY] WRITE} ...]
...
```

Desasegurar. Este comando llamado "UNLOCK TABLES" sirve para quitarle el seguro a una tabla o tablas que hayamos asegurado previamente.

```
UNLOCK TABLES
```

Seleccionar. Este comando llamado "SELECT" sirve para buscar información en una o varias tablas.

```
SELECT [STRAIGHT_JOIN]
       [SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]
       [SQL_CACHE | SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS] [HIGH_PRIORITY]
       [DISTINCT | DISTINCTROW | ALL]
    select_expression,...
    [INTO {OUTFILE | DUMPFILE} 'file_name' export_options]
    [FROM table_references
      [WHERE where_definition]
      [GROUP BY {unsigned_integer | col_name | formula} [ASC | DESC], ...
      [HAVING where_definition]
```

```
      [ORDER BY {unsigned_integer | col_name | formula} [ASC | DESC]
,...]
      [LIMIT [offset,] rows]
      [PROCEDURE procedure_name]
      [FOR UPDATE | LOCK IN SHARE MODE]]
```

Actualizar. Este comando llamado “UPDATE” actualiza el valor de una cierta columna o de varias columnas.

```
UPDATE [LOW_PRIORITY] [IGNORE] tbl_name
    SET col_name1=expr1 [, col_name2=expr2, ...]
    [WHERE where_definition]
    [LIMIT #]
```

Borrar. Este comando llamado “DELETE” borra uno o varios valores de una tabla.

```
DELETE [LOW_PRIORITY] [QUICK] FROM table_name
       [WHERE where_definition]
       [ORDER BY ...]
       [LIMIT rows]

or

DELETE [LOW_PRIORITY] [QUICK] table_name[.*] [,table_name[.*] ...]
       FROM table-references
       [WHERE where_definition]
```

Crear Tabla. Este comando llamado “CREATE TABLE” crea una tabla.

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name
[(create_definition,...)]
[table_options] [select_statement]

create_definition:
  col_name type [NOT NULL | NULL] [DEFAULT default_value]
[AUTO_INCREMENT]
            [PRIMARY KEY] [reference_definition]
  or    PRIMARY KEY (index_col_name,...)
  or    KEY [index_name] (index_col_name,...)
  or    INDEX [index_name] (index_col_name,...)
  or    UNIQUE [INDEX] [index_name] (index_col_name,...)
  or    FULLTEXT [INDEX] [index_name] (index_col_name,...)
  or    [CONSTRAINT symbol] FOREIGN KEY [index_name] (index_col_name,...)
            [reference_definition]
  or    CHECK (expr)

type:
        TINYINT[(length)] [UNSIGNED] [ZEROFILL]
  or    SMALLINT[(length)] [UNSIGNED] [ZEROFILL]
  or    MEDIUMINT[(length)] [UNSIGNED] [ZEROFILL]
  or    INT[(length)] [UNSIGNED] [ZEROFILL]
  or    INTEGER[(length)] [UNSIGNED] [ZEROFILL]
  or    BIGINT[(length)] [UNSIGNED] [ZEROFILL]
  or    REAL[(length,decimals)] [UNSIGNED] [ZEROFILL]
```

```
  or    DOUBLE[(length,decimals)] [UNSIGNED] [ZEROFILL]
  or    FLOAT[(length,decimals)] [UNSIGNED] [ZEROFILL]
  or    DECIMAL(length,decimals) [UNSIGNED] [ZEROFILL]
  or    NUMERIC(length,decimals) [UNSIGNED] [ZEROFILL]
  or    CHAR(length) [BINARY]
  or    VARCHAR(length) [BINARY]
  or    DATE
  or    TIME
  or    TIMESTAMP
  or    DATETIME
  or    TINYBLOB
  or    BLOB
  or    MEDIUMBLOB
  or    LONGBLOB
  or    TINYTEXT
  or    TEXT
  or    MEDIUMTEXT
  or    LONGTEXT
  or    ENUM(value1,value2,value3,...)
  or    SET(value1,value2,value3,...)
```

Borrar Tabla. Este comando llamado “DROP TABLE”. Elimina una o varias tablas.

```
DROP TABLE [IF EXISTS] tbl_name [, tbl_name,...] [RESTRICT | CASCADE]
```

Función de comparación de cadenas. Esta función llamada “LIKE” hace una comparación que no es “case sensitive”.

```
expr LIKE pat [ESCAPE 'escape-char']
```

Con “LIKE” se pueden usar dos caracteres especiales para hacer la comparación:

Tabla 3.1.1 Caracteres Especiales del comando “LIKE”

Caracter	Descripción
“%”	Compara cualquier cantidad de caracteres.
“_”	Compara exactamente un carácter.

3.2 Estructura del ICRELAP

En la sección 2.5 se mencionaron las características funcionales de la ICRELAP, sin embargo no se mencionaron las características operativas de la misma. Así las características operativas de la ICRELAP son las siguientes:

- Debido a que los códigos RELAP5 están codificados en FORTRAN, el ICRELAP debe ser compatible con el lenguaje FORTRAN.

- El ICRELAP debe ser llamado en cada iteración de los códigos.

- El ICRELAP debe ser compatible con algún API de MySQL[7], es decir que pueda establecer comunicación con el SGBD.

3.2.1 Funciones del API C de MySQL [M1]

Según las características operativas del ICRELAP el API que cumple con la compatibilidad con los códigos RELAP5 es el API C, debido a que el lenguaje C es compatible con FORTRAN. En este apartado mencionaremos los tipos de datos y funciones requeridas para el funcionamiento de ICRELAP.

Tipos de datos:

- MYSQL. Esta es una estructura que maneja la conexión a la base de datos. Es usada para la mayoría de funciones del API.

- MYSQL_RES. Esta estructura representa el resultado de una búsqueda a la base de datos que regresa renglones de datos (SELECT, DESCRIBE, entre otras). La información regresada es llamada “set de resultados”.

- MYSQL_ROW. Este tipo representa un renglón de datos[8]. Está implementado como un arreglo de cadenas de caracteres. Si el renglón contiene información binaria[9] no puede tratarse este arreglo como cadenas de caracteres terminadas en NULL. Los renglones son obtenidos llamando a la función mysql_fetch_row ().

- MYSQL_FIELD. Esta estructura contiene la información de un atributo[10], como su nombre, tipo y tamaño. Cada atributo se obtiene llamando a la función mysql_fetch_field ().

[7] Ver sección 1.10.3.
[8] Estos renglones de datos son resultado de una búsqueda o query en la base de datos.
[9] Esto pasa cuando uno o varios campos de un resultado son del tipo binary o blob.
[10] Estos atributos o campos son el resultado de una búsqueda en la base de datos.

Funciones:

- mysql_affected_rows (). Regresa el número de renglones afectados o borrados o insertados por la última búsqueda del tipo UPDATE, DELETE o INSERT.
- mysql_close (). Termina la conexión al servidor.
- mysql_errno (). Regresa el número de error de la última función MYSQL llamada.
- mysql_error (). Regresa el mensaje de error de la última función MYSQL llamada.
- mysql_fetch_field (). Regresa el tipo del siguiente campo de datos.
- mysql_fetch_row (). Trae el siguiente renglón del “set de resultados”.
- mysql_field_count (). Regresa el número de campos del “set de resultados” de la última búsqueda.
- mysql_free_result (). Libera la memoria usada por un “set de resultados”.
- mysql_get_host_info (). Regresa una cadena de caracteres que describe la conexión actual.
- mysql_get_server_info (). Regresa la versión del servidor.
- mysql_init (). Inicializa una estructura del tipo MYSQL.
- mysql_insert_id (). Regresa el ID generado por una columna del tipo auto_increment en la búsqueda previa.
- mysql_num_rows (). Regresa el número de renglones de un “set de resultados”.
- mysql_options (). Configura las opciones para mysql_real_connect ().
- mysql_ping (). Checa si la conexión con el servidor esta funcionando, reconecta de ser necesario.
- mysql_query (). Ejecuta una búsqueda SQL.
- mysql_real_connect (). Realiza una conexión a un servidor de MySQL.
- mysql_store_result (). Regresa un “set de resultados” de una búsqueda en la base de datos.

3.2.2 Descripción Funcional del ICRELAP

En la figura 3.2.1 se puede observar de flujo de la ICRELAP.

Diagrama de Flujo
De la ICRELAP

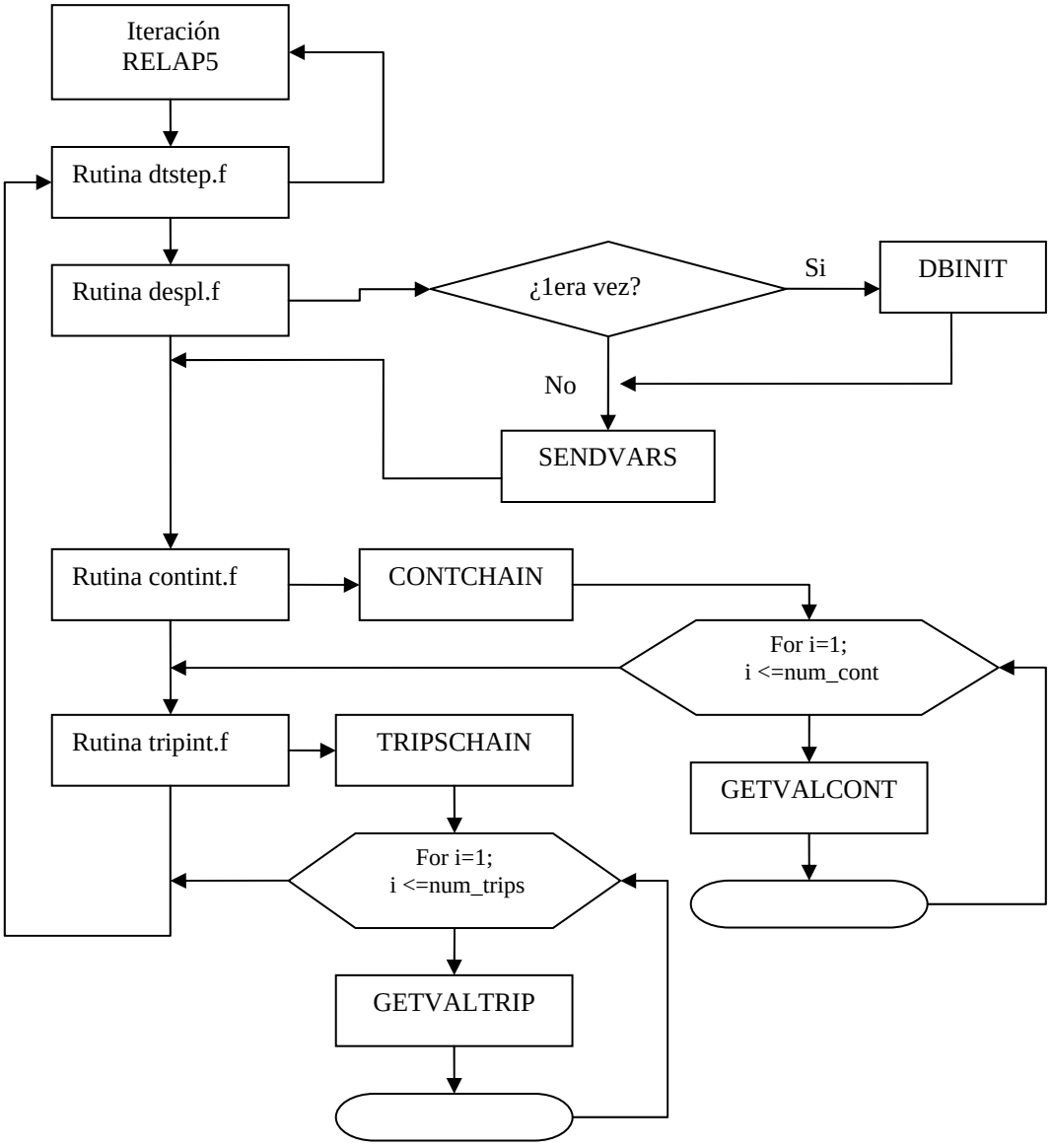

Figura 3.2.1

En la figura pueden apreciarse cuatro rutinas hechas en FORTRAN, contint.f, despl.f, dtstep.f y tripint.f. Estas rutinas forman parte de los códigos RELAP5 y son las que se modificaron[11] para lograr esta interfaz de comunicación.

[11] Sin alterar sus funciones originales.

Así mismo en la figura 3.2.1 se pueden observar las rutinas creadas en C[12] con el API de MySQL para lograr los objetivos del ICRELAP. Estas rutinas son: DBINIT, SENDVARS, CONTCHAIN, GETVALCONT, TRIPSCHAIN, GETVALTRIP.

3.2.3 Rutina contint.f

Se encarga de cambiar los valores de variables de control constantes declaradas en el modelo de entrada (usando los valores de las mismas de la base de datos) para lograr interactividad con los códigos RELAP5. La figura 3.2.2 muestra el diagrama de flujo de la rutina.

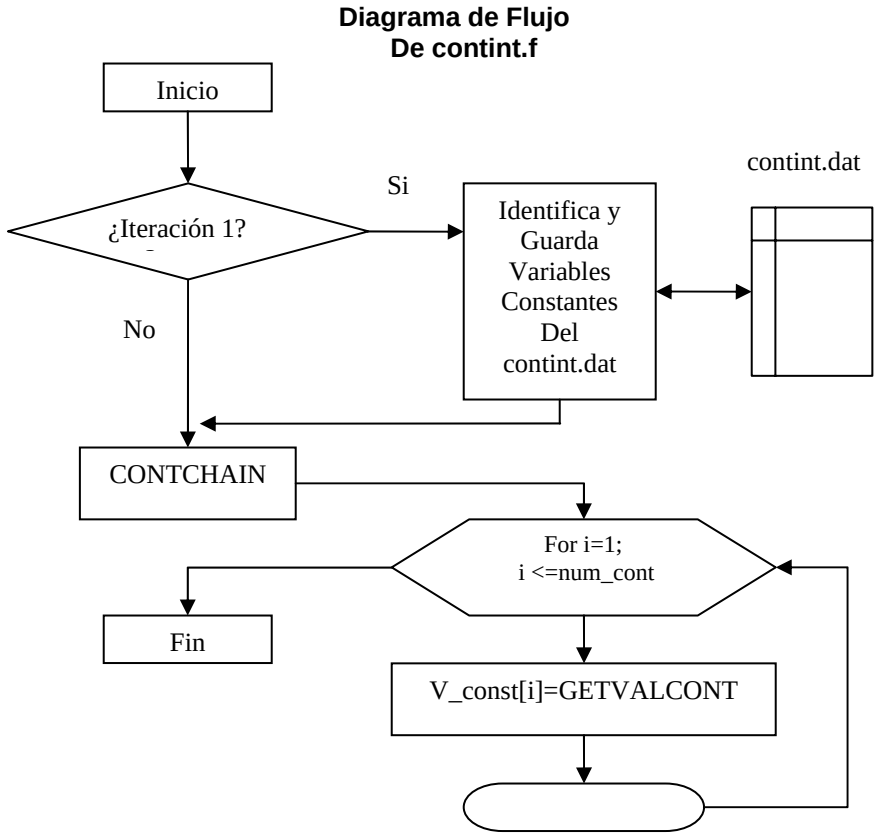

Figura 3.2.2

En la figura se puede observar el diagrama de flujo a *grosso modo* de contint. Para nuestros propósitos debe hacerse notar la llamada a CONTCHAIN Y GETVALCONT. La llamada a la primera función en C consulta en la base de datos los valores de las variables constantes, trayéndolas a los códigos RELAP5 y después la segunda función obtiene el valor de una variable específica de la base de datos para guardar este valor en su contraparte en los códigos RELAP5.

[12] Estas son llamadas desde las rutinas de FORTRAN.

3.2.4 Rutina dtstep.f

Esta rutina es llamada en cada paso de iteración de los códigos RELAP5 para imprimir la salida “outdta”. Por lo mismo se le escogió para llamar a las rutinas contint, despl y tripint. La figura 3.2.3 muestra el diagrama de flujo de dtstep.f que ilustra solamente los procesos de nuestro interés.

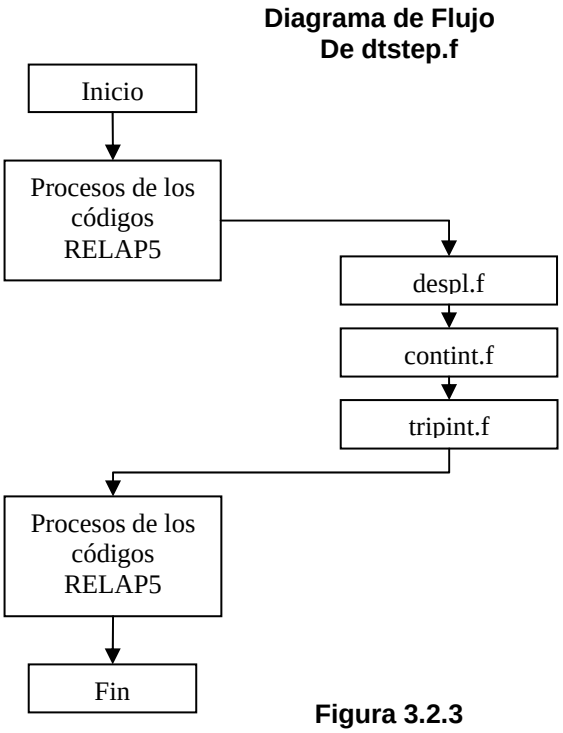

Figura 3.2.3

Como puede apreciarse la relevancia de esta rutina es la llamada a las demás rutinas en FORTRAN siguiendo el orden especificado en el mismo.

3.2.5 Rutina despl.f

Esta rutina se encarga recolectar los valores de las variables de control declaradas en el modelo de entrada y que se encuentran activas en la base de datos[13]. Una vez que se conocen los valores de estas variables son enviadas a la RELAP_DB. La figura 3.2.4 muestra el diagrama de flujo de la rutina despl.f.

Se puede apreciar que la rutina primero colecta las variables requeridas de sus valores en los códigos RELAP5 y las guarda en un arreglo de variables, cuando ha realizado lo anterior simplemente las envía a la base de datos llamando a la función en C llamada SENDVARS. La última se encarga de guardar las variables en la RELAP_DB.

[13] Quiere decir que estas variables activas son requeridas por un despliegue dado de alta previamente en la RELAP_DB.

Otro aspecto a destacar es la llamada a la función en C llamada DBINIT que tiene como función inicializar la conexión a la RELAP_DB y los archivos contint.dat, varcdes.dat y tripint.dat.

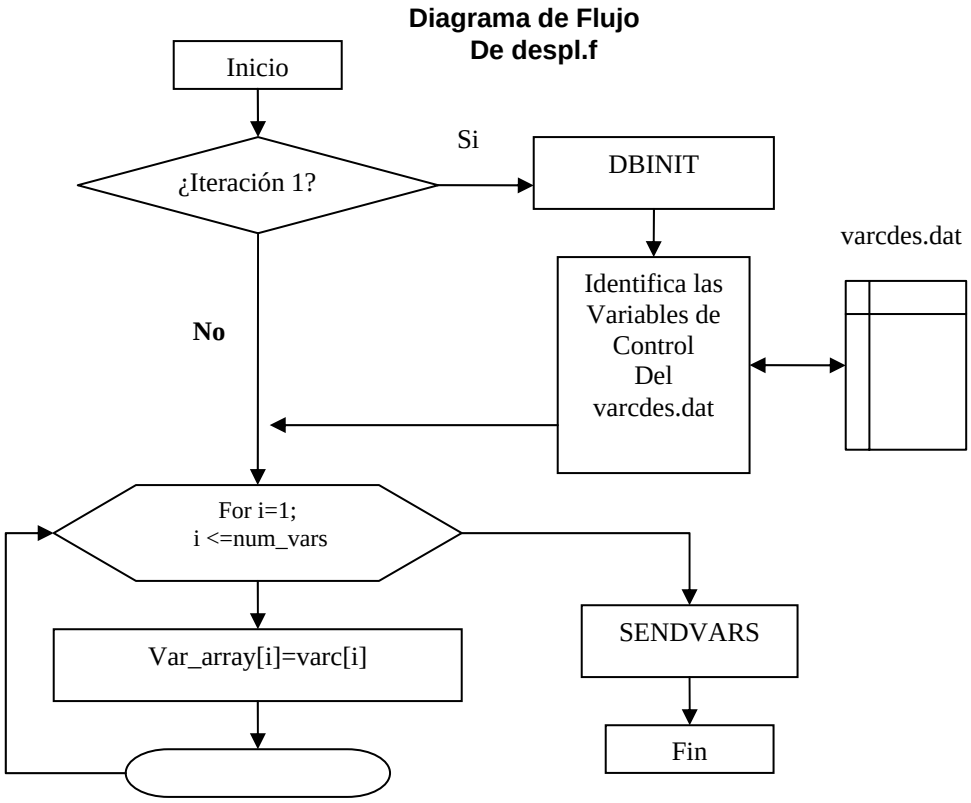

Figura 3.2.4

3.2.6 Rutina tripint.f

Se encarga de cambiar los valores de los trips activos del modelo para conseguir una interactividad con los códigos RELAP5. La figura 3.2.5 muestra el diagrama de flujo de tripint.f.

Como puede verse en su diagrama de flujo lo más significativo[14] de la rutina tripint.f son sus llamadas a las funciones en C llamadas TRIPSCHAIN y GETVALTRIP. La primera función obtiene los valores de los trips activos en la RELAP_DB mientras que la segunda función obtiene el valor de un trip específico. Con este valor se reemplaza el valor de su contraparte en los códigos RELAP5.

[14] Desde el punto de vista del ICRELAP.

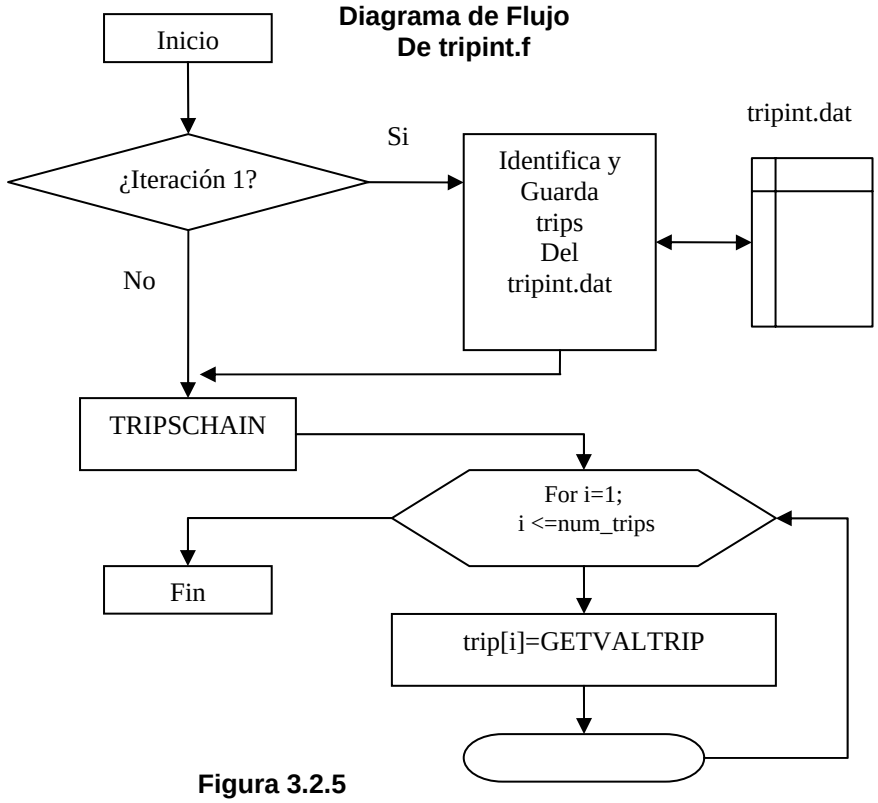

Figura 3.2.5

3.2.7 Rutina DBINIT

En la figura 3.2.1 se puede observar la llamada a esta función dentro de la ICRELAP, esta sólo se produce una vez. Así podemos decir que el objetivo principal de esta función es la inicialización de la conexión a la base de datos, los archivos necesarios para el funcionamiento del SADROV y regresar el nombre de la tabla donde se guardarán las variables de despliegue del modelo.

En la figura 3.2.6 se observa el diagrama de flujo de DBINIT. Las búsquedas[15] realizadas en la base de datos también pueden verse en la figura. Por lo que se pueden enumerar las funciones de la función DBINIT:

1. Inicializar conexión al SGBD MySQL.
2. Crear el archivo contint.dat.
3. Crear el archivo varcdes.dat.
4. Crear el archivo tripint.dat.
5. Regresar el nombre de la tabla de variables de despliegue.

[15] Ver sección 3.1 para ver la estructura de la base de datos.

