Centro de Investigación y de Estudios Avanzados del I.P.N.

Departamento de Ingeniería Eléctrica

Sección de Computación

Ambiente para el Desarrollo de Sistemas de Información (ADSI)

Tesis que presenta el Lic. en Informática Sergio Rivera Santiago para obtener el grado de Maestro en Ciencias dentro de la especialidad de Ingeniería Eléctrica con opción en Computación.

Trabajo dirigido por:

- Doctor en Ciencias Feliú Davino Sagols Troncoso (CINVESTAV, Departamento de Ingeniería Eléctrica, Sección de Computación.)

México, D.F.
Julio de 1998

Becario CONACYT: 62532

AMBIENTE PARA EL DESARROLLO DE SISTEMAS DE INFORMACIÓN (ADSI)

RESUMEN

El presente documento describe a ADSI, un ambiente enfocado hacia bases de datos, útil para desarrollar sistemas de información. ADSI contempla características típicas de los sistemas clásicos de bases de datos y aspectos relacionados. La arquitectura escalable es la fuerza de ADSI. Está formado por procesos cooperativos y utiliza una división eficiente de trabajo para manejar todos los servicios de las bases de datos. Además, ADSI permite diseñar diccionarios de datos, pantallas, informes y enunciados de consulta, y los usuarios pueden elaborar programas de aplicación. Así mismo, ADSI garantiza la integridad de los datos, la seguridad de la información, el control de la concurrencia, la alta velocidad de acceso a los datos, las respuestas optimizadas, los protocolos de enlace y aceptación, las operaciones sobre archivos, la distribución de carga de trabajo, la disponibilidad de temporizadores, la recuperación de información y la tolerancia a fallas. Adicionalmente, en ADSI el usuario puede proporcionar parámetros para optimizar recursos. Por último, ADSI posee facilidades naturales para instalarse en sistemas operativos multiusuario, multitarea y en red. El desarrollo original se llevó a cabo en el sistema operativo de tiempo real para computadoras personales QNX.

ENVIRONMENT TO DEVELOP INFORMATION SYSTEMS (EDIS)

ABSTRACT

Here we describe a database environment (EDIS), useful to develop information systems. EDIS contains some typical characteristics of classical database systems and related aspects. Its major characteristics relay on a scalable architecture. Consisting of a team of cooperating processes, EDIS employs an efficient scheduler to handle all database services. In addition, EDIS gives facilities to design data dictionaries, screens, reports and query macros as well as all the necessary to build applications. The integrity, security and concurrency control are all ensured under EDIS. Furthermore, EDIS supports high data access speedness, optimized answers, protocols for conectivity and commitment, operations on files, job distribution, timers availability, information recovery and failure tolerance. Moreover, users and system developers using EDIS may customize resource optimizing parameters. Finally, EDIS provides native facilities for being installed on multiuser, multitasking and networking operating systems. The original development was built on the realtime operating system for personal computers QNX.

Al Lic. Fernando Galindo Soria:
gracias por su amistad y su ayuda constante,
por inducirme al desarrollo de este trabajo, por
transmitirme sus ideas y experiencia, y por sus valiosas
aportaciones al diseño del proyecto.

Al Dr. Feliú Davino Sagols Troncoso:
agradezco su credibilidad en mí y en el proyecto,
su asesoría, sus consejos, sus enseñanzas y correcciones,
me son útiles en mi formación.

Al Dr. Morales Luna y al Dr. Coyote Estrada:
su revisión, sus observaciones y correcciones,
enriquecen al proyecto y a mi persona.

A mi amada esposa Araceli y a mis hijos Noemí Araceli y
Esteban Josué: gracias por su confianza, su paciencia, sus
consideraciones y apoyo.

A la memoria de mi papá y a mi madre gracias
por sus incesantes oraciones.

Al I.P.N., al CINVESTAV y al CONACYT gracias.

A la empresa DIAM, S.A. DE C.V.. gracias por
sostener económicamente el desarrollo del proyecto
durante los diez años requeridos, por proporcionar
el equipo, los libros, las licencias y los viajes necesarios.

pero por encima de todo, ..., GRACIAS A DIOS.

AMBIENTE PARA EL DESARROLLO DE SISTEMAS DE INFORMACIÓN (ADSI)

ÍNDICE

I. INTRODUCCIÓN

Se presenta un Ambiente para el Desarrollo de Sistemas de Información (ADSI) enfocado hacia bases de datos, que permite el desarrollo e implementación de sistemas de información a través de sus diversos componentes. ADSI contiene algunas características que son familiares en los sistemas clásicos de bases de datos y también contiene aspectos particulares del ambiente.