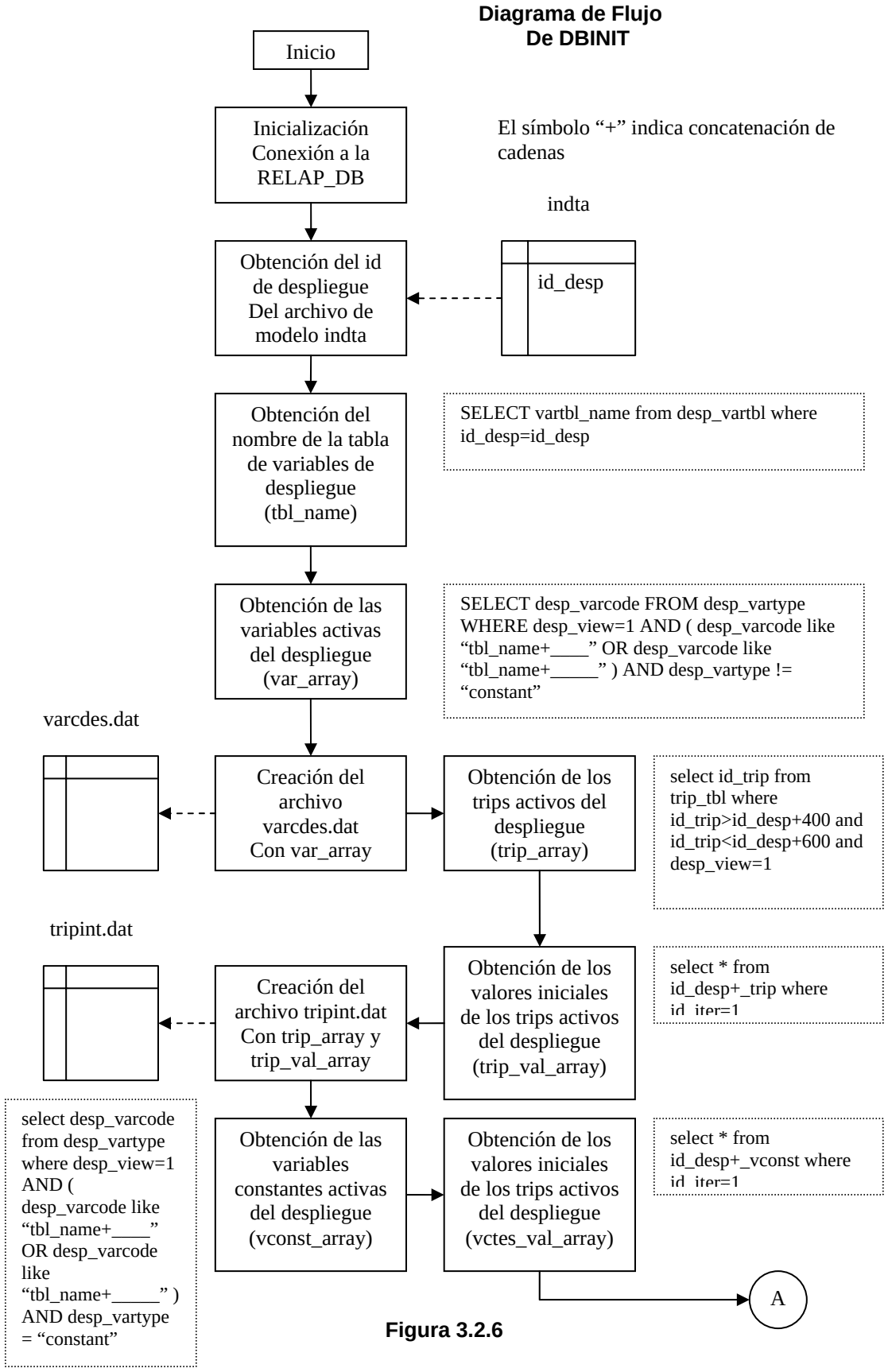
Diagrama de Flujo
De DBINIT
Inicio
Inicialización
Conexión a la
RELAP_DB
El símbolo "+" indica concatenación de cadenas
indta
Obtención del id
de despliegue
Del archivo de
modelo indta
id_desp
Obtención del
nombre de la tabla
de variables de
despliegue
(tbl_name)
SELECT vartbl_name from desp_vartbl where id_desp=id_desp
Obtención de las
variables activas
del despliegue
(var_array)
SELECT desp_varcode FROM desp_vartype WHERE desp_view=1 AND (desp_varcode like "tbl_name+____" OR desp_varcode like "tbl_name+_____") AND desp_vartype != "constant"
varcdes.dat
Creación del
archivo
varcdes.dat
Con var_array
Obtención de los
trips activos del
despliegue
(trip_array)
select id_trip from trip_tbl where id_trip>id_desp+400 and id_trip<id_desp+600 and desp_view=1
tripint.dat
Creación del
archivo tripint.dat
Con trip_array y
trip_val_array
Obtención de los
valores iniciales
de los trips activos
del despliegue
(trip_val_array)
select * from id_desp+_trip where id_iter=1
select desp_varcode from desp_vartype where desp_view=1 AND (desp_varcode like "tbl_name+____" OR desp_varcode like "tbl_name+_____") AND desp_vartype = "constant"
Obtención de las
variables
constantes activas
del despliegue
(vconst_array)
Obtención de los
valores iniciales
de los trips activos
del despliegue
(vctes_val_array)
select * from id_desp+_vconst where id_iter=1
A

Figura 3.2.6

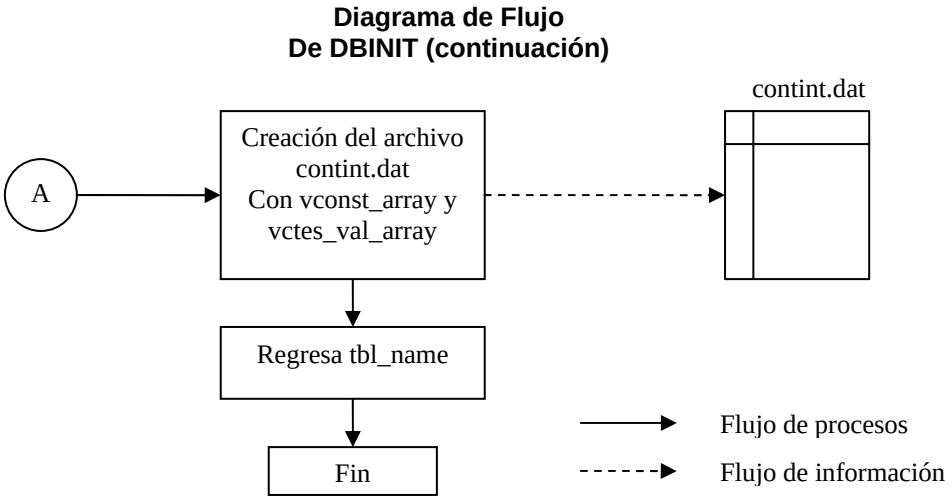

Figura 3.2.6

3.2.8 Rutina SENDVARS

Tiene por objetivo enviar los valores de las variables de despliegue activas a la RELAP_DB. La llamada a esta función escrita en lenguaje C se hace desde la rutina de FORTRAN despl.f en cada iteración de los códigos. La figura 3.2.7 muestra el diagrama de flujo de esta función.

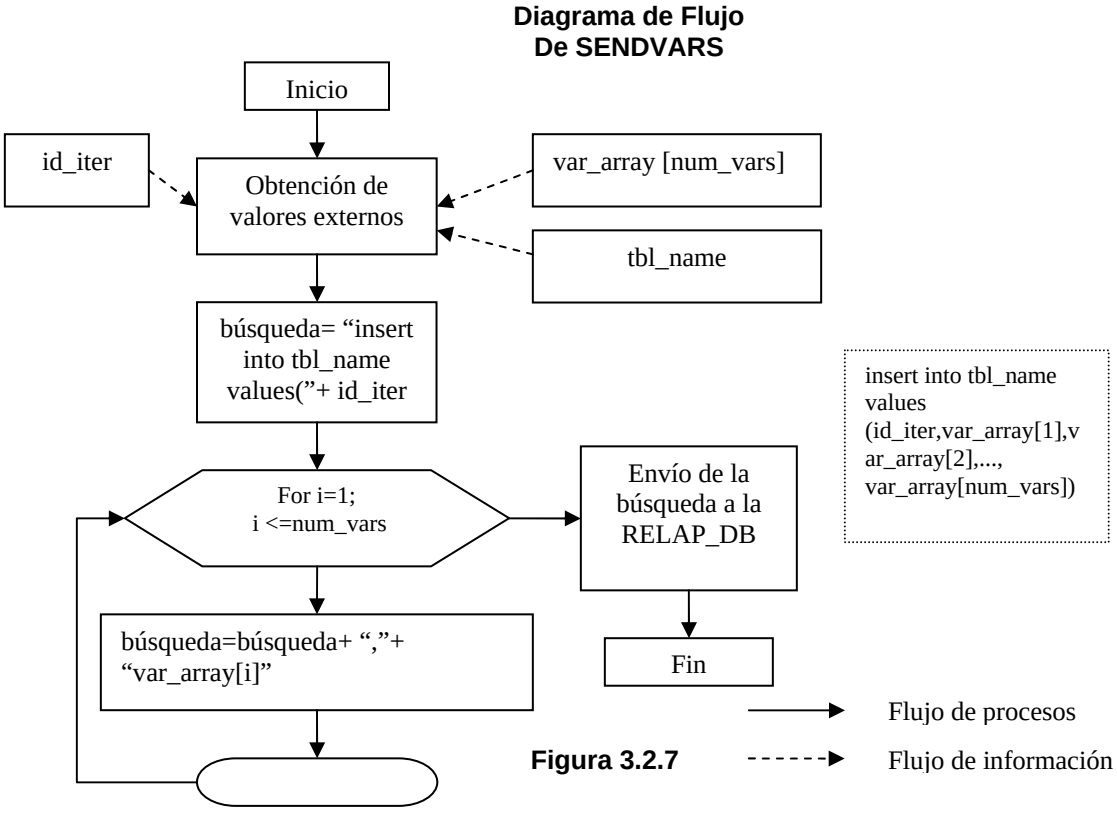

Figura 3.2.7

En el diagrama se aprecia la búsqueda realizada por la función, el símbolo “+” en este caso indica concatenación de cadenas de caracteres.

Así las funciones de SENDVARS son las siguientes:

1. Tomar los valores de las variables de despliegue activas en forma de arreglo.

2. Crear la búsqueda de inserción de datos en la RELAP_DB.

3. Realizar la búsqueda.

3.2.9 Rutina CONTCHAIN

Tiene por objetivo regresar los valores de las variables constantes activas del modelo que se encuentran en la RELAP_DB. Esta función escrita en lenguaje C es llamada por la rutina contint.f en cada iteración de los códigos como se aprecia en la figura 3.2.2. En la figura 3.2.8 se aprecia el diagrama de flujo de CONTCHAIN.

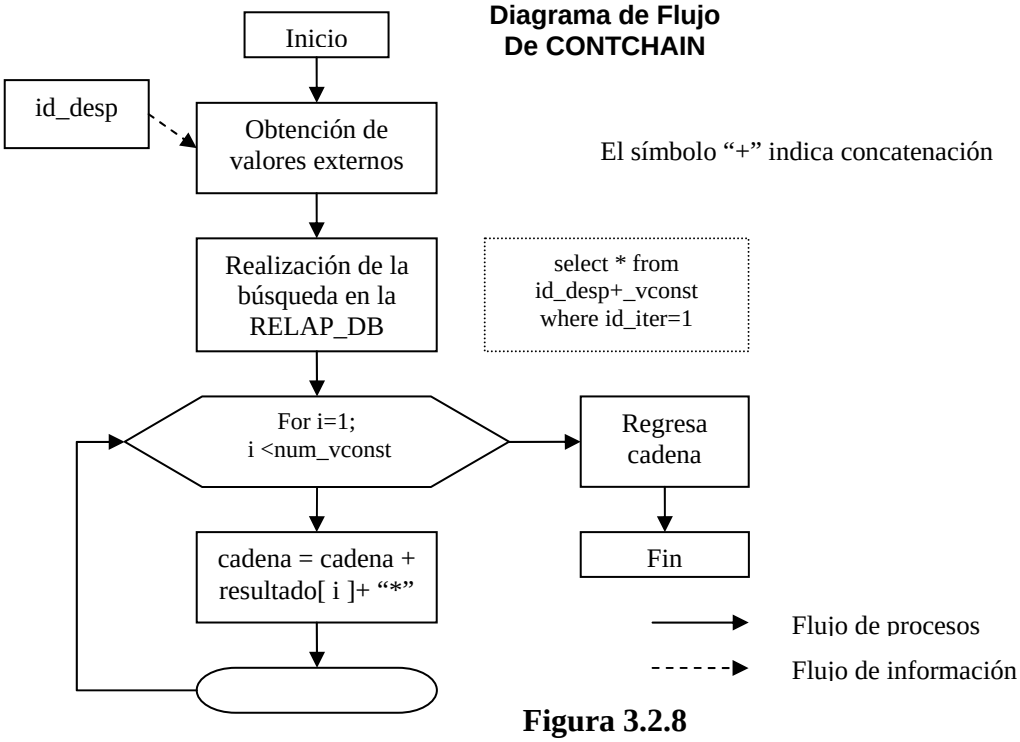

Figura 3.2.8

Se puede observar que CONTCHAIN regresa una cadena de caracteres, y los valores de las variables constantes activas están separados por un “*”. Así las funciones de CONTCHAIN pueden ser enumeradas de la siguiente forma:

1. Toma el identificador de despliegue.

2. Realiza la selección de los valores de las variables constantes activas en la RELAP_DB.

3. Le da formato a los valores obtenidos en forma de cadena de caracteres[16] con el símbolo “*” como separador.

3.2.10 Rutina GETVALCONT

Tiene por objetivo tomar la cadena regresada por CONTCHAIN y regresar un valor específico[17], el valor regresado depende de un identificador que recibe como dato externo. Esta función es llamada por la rutina de FORTRAN contint.f como se puede apreciar en la figura 3.2.2. La figura 3.2.9 muestra su diagrama de flujo.

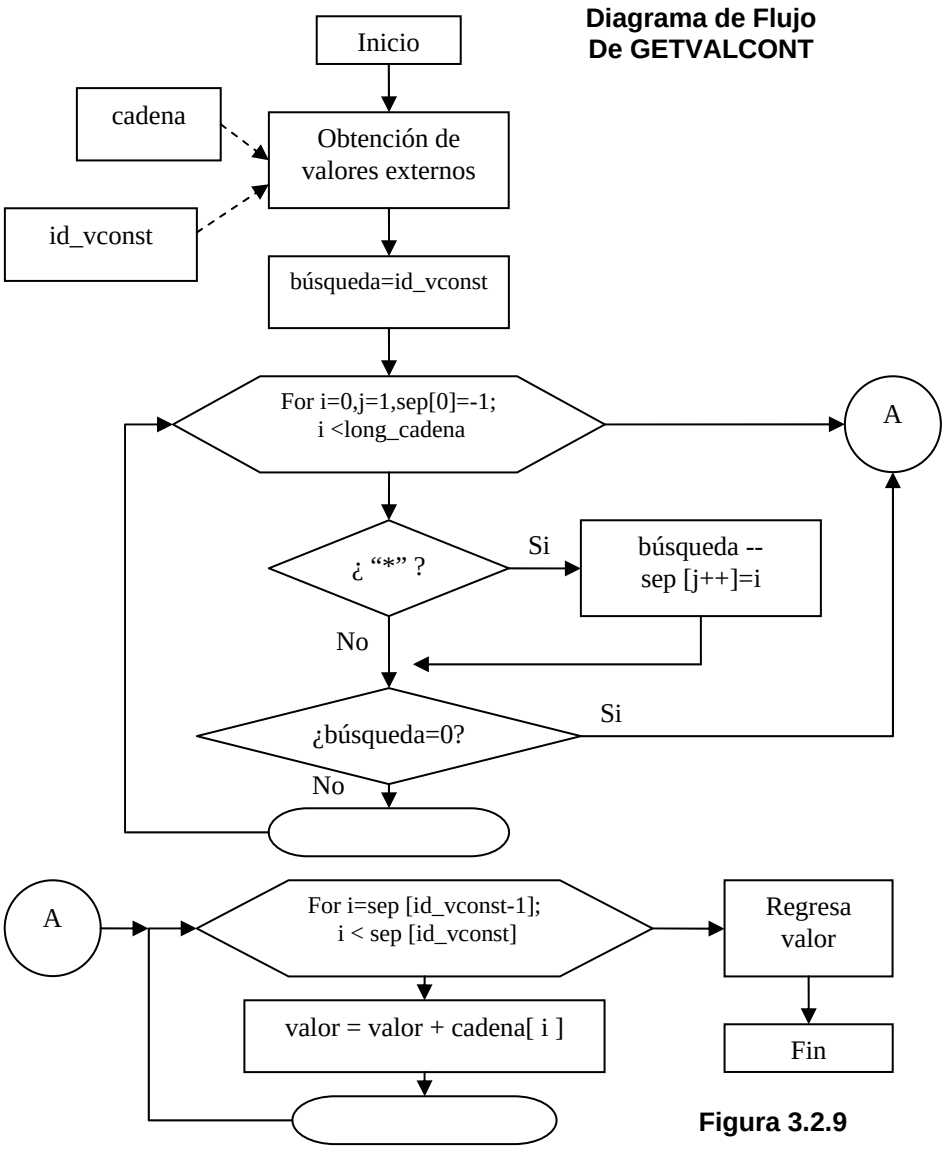

Figura 3.2.9

[16] También llamadas simplemente cadenas.
[17] Recordar que esta cadena esta formada por valores numéricos separados por el símbolo “*”.

Se puede observar del diagrama que GETVALCONT regresa un valor formado por caracteres[18] que es convertido a un tipo float[19].

3.2.11 Rutina TRIPSCHAIN

Tiene por objetivo regresar los valores de los trips activos del modelo que se encuentran en la RELAP_DB. Esta función escrita en lenguaje C es llamada por la rutina tripint.f en cada iteración de los códigos como se aprecia en la figura 3.2.5. En la figura 3.2.10 se aprecia el diagrama de flujo de TRIPSCHAIN.

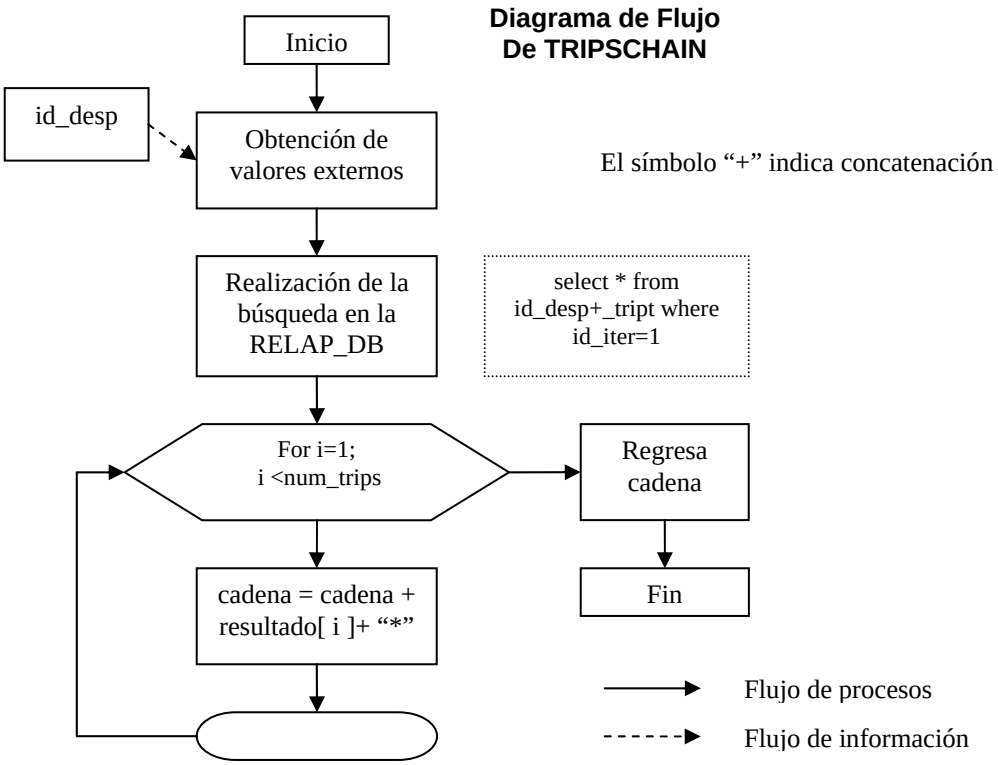

Figura 3.2.10

Se puede observar que TRIPSCHAIN regresa una cadena de caracteres, y los valores de las variables constantes activas están separados por un "*". Así las funciones de TRIPSCHAIN pueden ser enumeradas de la siguiente forma:

1. Toma el identificador de despliegue.

2. Realiza la selección de los valores de los trips activos en la RELAP_DB.

[18] El número de caracteres es variable.
[19] Tipo de dato del lenguaje C.

3. Le da formato a los valores obtenidos en forma de cadena de caracteres con el símbolo “*” como separador.

3.2.12 Rutina GETVALTRIP

Tiene por objetivo tomar la cadena regresada por TRIPSCHAIN y regresar un valor específico[20], el valor regresado depende de un identificador que recibe como dato externo. Esta función escrita en lenguaje C es llamada por la rutina de FORTRAN tripint.f como se puede apreciar en la figura 3.2.5. La figura 3.2.11 muestra el diagrama de flujo de esta función.

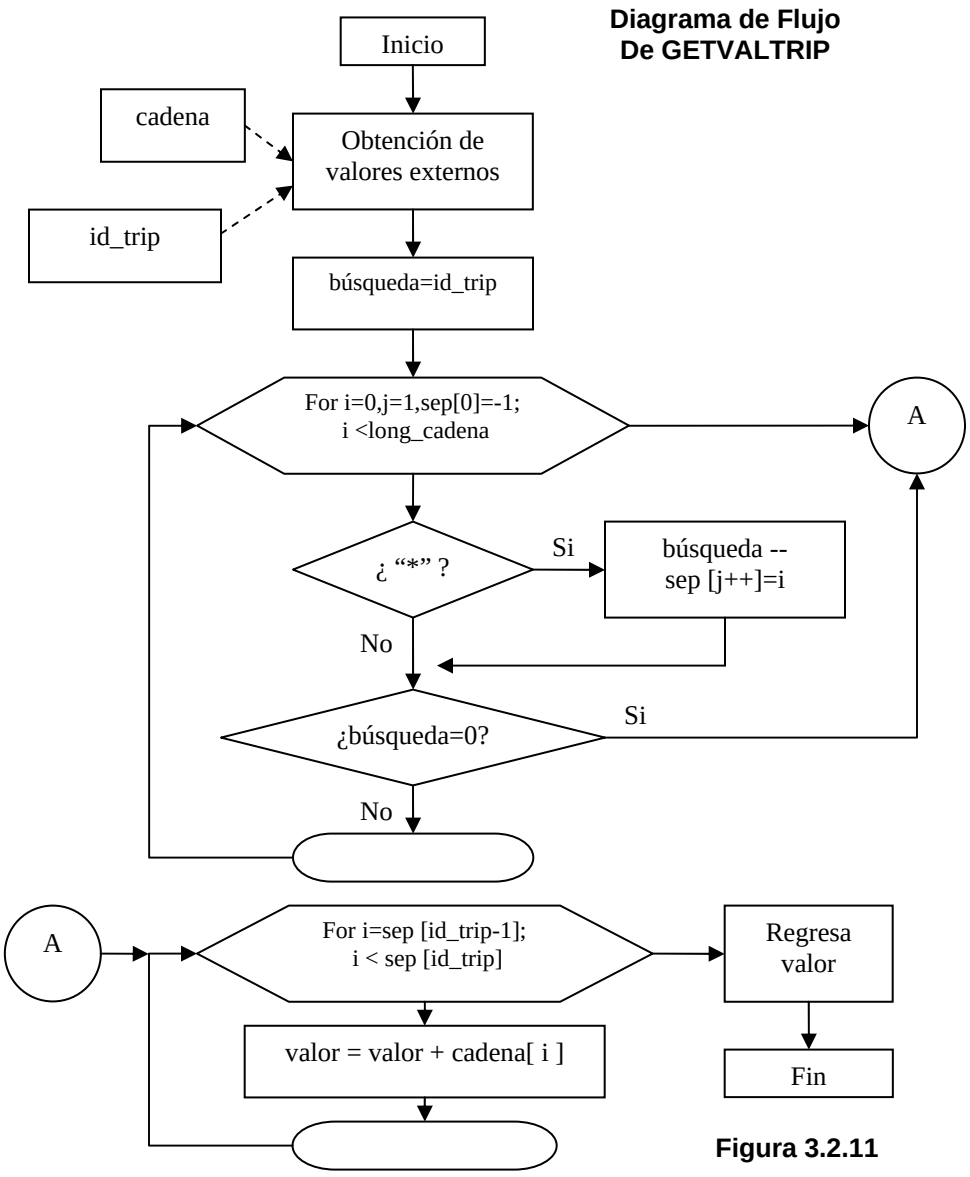

Figura 3.2.11

[20] Recordar que esta cadena esta formada por valores numéricos separados por el símbolo “*”.

Se puede observar del diagrama que GETVALTRIP regresa un valor formado por caracteres[21] que es convertido a un tipo float.

3.3 Estructura de la Interfaz Gráfica

En esta sección se abordará la estructura de la interfaz gráfica para conocer más a fondo sus funciones y características.

3.3.1 Diagrama de Flujo de la Interfaz Gráfica

En la figura 3.3.1 se puede apreciar el diagrama de flujo de la interfaz gráfica.

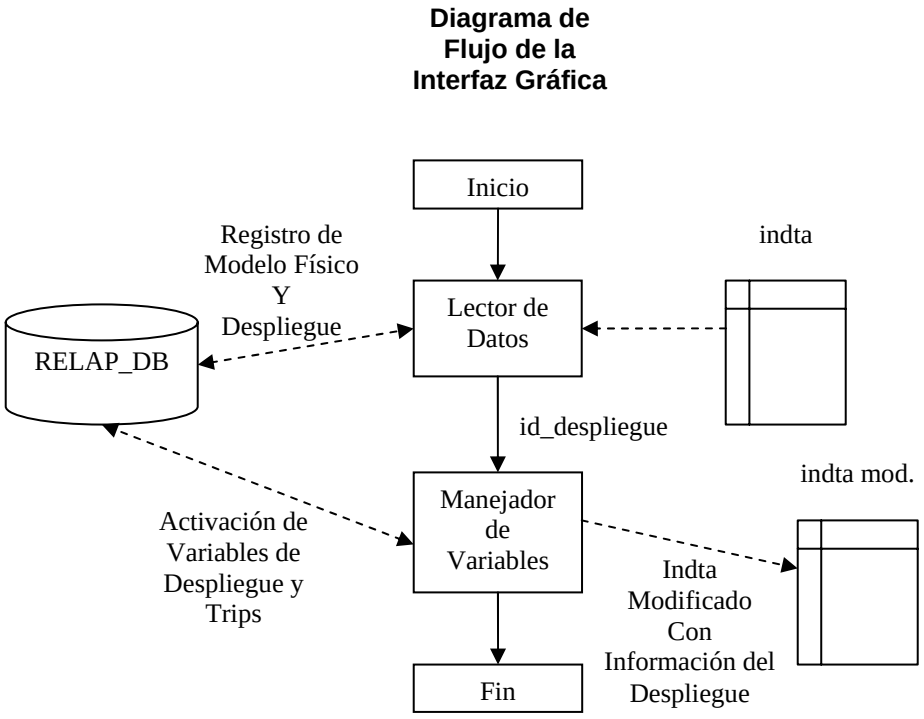

Figura 3.3.1

Como puede apreciarse en la figura son básicamente dos aplicaciones las que componen a la interfaz gráfica, el Lector de Datos (LDRELAP) y el Manejador de Variables (MVRELAP). Las funciones de ambas aplicaciones se discutirán a continuación.

[21] El número de caracteres es variable.

3.3.2 Estructura del LDRELAP

El Lector de datos de la Interfaz gráfica tiene las siguientes funciones principales:

- Dar de alta un "despliegue gráfico"[22] por modelo de entrada dado de alta en la RELAP_DB.
- Registrar los componentes termodinámicos del modelo.
- Registrar las unidades del modelo físico (entrada y salida).
- Registrar los "minor edits" del modelo físico.
- Registrar las variables de control del modelo físico.
- Registrar los trips del modelo físico.

Como se puede apreciar en el diagrama de flujo del lector de datos que aparece en la figura 3.3.2, de inicio debe dársele el archivo de modelo físico a usar en los códigos RELAP5 así como un nombre al despliegue[23]. Posteriormente el lector realizará las funciones antes mencionadas.

Debido a que esta aplicación es una aplicación Web y se sabe que los navegadores algunas veces fallan o los usuarios no hacen uso correcto de ellos. El LDRELAP mantiene un parámetro de error, éste es verdadero hasta que cada consulta a la base de datos resulta exitosa[24]. Si alguna consulta fallara el LDRELAP está diseñado para detener el proceso fallido en ese momento y por lo tanto dejar el parámetro de error verdadero. Entonces al final de sus procesos checa si este error es verdadero, si es así manda limpiar el despliegue erróneo y regresa al usuario al inicio.

La razón principal por la que el modelo se depura y posteriormente se inserta en la base de datos es el hacer uso de las funciones de comparación y búsqueda de datos del MySQL para reconocer las distintas tarjetas usadas en el archivo de modelo y registrarlas efectivamente en la RELAP_DB.

Como se aprecia en la figura 3.3.2 se han incluido las búsquedas usadas en cada proceso del LDRELAP con la finalidad de proporcionar un mejor entendimiento de los mismos y su relación con la estructura misma de la RELAP_DB.

[22] Se define como despliegue gráfico al modelo físico registrado en la RELAP_DB con variables de control y trips para el intercambio de información con un cliente de códigos RELAP5.
[23] Dependerá del usuario si este nombre es significativo al modelo a simular.
[24] Es decir no regresó algún error.

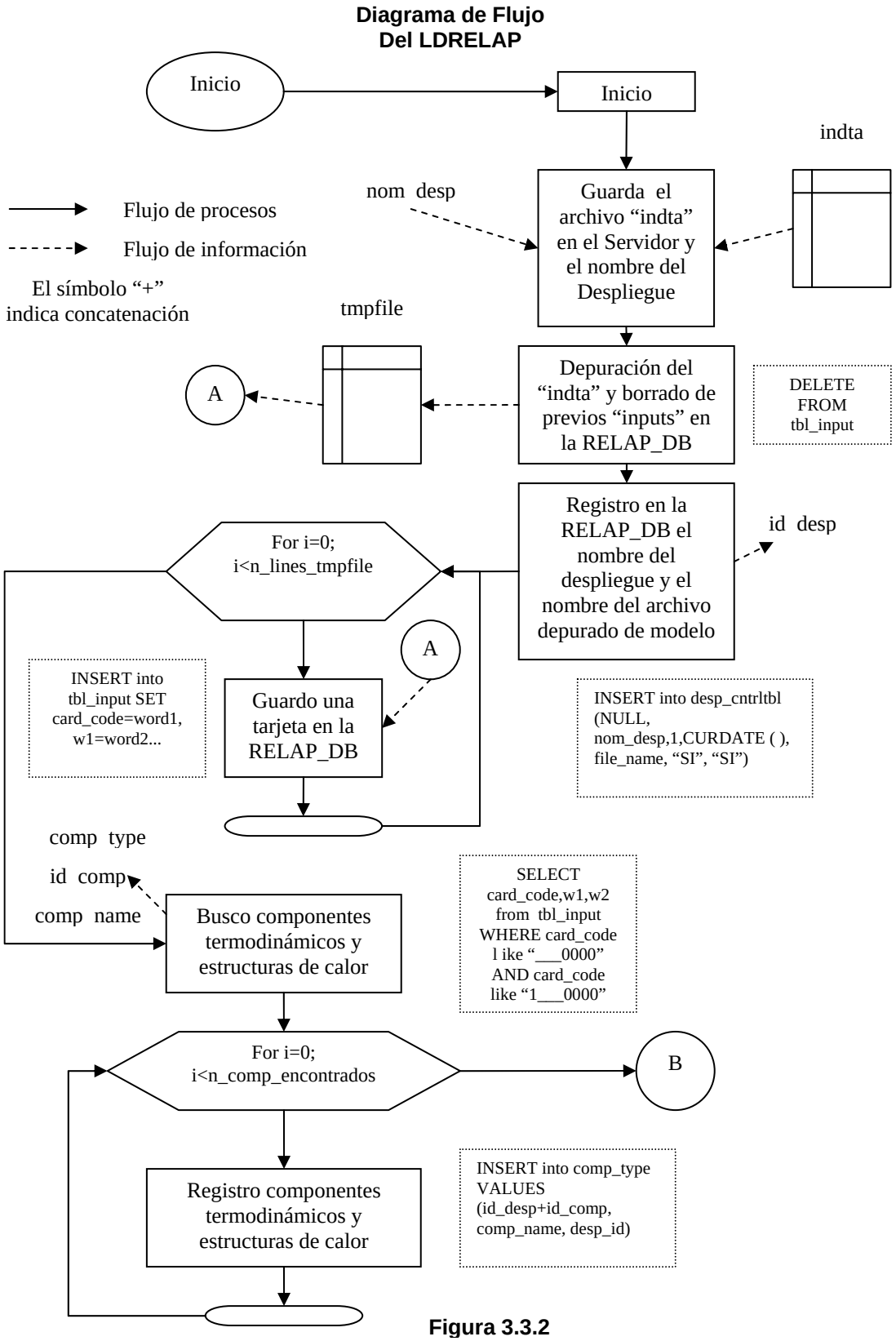
Diagrama de Flujo
Del LDRELAP
Inicio
Inicio
indta
nom desp
Guarda el archivo "indta" en el Servidor y el nombre del Despliegue
Flujo de procesos
Flujo de información
El símbolo "+" indica concatenación
tmpfile
A
Depuración del "indta" y borrado de previos "inputs" en la RELAP_DB
DELETE FROM tbl_input
Registro en la RELAP_DB el nombre del despliegue y el nombre del archivo depurado de modelo
id desp
For i=0; i<n_lines_tmpfile
A
INSERT into tbl_input SET card_code=word1, w1=word2...
Guardo una tarjeta en la RELAP_DB
INSERT into desp_cntrltbl (NULL, nom_desp,1,CURDATE ( ), file_name, "SI", "SI")
comp type
id comp
comp name
Busco componentes termodinámicos y estructuras de calor
SELECT card_code,w1,w2 from tbl_input WHERE card_code l ike "___0000" AND card_code like "1___0000"
For i=0; i<n_comp_encontrados
B
Registro componentes termodinámicos y estructuras de calor
INSERT into comp_type VALUES (id_desp+id_comp, comp_name, desp_id)

Figura 3.3.2

El símbolo “+”
indica concatenación

Diagrama de Flujo
Del LDRELAP
(continuación)

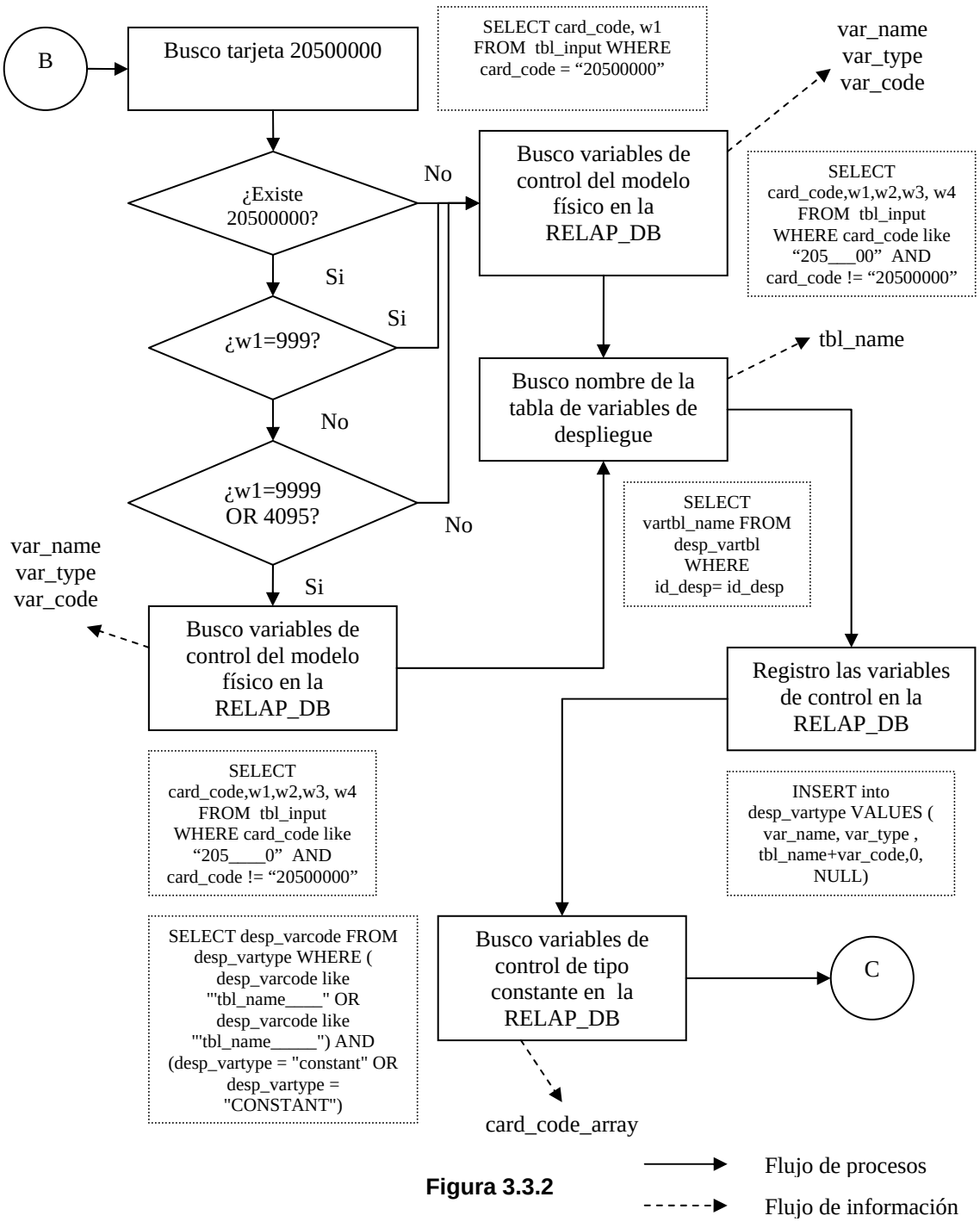

Figura 3.3.2

Diagrama de Flujo
Del LDRELAP
(continuación)

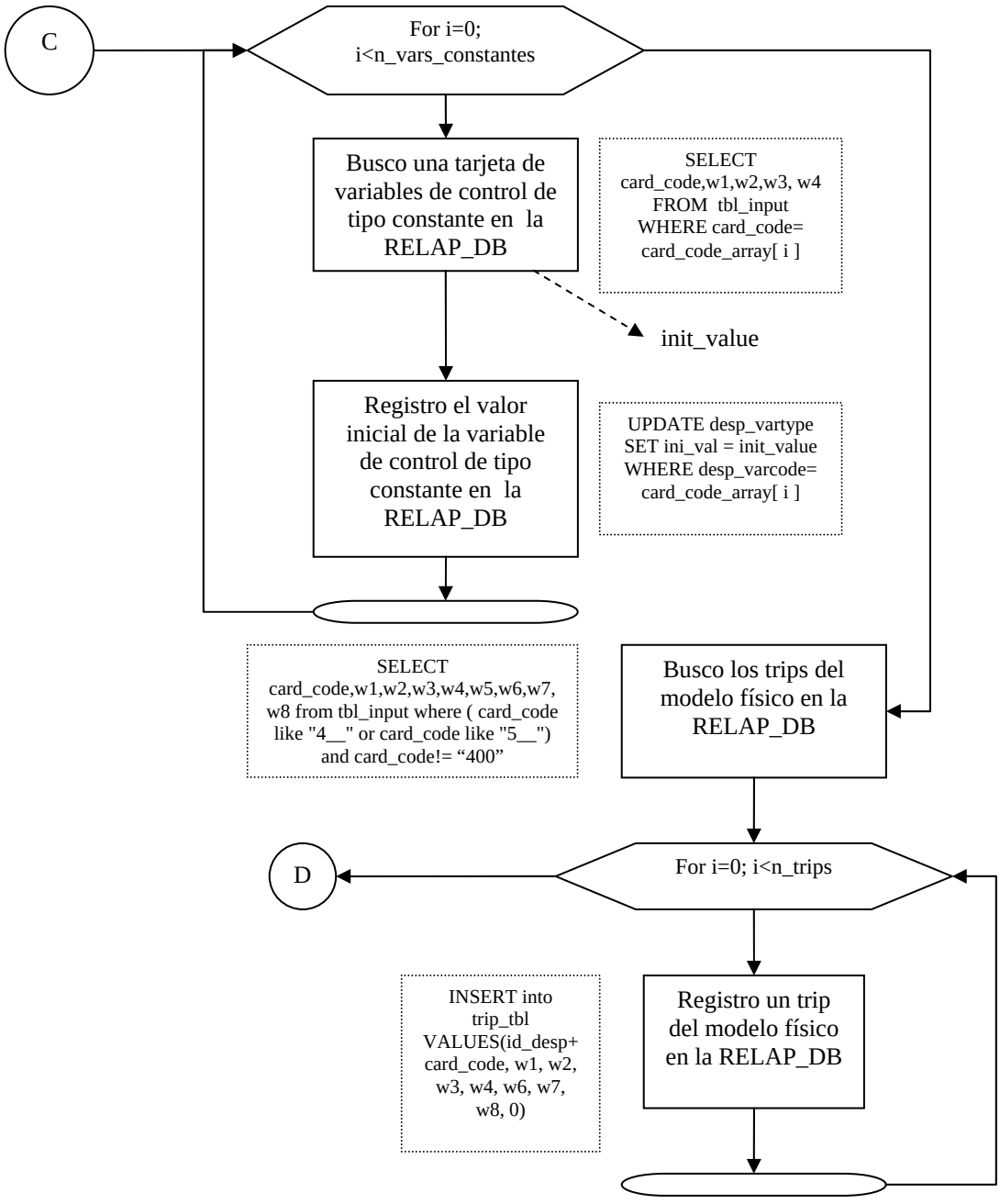

Figura 3.3.2

Diagrama de Flujo
Del LDRELAP
(continuación)

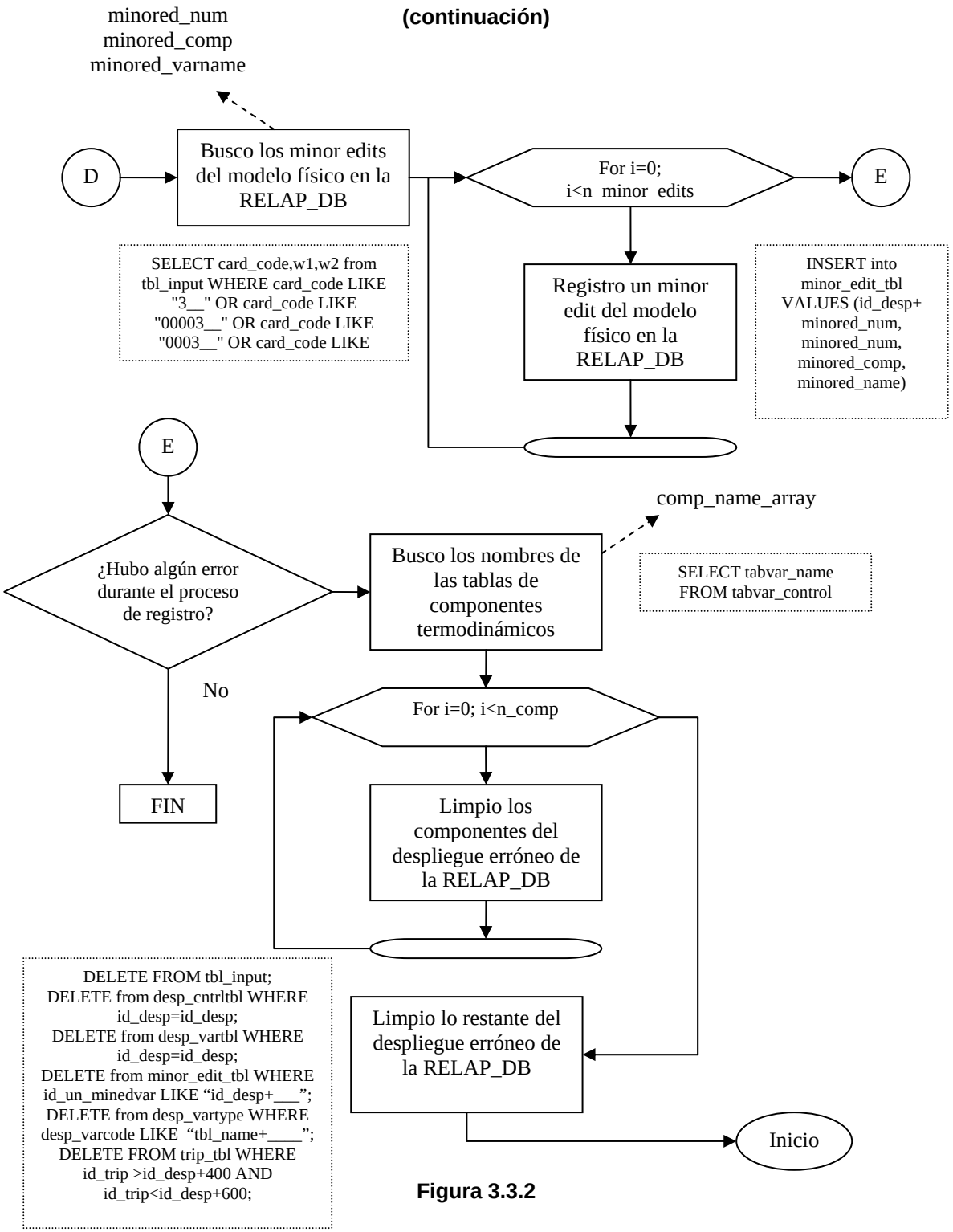

Figura 3.3.2

3.3.3 Estructura del MVRELAP

El Manejador de Variables de la Interfaz Gráfica tiene como funciones principales:

- Activar las variables de control y trips dados en el modelo físico, y registrados por el LDRELAP para el intercambio de datos entre los códigos RELAP5 y la RELAP_DB.

- Crear las tablas donde se alojarán los datos a intercambiar entre la base de datos y los clientes RELAP5. Para un despliegue dado.

- Modificar el archivo "indta" y regresarlo al usuario para que pueda simularlo en los códigos RELAP5.

Con respecto a los datos a intercambiar, el SADROV reconoce tres tipos de datos para el intercambio de información entre la RELAP_DB y los clientes RELAP5. La tabla 3.3.1 muestra estos tipos, su utilidad y el sentido del flujo de información.

Tabla 3.3.1 Tipos de Datos del SADROV

Tipo de Dato	Utilidad	Sentido del Intercambio
Variables de Despliegue	Sirven para mostrar datos provenientes de los códigos RELAP5, resultado de alguna simulación.	De los Códigos RELAP5 a la base de datos.
Trips	Sirven para lograr interactividad con los despliegues gráficos, pues permiten modificar parámetros dentro de los Códigos RELAP5.	De la base de datos a los Códigos RELAP5.
Variables constantes	Sirven para lograr interactividad con los Códigos RELAP5 al cambiar cantidades constantes para los códigos.	De la base de datos a los Códigos RELAP5.

Así mismo se decidió que estos datos se organizaran en la base de datos en tres tablas[25]. Los nombres y la estructura de estas tablas se describen a continuación en la figura 3.3.3. En el modelo entidad relación de la RELAP_DB se

[25] Una tabla para cada tipo de dato.

pueden observar estas tablas como la “tabla_de_variables”, la “tabla_de_trips” y la “tabla_de_variables_constantes”. Como se puede observar en este modelo entidad relación, estas tablas se forman con los datos de las variables de control, trips y los datos de control del despliegue dado.

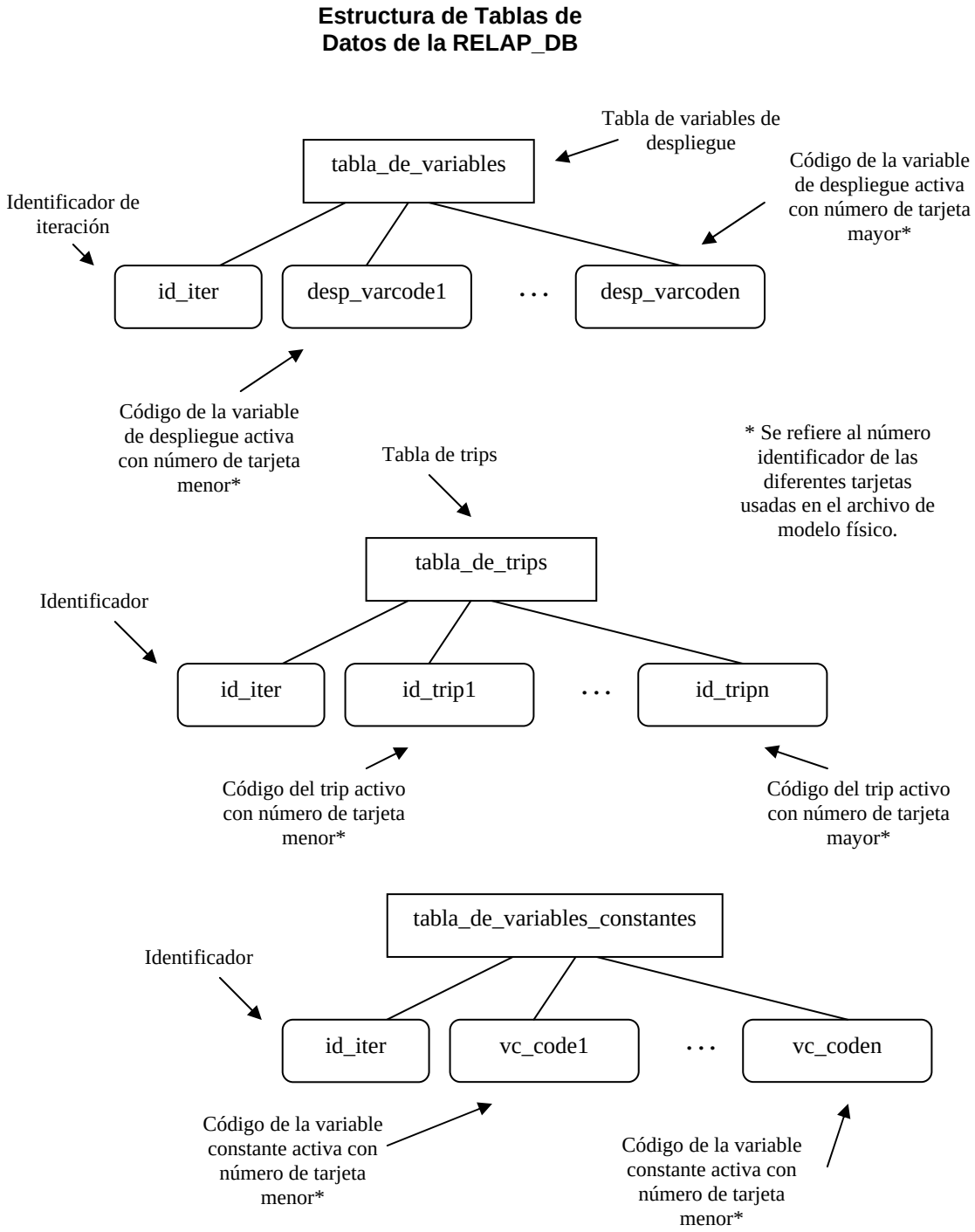

Figura 3.3.3

La figura 3.3.4 muestra el diagrama de flujo del MVRELAP.

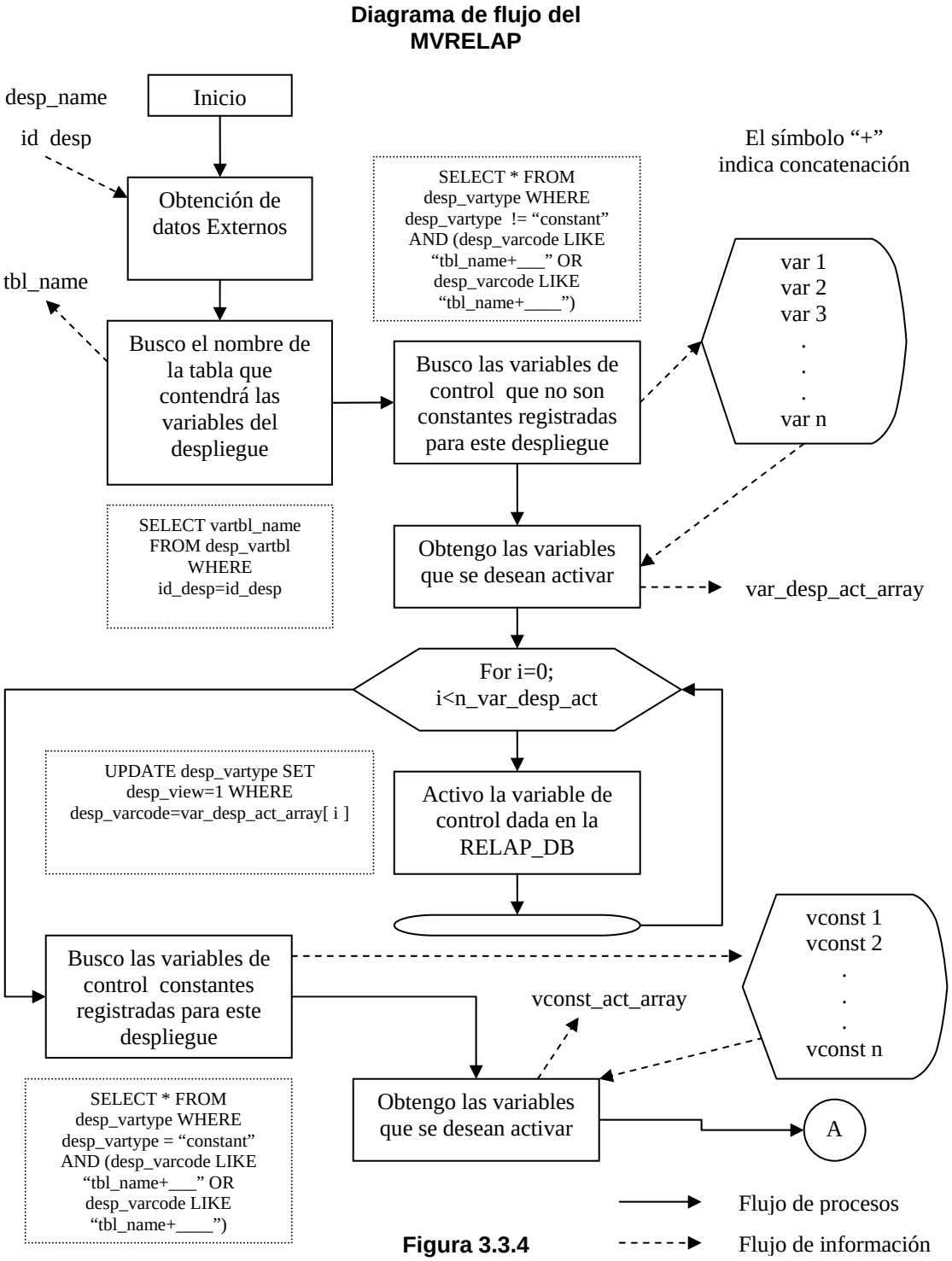

Figura 3.3.4

Diagrama de flujo del MVRELAP (continuación)

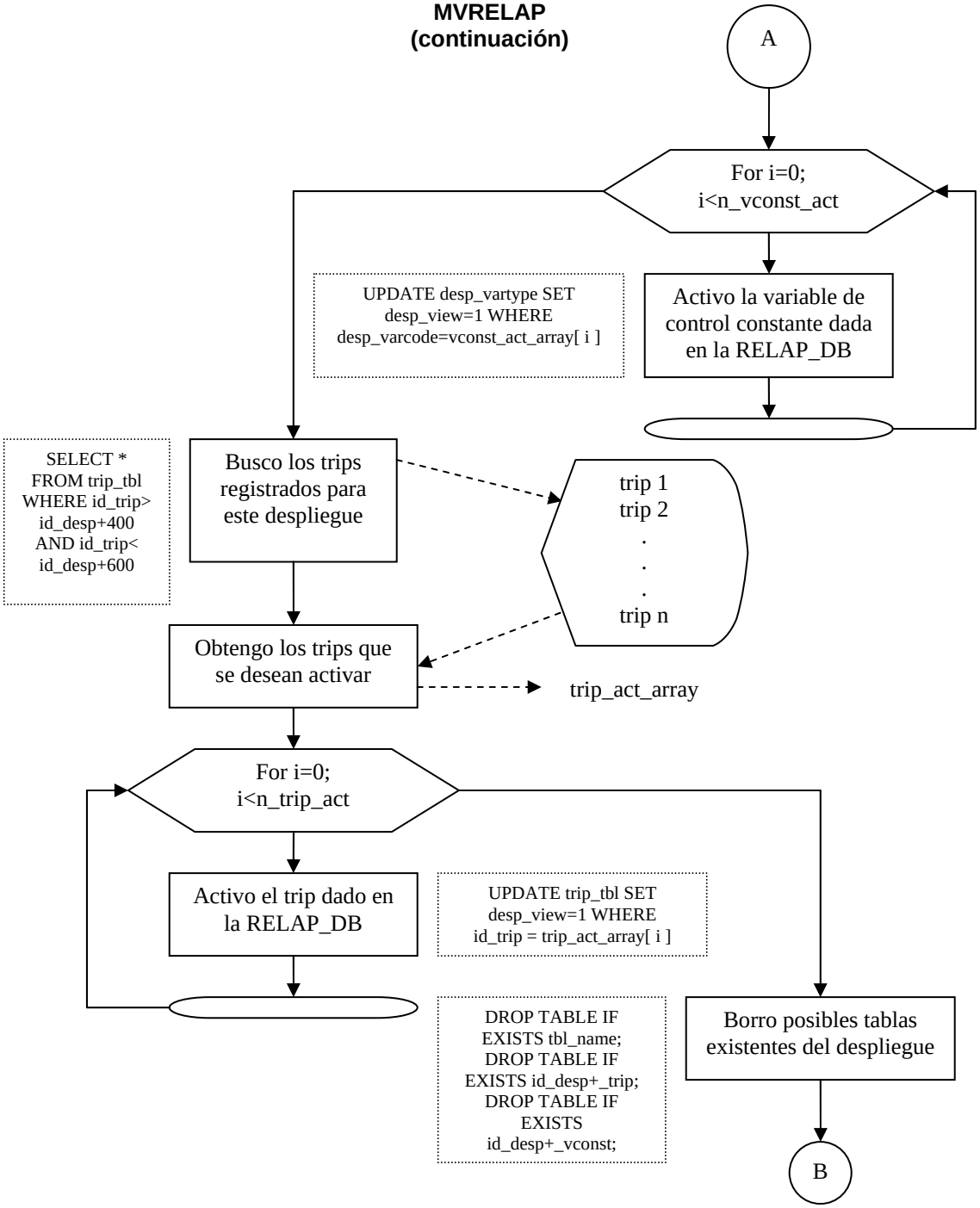

Figura 3.3.4

Diagrama de flujo del MVRELAP (continuación)