Este ambiente está integrado por seis clases de procesos que interactúan coordinadamente: núcleo, administradores, constructores, explotadores, utilerías y control. Algunos de estos procesos facilitan la elaboración de un sistema de información en el diseño de pantallas, informes o enunciados de consulta; otros procesos capturan, leen, actualizan y muestran datos; algunos otros procesos participan en la elaboración de un sistema y permanecen en las pruebas e implantación del mismo.

El diseño de ADSI es modular y escalable, contiene un conjunto de procesos cooperativos dentro del ambiente: algunos de ellos son fundamentales para la operación; otros se pueden integrar selectivamente; algunos más pueden ejecutarse en forma esporádica; el usuario puede agregar otros procesos que considere necesarios.

ADSI contiene mecanismos que garantizan la integridad de los datos, la seguridad de la información, la concurrencia de las transacciones, el acceso rápido y la respuesta óptima así como las facilidades naturales para instalarse en sistemas operativos multiusuario, multitarea y en red. El desarrollo original de ADSI se llevó a cabo en un sistema operativo de tiempo real para computadoras personales.

Finalmente, el Ambiente ofrece un método de configuración que permite optimizar los recursos, soportar concurrencia y tolerar fallas.

II. DESCRIPCIÓN DEL AMBIENTE

II.1. ANTECEDENTES

La Informática es la ciencia del tratamiento de la información que se ha dado a la tarea de establecer los vínculos entre las diferentes ciencias, campos de investigación y desarrollo tecnológico, con el objeto de crear un orden en el tráfico, intercambio y valoración de datos e información. Para ello la Informática se apoya constantemente en el desarrollo de tecnologías de cómputo, principalmente en materia de sistemas de información, programación de sistemas y comunicaciones.

La programación de sistemas ha desarrollado herramientas útiles y prácticas como son los sistemas operativos, los compiladores y los manejadores de bases de datos, entre otros.

Las computadoras han tenido importantes repercusiones en muchos aspectos de la sociedad y se espera que en los próximos años diversifiquen sus aplicaciones aumentando así su efecto en la vida cotidiana. Las telecomunicaciones y las bases de datos distribuidas son ramas que juegan un papel decisivo en estas nuevas aplicaciones; sus servicios cubren aspectos cada vez más amplios. Así, las bases de datos cobran mayor importancia.

Pero, ¿Qué es un sistema de bases de datos? En esencia, no es más que un sistema de administración de información basado en computadoras, es decir, un sistema cuyo propósito general es registrar, mantener y accesar información para consulta y/o actualización. Tal información puede estar relacionada con cualquier cosa que sea significativa para la toma de decisiones en la organización donde dicho sistema opera.

Los tópicos relacionados con bases de datos son numerosos y extensos, por lo cual, para el desarrollo de ADSI se revisó documentación sobre avances en materia de generadores de programas, diseño de sistemas y bases de datos así como sistemas administradores de bases de datos, entre otros más. Todo esto se combinó con ideas propias y se formuló un diseño inicial a principios de 1990.

Sin embargo, existen dos antecedentes que motivaron fundamentalmente el desarrollo de ADSI:

1. El interés por las actividades que conducen al diseño y fabricación de tecnología, ya que en ellas se planea, desde la inversión en la investigación científica, hasta las estrategias globales del comercio mundial.

2. La imperante necesidad de crear tecnología en México para disminuir la dependencia extranjera.

Con la integración de México al Tratado de Libre Comercio, se deriva la oportunidad de que los científicos mexicanos ayuden a la creación de campos propicios para la ciencia y la tecnología, acorten distancias entre la teoría y la aplicación, formulen planes de estudio e investigación donde se preparen expertos que lleven a la industria las innovaciones que

permiten mejorar la competencia y la calidad de productos y servicios que se producen en el país.

Aunque modestamente, ADSI es una base tecnológica cuyo desarrollo intenta cubrir las especificaciones de la ciencia en materia de bases de datos, ofrecer un servicio a la sociedad y ser capaz de asimilar las innovaciones que se presenten en la evolución de la Informática y la Computación particularmente.