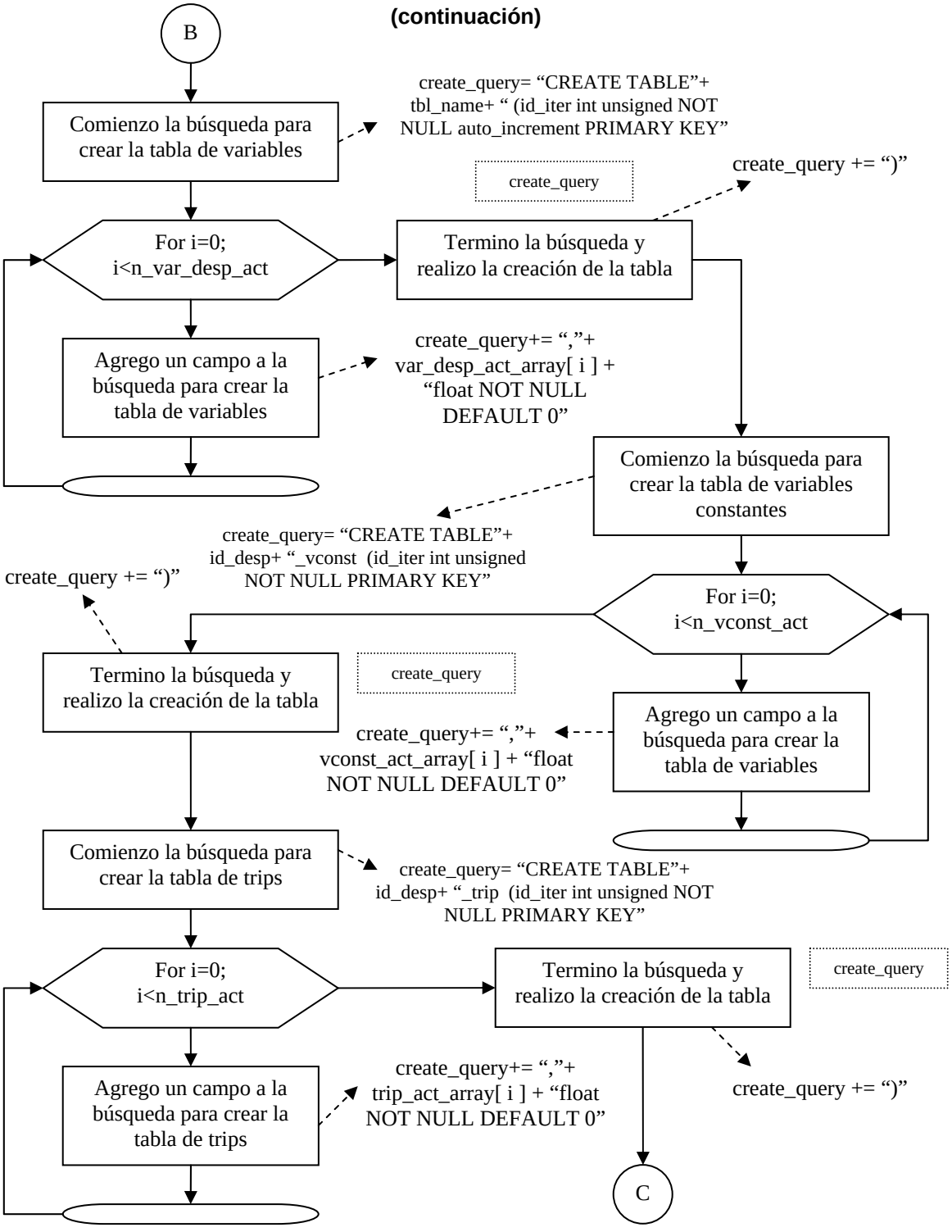

Figura 3.3.4

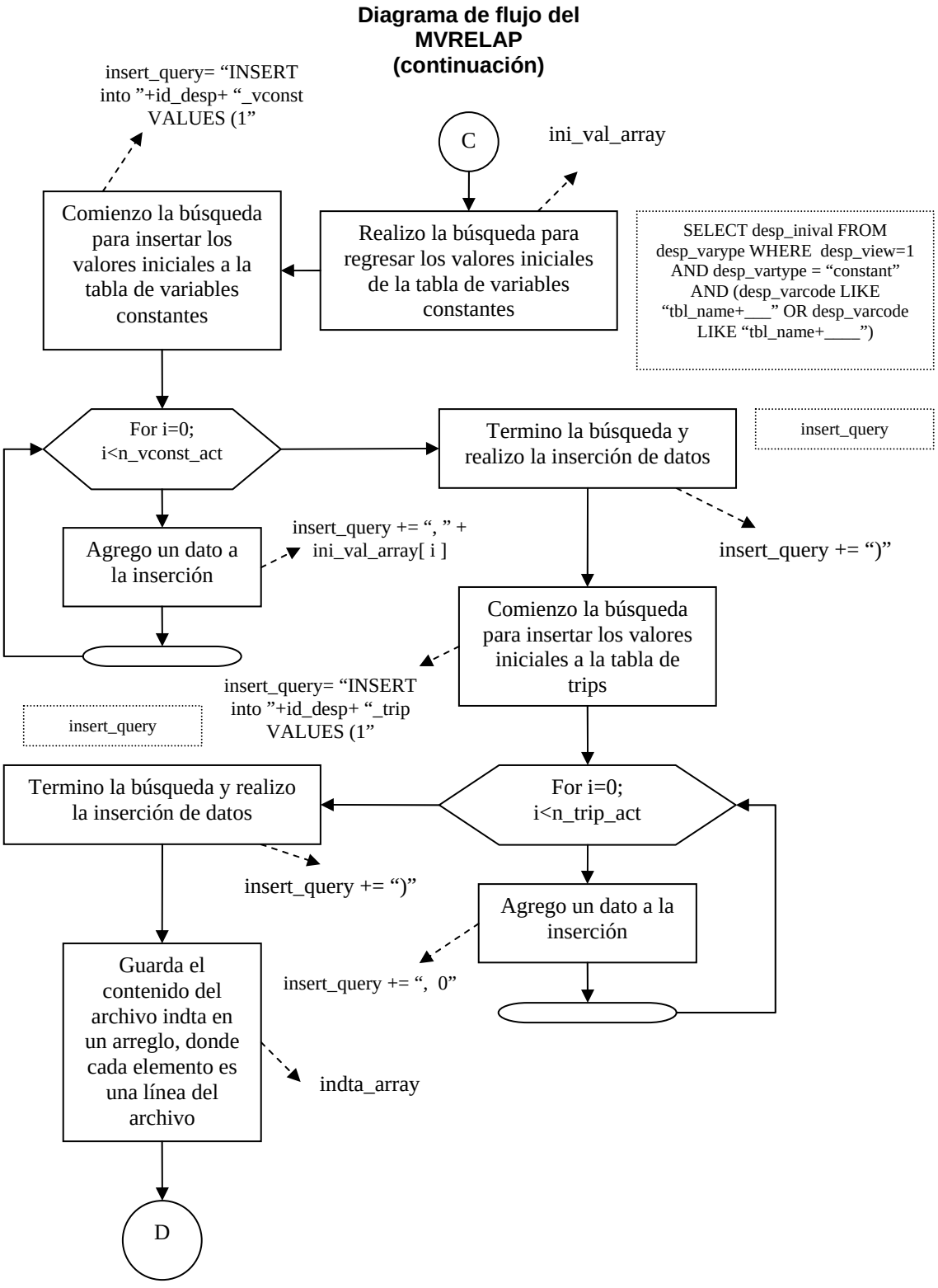
Diagrama de flujo del MVRELAP (continuación)
insert_query= “INSERT into ”+id_desp+ “_vconst VALUES (1”
C
ini_val_array
Comienzo la búsqueda para insertar los valores iniciales a la tabla de variables constantes
Realizo la búsqueda para regresar los valores iniciales de la tabla de variables constantes
SELECT desp_inival FROM desp_varype WHERE desp_view=1 AND desp_vartype = “constant” AND (desp_varcode LIKE “tbl_name+___” OR desp_varcode LIKE “tbl_name+____”)
For i=0; i<n_vconst_act
Termino la búsqueda y realizo la inserción de datos
insert_query
Agrego un dato a la inserción
insert_query += “, ” + ini_val_array[i]
insert_query += “)”
Comienzo la búsqueda para insertar los valores iniciales a la tabla de trips
insert_query= “INSERT into ”+id_desp+ “_trip VALUES (1”
insert_query
Termino la búsqueda y realizo la inserción de datos
For i=0; i<n_trip_act
insert_query += “)”
Agrego un dato a la inserción
Guarda el contenido del archivo indta en un arreglo, donde cada elemento es una línea del archivo
insert_query += “, 0”
indta_array
D

Figura 3.3.4

Diagrama de flujo del MVRELAP (continuación)

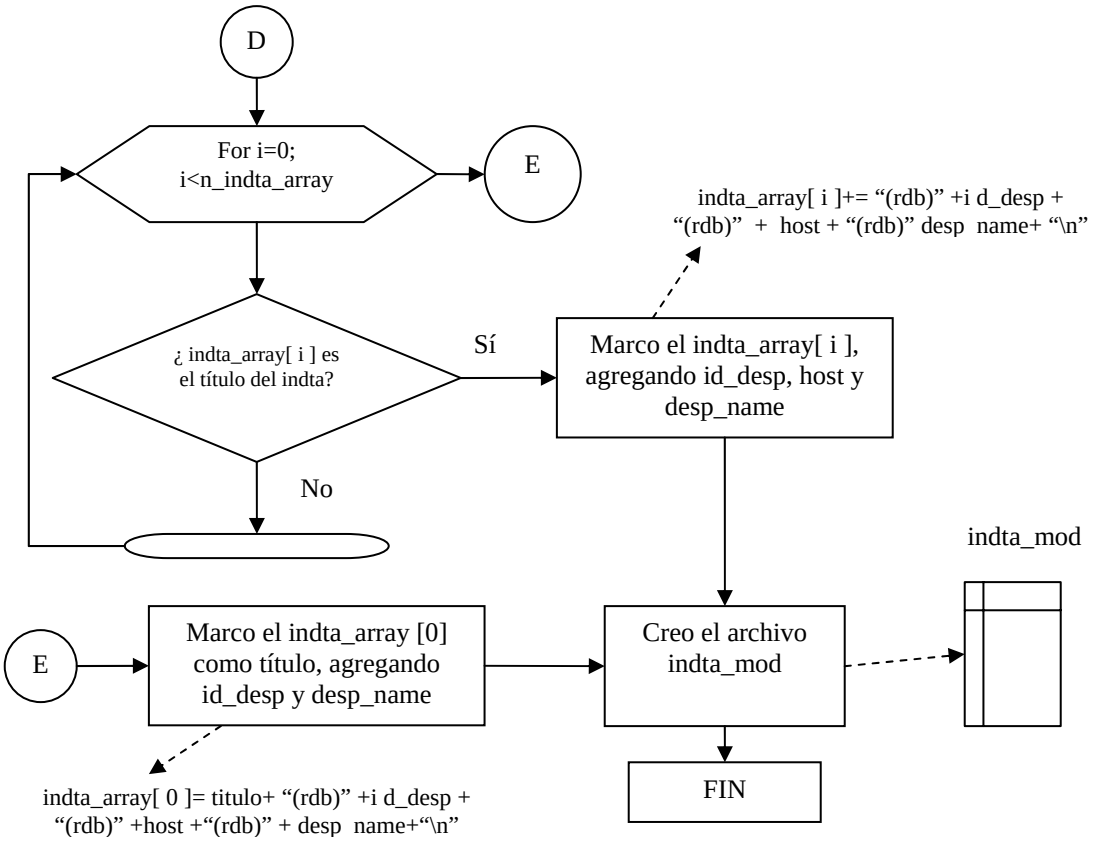

Figura 3.3.4

CAPÍTULO 4

IMPLEMENTACIÓN DEL SADROV

En la sección 1.4, específicamente en la figura 1.4.2 se observa el diagrama de implementación del SADROV. Como se puede observar se compone básicamente de el "Servidor" que se refiere a la base de datos, los "Clientes Despliegues Gráficos" que no forman parte integral del sistema, la "Interfaz Gráfica del SADROV" y los "Clientes RELAP5". En esta sección se describirá la implementación del SADROV.

4.1 Implementación del ICRELAP

En la sección 3.2 se explicaron las funciones que debería realizar la ICRELAP, así como las rutinas[1] que se modificaron y/o crearon para lograr su funcionamiento. En esta sección se dará una lista de todas las funciones implementadas para crear la ICRELAP así como una rápida descripción de su propósito, la rutina que la llama y el valor que devuelve.

- void dbinit_(char *desp_tbl,int length,int dummie)

    a) Esta función inicializa la conexión del ICRELAP con la RELAP_DB y crea los archivos varcdes.dat, contint.dat y tripint.dat necesarios para la transferencia de información.

    b) Recibe una variable entera *dummy*.

    c) Regresa el nombre de la tabla de variables de despliegue.

    d) Es llamada por la rutina en FORTRAN despl.f, la llama con el nombre desp_tbl = DBINIT (dummie)

- int get_id(void)

    a) Esta función obtiene el número de despliegue del modelo de entrada, de los códigos RELAP5.

    b) No recibe ningún parámetro.

    c) Regresa el número de despliegue.

    d) Es llamada por dbinit_ (…).

- void get_table(int id_desp)

    a) Esta función obtiene el nombre de la tabla de variables.

    b) Recibe el identificador de despliegue.
    c) No tiene valor de retorno.

    d) Es llamada por dbinit_ (…).

- int get_var_number(char codigo[])

[1] Tanto en FORTRAN como en lenguaje C.

a) Esta función obtiene el número de variable de control[2] que recibe del código de variable de la RELAP_DB.

b) Recibe el código de la variable con el formato en que se guarda en la RELAP_DB.

c) Regresa el número de la variable de control.

d) Es llamada por make_vardesc () y make_vconst ().

- void make_vardesc(int id_desp)

a) Esta función crea el archivo varcdes.dat necesario para el ICRELAP.

b) Recibe el identificador de despliegue.

c) No tiene valor de retorno.

d) Es llamada por dbinit_ (…).

- void make_trips(int id_desp)

a) Esta función crea el archivo tripint.dat necesario para el ICRELAP.

b) Recibe el identificador de despliegue.

c) No tiene valor de retorno.

d) Es llamada por dbinit_ (…).

- void make_vconst(int id_desp)

a) Esta función crea el archivo contint.dat necesario para el ICRELAP.

b) Recibe el identificador de despliegue.

c) No tiene valor de retorno.

d) Es llamada por dbinit_ (…).

- int get_trip(char codigo[])

a) Esta función obtiene el número de trip[3] que recibe del código de trip almacenado en la RELAP_DB.

[2] Según el formato de la tarjeta 205CCC00 ó 205CCCC0, donde CCC o CCCC es el número de variable.

b) Recibe el código de trip de la RELAP_DB.

c) Regresa el número de trip.

d) Llamada por make_trips (…).

- int getdespliegue_(char *tbl_name,int lenght)

a) Esta función obtiene el número de despliegue del nombre de la tabla de variables.

b) Recibe el nombre de la tabla de variables.

c) Regresa el número de despliegue.

d) Es llamada por la rutina en FORTRAN despl.f. La llama con el nombre id=GETDESPLIEGUE (tbl_name).

- void tripschain_(char *trips,int lenght,int *id_desp)

a) Obtiene los valores actuales de los trips activos en la RELAP_DB.

b) Recibe el identificador de despliegue.

c) Devuelve una cadena con los valores de los trips, separados por un carácter "*".

d) Llamada por la rutina en FORTRAN tripint.f. La llama con el nombre TRIPSCHAIN (id_desp).

- int getvaltrip_(char *tripchain,int *n_trip,int lenght)

a) Obtiene el valor de un trip en específico de la cadena de valores de los trips.

b) Recibe la cadena de trips y el número de trip.

c) Regresa el valor del trip especificado.

d) Es llamada por la rutina de FORTRAN tripint.f. La llama con el nombre GETVALTRIP (tripschain).

[3] Estos número de trip se refieren al número de tarjeta (que van del 401 a la 599) del modelo de entrada.

- void sendvars_(float vardesp[],int vardesp_size,int varnumber[],int vanumber_size,char *tblname,int *n_iter,int *n_varc,int lenght)

    a) Esta función envía las variables de despliegue al a RELAP_DB.

    b) Recibe un arreglo con los valores de las variables de despliegue, otro arreglo con los números de las variables de despliegue, el número de iteración, el número de variables de control y el nombre de la tabla de las variables.

    c) No tiene valor de retorno.

    d) Es llamada por la rutina en FORTRAN despl.f. La llama con el nombre SENDVARS (vardesp, vardesp_size, varnumber, varnumber_size, tbl_name, n_varc).

- void contchain_ (char *var_const, int lenght, int *id_desp)

    a) Obtiene los valores actuales de las variables constantes activas en la RELAP_DB.

    b) Recibe el identificador de despliegue.

    c) Devuelve una cadena con los valores de los trips, separados por un carácter “*”.

    d) Llamada por la rutina en FORTRAN contint.f. La llama con el nombre var_const = CONTCHAIN (id_desp).

- float getvalcont_ (char *contchain, int *n_vconst, int lenght)

    a) Obtiene el valor de una variable constante en específico de la cadena de valores de las variables constantes.

    b) Recibe la cadena de trips y el número de trip.

    c) Regresa el valor del trip especificado.

    d) Es llamada por la rutina de FORTRAN contint.f. La llama con el nombre GETVALCONT (contchain).

4.2 Implementación de la RELAP_DB

En esta sección mostraremos la implementación de las tablas de la base de datos RELAP_DB, describiremos la utilidad de cada tabla y los campos que la componen.

desp_cntrltbl. Esta tabla guarda los nombres de los despliegues que se han dado de alta mediante la Interfaz Gráfica. Sus campos son los siguientes:	
id_desp	Identificador de despliegue.
desp_name	Nombre del despliegue.
desp_date	Fecha de alta del despliegue en la RELAP_DB.
desp_error	Alerta de error en el despliegue, vale 0 si no hubo error al crear el despliegue y 1 si hubo error.
temp_filename	Nombre del archivo de modelo del despliegue en el servidor.
input_units	Sistema de unidades de entrada del modelo[4], puede tomar dos valores SI o BRITISH.
ouput_units	Sistema de unidades de salida del modelo, puede tomar dos valores SI o BRITISH.

desp_vartbl. Esta tabla guarda el nombre de la tabla donde se guardan los datos provenientes de las variables de despliegue de los Códigos RELAP5, este nombre se forma con los últimos 3 dígitos de un timestamp (NNN) y los primeros 3 caracteres del nombre del despliegue (CCC).	
id_desp	Identificador de despliegue.
vartbl_name	Nombre de la tabla donde se guardan los datos provenientes de los Códigos RELAP5, con formato NNN_CCC

desp_varop. Esta tabla guarda los nombres de las operaciones que se le aplican a las variables de despliegue (usadas para dar de alta las variables de despliegue).	
id_varop	Identificador de operación de variables de control, puede decirse que es el tipo de variable de control.
varop_name	Nombre de la operación.

logic_trip_op. Esta tabla guarda los nombres de las comparaciones lógicas que se le aplican a los trips.	
id_op	Identificador de operaciones lógicas de trips.
op_name	Nombre de la operación lógica.

[4] Ver sección 2.2.5.

desp_vartype. Esta tabla guarda los nombres de las variables de despliegue así como su tipo y les asigna una clave única.	
desp_varname	Nombre de la variable de despliegue[5].
desp_vartype	Tipo de la variable de control, todos los tipos de variables de control se encuentran en desp_varop.
desp_varcode	Código único de variable de despliegue, se forma por vartbl_name (NNN_CCC) + "_" + número de variable de control (MMM o MMMM).
desp_view	Alerta de despliegue, toma el valor de 1 si la variable está activa[6], 0 si no lo está.
desp_inival	Valor inicial de la variable, sólo aplica a las variables de tipo "constant". En los demás tipos de variables este campo queda con valor NULL.

tbl_input. Esta tabla es usada para guardar temporalmente el archivo de modelo de entrada (para registrar los componentes) cuando se está registrando un despliegue en la Interfaz Gráfica del SADROV. Este input es guardado sin ningún tipo de comentario[7].	
card_code	Código numérico de la tarjeta del modelo de entrada.
w1	Primera palabra de la tarjeta identificada por el card_code.
w2	Segunda palabra de la tarjeta identificada por el car_code.
. . .	
w19	Décimo novena palabra de la tarjeta identificada por el card_code. (Si alguna de estas palabras no existe en el modelo de entrada se pone un valor de default "---").

tabvar_control. Esta tabla guarda los nombres de los componentes termodinámicos de los Códigos RELAP5 (Ej. *turbine, mixer, pump, valve*, etc.) así como su código de tabla.	
tabvar_code	Código de la tabla del componente.
tabvar_name	Nombre de la tabla del componente.

[5] Nombre dado por el creador del modelo de entrada.
[6] Se desea extraer el valor de esa variable de los Códigos RELAP5.
[7] Así solo se guardan tarjetas y palabras, ver sección 2.2.

var_control. Esta tabla guarda los nombres de las variables termodinámicas de los Códigos RELAP5 (mismas que pueden ser usadas para dar de alta variables de despliegue) así como en que componente se encuentran (Ej. *turbine, pump, valve*, etc.).	
var_code	Código único de identificación de la variable por componente, el 1er número del código se refiere a la tabla de componente a la cual pertenece la variable en la base y los siguientes tres números es su número dentro del componente al que pertenece.
var_name	Nombre de la variable, según se declara en los "minor edits".
tabvar_code	Código de la tabla del componente al que pertenece la variable.

var_desc. Esta tabla guarda la descripción de las variables termodinámicas de los Códigos RELAP5 (significado físico y unidades ya sea en SI o en sistema inglés).	
var_code	Código único de identificación de la variable.
var_description	Descripción de la variable, tal como viene en los manuales de los Códigos RELAP5[8].
var_units_si	Unidades de la variable en SI.
var_units_english	Unidades de la variable en sistema inglés.

minor_edit_tbl. Esta tabla guarda los "minor edits" del modelo de entrada, asignándoles un código único de "minor edit". Este código se compone del id_desp (identificador de despliegue) + Número de "minor edit"[9].	
id_un_minedvar	Se forma con el identificador de despliegue y el número del "minor edit".
minedvar_number	Número del "minor edit" 3XX.
minedvar_comp	Código del componente del cual viene el "minor edit".
minedvar_name	Nombre del tipo de variable termodinámica, según el componente del que provenga.

[8] RELAP5/MOD3 code manual, Appendix A: Data Deck Organization and Data Card Requirements, p. A4-1 a A4-10.
[9] Este número va del 301 al 399, según se vio en la sección 2.2.4.

trip_tbl. Esta tabla guarda los “trips” encontrados del modelo de entrada, asignándoles un código único de identificación.	
id_trip	Identificación única de cada “trip”, se forma con el identificador de despliegue y el código del “trip” 4XX a 5XX.
var_code	Código de variable[10], según el Apéndice A del manual del RELAP5.
parameter	Parámetro del “trip”.
id_op	Código de operación lógica.
var_code2	Código de variable, según el Apéndice A del manual del RELAP5.
parameter_2	Parámetro del “trip”.
add_constant	Constante aditiva.
latch_ind	Indicador de retardo.[11]
timeof	Tiempo de retardo.
desp_view	Alerta de despliegue, toma el valor de 1 si el trip está activo[12], 0 si no lo está.

ACCUM. Esta tabla guarda los componentes termodinámicos del tipo ACCUM que encuentra en cada modelo de entrada, asignándoles un código único de componente, que se forma con el identificador de despliegue y el número de componente[13].	
id_un_comp	Código único de componente.
Id_comp	Número de componente que tiene en el modelo de entrada, éste se extrae de la tarjeta de la cual proviene.
comp_name	Nombre del componente que tiene en el modelo de entrada[14].
Id_desp	Identificador de despliegue.

[10] RELAP5/MOD3 code manual, Appendix A: Data Deck Organization and Data Card Requirements.
[11] Ver sección 2.2.9.
[12] Se desea manipular el valor del trip en la RELAP_DB.
[13] Ver sección 2.2.2.
[14] ídem.

ANNULUS. Esta tabla guarda los componentes termodinámicos del tipo ANNULUS que encuentra en cada modelo de entrada, asignándoles un código único de componente, que se forma con el identificador de despliegue y el número de componente.	
id_un_comp	Código único de componente.
Id_comp	Número de componente que tiene en el modelo de entrada, éste se extrae de la tarjeta de la cual proviene.
comp_name	Nombre del componente que tiene en el modelo de entrada.
Id_desp	Identificador de despliegue.

BRANCH. Esta tabla guarda los componentes termodinámicos del tipo BRANCH que encuentra en cada modelo de entrada, asignándoles un código único de componente, que se forma con el identificador de despliegue y el número de componente.	
id_un_comp	Código único de componente.
Id_comp	Número de componente que tiene en el modelo de entrada, éste se extrae de la tarjeta de la cual proviene.
comp_name	Nombre del componente que tiene en el modelo de entrada.
Id_desp	Identificador de despliegue.

CONTROL_S_Q. Esta tabla guarda los componentes termodinámicos del tipo CONTROL SYSTEM QUANTITIES que encuentra en cada modelo de entrada, asignándoles un código único de componente, que se forma con el identificador de despliegue y el número de componente.	
id_un_comp	Código único de componente.
Id_comp	Número de componente que tiene en el modelo de entrada, éste se extrae de la tarjeta de la cual proviene.
comp_name	Nombre del componente que tiene en el modelo de entrada.
Id_desp	Identificador de despliegue.

ECCMIX. Esta tabla guarda los componentes termodinámicos del tipo ECCMIX que encuentra en cada modelo de entrada, asignándoles un código único de componente, que se forma con el identificador de despliegue y el número de componente.	
id_un_comp	Código único de componente.
Id_comp	Número de componente que tiene en el modelo de entrada, éste se extrae de la tarjeta de la cual proviene.
comp_name	Nombre del componente que tiene en el modelo de entrada.
Id_desp	Identificador de despliegue.

GENERAL_Q. Esta tabla guarda los componentes termodinámicos del tipo GENERAL QUANTITIES que encuentra en cada modelo de entrada, asignándoles un código único de componente, que se forma con el identificador de despliegue y el número de componente.	
id_un_comp	Código único de componente.
Id_comp	Número de componente que tiene en el modelo de entrada, éste se extrae de la tarjeta de la cual proviene.
comp_name	Nombre del componente que tiene en el modelo de entrada.
Id_desp	Identificador de despliegue.

HEAT_S_Q. Esta tabla guarda los componentes termodinámicos del tipo HEAT STRUCTURE que encuentra en cada modelo de entrada, asignándoles un código único de componente, que se forma con el identificador de despliegue y el número de componente.	
id_un_comp	Código único de componente.
Id_comp	Número de componente que tiene en el modelo de entrada, éste se extrae de la tarjeta de la cual proviene.
comp_name	Nombre del componente que tiene en el modelo de entrada.
Id_desp	Identificador de despliegue.

JETMIXER. Esta tabla guarda los componentes termodinámicos del tipo JETMIXER que encuentra en cada modelo de entrada, asignándoles un código único de componente, que se forma con el identificador de despliegue y el número de componente.	
id_un_comp	Código único de componente.
Id_comp	Número de componente que tiene en el modelo de entrada, éste se extrae de la tarjeta de la cual proviene.
comp_name	Nombre del componente que tiene en el modelo de entrada.
Id_desp	Identificador de despliegue.

MTPLJUN. Esta tabla guarda los componentes termodinámicos del tipo MTPLJUN que encuentra en cada modelo de entrada, asignándoles un código único de componente, que se forma con el identificador de despliegue y el número de componente.	
id_un_comp	Código único de componente.
Id_comp	Número de componente que tiene en el modelo de entrada, éste se extrae de la tarjeta de la cual proviene.
comp_name	Nombre del componente que tiene en el modelo de entrada.
Id_desp	Identificador de despliegue.

PIPE. Esta tabla guarda los componentes termodinámicos del tipo PIPE que encuentra en cada modelo de entrada, asignándoles un código único de componente, que se forma con el identificador de despliegue y el número de componente.	
id_un_comp	Código único de componente.
Id_comp	Número de componente que tiene en el modelo de entrada, éste se extrae de la tarjeta de la cual proviene.
comp_name	Nombre del componente que tiene en el modelo de entrada.
Id_desp	Identificador de despliegue.

PUMP. Esta tabla guarda los componentes termodinámicos del tipo PUMP que encuentra en cada modelo de entrada, asignándoles un código único de componente, que se forma con el identificador de despliegue y el número de componente.	
id_un_comp	Código único de componente.
Id_comp	Número de componente que tiene en el modelo de entrada, éste se extrae de la tarjeta de la cual proviene.
comp_name	Nombre del componente que tiene en el modelo de entrada.
Id_desp	Identificador de despliegue.

REACTOR_K_Q. Esta tabla guarda los componentes termodinámicos del tipo REACTOR KINETIC QUANTITIES que encuentra en cada modelo de entrada, asignándoles un código único de componente, que se forma con el identificador de despliegue y el número de componente.	
id_un_comp	Código único de componente.
Id_comp	Número de componente que tiene en el modelo de entrada, éste se extrae de la tarjeta de la cual proviene.
comp_name	Nombre del componente que tiene en el modelo de entrada.
Id_desp	Identificador de despliegue.

REFLOOD_R_Q. Esta tabla guarda los componentes termodinámicos del tipo REFLOOD RELATED QUANTITIES que encuentra en cada modelo de entrada, asignándoles un código único de componente, que se forma con el identificador de despliegue y el número de componente.	
id_un_comp	Código único de componente.
Id_comp	Número de componente que tiene en el modelo de entrada, éste se extrae de la tarjeta de la cual proviene.
comp_name	Nombre del componente que tiene en el modelo de entrada.
Id_desp	Identificador de despliegue.

SEPARATR. Esta tabla guarda los componentes termodinámicos del tipo SEPARATR que encuentra en cada modelo de entrada, asignándoles un código único de componente, que se forma con el identificador de despliegue y el número de componente.	
id_un_comp	Código único de componente.
Id_comp	Número de componente que tiene en el modelo de entrada, éste se extrae de la tarjeta de la cual proviene.
comp_name	Nombre del componente que tiene en el modelo de entrada.
Id_desp	Identificador de despliegue.

SNGLJUN. Esta tabla guarda los componentes termodinámicos del tipo SNGLJUN que encuentra en cada modelo de entrada, asignándoles un código único de componente, que se forma con el identificador de despliegue y el número de componente.	
id_un_comp	Código único de componente.
Id_comp	Número de componente que tiene en el modelo de entrada, éste se extrae de la tarjeta de la cual proviene.
comp_name	Nombre del componente que tiene en el modelo de entrada.
Id_desp	Identificador de despliegue.

SNGLVOL. Esta tabla guarda los componentes termodinámicos del tipo SNGLVOL que encuentra en cada modelo de entrada, asignándoles un código único de componente, que se forma con el identificador de despliegue y el número de componente.	
id_un_comp	Código único de componente.
Id_comp	Número de componente que tiene en el modelo de entrada, éste se extrae de la tarjeta de la cual proviene.
comp_name	Nombre del componente que tiene en el modelo de entrada.
Id_desp	Identificador de despliegue.

TMDPJUN. Esta tabla guarda los componentes termodinámicos del tipo TMDPJUN que encuentra en cada modelo de entrada, asignándoles un código único de componente, que se forma con el identificador de despliegue y el número de componente.	
id_un_comp	Código único de componente.
Id_comp	Número de componente que tiene en el modelo de entrada, éste se extrae de la tarjeta de la cual proviene.
comp_name	Nombre del componente que tiene en el modelo de entrada.
Id_desp	Identificador de despliegue.

TMDPVOL. Esta tabla guarda los componentes termodinámicos del tipo TMDPVOL que encuentra en cada modelo de entrada, asignándoles un código único de componente, que se forma con el identificador de despliegue y el número de componente.	
id_un_comp	Código único de componente.
Id_comp	Número de componente que tiene en el modelo de entrada, éste se extrae de la tarjeta de la cual proviene.
comp_name	Nombre del componente que tiene en el modelo de entrada.
Id_desp	Identificador de despliegue.

TURBINE. Esta tabla guarda los componentes termodinámicos del tipo TURBINE que encuentra en cada modelo de entrada, asignándoles un código único de componente, que se forma con el identificador de despliegue y el número de componente.	
id_un_comp	Código único de componente.
Id_comp	Número de componente que tiene en el modelo de entrada, éste se extrae de la tarjeta de la cual proviene.
comp_name	Nombre del componente que tiene en el modelo de entrada.
Id_desp	Identificador de despliegue.

VALVE. Esta tabla guarda los componentes termodinámicos del tipo VALVE que encuentra en cada modelo de entrada, asignándoles un código único de componente, que se forma con el identificador de despliegue y el número de componente.	
id_un_comp	Código único de componente.
Id_comp	Número de componente que tiene en el modelo de entrada, éste se extrae de la tarjeta de la cual proviene.
comp_name	Nombre del componente que tiene en el modelo de entrada.
Id_desp	Identificador de despliegue.

Estas tablas competen exclusivamente a la parte fija de la base de datos, sin embargo con cada despliegue registrado se crean 3 tablas para el intercambio de datos con los "Clientes RELAP5" como se vio en la figura 3.3.3. A continuación se describen las características de esas tablas.

NNN_CCC[15]. Esta tabla es creada al registrar un despliegue y es usada para albergar las variables de despliegue provenientes de los "Clientes RELAP5".	
id_iter	Número de iteración de los datos.
vard_code1[16]	Valor de la variable de despliegue activa identificada por su var_code.
vard_code2	Valor de la variable de despliegue activa identificada por su var_code.
. . .	
vard_coden	Valor de la variable de despliegue activa identificada por su var_code.

[15] Este nombre está formado por un número aleatorio de tres cifras (NNN) y los tres primeros caracteres del nombre del despliegue (CCC).
[16] Este vard_code se forma con las últimas 7 u 8 cifras del var_code de la variable, es decir CCC_NNN ó CCC_NNNN.

X_trip[17]. Esta tabla es creada al registrar un despliegue y es usada para albergar los valores de los trips activos que se desean enviar a los “Clientes RELAP5”.	
id_iter	Identificador
trip_code1[18]	Valor del trip activo identificada por su id_trip.
trip_code1	Valor del trip activo identificada por su id_trip.
. . .	
trip_coden	Valor del trip activo identificada por su id_trip.

X_vconst[19]. Esta tabla es creada al registrar un despliegue y es usada para albergar los valores de las variables de control constantes activas que se desean enviar a los “Clientes RELAP5”.	
id_iter	Identificador.
vard_code1[20]	Valor de la variable de despliegue activa identificada por su var_code.
trip_code1	Valor de la variable de despliegue activa identificada por su var_code.
. . .	
trip_coden	Valor de la variable de despliegue activa identificada por su var_code.

La implementación de la RELAP_DB va estrechamente ligada al LDRELAP que es parte de la Interfaz Gráfica, lo que implica que para realizarle alguna mejora se tendría que cambiar el LDRELAP, debido al enfoque que se le dio a la RELAP_DB[21].

4.3 Implementación de la Interfaz Gráfica

Como se vio anteriormente la Interfaz Gráfica del SADROV se divide en dos aplicaciones, el LDRELAP y el MVRELAP. Ambas son aplicaciones Web implementadas en PHP cuyas características se han descrito en las secciones 2.4

[17] Este nombre está formado por el identificador de despliegue (X) seguido por la palabra “_trip”.
[18] Este trip_code se forma con el número de trip seguido por la palabra “_trip”.
[19] Este nombre está formado por el identificador de despliegue (X) seguido por la palabra “_vconst”.
[20] Este vard_code se forma con las últimas 7 u 8 cifras del var_code de la variable, es decir CCC_NNN ó CCC_NNNN.
[21] Ver sección 2.3.

y 3.3. En esta sección se dará una lista de las funciones implementadas en el lenguaje antes mencionado.

El LDRELAP que es el que se encarga básicamente de registrar los "despliegues gráficos" y el modelo físico de entrada del mismo. Antes de dar la lista de funciones que lo hacen funcionar debemos considerar dos aspectos:

- Esta es una aplicación Web, por lo que su funcionamiento estará distribuido en diferentes páginas Web.

- Es una aplicación implementada en POO[22], por lo que la mayoría de las funciones implementadas se encuentran en una clase llamada "lector".

En la figura 4.3.1 se observan las características de un objeto del tipo lector.

Objeto Lector

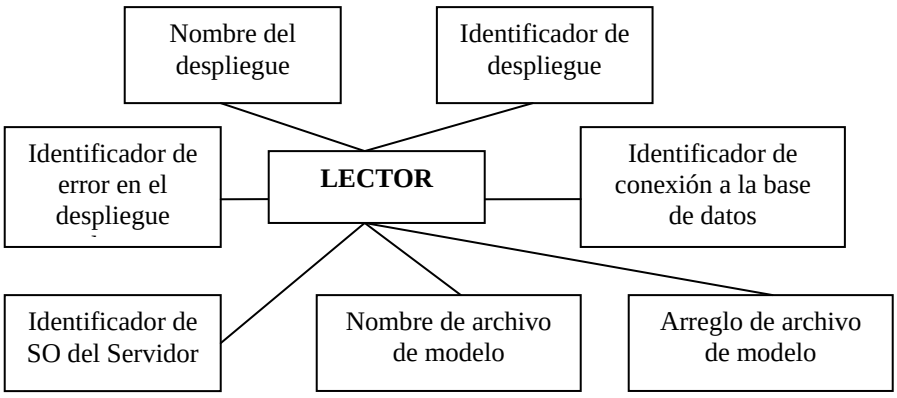

Figura 4.3.1

[22] Programación Orientada a Objetos.

Las funciones que componen a la clase “lector” son las siguientes:

lector ($direccion, $sistema, $id_despliegue)	
Funciones	Inicializa un objeto del tipo lector.
Parámetros que recibe	SO[23], identificador de despliegue, este parámetro es opcional, si no se da se asume que es un nuevo despliegue.
Valor de Retorno	Ninguno
Llamada desde la página o función	redir.php, despliegue.php

db_link_init()	
Funciones	Inicializa una conexión a la base de datos para el objeto lector.
Parámetros que recibe	Ninguno
Valor de Retorno	Ninguno
Llamada desde la página o función	redir.php, despliegue.php

depurador()	
Funciones	Depura el archivo de entrada de modelo.
Parámetros que recibe	Ninguno
Valor de Retorno	Ninguno
Llamada desde la página o función	redir.php

save()	
Funciones	Guarda el archivo de modelo de entrada en el servidor.
Parámetros que recibe	Ninguno
Valor de Retorno	Ninguno
Llamada desde la página o función	redir.php
insert_db()	
Funciones	Guarda el archivo de modelos depurado en el RELAP_DB e inicializa las unidades del despliegue en la misma.
Parámetros que recibe	Ninguno
Valor de Retorno	Ninguno
Llamada desde la página o función	redir.php

[23] Como se desarrollo en SO Windows y en Linux, éste se usa para distinguir el tipo de SO en que está funcionando.

comp_dbreg()	
Funciones	Busca los componentes termodinámicos existentes en el modelo, los registra en la RELAP_DB y los muestra en una tabla.
Parámetros que recibe	Ninguno
Valor de Retorno	Ninguno
Llamada desde la página o función	redir.php

clean_db()	
Funciones	Limpia la RELAP_DB de despliegues en los que se detecto un error.
Parámetros que recibe	Ninguno
Valor de Retorno	Ninguno
Llamada desde la página o función	despliegue.php

clean_inputdb()	
Funciones	Limpia la tabla tbl_input de modelos anteriores.
Parámetros que recibe	Ninguno
Valor de Retorno	Ninguno
Llamada desde la página o función	redir.php

despreg_db($desp_name)	
Funciones	Registra un despliegue en la RELAP_DB.
Parámetros que recibe	Recibe en nombre del despliegue.
Valor de Retorno	Regresa el identificador de despliegue, generado en el registro.
Llamada desde la página o función	redir.php

vardespreg_db()	
Funciones	Registra en la RELAP_DB las variables de control encontradas.
Parámetros que recibe	Ninguno
Valor de Retorno	Ninguno
Llamada desde la página o función	redir.php

bad_insert($bad_inserts)	
Funciones	Si algún componente termodinámico no es registrado correctamente hace un análisis minucioso e intenta registrarlo en la RELAP_DB.
Parámetros que recibe	Arreglo con los códigos de las inserciones que no se pudieron llevar a cabo.
Valor de Retorno	Ninguno
Llamada desde la página o función	comp_dbreg()

bad_regvars ($bad,$name_tbl,$type,$cards_type)	
Funciones	Si alguna variable de control no es registrada correctamente
Parámetros que recibe	Código de la tarjeta, nombre de la tabla de variables, tipo de variable y tipo de código de variable.
Valor de Retorno	Ninguno
Llamada desde la página o función	vardespreg_db()

desp_error()	
Funciones	Checa si hubo algún error en el registro del despliegue[24]
Parámetros que recibe	Ninguno
Valor de Retorno	Ninguno
Llamada desde la página o función	redir.php

get_status()	
Funciones	Función que obtiene el status del despliegue, principalmente checa si tiene error.
Parámetros que recibe	Ninguno
Valor de Retorno	Ninguno
Llamada desde la página o función	despliegue.php

[24] Si hubo algún error en cualquiera de las funciones del LDRELAP los detecta.

get_inputfile()	
Funciones	Función que pone el contenido del archivo de entrada de modelo en el arreglo de archivo de modelo[25]
Parámetros que recibe	Ninguno
Valor de Retorno	Ninguno
Llamada desde la página o función	

set_markfile()	
Funciones	Función que marca el arreglo de archivo “indta” con el identificador de despliegue y nombre de despliegue.
Parámetros que recibe	Ninguno
Valor de Retorno	Ninguno
Llamada desde la página o función	

write_inputdecfile()	
Funciones	Función que rescribe el archivo “indta” del arreglo de archivo, para ser regresado al usuario de la Interfaz Gráfica.
Parámetros que recibe	Ninguno
Valor de Retorno	Ninguno
Llamada desde la página o función	

set_inputend()	
Funciones	Función que pone el terminador del “indta” en el arreglo del archivo.
Parámetros que recibe	Ninguno
Valor de Retorno	Ninguno
Llamada desde la página o función	

unset_inputend()	
Funciones	Función que busca el terminador del “indta” en el arreglo del archivo y lo quita para poder agregarle líneas útiles al modelo de entrada.
Parámetros que recibe	Ninguno
Valor de Retorno	Ninguno
Llamada desde la página o función	

[25] Esto facilita la manipulación del archivo.

reg_trips()	
Funciones	Función que registra los trips encontrados en el modelo de entrada en la RELAP_DB.
Parámetros que recibe	Ninguno
Valor de Retorno	Ninguno
Llamada desde la página o función	redir.php

vconst_init()	
Funciones	Función que inicializa los valores de las variables constantes encontradas en el modelo de entrada en la RELAP_DB.
Parámetros que recibe	Ninguno
Valor de Retorno	Ninguno
Llamada desde la página o función	vardespreg_db()

Las funciones antes mencionadas son las que componen básicamente al LDRELAP y lo hacen funcionar.

Se decidió implementar el LDRELAP en POO debido a la facilidad, que tiene esta forma de programación, para modificar la aplicación en cuando sea necesario. Al pensar que en trabajos futuros se puede profundizar el análisis de los modelos de entrada, va a ser relativamente sencillo ampliar

En la implementación del MVRELAP no se usaron funciones como tales, tiene sus actividades distribuidas en varias páginas Web. Así a continuación damos una lista de las páginas que se usaron para lograr la implementación de “La Interfaz Gráfica del SADROV”:

upload.php	
Funciones	Recibe un nombre de despliegue y el archivo de entrada de modelo a ser utilizado en el proceso y registra los componentes en la RELAP_DB.
Parámetros que recibe	Archivo de entrada de modelo.
Valor de Retorno	id[26]. Identificador que se le asigna al modelo a simular.

[26] id. Número que se asigna de manera automática al modelo que será simulado, este id será de gran importancia porque mediante él se identificará el modelo en proceso.

redir.php	
Funciones	Muestra los componentes hallados en el archivo de modelo como lo es el tipo de componente, número del componente, minor edits, y trips. Guarda en la RELAP_DB los componentes, minor edits y trips. Muestra el nombre del despliegue (dado en upload.php), el id del mismo y los trips encontrados.
Parámetros que recibe	id. Identificador el cual es un número que se asigna al modelo a simular.
Valor de Retorno	Identificador (id).

despliegue.php	
Funciones	Permite la selección de dos opciones que son: Manejo de variables contenidas en RELAP_DB. Registro de variables en RELAP_DB.
Parámetros que recibe	Id.
Valor de Retorno	Id.

mvrelap.php	
Funciones	Permite elegir las variables despliegue así como las variables constantes que se encuentran en la RELAP_DB.
Parámetros que recibe	Id.
Valor de Retorno	Id y tipo[27].

proceso.php	
Funciones	Activa[28] en la RELAP_DB las variables de control y trips seleccionados en mvrelap.php y trips.php.
Parámetros que recibe	Id y tipo.
Valor de Retorno	Id y fin[29] (opcional).

[27] tipo. Su valor es 1 si va a registrar variables de despliegue y constantes. Su valor es 2 si va a registrar trips.
[28] Coloca un valor de 1 en el campo desp_view de la tabla correspondiente de acuerdo a las variables o trips seleccionados.
[29] fin. Identificador que indica el fin del proceso de la interfaz.

trips.php	
Funciones	Permite elegir los trips hallados en la RELAP_DB.
Parámetros que recibe	Id y fin.
Valor de Retorno	Id.

despliegue.php	
Funciones	Muestra el identificador del despliegue (id), así como el nombre del mismo. Muestra el status de registro. Permite descargar el archivo de modelo de entrada ya modificado para su uso. Permite redireccionar a upload.php.
Parámetros que recibe	Id y fin.
Valor de Retorno	Ninguno

4.3.1 Diagrama de implementación de la Interfaz Gráfica del SADROV.

En la figura 4.3.2 se puede observar cada una de las páginas que conforman la interfaz gráfica además de una pequeña descripción de las funciones principales de cada una de ellas. Como puede observarse algunas páginas simplemente se usan como pantallas de información[30], mientras que en otras, como proceso.php[31] el usuario ni siquiera notará que estuvo en ellas, pues es redireccionado inmediatamente a otra página.

[30] Como mvrelap.php, trips.php entre otras.
[31] En este tipo de páginas se realiza casi toda la interacción con la RELAP_DB.

Interfaz Gráfica del SADROV

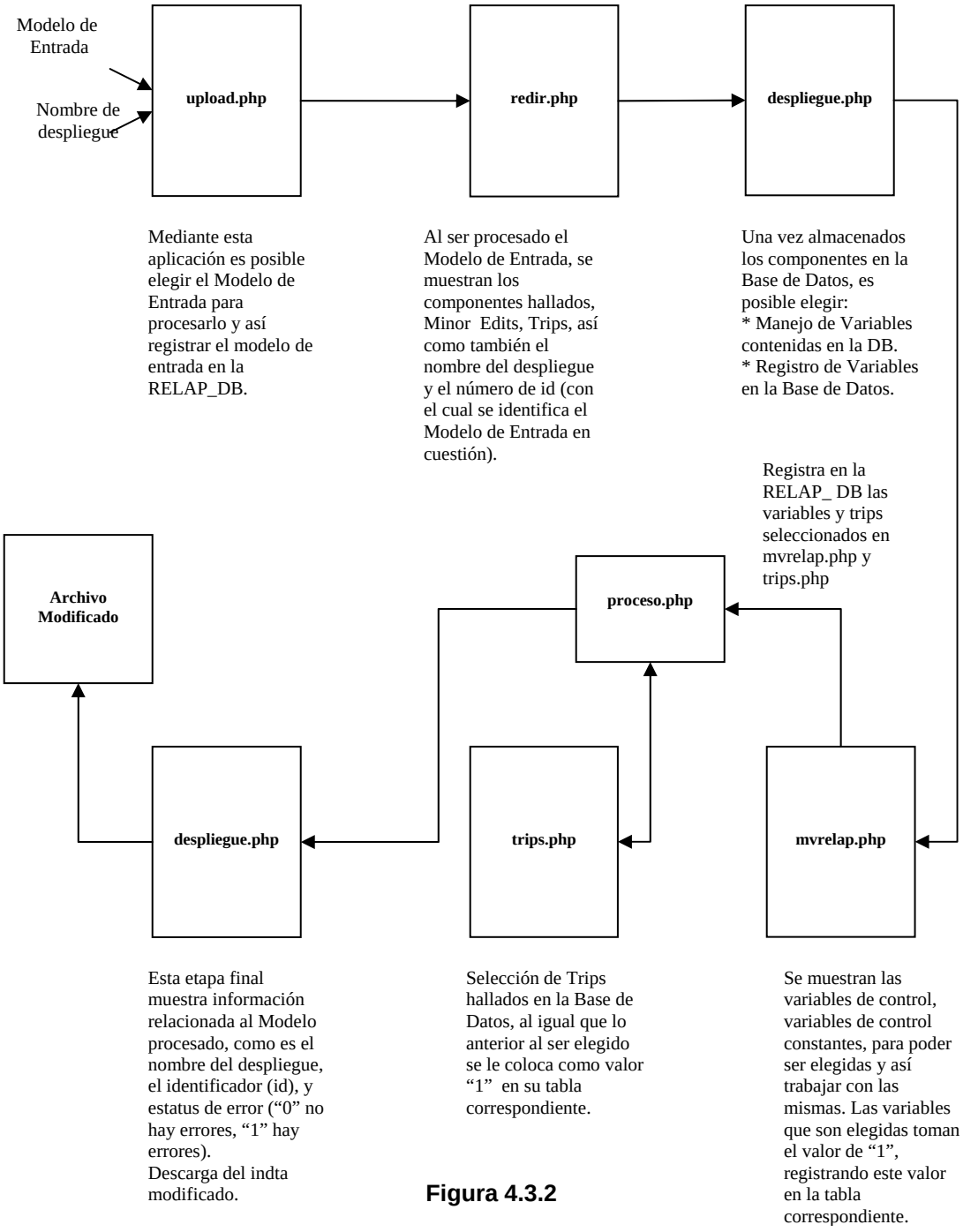

Figura 4.3.2

CAPÍTULO CINCO

PRUEBAS DEL SADROV Y CONCLUSIONES

En esta sección veremos las pruebas que se le realizaron al SADROV así como las conclusiones a las que nos llevó el desarrollo del presente trabajo.

5.1 Pruebas del Sistema

En esta sección veremos las pruebas que se le realizaron al SADROV, que incluyen el proceso completo de registro, en la RELAP_DB, del modelo de entrada a simular[1] en los Códigos RELAP5, la activación de las variables de despliegue, trips y variables constantes, así como las corridas de los clientes de prueba[2]. Estas últimas son un parámetro importante que nos permite definir si el SADROV es adecuado para responder a las necesidades propias del "simulador de aula". Sobre todo por los tiempos necesarios para la actualización de despliegues gráficos.

La tabla 5.1.1 contiene la información necesaria sobre los recursos físicos computacionales usados en cada parte de las pruebas del SADROV.

Tabla 5.1.1 Recursos Usados en las pruebas del SADROV

Componente	Descripción	Tipo de Conexión	Dirección IP	Localización física
SERVIDOR LAMP	Servidor de MySQL que alberga la RELAP_DB y a la interfaz Gráfica.	10 baseT (10 Mbps)	132.248.52.222	Laboratorio LAPE ubicado en la DEPFI (México, D.F)
Cliente RELAP5	Estación de trabajo de tipo (Alpha) UNIX.	ADSL 256 kbps.	201.129.0.166	Laboratorio LAIRN en la DEPFI (Jiutepec, Mor.)
Cliente de Despliegue.	PC corriendo en SO Windows XP.	ADSL 256 kbps.	201.129.0.166	Laboratorio LAIRN en la DEPFI (Jiutepec, Mor.)
Cliente de Interacción.	PC corriendo en SO Windows XP.	ADSL 256 kbps.	201.129.0.166	Laboratorio LAIRN en la DEPFI (Jiutepec, Mor.)
Cliente de Despliegue.	PC corriendo en SO Windows XP.	Modem 56kbps.	200.228.161	Sitio de Pruebas (Algún lugar de México, DF)
Cliente de Interacción.	PC corriendo en SO Windows XP.	Modem 56kbps.	200.228.161	Sitio de Pruebas (Algún lugar de México, DF)

[1] Ver apéndice A.
[2] El código fuente de ambos clientes se incluye en los apéndices B y C.

Como se puede observar en la tabla 5.5.1 el único componente “fijo” en este sistema se trata del servidor LAMP, que alberga la Interfaz Gráfica del SADROV y el SGBD que contiene a la RELAP_DB. Los demás componentes al tratarse de clientes, pueden ser implementados en cualquier parte del orbe y estar corriendo al mismo tiempo.

La tabla 5.1.2 contiene los tiempos de respuesta entre el servidor y los clientes de despliegue. Las condiciones de estas pruebas fueron las siguientes:

1. EL Cliente RELAP5 registró un modelo de entrada que se encuentra en el Apéndice A.

2. Las variables de despliegue activadas en la Interfaz Gráfica del SADROV se muestran en la figura 5.1.4.

3. Los trips activados en la Interfaz Gráfica del SADROV se muestran en la figura 5.1.5.

4. Las variables constantes activas se muestran en la figura 5.1.4.

5. El cliente RELAP5 se echó a andar simultáneamente con los Clientes de Despliegue de Prueba y el Cliente de Interacción[3].

6. En el caso del cliente de prueba se tomó la decisión de sólo tomar una muestra con las primeras 200 iteraciones de los códigos.

7. Las corridas de todos los clientes se encuentran en el Apéndice A.

8. Los archivos contint.dat, varcdes.dat y tripint.dat se encuentran en el Apéndice A.

Tabla 5.1.2 Tiempos de Respuesta de prueba del SADROV

Cliente de Despliegue	Tiempo promedio por iteración.[ms]
Sitio de Pruebas (México, D.F)	143.16
LAIRN (Jiutepec, Mor.)	32.41

Como puede observarse en la tabla 5.1.2 se tienen los tiempos de respuesta para regresar las variables de la simulación que se estaba realizando en ese mismo momento en el Cliente RELAP5. Sólo se tomaron los tiempos de respuesta de los Clientes de Despliegues por las siguientes razones:

[3] En este caso fueron dos pruebas diferentes, una corriendo ambos clientes en el LAIRN ubicado en Jiutepec, Mor. y la otra corriendo en el Sitio de Pruebas en México D.F.

- Los 3 Clientes corrieron simultáneamente. Así que el tiempo de respuesta más significativo es el de retorno de los valores por los Clientes de Despliegue.

- El tiempo de respuesta de los Clientes RELAP5 será menor al tiempo de respuesta de los Clientes de Despliegue, dado que los primeros alimentan los datos a la RELAP_DB, de donde los segundos toman estos datos.

Para implementar los clientes de prueba se usó el API C de MySQL, mientras que para medir el tiempo de respuesta se usó la función del lenguaje C llamada clock ()[4] de la siguiente manera:

```
inicio = clock ();

busqueda_mysql ();

tiempo_busqueda = (clock () - inicio) / (CLOCKS_PER_SEC);
```

Esas tres sentencias nos permiten calcular el tiempo en segundos[5], posteriormente se obtienen los milisegundos (que es una medida más acorde a los tiempos requeridos):

```
tiempo_busqueda = 1000*tiempo_busqueda;
```

[4] Que se encuentra en la librería “time.h”.
[5] De las funciones de la librería “time.h” la función clock ( ) es la que permite calcular tiempos de la manera más precisa (fracciones de segundo).