II.2. MODELO CONCEPTUAL

II.2.1. PANORAMA GENERAL

La figura 2.1 representa al Ambiente para el Desarrollo de Sistemas de Información (ADSI). En el diagrama aparecen los diferentes componentes que interactúan cooperativamente. En el centro están los administradores de recursos; a la izquierda se notan los constructores de esquemas, pantallas, informes y enunciados (macros) de consulta; arriba se muestran los procesos de manipulación de datos; a la derecha están aquellos que agregan accesorios y herramientas de apoyo y; abajo se aprecian los comandos de control que dirigen la operación de ADSI.

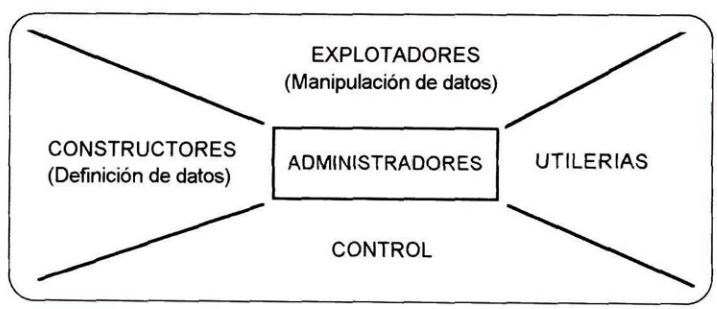


Fig. 2.1 Esquema general de ADSI

Las habilidades fundamentales de ADSI son las siguientes:

1. Maneja datos persistentes (duraderos).

2. Trabaja eficientemente con grandes cantidades de datos.

ADSI logra esto gracias a las siguientes propiedades:

a) Soporta un modelo de datos para describir aplicaciones.

b) Permite definir, accesar y manipular datos.

c) Maneja transacciones concurrentes, es decir, varios usuarios a la vez pueden utilizar su información.

d) Controla y limita el acceso a usuarios no autorizados.

e) Verifica la validez de los datos.

f) Posee la habilidad de recuperarse de fallas sin que haya pérdida de información.

Es adecuado referirse al modelo de datos como una manera de concebir los datos almacenados en forma entendible al ser humano, como tablas y registros, y no como un conjunto abstracto de pulsos eléctricos o bits. A un nivel relativamente bajo ADSI permite visualizar los datos como un conjunto de archivos y la relación entre ellos se establece bajo el marco del modelo relacional.

ADSI no administra archivos únicamente, su mayor poder radica en el manejo de transacciones. Esto involucra el acceso concurrente a los datos, quizá para responder a una consulta o para actualizar los datos, lo cual exige un tiempo pequeño y preferentemente constante e independiente del tamaño de los archivos. Lo último se ofrece mediante un lenguaje propio accesible desde los programas de aplicación y desde un lenguaje semejante a SQL.

También se ofrecen herramientas para el diseño y encadenamiento de pantallas a fin de manipular datos desde consolas o terminales; igualmente se tienen herramientas para diseñar reportes.

Actualmente ADSI soporta un sistema de archivos centralizados.

En materia de seguridad, ADSI garantiza que bajo ninguna circunstancia haya pérdida de datos. Así mismo la información está protegida contra usuarios no autorizados y para ello se ha incorporado un árbol interactivo para la asignación y remoción de privilegios.

II.2.2. NIVELES DE ABSTRACCIÓN

En la literatura de sistemas clásicos de bases de datos se distingue el significado de la dicotonomía *esquema/instancia* (scheme/instance), donde *esquema* es la estructura de alguna cosa mientras que *instancia* es el valor que una porción o el total de esa cosa tiene actualmente.

Los niveles de abstracción son diferentes formas de apreciar los datos: la computadora por su parte, trata a los datos como bits; mientras que el usuario se refiere a ellos por conceptos como nombre, dirección, teléfono, etc.

En la figura 2.2 aparece una forma adecuada de observar los tres niveles de abstracción en un sistema de base de datos.