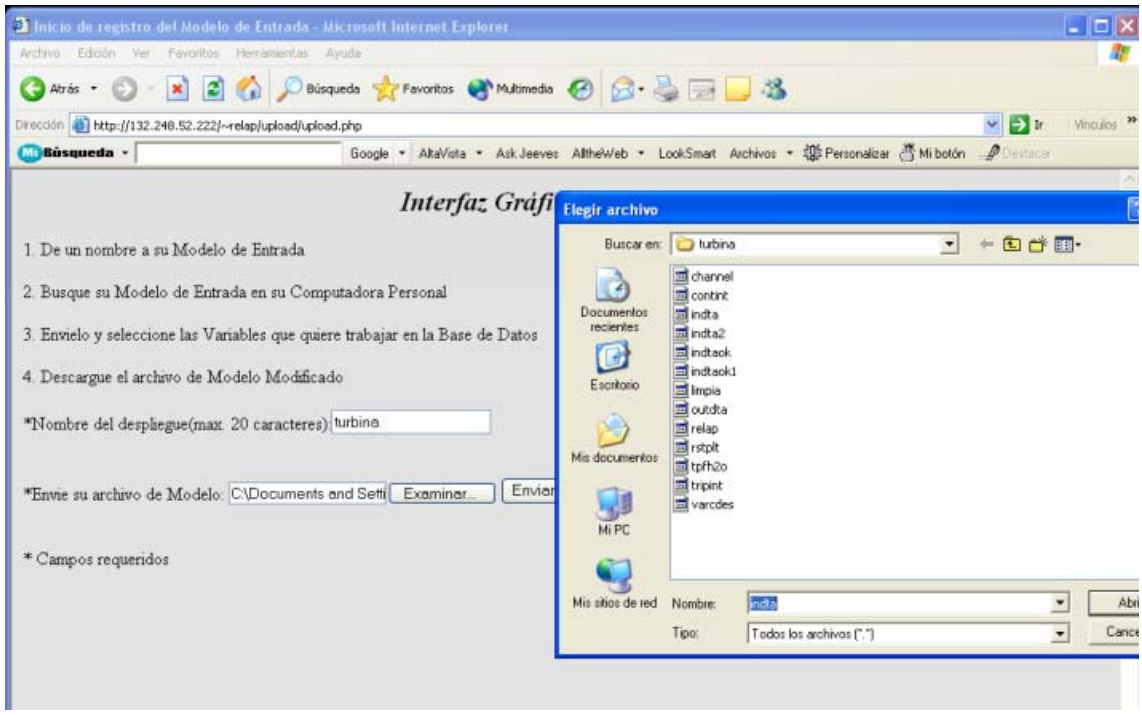

Figura 5.1.1 Página upload.php

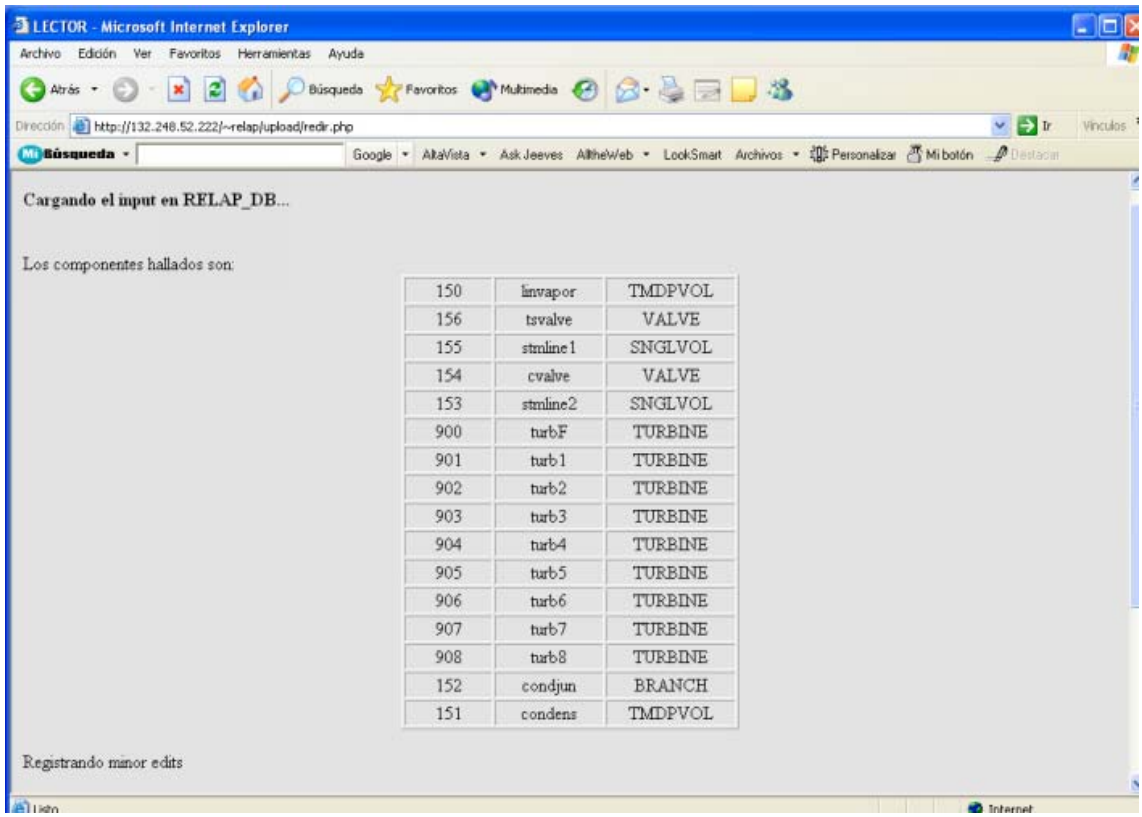

Figura 5.1.2 Página redir.php

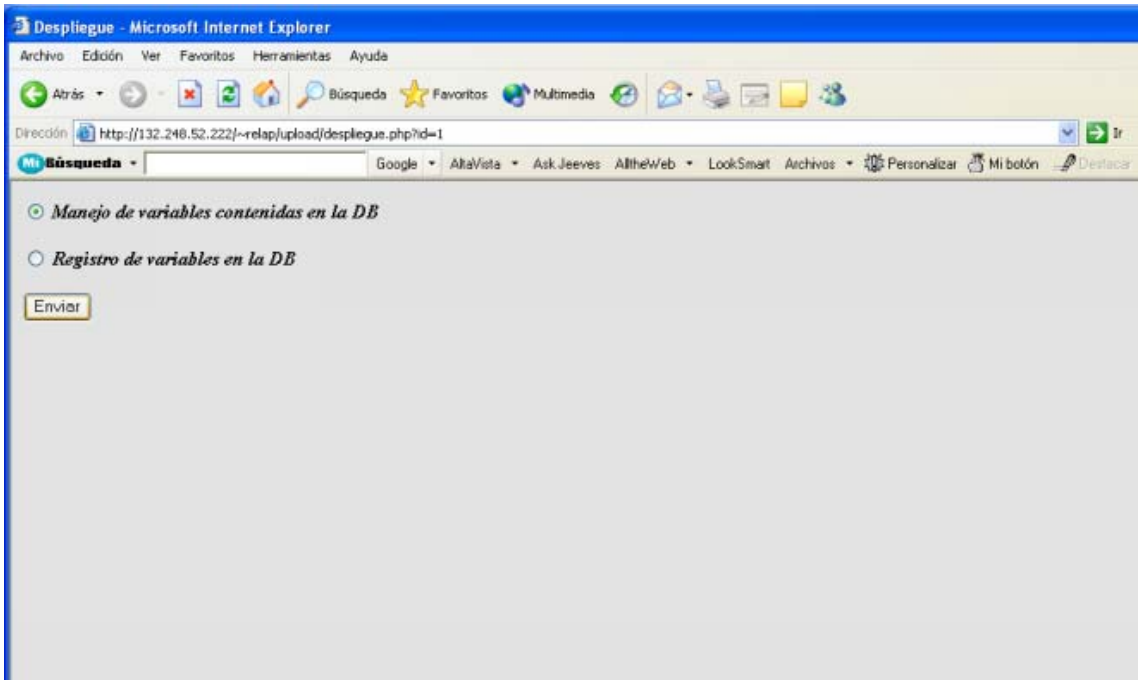

Figura 5.1.3 Página despliegue.php

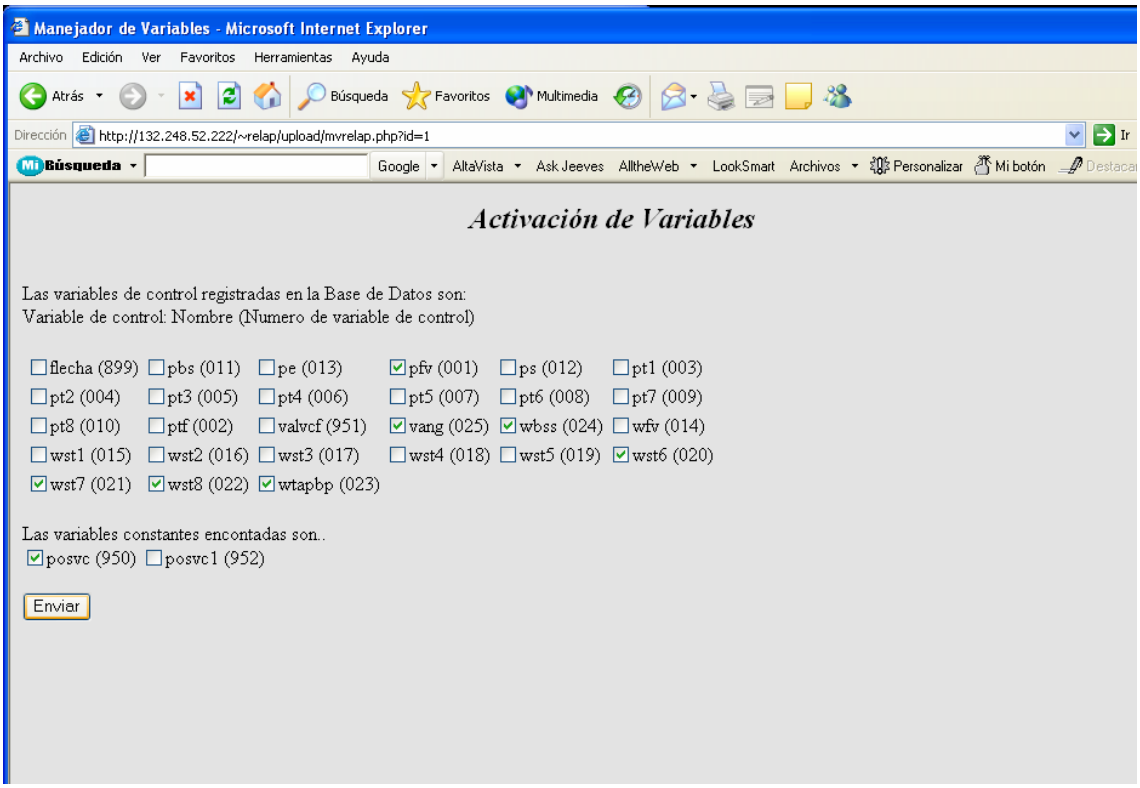

Figura 5.1.4 Página mvrelap.php

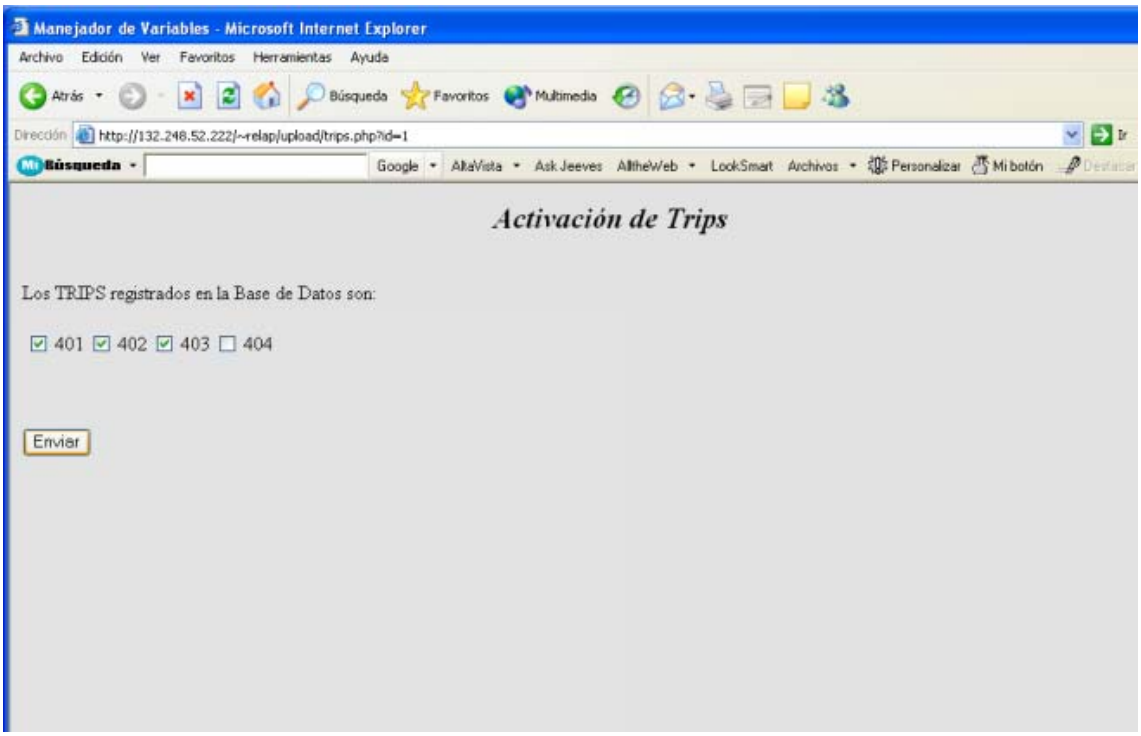

Figura 5.1.5 Página trips.php

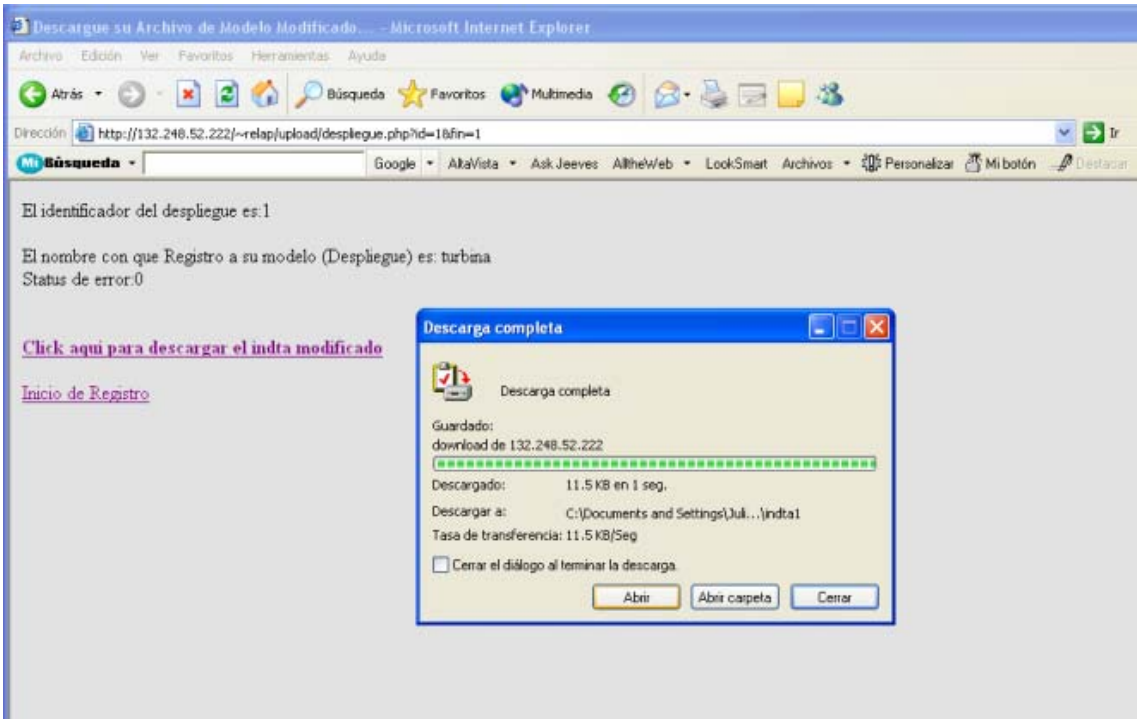

Figura 5.1.6 Página despliegue.php

El registro completo de un modelo de entrada al SADROV se muestra en las figuras 5.1.1 a la figura 5.1.6. Los pasos a seguir para registrar el modelo son los siguientes:

1. Conectarse a la Interfaz Gráfica del SADROV.

2. Dar un nombre de Despliegue al archivo de entrada de modelo de los códigos RELAP5.

3. Enviar el archivo de modelo a simular.

4. El MVRELAP mostrará las variables de control encontradas en el modelo y registradas en la base de datos, las variables seleccionadas se convertirán en “variables de despliegue activas”.

5. EL MVRELAP mostrará las variables de control constantes encontradas en el modelo y registradas en la base de datos, las variables seleccionadas se convertirán en “variables constantes activas”.

6. EL MVRELAP mostrará los trips encontrados en el modelo y registrados en la base de datos, los trips seleccionados se convertirán en “trips activos”.

7. EL MVRELAP dará un “link” para descargar el archivo de modelo de entrada modificado, listo para que la ICRELAP del Cliente RELAP5 lo reconozca.

5.2 Conclusiones

Con base en las pruebas que se le realizaron al SADROV, podemos llegar a las siguientes conclusiones:

- El proceso de registro de un modelo en el SADROV, es sencillo e intuitivo, sin embargo debe hacerse notar que la persona que registre un modelo de entrada, en la RELAP_DB, a simular en los Códigos RELAP5 debe tener un buen conocimiento en modelación, debido a que la forma de extracción de datos e interacción con los códigos es específica con “Variables de control” y “Trips”.

- De la tabla 5.1.2 se puede observar que los tiempos promedio de respuesta son bastante aceptables, sobre todo el que se obtuvo en el LAIRN. Viendo la tabla 5.1.1 se observa que la conexión a Internet en este caso es mucho mejor que la del “Sitio de Pruebas”. Lo que nos lleva a decir que este tiempo de respuesta depende en parte del tipo de conexión a Internet.

- De la tabla 5.1.2 podemos concluir que el SADROV cumple sus funciones en tiempos razonables por lo que es factible la migración de los "despliegues" del simulador de aula que actualmente utilizan memoria compartida, a "Clientes de Despliegue" que utilicen la ICRELAP del SADROV.

El SADROV provee las siguientes ventajas para el simulador de aula:

- Conectividad, a través de redes LAN (Local Area Network) y WAN (Wide Area Network).

- EL SADROV es un sistema multiusuario, lo que permitirá el tener funcionando en un mismo tiempo varios despliegues del simulador de aula (ya sean iguales o diferentes), en diferentes máquinas[6] y en diferentes lugares, conectadas al servidor que contiene a la RELAP_DB.

- Guardar información histórica (simulaciones completas), evitando el tener funcionando algún Cliente RELAP5. Así, simplemente, se llama a la BD cuando se quiera visualizar un análisis de alguna simulación previamente realizada. Esto trae como ventaja que no tiene que ser exactamente en tiempo real, se puede manejar la velocidad del despliegue de los datos de interés[7].

- El SADROV es un sistema robusto, ya que la RELAP_DB provee una forma de organización de las variables, por lo que todos los Clientes de Despliegue seguirán el mismo método de obtención de las mismas.

5.3 Trabajos Futuros

Durante el desarrollo del SADROV se distinguieron dos grandes áreas donde se puede aprovechar mucho el uso de este sistema:

- Realizar "Clientes de Despliegue" para el simulador de aula, hechos con gráficos usando DATAVIEWS compatibles con la RELAP_DB que utilicen el SADROV.

- Ampliar la Interfaz Gráfica del SADROV para facilitar el desarrollo de nuevos despliegues para el simulador de aula. Ya sea con modelos ya hechos o desarrollados en la misma Interfaz Gráfica del SADROV.

[6] Ya sea en una misma locación física o en diferentes locaciones.
[7] Ya sea más rápido o más lento.

REFERENCIAS

[A1]. Álvarez, Miguel Ángel. “Breve historia del PHP” [en línea]. <http://www.desarrolloweb.com/articulos/436.php?manual=12> [Consulta: diciembre 2003].

[A2]. Álvarez, Rubén. “Tareas principales del PHP” [en línea]. <http://www.desarrolloweb.com/articulos/304.php?manual=12> [Consulta: diciembre 2003].

[D1].Dubois, Paul. “Edición Especial de MySQL”. .Prentice Hall, Madrid 2001

[F1]. Fernández Sarti, Marian. “Áreas de Potencial Desarrollo de los simuladores de Entrenamiento en la Industria Nuclear” [en línea]. <http://www.ct3.es/areas_pot.html > [Consulta: diciembre 2003].

[F2]. “Tutorial de Fortran” [en línea]. <http://fismat.umich.mx/mn1/tutor_fort/preface.html> [Consulta: diciembre 2003].

[G1].Gardarin, Georges. “Bases de Datos”. Editorial PARAINFO. Segunda Edición, Madrid, 1990.

[G2]. Gilbert; Troitzsch. “Simulation for the Social Scientist”. Open University Press, 1999.

[G3].Gordo Saez, Roberto. “El Funcionamiento de Internet” [en línea]. 1998 <http://www.geocities.com/SiliconValley/Bay/8259/contenido.html > [Consulta: diciembre 2003].

[G4].Greenspan, Jay; Bulger, Brad. “MySQL/PHP Database aplications”. M&T Books, 2001, P. XXIII – XXIX.

[G5].Groff, R. “Using SQL”. Mc Graw Hill, EU, 1990.

[I1]. INEL. “RELAP5/MOD3 code manual, Volume I: Code Structure, System Models, and Solution Methods, p. XV.

[I2].INEL. “RELAP5/MOD3 code manual, Appendix A: Data Deck Organization and Data Card Requirements”, p. A1-1, A1-2.

[I3]. Ídem, p. A2-3.

[I4]. Ídem, p. A4-1 – A1-3.

[I5]. Ídem, p. A5-1 - A5-2.

[I6]. Ídem, p. A7-1 – A7-7.

[I7]. Ídem, p. A7-11 – A7-16.

[I8]. Ídem, p. A7-27 – A7-28.

[I9]. Ídem, p. A7-41 – A7-43.

[I10]. Ídem, p. A8-1.

[I11]. Ídem, p. A7-49, A7-60, A7-64 - A7-65.

[I12]. Ídem, p. A14-1, A14-2.

[M1]. Miravet Bonet, J. “Protocolos TCP/IP” [en línea]. 1999 <http://www4.uji.es/~al019803/Tcpip.htm > [Consulta: diciembre 2003].

[M2].“MySQL Reference Manual for version 4.0.5”. Disponible en Internet: MySQL <http://www.mysql.com> Enero 2004.

[P1]. Pablo, Federico. “Simulación en Economía” [en línea]. 2002 <http://www.economiaindustrial.com/simulacion.htm > [Consulta: diciembre 2003].

[S1]. Salazar Cravioto, José Humberto. "Incorporación de los Códigos RELAP/SCDAP al Simulador de Procesos Nucleares para Análisis y Entrenamiento en Aula". Tesis de Licenciatura. Facultad de Ingeniería UNAM. 2002

[S2]. “La Suite de Protocolos TCP/IP” [en línea]. <http://minter.cieamer.conacyt.mx/internet/tcp_ip.html > [Consulta: diciembre 2003].

ACRÓNIMOS Y GLOSARIO

En este apartado se dan algunas definiciones de términos usados en el presente trabajo.

A

Administrador de Bases de Datos: Ver Manejador de Bases de Datos.

ADSL: Asymmetric Digital Subscriber Line. Tecnología que permite el uso de una línea de cobre para transmisión de datos de alta velocidad y, a la vez, para el uso normal como línea telefónica.

Apache: Servidor Web, aplicación que es responsable de la comunicación del servidor con el navegador del cliente.

Archivo contint.dat: Archivo que contiene información sobre variables de despliegue activas que permiten interactividad con los códigos.

Archivo indta: Archivo de entrada de modelo para los códigos RELAP5. Ver inputdeck.

Archivo tripint.dat: Archivo que contiene información sobre los trips activos en el despliegue.

Archivo vardesc.dat: Archivo que contiene información sobre las variables de despliegue activas.

Archivo outdta: Archivo que contiene la salida de los códigos RELAP5.

B

Base T, 10: Estándar de transmisión de Ethernet sobre cable de par trenzado a 10 Mbps.

BD: Base de Datos.

C

Cliente de Despliegue: Aplicación que se conecta a la RELAP_DB para obtener información de los códigos RELAP5.

Cliente de Interacción: Aplicación que se conecta a la RELAP_DB para interactuar con los códigos RELAP5 mediante trips y variables de control.

Cliente RELAP5: Computadora corriendo los códigos RELAP5 con el ICRELAP implementado.

CONACYT: Consejo Nacional de Ciencia y Tecnología.

Cookies: Galletitas, porción de información que es mandada desde el servidor hacia nuestra máquina para ser almacenada durante la gestión de una página Web.

D

Database: Base de datos.

Datagrama: También llamado "paquete" unidad de información básica en una comunicación a través del Internet.

DEPFI: División de Estudios de Posgrado de la Facultad de Ingeniería.

Despliegue: Término que engloba a los despliegues gráficos, mímicos y representaciones virtuales de tableros que pueden ser utilizados para el simulador de aula.

E

Ethernet: Estándar de red más popular e implementado. Utiliza CSMA/CD con una velocidad de 10 Mbps.

F

FORTRAN: Lenguaje de programación de alto nivel para propósitos matemáticos y científicos, el acrónimo viene de FORmula TRANslator.

H

Hardware: Componentes electrónicos, equipos periféricos y otros componentes de un sistema de computadoras.

Host: Servidor con funciones centralizadas que hace disponibles programas a otras computadoras.

HTML: Hypertext Markup Language. Lenguaje que provee formato de texto especial usado por los navegadores para visualizar páginas Web.

I

ICRELAP: Interfaz de Comunicación para los Códigos RELAP5.

INEL: Idaho National Engineering Laboratory.

Inputdeck: Nombre dado a la entrada de los códigos RELAP5, o archivo de tarjetas de entrada.

Inputs: Datos de entrada de los códigos de simulación.

IP: Internet Protocol o protocolo de Internet.

L

LAIRN: Laboratorio de Análisis en Ingeniería de Reactores Nucleares. Jiutepec, Mor.

LAPE: Laboratorio de Proyectos Especiales. Ciudad Universitaria, México DF.

Latched: Cerrado.

LDRELAP: Lector de Datos de la Interfaz Gráfica del SADROV.

LINUX: Sistema operativo, con licencia de software libre.

LWR: Light Water Reactor o reactor de agua ligera.

M

Manejador de Bases de Datos o Administrador de Bases de Datos: Conjunto de programas y dispositivos que permiten organizar y estructurar datos para que puedan ser manipulados por usuarios y programas.

Memoria Compartida: Junto con los semáforos y las colas de mensajes, son los recursos compartidos que pone UNIX a disposición de los programas para que puedan intercambiarse información.

Middleware: Lenguaje para mandar las peticiones entre el Servidor Web y el Servidor de Base de Datos, también puede ser usado para realizar tareas sobre la información.

Minor edit: Componente que permite la impresión de una cierta variable física en el archivo de salida de los códigos RELAP5.

Modelador: Se refiere a la persona que define el modelo físico a simular.

MVRELAP: Manejador de Variables de la Interfaz Gráfica del SADROV.

O

ODBC: Open Database Connectivity, Conectividad de Base de Datos.

OIEA: Organismo Internacional de Energía Atómica.

Ouputs: Datos de salida de los códigos de simulación.

P

PAPIIT: Programa de Apoyo para Proyectos de Investigación e Innovación Tecnológica.

PC: Personal Computer o Computadora Personal.

POO: Programación Orientada a Objetos.

Q

Query: Petición, en este caso se refiere a petición de alguna acción al SGBD.

R

RDBMS: Relational Data Base Manage System o Manejador de Bases de Datos Relacionales.

Router: Dispositivo que permite la conexión entre redes encargándose además de que los paquetes en que se divide la información llegue a su destino. Coloquialmente llamado “enrutador”.