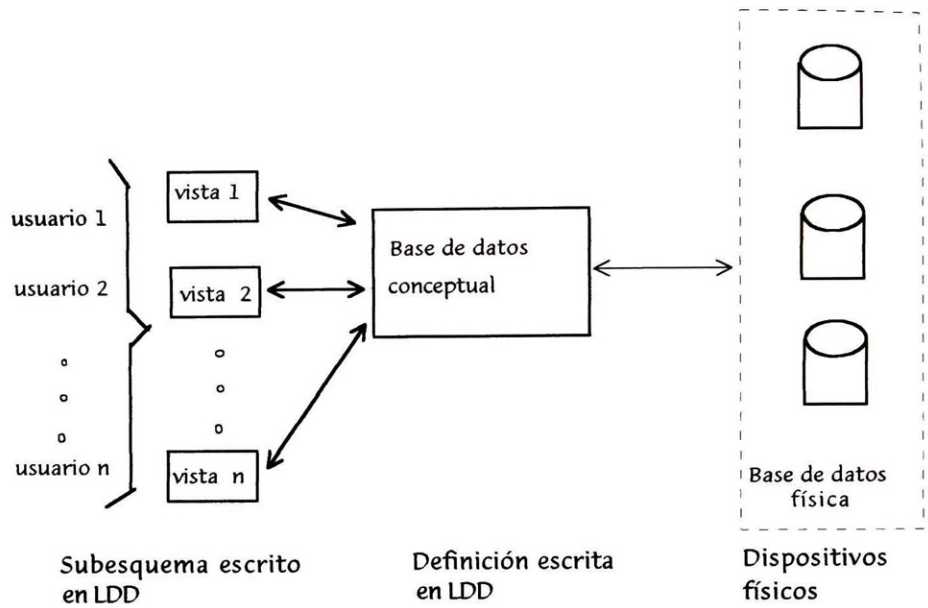


Fig. 2.2 Niveles de abstracción de un sistema de bases de datos

El *nivel físico* se refiere a la colección de archivos, índices y otras estructuras de almacenamiento empleadas para accesarlos eficientemente, y residen en dispositivos de almacenamiento secundario tales como discos.

El *nivel conceptual* es una abstracción del mundo real, es decir, del mundo de los usuarios. ADSI proporciona un lenguaje de definición de datos (LDD) para describir el esquema conceptual y su implementación por el esquema fisico.

Las bases de datos conceptuales unifican los datos usados en una organización. En ADSI cada aplicación coloca todos sus archivos de información juntos y los aprecia bajo el modelo descrito por las bases de datos conceptuales. Al proceso de unificar todos los datos y formatos de una organización con el fin de obtener la conceptualización de las bases de datos, se le denomina *integración de bases de datos*.

El *nivel de vista o subesquema* es una porción de las bases datos conceptuales o una abstracción de una parte de las mismas. En cierta forma la construcción de vistas es lo contrario del proceso de integración de bases de datos; para cada colección de datos que contribuyeron a las bases de datos conceptuales, se puede construir una vista que contiene únicamente esos datos. Las vistas son importantes para garantizar la seguridad en un sistema

de bases de datos, permitiendo que subconjuntos de datos puedan ser vistos por aquellos usuarios con privilegios para accederlos.

A menudo una vista es una pequeña base de datos conceptual y está al mismo nivel de abstracción de las bases de datos conceptuales. Sin embargo, existen situaciones donde las vistas son más abstractas que las bases de datos conceptuales, como en el caso de los datos que una vista genera a partir de bases de datos conceptuales donde realmente dichos datos no están en esas bases de datos.

La cadena de abstracciones de la figura 2.2 va desde el nivel de vistas hasta el nivel conceptual y desde ahí va al nivel físico. Existen dos niveles de independencia de datos. Lo más obvio es que el esquema físico de las bases de datos bién diseñadas pueda cambiarse sin alterar el esquema conceptual ni requerir una redefinición de subesquemas. Esta independencia se denomina *independencia de datos física*, y una de sus consecuencias es que al modificar a la organización física de los archivos con fines de optimización, se afecte la eficiencia de las aplicaciones pero nunca requiere modificar los programas basados en el esquema conceptual de las bases de datos.

La relación entre las vistas y las bases de datos conceptuales es la *independencia de datos lógica*, y así se pueden aplicar muchas modificaciones a las bases de datos conceptuales sin afectar los subesquemas existentes. Nuevamente, no se requiere modificar los programas. Las únicas modificaciones que pueden afectar el diseño conceptual ocurren cuando se altera la información estructural de los esquemas o subesquemas.

II.2.3. LENGUAJES DE DEFINICIÓN Y MANIPULACIÓN DE DATOS

En el diseño de ADSI se contemplan dos lenguajes particulares del Ambiente: el lenguaje de definición de datos (LDD) y el lenguaje de manipulación de datos (LMD). Ambos lenguajes se incluyen debido a que en la programación ordinaria, las declaraciones e instrucciones ejecutables son parte del mismo lenguaje, y en consecuencia los datos de un programa existen únicamente cuando se ejecuta. En cambio, en ADSI los datos son permanentes sin importar cuál o cuántos programas se ejecutan.

Para la declaración del esquema conceptual se emplea un lenguaje de definición de datos no procedural, que describe los tipos de entidades y sus interrelaciones.

Para ejecutar operaciones sobre archivos se requiere del lenguaje de manipulación de datos. En el se expresan comandos tales como leer, encontrar, escribir, actualizar, borrar, etc.

A menudo la manipulación de archivos es hecha por programas del usuario escritos en algún lenguaje de alto nivel como 'C' Estos programas manipulan datos en tareas dedicadas que requieren tomar decisiones, efectuar cálculos e invocar operaciones fundamentales de ADSI sobre archivos. La figura 2.3 muestra los datos vistos por un programa de usuario. Los datos locales del programa se manipulan en forma ordinaria, mientras que los datos persistentes se manejan a través de servicios que ofrece ADSI. Cada transacción deposita sus datos dentro del área de trabajo local del programa a fin de ser modificados y actualizados los archivos ADSI.

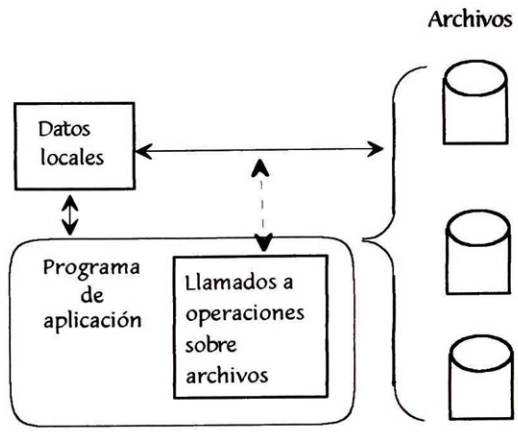


Fig. 2.3 Los datos vistos por un programa de aplicación

II.2.4. INTERACCIÓN DE COMPONENTES

La figura 2.4 muestra como interactúan los diversos componentes y lenguajes de un Sistema Administrador de Base de Datos (SABD). En el lado derecho aparece el esquema de la base de datos (diseño) alimentando al compilador del LDD para producir una descripción interna de las bases de datos.

El procesador del LMD recibe en diversas maneras las operaciones solicitadas por los usuarios: esto incluye a las operaciones de navegación interactivas, la generación de informes, la atención de consultas y programas de aplicación, etc. Para cada una de estas operaciones hay procesos controladores dedicados que funcionan como interfases entre los usuarios y el LMD.

El administrador de bases de datos (ver figura 2.4) toma comandos en el nivel conceptual y los traduce al nivel físico, mantiene y accede tablas de información sobre autorizaciones y control de concurrencia. Así puede saber si un usuario tiene o no privilegios de acceso sobre los datos solicitados; la tabla de control de concurrencia impide conflictos al actualizar simultáneamente los datos.

El administrador de archivos lleva a cabo las operaciones solicitadas por el usuario en dispositivos de almacenamiento secundario.

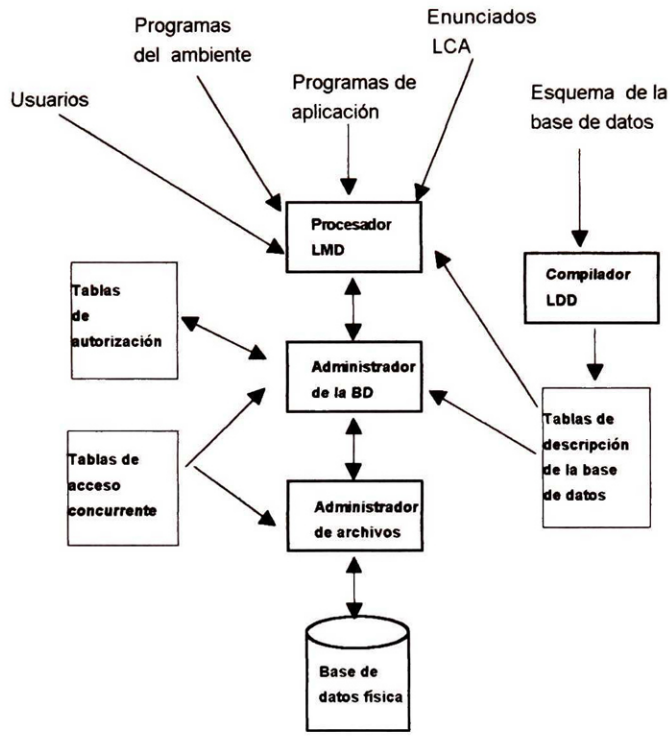


Fig. 2.4 Diagrama general de un sistema de bases de datos

II.3. NÚCLEO (ADMCEN)

En la figura 2.5 se muestra el esquema general de ADSI. En el centro aparece su parte más importante que denominamos NÚCLEO o ADMINISTRADOR CENTRAL (ADMCEN). Los administradores son procesos concurrentes de ADSI, y la comunicación con ellos para interactuar con el ambiente se hace por medio de mensajes (Ver anexo B).