S

SADROV: Sistema de Adquisición de Datos y Registro de Variables.

SCRIPT: Conjunto de instrucciones que se guardan en un fichero y que son interpretadas tal y como están por un programa.

Servidor LAMP: Servidor que tiene LINUX como sistema operativo, Apache como servidor Web y PHP como lenguaje del lado del servidor para páginas dinámicas.

SGBD: Sistema Gestor de Bases de Datos, sinónimo de Manejador de Bases de Datos.

SI: Sistema Internacional.

SO: Sistema Operativo.

Socket: Punto final de un enlace de comunicación de dos vías entre dos programas que se ejecutan a través de la red.

Software: Colección de códigos de programación y datos de un sistema de computadoras.

SQL: Structured Query Language o Lenguaje de búsqueda estructurado para bases de datos relacionales.

T

Tag: Etiqueta.

Tarjeta: Componente básico con que se crean los modelos de entrada para los códigos RELAP5.

TCP: Transport Control Protocol o Protocolo de Control de Transporte.

Trip: Disparo, termino usado para definir un componente termodinámico de los códigos RELAP5 que define el disparo o cambio de valor de una variable mediante una condición lógica o por tiempo.

U

UDP: User Datagram Protocol.

UNIX: Estándar para crear Sistemas Operativos.

Unlatched: Abierto.

URL: Uniform Resource Location. Es la dirección estandarizada y detallada de páginas Web.

V

Variable de control: Componente de los códigos RELAP5 que permite implementar un sistema de control en el modelo de entrada. Usada para intercambiar datos con la base de datos.

Variable de despliegue: Variable de control registrada en la RELAP_DB, sirve para mostrar datos provenientes de los códigos RELAP5, resultado de alguna simulación.

W

Web: Red, sinónimo de Internet.

Window: Ventana.

Windows: Sistema Operativo

Word: Palabra.

APÉNDICE A

En este apéndice se encuentran los diversos archivos utilizados para realizar las pruebas del SADROV, así como las corridas de los Clientes de Despliegue y de Interacción usados para realizar las mismas.

A.1 Archivo de Modelo

Las palabras letra cursiva indican la parte del modelo procesado por el SADROV.

```
= turbina LV(rdb)1(rdb)132.248.52.222(rdb)turbina
*------------------------------------------------------------------
*------------------------------------------------------------------
*
0000100   new    transnt
0000101   run
0000102   british   si
*0000105   020.     035.
*------------------------------------------------------------------
*crdno    noncond 1     noncond 2    noncond 3    noncond 4    noncond 5
0000110   hydrogen      helium       xenon        krypton      air  *quitar en rst
0000115   0.0           0.0          0.0          0.0          1.0   *quitar en rst
*------------------------------------------------------------------
*0000120   151000000  0.0       h2o  "reactor"
*------------------------------------------------------------------
*card     time.end  min       max       control   minor   major   rstplt
0000201      200.e0  1.0e-8    0.00375       3        100      1000   90000
*0000202      20.   1.0e-8    0.001875      3        100      1000   90000
*------------------------------------------------------------------
*
20589900  flecha shaft   1.0  1800.0  0  0
20589901  0  1000000.0  10000.0 generatr  899  turbine  900
20589902  turbine   901  turbine   902
20589903  turbine   903  turbine  904
20589904  turbine   905  turbine  906
20589905  turbine   907  turbine  908
20589906  1800.0  1800.0  1.0e7  1.e3 401 402
*
*TRIP DATA
*
401 time 0 gt    null  0  1.e20  n    *trip on time red sincrona
402 time 0 gt    null  0  1.e20  n    *trip on time desacoplado generador - turbina
403 time 0 gt    null  0  1.e20  n    *trip on time disparo de turbina
404 cntrlvar 25 gt null 0 1850.  l
601 -403 and -404 n 0.
*
*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*
*  Minor Edit Requests
*
0000301   mflowj    900010000                    *junction flow
0000302   mflowj    901010000                    *junction flow
0000303   mflowj    901020000                    *junction flow
0000304   mflowj    902010000                    *junction flow
```

```
0000305  mflowj   902020000          *junction flow
0000306  mflowj   903010000          *junction flow
0000307  mflowj   903020000          *junction flow
0000308  mflowj   904010000          *junction flow
0000309  mflowj   904020000          *junction flow
0000310  mflowj   905010000          *junction flow
0000311  mflowj   905020000          *junction flow
0000312  mflowj   906010000          *junction flow
0000313  mflowj   906020000          *junction flow
0000314  mflowj   907010000          *junction flow
0000315  mflowj   907020000          *junction flow
0000316  mflowj   908010000          *junction flow
0000317  cntrlvar 25                 *vel angular
0000318  mflowj   152010000          *junction flow
0000319  mflowj   152020000          *junction flow
0000320  cntrlvar 1                  *vol pressure
0000321  cntrlvar 2                  *vol Pressure
0000322  cntrlvar 3                  *vol Pressure
0000323  cntrlvar 4                  *vol Pressure
0000324  cntrlvar 5                  *vol Pressure
0000325  cntrlvar 6                  *vol pressure
0000326  cntrlvar 7                  *vol pressure
0000327  cntrlvar 8                  *vol pressure
0000328  cntrlvar 9                  *vol pressure
0000329  cntrlvar 10                 *vol pressure
0000330  cntrlvar 11                 *vol pressure
0000331  cntrlvar 12                 *vol pressure
0000332  cntrlvar 13                 *tot pow
0000333  dt       0
0000334  dtcrnt   0
0000335  emass    0
0000336  errmax   0
*  steam source
1500000  linvapor tmdpvol
1500101  8.20      100.      820.     0.    0.0    0.
1500102  0.0       0.0       00
1500200  2
1500201  0.       950.   1.0
1500202  10.      950.    1.0
1500203  1000.0   950.   1.0
*
1560000  tsvalve  valve
1560101  150010002  155010001 8.2  0.0  0.0   120
1560201  1   2246.3 0.0
1560300  trpvlv
1560301  601
*
```

```
1550000   stmline1  snglvol
1550101    8.2 100. 0. 0. 0. 0. 0. 0. 11010
1550200    102 950. 1.0
*
1540000   cvalve   valve
1540101   155010002  153010001 8.2  0.0  0.0   120
1540201   1   2246.3 0.0
1540300   srvvlv
1540301   951
*
1530000   stmline2  snglvol
1530101    8.2 100. 0. 0. 0. 0. 0. 0. 11010
1530200    102 950. 1.0
*  turbINA FICTICIA
9000000   turbF  turbine
9000001   1  1
9000101   8.20  100.  820.  0.0  0.0  0.0  0.0  0.0  11010
9000200   102  950.  1.0  * pres (psia) calidad en fracción
9001101   153010002  900010001  8.20  0.0 0.0  1020
9001201   0.0  2246.3   0.0
9000300   1800.0  1000.0   0.0  899  0 0
9000400   2    0.0  0.5   1.0
* PRIMERA turbINA REAL
9010000   turb1  turbine
9010001   2  1
9010101   2.1  100.  210.  0.0  0.0  0.0  0.0  0.0  11010
9010200   102  473.9  1.0  * pres (psia) calidad en fracción
9011101   900010002  901010001  2.69 0.0 0.0 1020
9012101   901010002  152000000  0.528  1.815e+02 1.815e+02 0023
9011201   0.0  2246.3  0.0
9012201   0.0  82.94  0.0
9012112   0. 0. 0. 0.
9010300   1800.0 1.e7 100.0  899  0  0
9010400   2 0.90 0.5 1.
* SEGUNDA turbINA REAL
9020000   turb2  turbine
9020001   2  1
9020101   20.14  10.0  201.4  0.0  0.0  0.0  0.0  0.0 11010
9020200   102  352.1  1.0
9021101   901010002  902010001  4.21  0.0 0.0 1020
9022101   902010002  152000000  0.528  4.158e+01   4.158e+01 1023
9021201   0.0  2163.36  0.0
9022201   0.0  128.8  0.0
9022112   0. 0. 0. 0.
9020300   1800.0  1.e7  100.0  899  0  0
9020400   2  0.90  0.5  1.
* TERCER turbINA REAL
```

```
9030000   turb3  turbine
9030001   2  1
9030101   8.5  100.  850. 0.0  0.0  0.0  0.0  0.0  11010
9030200   102  198.7  1.0  * pres (psia) calidad en fracción
9031101   902010002  903010001  4.13 0.0  0.0  1020
9032101   903010002  152000000  2.113  2.1784e+1    2.1784e+1     1023
9031201   0.0  2034.56    0.0
9032201   0.0  404.  0.0
9032112   0. 0. 0. 0.
9030300   1800.0  1.e7 100.0  899  0  0
9030400   2 0.90 0.5  1.0
* CUARTA turbINA REAL
9040000   turb4  turbine
9040001   2  1
9040101   15.  100. 1500.  0.0  0.0  0.0  0.0  0.0  11010
9040200   102  89.49  1.0  * pres (psia) calidad en fracción
9041101   903010002  904010001  4.96  1.694e-1 1.694e-1   1020
9042101   904010002  152000000  2.113  6.5119e+1  6.5119e+1  1023
9041201   0.0  1630.56  0.0
9042201   0.0  107.0  0.0
9040300   1800.0  1.e7 100.0  899  0  0
9042112   0. 0. 0. 0.
9040400   2 0.90 0.5  1.0
* QUINTA turbINA REAL
9050000   turb5  turbine
9050001   2  1
9050101   32.  10.  320.  0.0  0.0  0.0  0.0  0.0  11010
9050200   102  36.21  1.0  * pres (psia) calidad en fracción
9051101   904010002  905010001  8.12 0.0  0.0  1020
9052101   905010002  152000000  4.753 7.387e+1  7.387e+1  1023
9051201   0.0  1523.56  0.0
9052201   0.0  93.01  0.0
9052112   0. 0. 0. 0.
9050300   1800.0  1.e7 100.0  899  0  0
9050400   2 0.90 0.5  1.0
* SEXTA turbINA REAL
9060000   turb6  turbine
9060001   2  1
9060101   72.  10.  720.   0.0  0.0  0.0  0.0  0.0  11010
9060200   102  13.53  1.0  * pres (psia) calidad en fracción
9061101   905010002  906010001  15.91  0.0  0.0  1020
9062101   906010002  152010001  8.45  64.68   64.68  1023
9061201   0.0  1430.55  0.0
9062201   0.0  66.69  0.0
9062112   0. 0. 0. 0.
9060300   1800.0  1.e7 100.0  899  0  0
9060400   2 0.90 0.5  1.0
```

```
* SEPTIMA turbINA REAL
9070000   turb7  turbine
9070001   2  1
9070101   150.  10.  1500.  0.0  0.0  0.0  0.0  0.0  11010
9070200   102  5.82  1.0  * pres (psia) calidad en fracción
9071101   906010002  907010001  34.88   0.0  0.0  1020
9072101   907010002  152000000  19.011  42.43   42.43   1023
9071201   0.0  1363.86  0.0
9072201   0.0  77.40  0.0
9072112   0. 0. 0. 0.
9070300   1800.0  1.e7 100.0  899  0  0
9070400   2 0.90 0.5  1.0
* OCTAVA turbINA REAL
9080000   turb8  turbine
9080001   1  1
9080101   200. 100.  20000.  0.0  0.0  0.0  0.0  0.0  11010
9080200   102  2.742  1.0  * pres (psia) calidad en fracción
9081101   907010002  908010001   68.14 0.0 0.0  1020
9081201   0.0  1286.46    0.0
9080300   1800.0  1.e7 100.0  899  0  0
9080400   2 0.90 0.5  1.0
* salida principal a condensador turbina 8
1522101   908010002  152000000  465.3  16.58  16.58 1020
1522201   0.0  1286.46 0.0
* branch del sumidero
1520000   condjun  branch
1520001   2    1
1520101   10000.  100.  1000000.  0.0  0.0  0.0  0.0 0.0 11010
1520200   102   0.98   1.0
1521101   152010002  151000000  645.86  4.657e-2 4.657e-2 1020
1521201   0.0 2246.34  0.0
*  sumidero
1510000   condens tmdpvol
1510101   1.e6      0.       1.e6      0.    0.0     0.
1510102   0.0       0.0       00
1510200   2
1510201   0.       0.65   1.0
1510202   1000.    0.65   1.0
*
20500100   pfv   sum  1.  0.0   1
20500101   0.0 1.e-6  p 150010000
20500200   ptf   sum  1.  0.0   1
20500201   0.0 1.e-6  p 900010000
20500300   pt1   sum  1.  0.0   1
20500301   0.0 1.e-6  p 901010000
20500400   pt2   sum  1.  0.0   1
20500401   0.0 1.e-6  p 902010000
```

```
20500500  pt3   sum  1.  0.0   1
20500501   0.0 1.e-6  p 903010000
20500600  pt4   sum  1.  0.0   1
20500601   0.0 1.e-6  p 904010000
20500700  pt5   sum  1.  0.0   1
20500701   0.0 1.e-6  p 905010000
20500800  pt6   sum  1.  0.0   1
20500801   0.0 1.e-6  p 906010000
20500900  pt7   sum  1.  0.0   1
20500901   0.0 1.e-6  p 907010000
20501000  pt8   sum  1.  0.0   1
20501001   0.0 1.e-6  p 908010000
20501100  pbs   sum  1.  0.0   1
20501101   0.0 1.e-6  p 152010000
20501200  ps   sum  1.  0.0   1
20501201   0.0 1.e-6  p 151010000
20501300  pe   sum  1.  0.0   1
20501301   0.0  1.e-6 turpow 901 1.e-6 turpow 902
20501302       1.e-6 turpow 903 1.e-6 turpow 904
20501303       1.e-6 turpow 905 1.e-6 turpow 906
20501304       1.e-6 turpow 907 1.e-6 turpow 908
20501400  wfv  sum  1.  1021.038 1
20501401  0. 1. mflowj 901010000
20501500  wst1 sum 1. 37.621 1
20501501  0. 1. mflowj 901020000
20501600  wst2 sum 1. 58.423 1
20501601  0. 1. mflowj 902020000
20501700  wst3 sum 1. 63.321 1
20501701  0. 1. mflowj 903020000
20501800  wst4 sum 1. 48.534 1
20501801  0. 1. mflowj 904020000
20501900  wst5 sum 1. 42.184 1
20501901  0. 1. mflowj 905020000
20502000  wst6 sum 1. 30.255 1
20502001  0. 1. mflowj 906020000
20502100  wst7 sum 1. 35.108 1
20502101  0. 1. mflowj 907020000
20502200  wst8 sum 1. 583.593 1
20502201  0. 1. mflowj 152020000
20502300  wtapbp sum 1. 739.674 1
20502301  0. 1. mflowj 904010000
20502400  wbss sum 1. 899.04 1
20502401  0. 1. mflowj 152010000
20502500  vang sum 1. 1800. 1
20502501  0. 9.5493 cntrlvar 899
20595000  posvc constant 100.
20595200  posvc1 constant 1.
```

```
20595100  valvcf   function  1.0  1.0  1  3   0.0  1.0
20595101 cntrlvar  950   2
20200200  reac-t
20200201  0.0    0.0
20200202  10.    0.013
20200203  20.    0.027
20200204  30.    0.040
20200205  40.    0.053
20200206  50.    0.070
20200207  60.    0.092
20200208  70.    0.131
20200209  80.    0.175
20200210  90.    0.256
20200211  100.   1.
**********************************
**                              **
**  RELAP DB                    **
** variables de despliegue      **
**                              **
**********************************
.end
```

A.2 Archivo contint.dat

El archivo generado por el Cliente RELAP5 para las pruebas del SADROV.

```
vconstantes  CMC   const_inival
950     260     100
```

A.3 Archivo tripint.dat

El archivo generado por el Cliente RELAP5 para las pruebas del SADROV.

```
trips    CMC   DBCode
401     250     0       1_trip
402     251     0       1_trip
403     252     0       1_trip
```

A.4 Archivo varcdes.dat

El archivo generado por el Cliente RELAP5 para las pruebas del SADROV.

```
var_control   CMC   DBCode
1       200    157_tur_001
20      201    157_tur_020
21      202    157_tur_021
22      203    157_tur_022
23      204    157_tur_023
24      205    157_tur_024
25      206    157_tur_025
```

A.5 Corridas de los Clientes en el LAIRN[1]

El Cliente de de Despliegue arrojó los siguientes resultados:

```
Iniciando una conexion al servidor!!!
Cliente de MySQl usado:0x00401A72
Host:132.248.52.222 via TCP/IP
Servidor:4.0.16-log
Estatus:
Uptime: 308233  Threads: 2  Questions: 50931  Slow queries: 0  Opens: 36  Flush
tables: 1  Open tables: 30  Queries per second avg: 0.046
Tarjetas de control 205CCC0
Las variables de control (despliegue) encontradas son:
1: 2050010
2: 2050200
3: 2050210
4: 2050220
5: 2050230
6: 2050240
7: 2050250
teclea el numero para desplegar la variable 1-7
Num:6
Otro y/n?
y
Num:7
Otro y/n?
n
1       1018.92 1800          31 ms.
2       932.82  1800          32 ms.
3       1000.39 1800          31 ms.
4       1026.18 1800          31 ms.
5       1035.55 1800          31 ms.
```

[1] Jiutepec, Mor.

6	1039.45	1800	32 ms.
7	1041.06	1800	31 ms.
8	1041.96	1800	31 ms.
9	1042.62	1800	16 ms.
10	1043.09	1800	78 ms.
11	1043.4	1800	15 ms.
12	1043.58	1800	32 ms.
13	1043.66	1800	31 ms.
14	1043.69	1800	31 ms.
15	1043.68	1800	31 ms.
16	1043.67	1800	32 ms.
17	1043.66	1800	31 ms.
18	1043.66	1800	31 ms.
19	1043.66	1800	31 ms.
20	1043.66	1800	31 ms.
21	1043.66	1800	47 ms.
22	1043.66	1800	16 ms.
23	1043.66	1800	31 ms.
24	1043.66	1800	31 ms.
25	1043.66	1800	47 ms.
26	1043.66	1800	47 ms.
27	1043.66	1800	47 ms.
28	1043.66	1800	31 ms.
29	1043.66	1800	31 ms.
30	1043.66	1800	31 ms.
31	1043.66	1800	31 ms.
32	1043.66	1800	15 ms.
33	1043.66	1800	32 ms.
34	1043.66	1800	31 ms.
35	1043.66	1800	31 ms.
36	1043.66	1800	31 ms.
37	1043.66	1800	31 ms.
38	1043.66	1800	47 ms.
39	1043.66	1800	32 ms.
40	1043.66	1800	32 ms.
41	1043.66	1800	31 ms.
42	1043.66	1800	31 ms.
43	1043.66	1800	47 ms.
44	1043.66	1800	47 ms.
45	1043.66	1800	15 ms.
46	1043.66	1800	31 ms.
47	1043.66	1800	79 ms.
48	1043.66	1800	32 ms.
49	1043.66	1800	31 ms.
50	859.404	1800	31 ms.
51	597.79	1800	62 ms.
52	414.57	1800	31 ms.

53	287.772	1800	63 ms.
54	202.666	1800	31 ms.
55	142.842	1800	31 ms.
56	101.19	1800	31 ms.
57	72.5709	1800	31 ms.
58	52.4807	1800	31 ms.
59	38.2273	1800	32 ms.
60	27.7703	1800	31 ms.
61	20.3507	1800	31 ms.
62	15.1458	1800	31 ms.
63	11.2372	1800	62 ms.
64	8.14948	1800	31 ms.
65	5.87332	1800	31 ms.
66	4.20964	1800	31 ms.
67	3.13239	1800	32 ms.
68	2.28921	1800	31 ms.
69	1.59674	1800	31 ms.
70	1.02895	1800	32 ms.
71	0.57677	1800	32 ms.
72	0.344803	1800	31 ms.
73	0.202498	1800	31 ms.
74	0.019049	1800	32 ms.
75	2466.34	1800	47 ms.
76	1383.04	1800	31 ms.
77	1156.32	1800	32 ms.
78	1078.79	1800	31 ms.
79	1046.68	1800	31 ms.
80	1035.33	1800	94 ms.
81	1035.52	1800	15 ms.
82	1036.79	1800	31 ms.
83	1039.37	1800	16 ms.
84	1041.68	1800	31 ms.
85	1042.92	1800	31 ms.
86	1043.64	1800	31 ms.
87	1043.93	1800	32 ms.
88	1043.92	1800	32 ms.
89	1043.83	1800	31 ms.
90	1043.74	1800	16 ms.
91	1043.68	1800	31 ms.
92	1043.64	1800	31 ms.
93	1043.64	1800	31 ms.
94	1043.64	1800	31 ms.
95	1043.65	1800	47 ms.
96	1043.66	1800	31 ms.
97	1043.66	1800	32 ms.
98	1043.66	1800	31 ms.
99	1043.66	1800	31 ms.

100	1043.66	1800	31 ms.
101	1043.66	1800	32 ms.
102	1043.66	1800	32 ms.
103	1043.66	1800	46 ms.
104	1043.66	1800	32 ms.
105	1043.66	1800	16 ms.
106	1043.66	1800	31 ms.
107	1043.66	1800	31 ms.
108	1043.66	1800	31 ms.
109	1043.66	1800	31 ms.
110	1043.66	1800	31 ms.
111	1043.66	1800	32 ms.
112	1043.66	1800	32 ms.
113	1043.66	1800	31 ms.
114	1043.66	1800	31 ms.
115	1043.66	1800	31 ms.
116	1043.66	1800	31 ms.
117	1043.66	1800	31 ms.
118	1043.66	1800	31 ms.
119	1043.66	1800	47 ms.
120	1043.66	1800	31 ms.
121	1043.66	1800	16 ms.
122	1043.66	1800	47 ms.
123	1043.66	1800	31 ms.
124	1043.66	1800	15 ms.
125	1043.66	1800	16 ms.
126	1043.66	1800	31 ms.
127	1043.66	1800	31 ms.
128	1043.66	1800	31 ms.
129	1043.66	1800	16 ms.
130	1043.66	1800	32 ms.
131	851.475	1800	31 ms.
132	592.083	1800	32 ms.
133	417.864	1800	32 ms.
134	290.928	1800	31 ms.
135	203.915	1800	31 ms.
136	143.491	1800	31 ms.
137	101.543	1800	47 ms.
138	73.36	1800	15 ms.
139	52.8648	1800	31 ms.
140	38.5486	1800	47 ms.
141	28.0234	1800	31 ms.
142	20.6556	1800	31 ms.
143	15.221	1800	15 ms.
144	11.2763	1800	31 ms.
145	8.17555	1800	47 ms.
146	5.99833	1800	31 ms.

147 4.27863 1800 47 ms.
148 3.11435 1800 32 ms.
149 2.26294 1800 47 ms.
150 1.57134 1800 31 ms.
151 1.0271 1800 31 ms.
152 0.571835 1800 16 ms.
153 0.340839 1800 31 ms.
154 0.193783 1800 31 ms.
155 0.019918 1800 31 ms.
156 -0.158934 1800 31 ms.
157 -0.2919 1800 31 ms.
158 -0.255493 1800 31 ms.
159 -0.143555 1800 16 ms.
160 -0.01658 1800 31 ms.
161 2226.83 1800 15 ms.
162 1331.8 1800 15 ms.
163 1126.09 1800 32 ms.
164 1063.74 1800 16 ms.
165 1036.91 1800 31 ms.
166 1030.56 1800 32 ms.
167 1032.77 1800 16 ms.
168 1036.88 1800 31 ms.
169 1040.35 1800 31 ms.
170 1042.74 1800 16 ms.
171 1043.84 1800 31 ms.
172 1044.15 1800 32 ms.
173 1044.05 1800 32 ms.
174 1043.9 1800 32 ms.
175 1043.75 1800 31 ms.
176 1043.65 1800 47 ms.
177 1043.63 1800 16 ms.
178 1043.63 1800 47 ms.
179 1043.63 1800 46 ms.
180 1043.65 1800 32 ms.
181 1043.66 1800 31 ms.
182 1043.66 1800 31 ms.
183 1043.66 1800 15 ms.
184 1043.66 1800 31 ms.
185 1043.66 1800 62 ms.
186 1043.66 1800 31 ms.
187 1043.66 1800 32 ms.
188 1043.66 1800 31 ms.
189 1043.66 1800 31 ms.
190 1043.66 1800 31 ms.
191 1043.66 1800 31 ms.
192 1043.66 1800 31 ms.
193 1043.66 1800 63 ms.

194 1043.66 1800 32 ms.
195 1043.66 1800 47 ms.
196 1043.66 1800 31 ms.
197 1043.66 1800 31 ms.
198 1043.66 1800 31 ms.
199 1043.66 1800 31 ms.

EL tiempo promedio es: 32.41 ms.
Press any key to continue

El Cliente de Interacción arrojó los siguientes resultados:

Trips Activos encontrados:
1 =>401 value : 0
2 =>402 value : 0
3 =>403 value : 0
Variables Constantes encontrados:
4 =>950 value : 100
teclea el numero para Cambiar el valor actual (1-4)
teclea un 0 para terminar...
Opcion: 3
UPDATE 1_trip SET 403_trip=1

Trip 403 val :1
Opcion: 3
UPDATE 1_trip SET 403_trip=0

Trip 403 val :0
Opcion: 3
UPDATE 1_trip SET 403_trip=1

Trip 403 val :1
Opcion: 3
UPDATE 1_trip SET 403_trip=0

Trip 403 val :0
Opcion:

A.5 Corridas de los Clientes en el Lugar de Trabajo[2]

El Cliente de de Despliegue arrojó los siguientes resultados:

Iniciando una conexion al servidor!!!
Cliente de MySQl usado:0x00401A72
Host:132.248.52.222 via TCP/IP

[2] Lugar con conexión a Internet mediante Modem a 56kbps, México, D.F.

Servidor:4.0.16-log
Estatus:
Uptime: 395424 Threads: 1 Questions: 51299 Slow queries: 0 Opens: 36 Flush tables: 1 Open tables: 30 Queries per second avg: 0.130
Tarjetas de control 205CCC0
Las variables de control (despliegue) encontradas son:
1: 2050010
2: 2050200
3: 2050210
4: 2050220
5: 2050230
6: 2050240
7: 2050250
teclea el numero para desplegar la variable 1-7
Num:5
Otro y/n?
y
Num:6
Otro y/n?
y
Num:7
Otro y/n?
n

1	739.61	1018.92	1800	188 ms.
2	550.014	731.563	1800	203 ms.
3	453.574	609.069	1800	156 ms.
4	413.009	555.023	1800	172 ms.
5	395.908	534.313	1800	172 ms.
6	387.985	526.042	1800	156 ms.
7	384.337	522.792	1800	157 ms.
8	382.658	521.519	1800	157 ms.
9	381.895	521.036	1800	141 ms.
10	381.53	520.857	1800	156 ms.
11	381.364	520.806	1800	141 ms.
12	381.287	520.802	1800	141 ms.
13	381.252	520.812	1800	172 ms.
14	381.235	520.825	1800	140 ms.
15	381.228	520.836	1800	140 ms.
16	381.224	520.844	1800	156 ms.
17	381.223	520.85	1800	156 ms.
18	381.222	520.854	1800	156 ms.
19	381.222	520.857	1800	141 ms.
20	381.222	520.858	1800	141 ms.
21	381.222	520.86	1800	171 ms.
22	381.221	520.86	1800	141 ms.
23	381.221	520.861	1800	125 ms.
24	381.221	520.861	1800	156 ms.