ADMCEN se integra al sistema operativo (SO) como un proceso privilegiado.

El SO le proporciona los canales y recursos para la comunicación y control de los demás procesos. En el anexo D se muestra la lista de mensajes para interactuar con ADMCEN.

ADMCEN realiza el control de la concurrencia y se encarga de ejecutar el protocolo de aceptación en dos fases (2-phase commitment protocol), al mismo tiempo ADMCEN sostiene un estrecho contacto con los administradores de archivos ADMARCS, para realizar las operaciones de acceso a los datos. También controla un archivo bitácora (archivo log) para reconstruir la información en caso de fallas.

ADMCEN recibe todos los requerimientos que los procesos envían, los evalúa y decide rechazar o delegar la función solicitada en alguno de los administradores.

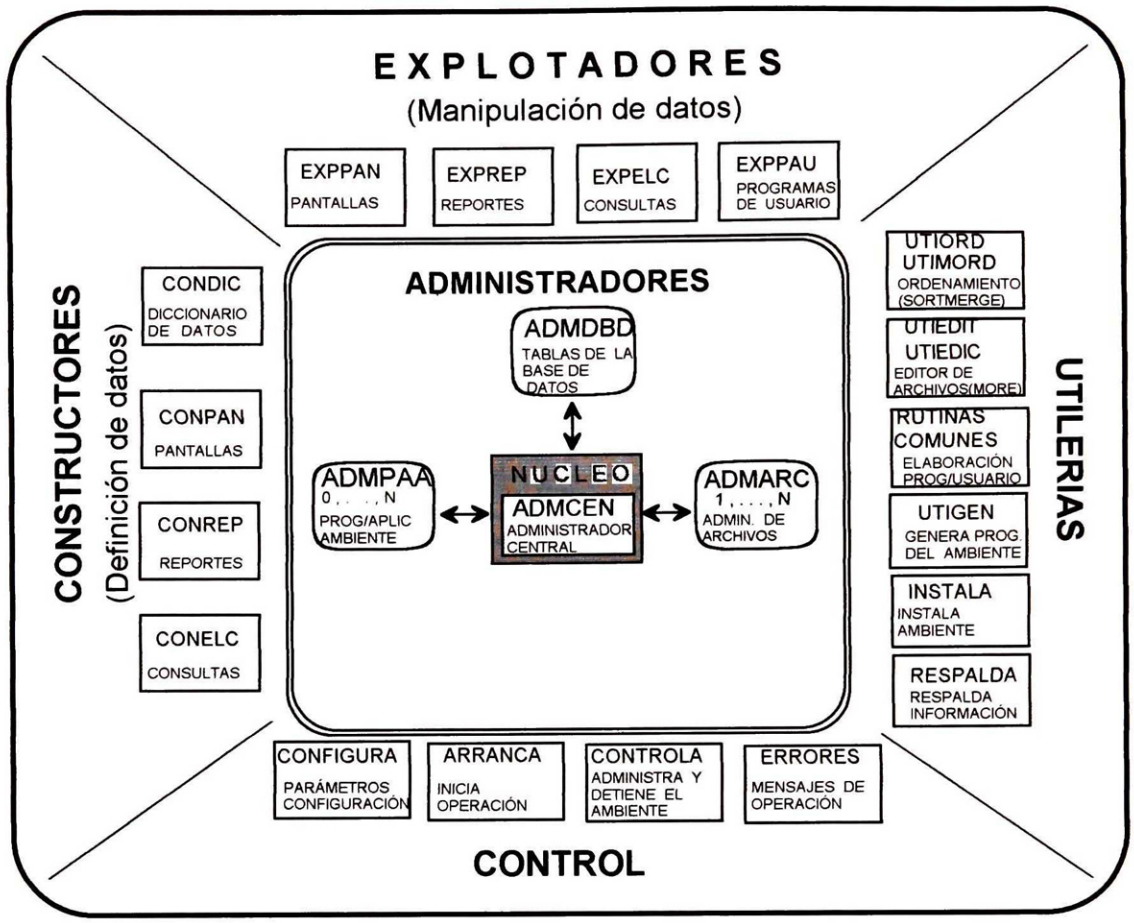


Fig. 2.5 Ambiente para el desarrollo de sistemas de información (ADSI)

II.4. ADMINISTRADORES

Los administradores son procesos fundamentales de ADSI y se encuentran en operación permanente interactuando con ADMCEN, mientras el ambiente no sea detenido por el usuario.

II.4.1. ADMARCS

Los administradores de archivos son procesos que toman bajo su custodia un número determinado de archivos del sistema (1 o más). Pueden existir varios ADMARCS pero un archivo cualquiera únicamente puede aparecer en un ADMARC. Cada ADMARC es responsable de mantener el control y acceso de los datos e índices de cada archivo bajo su custodia. Los mensajes de las operaciones sobre archivos se señalan en el anexo E.

Los detalles sobre la interacción de ADMARC y ADMCEN para garantizar el control de concurrencia aparecen en el capítulo IV

II.4.2. ADMDBD

Este administrador mantiene en memoria el diccionario de datos y lleva a cabo el control estructural de la aplicación.

Otros procesos del ambiente pueden comunicarse con ADMDBD para solicitar información sobre archivos, tipos de llaves, longitud de campos, etc. Particularmente, el explotador de pantallas (EXPPAN) solicita a este administrador la trayectoria detallada de acciones a ejecutar incluyendo la integridad referencial, para los movimientos de altas, bajas y cambios.

Véanse en el anexo F los mensajes referentes a ADMDBD.

II.4.3. ADMPAAS

Los administradores de programas de aplicación del ambiente, activan como transacciones (en el sentido más amplio del término), el código desarrollado por los programadores de aplicaciones. Estos procesos pueden utilizar todos los servicios de los demás administradores y comunicarse con cualquier otro proceso del sistema pasando argumentos y utilizando mensajes. Ver anexo K.

II.5. CONSTRUCTORES

Los constructores son procesos que sirven para elaborar componentes de las aplicaciones, se usan generalmente en la etapa de diseño de las mismas, no están en operación permanente como ocurre con los administradores.

II.5.1. CONDIC

El constructor CONDIC permite que el usuario diseñe el diccionario de datos de su aplicación, es decir, el esquema conceptual. El diccionario de datos se define en un archivo de texto y contiene la información estructural de atributos, índices, llaves e integridad referencial de los archivos. Así mismo, se agregan los nombres de los administradores de programas de aplicación del ambiente ADMPAA con la especificación de sus parámetros.

El diccionario de datos escrito en el archivo de texto (*.DD) es revisado por CONDIC para producir un archivo compilado (*.DIC). Un caso se muestra y se documenta en el anexo N.

II.5.2. CONPAN

Este constructor permite diseñar y modificar pantallas que funcionan como interfase entre ADSI y el usuario. Las pantallas se invocan desde el explotador de pantallas (EXPPAN) y se ligan entre sí para crear cascadas. CONPAN cuenta con ayudas al usuario que se activan al oprimir simultáneamente las teclas CTRL/Y. El menú ofrece las siguientes acciones:

1. EDITA FORMA. Invoca una pantalla o forma previamente elaborada.

2. NUEVA FORMA. Diseña una pantalla con determinadas dimensiones: fila superior, columna inicial izquierda, ancho del marco, altura del marco, ancho del texto, memoria requerida.

3. CARGA FORMA. Pide el nombre de la pantalla a mostrar.

4. GUARDA FORMA. Pide el nombre para guardar la pantalla actual.

5. ETIQUETA FORMA. Pide una etiqueta de 5 caracteres para colocarla en el centro del marco superior de la pantalla.

6. ASIGNA CAMPOS. Pide los atributos de cada campo:
   -nombre,
   -justificación,
   -caracter de limpieza,
   -si se rellena a ceros,
   -valor por omisión,
   -mensaje de ayuda,
   -tabulación automática,
   -de captura,
   -el usuario debe llenarlo,
   -de salida,
   -eco (datos visibles),
   -tipo de dato (int, unsigned, long, float, double, char),
   -notación científica,
   -importación de datos (valor constante, moverse de otro campo, acceder un archivo),
   -ignora captura,
   -saltar hacia otro campo,
   -respeta formato,
   -hipermedias asociadas.