25	381.221	520.861	1800	156 ms.
26	381.221	520.861	1800	140 ms.
27	381.221	520.862	1800	125 ms.
28	381.221	520.862	1800	141 ms.
29	381.221	520.862	1800	125 ms.
30	381.221	520.862	1800	125 ms.
31	381.221	520.862	1800	125 ms.
32	381.221	520.862	1800	141 ms.
33	381.221	520.862	1800	141 ms.
34	381.221	520.862	1800	140 ms.
35	381.221	520.862	1800	141 ms.
36	381.221	520.862	1800	125 ms.
37	381.221	520.862	1800	125 ms.
38	381.221	520.862	1800	125 ms.
39	381.221	520.862	1800	125 ms.
40	381.221	520.862	1800	140 ms.
41	381.221	520.862	1800	141 ms.
42	381.221	520.862	1800	141 ms.
43	381.221	520.862	1800	141 ms.
44	381.221	520.862	1800	157 ms.
45	381.221	520.862	1800	141 ms.
46	381.221	520.862	1800	172 ms.
47	381.221	520.862	1800	141 ms.
48	381.221	520.862	1800	156 ms.
49	381.221	520.862	1800	125 ms.
50	326.818	472.577	1800	125 ms.
51	235.983	348.374	1800	125 ms.
52	165.384	242.718	1800	140 ms.
53	114.734	166.255	1800	125 ms.
54	80.0676	114.365	1800	156 ms.
55	56.7072	80.1024	1800	141 ms.
56	40.1292	56.2089	1800	140 ms.
57	28.8048	40.1749	1800	140 ms.
58	20.7569	28.4615	1800	141 ms.
59	15.3258	20.7857	1800	125 ms.
60	11.2625	15.1692	1800	141 ms.
61	8.36737	10.7403	1800	140 ms.
62	6.28762	7.70401	1800	141 ms.
63	4.78657	5.46245	1800	125 ms.
64	3.70183	3.90865	1800	125 ms.
65	2.96756	2.73003	1800	125 ms.
66	2.50819	1.94235	1800	125 ms.
67	2.0686	1.31559	1800	125 ms.
68	1.64025	0.788418	1800	140 ms.
69	1.17206	0.420008	1800	141 ms.
70	0.734777	0.263415	1800	140 ms.
71	0.376736	0.084388	1800	141 ms.

72	0.070166	-0.102644	1800	141 ms.
73	-0.161169	-0.248186	1800	141 ms.
74	-0.206213	-0.213323	1800	140 ms.
75	-0.162873	-0.107885	1800	156 ms.
76	-0.087557	0.015217	1800	141 ms.
77	0.047147	0.161388	1800	141 ms.
78	0.124939	0.1526	1800	157 ms.
79	0.134066	0.087088	1800	141 ms.
80	0.08462	-0.01007	1800	157 ms.
81	0.005385	-0.069295	1800	172 ms.
82	-0.039203	-0.073907	1800	156 ms.
83	-0.04582	-0.037456	1800	125 ms.
84	-0.035583	0.011632	1800	140 ms.
85	-0.003529	0.055679	1800	125 ms.
86	0.022581	0.055903	1800	141 ms.
87	0.029556	0.023613	1800	125 ms.
88	0.022947	-0.01213	1800	125 ms.
89	0.004708	-0.031719	1800	140 ms.
90	-0.007695	-0.025792	1800	125 ms.
91	-0.010311	-0.006894	1800	125 ms.
92	-0.007199	0.01805	1800	125 ms.
93	0.001319	0.028742	1800	157 ms.
94	0.006981	0.018499	1800	141 ms.
95	0.007462	-0.000923	1800	140 ms.
96	0.004789	-0.01345	1800	156 ms.
97	-9.4e-05	-0.015358	1800	156 ms.
98	-0.002558	-0.006833	1800	140 ms.
99	-0.002268	0.005645	1800	125 ms.
100	-0.000718	0.014259	1800	156 ms.
101	0.001596	0.011214	1800	125 ms.
102	0.002684	0.002119	1800	141 ms.
103	0.002146	-0.005577	1800	172 ms.
104	0.001036	-0.008154	1800	141 ms.
105	-9.8e-05	-0.004403	1800	125 ms.
106	-0.000436	0.001521	1800	157 ms.
107	-4.7e-05	0.006607	1800	141 ms.
108	163.567	196.982	1800	172 ms.
109	258.844	340.102	1800	157 ms.
110	315.965	418.941	1800	141 ms.
111	350.445	467.316	1800	140 ms.
112	370.296	496.629	1800	156 ms.
113	381.902	514.671	1800	156 ms.
114	388.619	525.71	1800	141 ms.
115	392.679	532.812	1800	125 ms.
116	395.063	537.248	1800	140 ms.
117	396.449	539.986	1800	172 ms.
118	397.242	541.648	1800	156 ms.

119	397.715	542.699	1800	125 ms.
120	397.988	543.341	1800	141 ms.
121	398.146	543.736	1800	157 ms.
122	398.236	543.973	1800	125 ms.
123	398.289	544.123	1800	141 ms.
124	398.321	544.215	1800	687 ms.
125	398.339	544.272	1800	140 ms.
126	398.349	544.307	1800	141 ms.
127	398.355	544.328	1800	141 ms.
128	398.359	544.342	1800	125 ms.
129	398.361	544.35	1800	141 ms.
130	398.362	544.355	1800	141 ms.
131	398.362	544.358	1800	125 ms.
132	398.363	544.36	1800	141 ms.
133	398.363	544.361	1800	141 ms.
134	398.363	544.362	1800	140 ms.
135	398.363	544.362	1800	140 ms.
136	398.363	544.362	1800	172 ms.
137	398.363	544.362	1800	157 ms.
138	398.363	544.363	1800	141 ms.
139	398.363	544.363	1800	157 ms.
140	398.363	544.363	1800	141 ms.
141	398.363	544.363	1800	140 ms.
142	398.363	544.363	1800	188 ms.
143	398.363	544.363	1800	125 ms.
144	398.363	544.363	1800	125 ms.
145	398.363	544.363	1800	141 ms.
146	398.363	544.363	1800	125 ms.
147	398.363	544.363	1800	141 ms.
148	398.363	544.363	1800	157 ms.
149	398.363	544.363	1800	125 ms.
150	398.363	544.363	1800	125 ms.
151	398.363	544.363	1800	141 ms.
152	398.363	544.363	1800	141 ms.
153	398.363	544.363	1800	156 ms.
154	398.363	544.363	1800	141 ms.
155	398.363	544.363	1800	125 ms.
156	398.363	544.363	1800	156 ms.
157	398.363	544.363	1800	125 ms.
158	398.363	544.363	1800	125 ms.
159	398.363	544.363	1800	141 ms.
160	398.363	544.363	1800	140 ms.
161	398.363	544.363	1800	125 ms.
162	398.363	544.363	1800	125 ms.
163	398.363	544.363	1800	125 ms.
164	398.363	544.363	1800	125 ms.
165	398.363	544.363	1800	140 ms.

```
166    398.363 544.363 1800        141 ms.
167    398.363 544.363 1800        141 ms.
168    398.363 544.363 1800        125 ms.
169    398.363 544.363 1800        125 ms.
170    398.363 544.363 1800        125 ms.
171    398.363 544.363 1800        156 ms.
172    398.363 544.363 1800        125 ms.
173    398.363 544.363 1800        125 ms.
174    398.363 544.363 1800        125 ms.
175    398.363 544.363 1800        125 ms.
176    398.363 544.363 1800        125 ms.
177    398.363 544.363 1800        141 ms.
178    398.363 544.363 1800        141 ms.
179    398.363 544.363 1800        140 ms.
180    398.363 544.363 1800        125 ms.
181    398.363 544.363 1800        156 ms.
182    398.363 544.363 1800        140 ms.
183    398.363 544.363 1800        109 ms.
184    398.363 544.363 1800        141 ms.
185    398.363 544.363 1800        140 ms.
186    398.363 544.363 1800        141 ms.
187    398.363 544.363 1800        141 ms.
188    398.363 544.363 1800        140 ms.
189    398.363 544.363 1800        125 ms.
190    398.363 544.363 1800        125 ms.
191    398.363 544.363 1800        140 ms.
192    398.363 544.363 1800        125 ms.
193    398.363 544.363 1800        125 ms.
194    398.363 544.363 1800        141 ms.
195    398.363 544.363 1800        125 ms.
196    398.363 544.363 1800        141 ms.
197    398.363 544.363 1800        219 ms.
198    398.363 544.363 1800        140 ms.
199    398.363 544.363 1800        125 ms.

EL tiempo promedio es: 143.16 ms.
Press any key to continue
```

El Cliente de Interacción arrojó los siguientes resultados:

```
Trips Activos encontrados:
1 =>401 value : 0
2 =>402 value : 0
3 =>403 value : 0
Variables Constantes encontrados:
4 =>950 value : 100
teclea el numero para Cambiar el valor actual (1-4)
```

```
teclea un 0 para terminar...
Opcion: 3
UPDATE 1_trip SET 403_trip=1

Trip 403        val :1
Opcion: 4
Var const 950   val :100
value :90.50
UPDATE 1_vconst SET tur_950=90.500000

 Var const950   val :90.5
Opcion:
```

APÉNDICE B

En este apéndice se presenta el código fuente del Cliente de Despliegue de prueba. Básicamente se compone de tres archivos, dos de ellos contienen una clase implementada para manejar la conexión a la base de datos y las búsquedas, mientras que el tercero se refiere básicamente al programa en sí.

B.1 Clase conexion4

En el archivo conexion4.h se pueden observar todas las características de la clase, así como sus métodos.

conexion4.h

```
#define MI_CONEXION
#include <windows.h>
#include <iostream.h>
#include <c:\mysql\include\mysql.h>
#include <string.h>
#include <stdlib.h>
#include <fstream.h>
#include <c:\mysql\include\consola\consola.h>
#include <stdio.h>
#include <time.h>


class conexion4{

        unsigned int num_renglones;             //numero de renglones del resultado...
        unsigned int num_columnas;              //numero de columnas del resultado...
        char QueryFromFile[30][300];    //querys leidas desde un archivo...
        clock_t inicio,fin;                             //estructuras de tiempo
        double tiempoBusqueda;                  //tiempo usado por la ultima busqueda...

public:

        MYSQL conexion;                                 //tipo de  conexion al Mysql!!
        MYSQL_RES *resultados;                  //tipo de resultado de MYSQL!!
        MYSQL_ROW renglon;                      //regresa un arreglo de cadenas que representa los resultados...
        MYSQL_FIELD *campos;                    //regresa los campos del MYSQL...
        bool estatusConexion;           //cierto si hay conexion...

        conexion4(void);                                //constructor..
        conexion4(char database[20]);                                   //constructor...
        conexion4(char host[20],char user[20],char password[20],char database[20]);   //constructor...
        //inicia conexion
        void fin_conexion(void);  //finaliza la conexion...
        bool busqueda(char query[]);        //realiza una busqueda...
        bool ping(void);  //revisa si sigue viva la conexion
        void libera_resultado(void);        //libera la memoria del resultado...
        unsigned int getRenglones(void);   //regresa los renglones del query...
        unsigned int getColumnas(void);             //regresa las columnas del query...
        void getColumName(char *name,unsigned int numCampo);                    //regresa el nombre de la columna...dandole un numero...
        //debes darle el apuntador a donde regresa el nombre... **<>**
        bool getRenglon(void);     //inicializa el renglon del resultado...y regresa cierto si existe
        bool QuerysFromFile(char *nombre);
        double getQueryTime(void);          //regresa el tiempo que tardo la última búsqueda...
        ~conexion4();      //destructor..
```

```
};
```

En el archivo conexion4.cpp se puede observar la implementación de los métodos de la clase conexion4.

conexion4.cpp

```
#ifndef MI_CONEXION
	#include <windows.h>
	#include <iostream.h>
	#include <c:\mysql\include\mysql.h>
	#include <string.h>
	#include "conexion4.h"
#endif

/*****************************************/

conexion4::conexion4(char database[20]){

	estatusConexion=true;	//hubo conexion...
	this->num_renglones=0;
	this->num_columnas=0;
	this->tiempoBusqueda=-1;		//inicializo el tiempo a -1(no ha habido busqueda...)

	mysql_init(&conexion);
	mysql_options(&conexion,MYSQL_OPT_CONNECT_TIMEOUT,"20");
	mysql_options(&conexion,MYSQL_READ_DEFAULT_GROUP,"prueba");

	cout<<"Iniciando una conexion al servidor!!!"<<endl;


	if(!mysql_real_connect(&conexion,"localhost","ontananza","adios",database,0,NULL,0)){

		cout<<"\nNO SE PUDO CONECTAR A LA BASE DE DATOS\n"<<"Error:"<<mysql_error(&conexion)<<endl;

		estatusConexion=false;
	}

}
/*****************************************/

conexion4::conexion4(char host[20],char user[20],char password[20],char database[20]){

	estatusConexion=true;	//hubo conexion...
	this->num_renglones=0;	//
	this->num_columnas=0;	//
	this->tiempoBusqueda=-1;		//inicializo el tiempo a -1(no ha habido busqueda...)

	mysql_init(&conexion);	//inicio la conexion...
	mysql_options(&conexion,MYSQL_OPT_CONNECT_TIMEOUT,"20");	//opciones
```

```
	mysql_options(&conexion,MYSQL_READ_DEFAULT_GROUP,"prueba");	//opciones

	cout<<"Iniciando una conexion al servidor!!!"<<endl;
	if(!mysql_real_connect(&conexion,host,user,password,database,0,NULL,0)){

		cout<<"\nNO SE PUDO CONECTAR A LA BASE DE DATOS\n"<<"Error:"<<mysql_error(&conexion)<<endl;

		estatusConexion=false;
	}

}

/*****************************************/

void conexion4::fin_conexion(void){

	mysql_close(&conexion);

}

/*********************************************/

bool conexion4::busqueda(char query[]){

	this->inicio=clock();
	bool estatus=true;		//se logro el query
	if(mysql_query(&conexion,query)){		//el query fallo!!!...

		cout<<"\n\nError en la busqueda: "<<mysql_error(&conexion)<<endl;
		estatus=false;
	}else{

		this->resultados=mysql_store_result(&conexion);

		if(resultados){	//regreso algo...

			this->num_renglones=mysql_num_rows(this->resultados);
			this->num_columnas=mysql_num_fields(this->resultados);
		}else{	//no regreso nada...

			if(mysql_errno(&conexion)){	//si hubo error en el resultado...

				cout<<"\n\nError: "<<mysql_error(&conexion)<<endl;
				this->num_renglones=0;
				this->num_columnas=0;

			}else if(mysql_field_count(&conexion)==0)		//no fue un select...
				this->num_renglones=mysql_affected_rows(&conexion);	//regreso los renglones afectados...

		}

	}
```

```
	//calculo el tiempo que tardo en ejecutar la busqueda...
	this->tiempoBusqueda=(clock()-this->inicio)/(double)CLOCKS_PER_SEC;

	return estatus;
}

/*********************************************/

conexion4::~conexion4(){

}

/*********************************************/

conexion4::conexion4(void){

	estatusConexion=false;
	this->num_renglones=0;
	this->num_columnas=0;
	mysql_init(&conexion);    //inicio la conexion...
	mysql_options(&conexion,MYSQL_OPT_CONNECT_TIMEOUT,"20");        //opciones
	mysql_options(&conexion,MYSQL_READ_DEFAULT_GROUP,"prueba");    //opciones

}

/*********************************************/

bool conexion4::ping(void){

	bool estatus=true;

	if(mysql_ping(&conexion)){
		cout<<"\n\nError en la conexion..."<<mysql_error(&conexion)<<endl;
		estatus=false;
	}
	return estatus;
}

/*********************************************/

void conexion4::libera_resultado(void){
	mysql_free_result(this->resultados);

}

/*********************************************/

unsigned int conexion4::getRenglones(void){          //regresa los renglones del query...

	return this->num_renglones;

}

/*********************************************/
```

```
unsigned int conexion4::getColumnas(void){                    //regresa las columnas del query...

        return this->num_columnas;

}

/*********************************************/

void conexion4::getColumName(char *name,unsigned int numCampo){
        if(resultados){             //hay algo en el resultado...
                this->campos=mysql_fetch_field_direct(this->resultados,numCampo);
        }
        strcpy(name,campos->name);


}
/*********************************************/

bool conexion4::getRenglon(void){

        bool estatus=false;
        if(this->resultados){       //si hay algo en resultados...

                renglon=mysql_fetch_row(resultados);      //arroja siguiente renglon de resultados

                if(this->renglon)  //si renglon no es nulo.....
                        estatus=true;

        }

        return estatus;
}

/*********************************************/

bool conexion4::QuerysFromFile(char *nombre){

        char **QueryFromFile;
        int i=0,j=0,cont=0,longitud=0;
        int sizes[600];

        char ch;
        bool estatus;
   ifstream fe(nombre);     //si existe el fichero lo pone en fe
        while(fe.get(ch)){

                longitud++;
                if(ch==';'){
                        sizes[cont]=longitud;
                        longitud=0;
                        cont++;
                }
```

```
	}
	sizes[cont]=longitud;	//para lo sobrante del archivo!!
	fe.close();
	QueryFromFile=new (char *[(cont+1)]);
	for(int l=0;l<=cont;l++)
		QueryFromFile[l]=new (char[ (sizes[l]+1) ]);

	fe.open(nombre);
	while(fe.get(ch)){

		if(ch!=';'){

			if(ch=='\n')
				ch=' ';
				QueryFromFile[j][i]=ch;

		}else{
			QueryFromFile[j][i]='\0';
			j++;
			i=0;
			continue;
		}
		i++;
	}
	fe.close();

	for(int k=0;k<j;k++)
		estatus=this->busqueda(QueryFromFile[k]);		//aqui realizo las busquedas...
	delete QueryFromFile;
	return estatus;
}
/*************************************************/
double conexion4::getQueryTime(){
	return this->tiempoBusqueda;
}
/*************************************************/
```

B.2 Cliente

El archivo cliente.cpp contiene el main del programa del Cliente de Despliegue.

cliente.cpp

```
#include "conexion4.h"

conexion4  zeus; //conexion global...
void get_data(int id);
int get_vars(char temp[]);
double despliega_vars(int num_vars,int iter);
char var_tbl[8]="";		//tabla de variables....
char vardesp[100][10];
bool var_activas[100];
main(){
```

```
	double promedio=0;
	char query[260]="";
	char *temp;	//variables temporal...
	int id_desp=1;	//id del despliegue...
	int nvars;			//numero de variables...

	do{


		conexion4 ares("132.248.52.222","user","holaa","RELAP_DB");	//llamo a un objeto conexion4

		zeus=ares;
	//lo igualo al global..
		ares.~conexion4;

	}while(!zeus.estatusConexion);	//hazlo mientras no haya conexion al servidor
	//temporal.....

	cout<<"Cliente de MySQl usado:"<<mysql_get_client_info()<<endl;
	cout<<"Host:"<<mysql_get_host_info(&zeus.conexion)<<endl;
	cout<<"Servidor:"<<mysql_get_server_info(&zeus.conexion)<<endl;
	cout<<"Estatus:"<<endl<<mysql_stat(&zeus.conexion)<<endl;
	sleep(3);

	//-------------------------------------------------------
	//---------Pido las variables diponibles para verlas...
	get_data(id_desp);
	nvars=get_vars(var_tbl);
	cout<<"Tarjetas de control 205CCC0"<<endl;
	cout<<"Las variables de control (despliegue) encontradas son: "<<endl;
	for(int i=1;i<=nvars;i++){
		temp=strchr(vardesp[i],'_');
		temp++;
		cout<<i<<": 205"<<temp<<"0"<<endl;
	}
	//-------------------------------------------------------
	cout<<"teclea el numero para desplegar la variable 1-"<<nvars<<endl;
	char c='y';
	do{
		int response;
		cout<<"Num:";
		cin>>response;
		if(response>=1 && response<=nvars){
			var_activas[response]=true;
		}else
			cout<<"Num no existente"<<endl;
		cout<<"Otro y/n?"<<endl;
		cin>>c;
	}while(c != 'n' && c != 'N');
	//-------------------------------------------------------
	for(int cont=1;cont<200;cont++)
		promedio+=despliega_vars(nvars,cont);
	cout<<"\nEL tiempo promedio es: "<<(promedio/cont)<<" ms."<<endl;
```

```
	//en la variable promedio voy sumando los tiempos necesario de las Querys que muestra las variables...

	return 0;
}
//-------------------------------------------------------
void get_data(int id){


	char query[100]="";
	sprintf(query,"SELECT vartbl_name from desp_vartbl where id_desp=%d",id);
//	cout<<query;
	if(zeus.busqueda(query)){ //realizo la busqueda.....
		if(zeus.getRenglon())//si hay un renglon de resultado.....
			//cout<<"\nLa tabla que alberga las variables es: "<<zeus.renglon[0]<<endl;
			strcpy(var_tbl,zeus.renglon[0]);
		else;
			//cout<<"La tabla que alberga las variables es: "<<zeus.renglon[0]<<endl;
		zeus.libera_resultado();
	}

}
//-------------------------------------------------------
int get_vars(char temp[]){

	char query[70]="";
	int i=1;
	sprintf(query,"describe %s",temp);//describe tbl

	//-----------inicio el arreglo de variables....
	for(int j=0;j<100;j++){
		sprintf(vardesp[j],"");
		var_activas[j]=false;
	}
	//---------------------------------------------

	if(zeus.busqueda(query)){
		zeus.getRenglon();

		while(zeus.getRenglon()){
			strcpy(vardesp[i],zeus.renglon[0]);
			i++;
		}
		zeus.libera_resultado();
	}

	return i-1;
}
//-------------------------------------------------------
double despliega_vars(int num_vars,int iter){

	char query[280]="select id_iter";
	char final[330]="";
	char temp[10]="";
```

```
	double retorno;
	for(int i=1;i<=num_vars;i++){
		if(var_activas[i]){
			sprintf(temp,", %s",vardesp[i]);
			strcat(query,temp);
		}
	}

	sprintf(final,"%s from %s where id_iter=%d",query,var_tbl,iter);
	//cout<<final<<endl;
	do{
		if(zeus.busqueda(final)){
			if(zeus.getRenglon()){

				for(unsigned int i=0;i<zeus.getColumnas();i++){
					cout<<zeus.renglon[i]<<"\t";
				}
				retorno=1000*zeus.getQueryTime();
				cout<<"\t"<<retorno<<" ms."<<endl;
				zeus.libera_resultado();
			}
		}
		delay(100);

	}while(zeus.getRenglones()==0);
	return retorno;
}
//-------------------------------------------------------------
```

Apéndice C

En este apéndice se presenta el código fuente del cliente de interacción, usado para las pruebas. Básicamente se compone de tres archivos, dos de ellos contienen una clase implementada para manejar la conexión y las búsquedas en la base de datos[1] y el tercero contiene al programa en sí.

[1] Ver clase conexion4 en el Apéndice B.

C.1 Cliente de Interacción

El archivo cliente2.cpp contiene el main del Cliente de Interacción.

cliente2.cpp

```
#include <stdio.h>
#include "conexion4.h"

conexion4  zeus; //conexion global...
int id_desp=1;                  //identificador de despliegue...
int get_trips(void);            //funcion que regresa los trips activos....
int get_vconst(void);           //funcion que regresa las variables activas....
void disparo(int opcion);               //cambia el valor del trip especificado...
void cambio(int opcion);                //cambia el valor de la v constante especificada...
char trips[100][9];                     //trips activos...
int iniValTrips[100];           //valor inicial de los trips activos...
char vconst[100][9];            //var constantes activas...
float iniValVConst[100];  //valor inicial de las vconst...

main(){

        //long int i=0;
        int ntrips,nvconst;
        char *temp;         //variable temporal...
        char temp2[9]="";

        do{

                conexion4 ares("132.248.52.222","user","holaa","RELAP_DB");        //llamo a un objeto
conexion4

                zeus=ares;
        //lo igualo al global..
                ares.~conexion4;

        }while(!zeus.estatusConexion);      //hazlo mientras no haya conexion al servidor

        //------------
        cout<<"Cliente de MySQl usado:"<<mysql_get_client_info()<<endl;
        cout<<"Host:"<<mysql_get_host_info(&zeus.conexion)<<endl;
        cout<<"Servidor:"<<mysql_get_server_info(&zeus.conexion)<<endl;
        cout<<"Estatus:"<<endl<<mysql_stat(&zeus.conexion)<<endl;
        //------------
        ntrips=get_trips();                     //obtengo los trips...
        nvconst=get_vconst();       //obtengo las vconst...
        //------------
        system("cls");
        //-despliego los componentes encontrados...
        cout<<"Trips Activos encontrados: "<<endl;
        for(int i=0;i<ntrips;i++){
                strcpy(temp2,trips[i]);
                temp=strtok(temp2,"_");
                cout<<i+1<<" =>"<<temp<<"\tvalue : "<<iniValTrips[i]<<endl;
        }
```

```
	cout<<"Variables Constantes encontrados: "<<endl;
	for(int j=0;j<nvconst;j++){
		temp=strchr(vconst[j],'_');
		temp++;
		cout<<j+ntrips+1<<" =>"<<temp<<"\tvalue : "<<iniValVConst[j]<<endl;
	}
	//---------------------------------------
	cout<<"teclea el numero para Cambiar el valor actual (1-"<<ntrips+nvconst<<")"<<endl;
	cout<<"teclea un 0 para terminar..."<<endl;
	int option;
	do{
		cout<<"Opcion: ";
		cin>>option;
		if(option==0)
			break;
		else if(option>0 && option<=ntrips)			//se dispara...
			disparo(option);  //...
		else if(option<=(ntrips+nvconst))   //se cambia el valor de la cte...
			cambio((option-ntrips));			//...

	}while(TRUE);
	cout<<"adios...";

	return 0;

}

/************************/
int get_trips(void){

	char query[90]="";
	int cont=0;
	sprintf(query,"describe %d_trip",id_desp);
	//--------------------
	if(zeus.busqueda(query)){ //realizo la busqueda.....

		zeus.getRenglon();
		while(zeus.getRenglon())
			strcpy(trips[cont++],zeus.renglon[0]);
		zeus.libera_resultado();

	}
	//--------------------
	sprintf(query,"SELECT * FROM %d_trip WHERE id_iter=1",id_desp);
	if(zeus.busqueda(query)){
		zeus.getRenglon();		//obtengo un renglon del resultado...
		for(unsigned int i=1;i<zeus.getColumnas();i++){
			iniValTrips[i-1]=atoi(zeus.renglon[i]);
		}
	}

	return cont;		//regreso el numero de trips...
}
/************************/
int get_vconst(void){
```

```
	char query[90]="";
	int cont=0;
	sprintf(query,"describe %d_vconst",id_desp);
	//--------------------
	if(zeus.busqueda(query)){ //realizo la busqueda.....

		zeus.getRenglon();
		while(zeus.getRenglon())
			strcpy(vconst[cont++],zeus.renglon[0]);
		zeus.libera_resultado();
	}
	//--------------------
	sprintf(query,"SELECT * FROM %d_vconst WHERE id_iter=1",id_desp);
	if(zeus.busqueda(query)){
		zeus.getRenglon();		//obtengo un renglon del resultado...
		int j,k;
		for(unsigned int i=1;i<zeus.getColumnas();i++){

			//-------------------------
			char *temp;
			temp=zeus.renglon[i];
			for(k=0,j=0;k<strlen(temp);k++){
				if(temp[i]=='.'){   //si tiene .
					j=1;
					break;
				}
			}
			if(j==0)
				sprintf(temp,"%s.00",zeus.renglon[i]);

			iniValVConst[i-1]=atof(temp);
			//-------------------------
		}
	}
	//--------------------
	return cont;	//regreso el numero de constantes activas...
}
/***********************/
void disparo(int opcion){
	char query[100]="";
	char temp[9]="";
	int value=0;
	if(iniValTrips[opcion-1]==value)
		value=1;
	sprintf(query,"UPDATE %d_trip SET %s=%d",id_desp,trips[opcion-1],value);
	cout<<query<<endl;

	if(zeus.busqueda(query)){ //realizo la busqueda...
		strcpy(temp,trips[opcion-1]);
		cout<<"\nTrip "<<strtok(temp,"_")<<"\tval :"<<value<<endl;
	}
	iniValTrips[opcion-1]=value;
}
/***********************/
void cambio(int opcion){
	char query[100]="";
```

```
		//--------------------------------
		float value=iniValVConst[opcion-1];
		char *temp=strchr(vconst[opcion-1],'_');
		temp++;
		//--------------------------------

		cout<<"Var const "<<temp<<"\tval :"<<value<<endl;
		cout<<"value :";
		cin>>value;
		iniValVConst[opcion-1]=value;
		sprintf(query,"UPDATE %d_vconst SET %s=%f",id_desp,vconst[opcion-1],value);
		cout<<query<<endl;
		if(zeus.busqueda(query)){ //realizo la busqueda...
			cout<<"\n Var const"<<temp<<"\tval :"<<value<<endl;
		}
}
/**********************/
```