7. CAMBIA CAMPOS. Cambia el color o atributos.

8. APLICACIÓN. Pide la aplicación: alta, baja, cambio, consulta, menú, ayuda, procesos a ejecutar, claves de acceso.

9. CAMPO CLAVE. Nombre del campo llave para aplicaciones de alta, baja, cambio y consulta.

10. CÁLCULOS. Operaciones aritméticas sobre los valores de los campos.

11. FUNCIONES. Invoca funciones del Ambiente. Ver anexo J.

12. HIPERMEDIAS. Pide acciones o eventos que se disparan durante el proceso:
    -invocar la calculadora,
    -funciones del ambiente,
    -asociar pantallas,
    -ejecutar un programa en línea,
    -ejecutar un programa de imágenes,
    -validar el dato capturado,
    -mostrar dato si cumple con la condición,
    -ejecutar un programa en lote (batch),
    -ejecutar un programa de sonido,
    -ir al SO,
    -ejecutar un comando del SO,
    -ejecutar un programa de gráfica,
    -mostrar documentación (ayuda),
    -validar formato de fecha,
    -acceder un archivo.

13. CAMBIA APLICACIÓN. Indica la posibilidad de cambiar la aplicación durante la ejecución. Permitido para altas, bajas, cambios y consultas.

14. PROCESO FINAL. Invoca recursos del ambiente en altas, bajas, cambios y consultas.

15. COMANDOS DE CONTROL DE EJECUCIÓN. Invoca recursos del ambiente en aplicaciones que ejecutan procesos.

Además de las acciones que se efectúan en el menú, las teclas de función tienen una acción programada:

F1. Lista los caracteres gráficos.

F2. Da acceso a la edición de pantallas en modo gráfico.

F3. Cambia el color del marco de la pantalla o del texto.

F4. Cambia el marco a sencillo, doble o sin marco.

F5. Define la naturaleza funcional de un atributo: calculadora, función o dato. Para ello se piden sus dimensiones: fila superior, columna inicial izquierda, ancho del marco, altura del marco, ancho del texto, memoria requerida.

F6. Elimina un campo de la pantalla.

F7. Guarda un campo en memoria con todos sus atributos.

F8. Copia un campo de memoria sobre un lugar en la pantalla.

F9. Presenta la ayuda para usar CONPAN.

F10. Muestra la ayuda para usar las rutinas comunes de los programas de aplicación. Ver anexo K.

II.5.3. CONREP

Este constructor diseña y modifica informes. Funciona como interfase entre ADSI y el usuario. Los informes se ligan en el diseño de pantallas y su salida se puede ver en una impresora o terminal de la red. La definición de las impresoras se proporciona en un archivo LPT.INFO. Ver anexo M.

Los procesos CONREP y EXPREP facilitan el diseño de informes automáticamente, evalúan los registros de entrada y las líneas de detalle, copian información entre campos, procesan sumatorias y cortes, producen salidas complejas y rápidas a la terminal de video o a alguna impresora. Los diseños se invocan desde el explotador de informes (EXPREP). CONREP proporciona ayuda al oprimir simultáneamente las teclas CTRL/Y. El menú de acciones es el siguiente:

1. EDITA REPORTE. Invoca un informe previamente elaborado.

2. NUEVO REPORTE. Similar a CONPAN.

3. CARGA REPORTE. Pide el nombre del informe a mostrar.

4. GUARDA REPORTE. Pide el nombre para guardar el informe actual.

5. INFORMACIÓN GENERAL. Pide información referente a:
   -número de parámetros de entrada,
   -número de página a iniciar,
   -número de líneas por página,
   -nombre del archivo a leer,
   -campo llave del archivo,
   -identificación del informe,
   -número de columnas por línea de impresión,
   -número de líneas por pulgada a imprimir.

6. ASIGNA CAMPOS. Pide información para asociar a cada campo:
   -nombre,
   -justificación,
   -importación de datos (valor constante, copiar campo, acceder un archivo),
   -tipo de campo (int, unsigned, long, float, double, char),
   -notación científica,
   -campo acumulador,

-hipermedias asociadas.

7. CAMBIA CAMPO. Similar a CONPAN.

8. ASIGNA LÍNEAS. Pide información de cada línea:
   -tipo (línea a ignorar, línea de trabajo, encabezado de página, corte de página, encabezado de corte de control, corte de control, línea de detalle, línea de detalle simulada, corte final, encabezado de la página final, corte de la página final),
   -número de páginas a saltar antes de escribir,
   -número de líneas a saltar antes de escribir,
   -campo de corte,
   -salta una página antes de escribir,
   -salta una página después de escribir.

9. CALCULADORA. Similar a CONPAN.

10. FUNCIÓN. Similar a CONPAN.

11. EVALUACIÓN DE LA LÍNEA DE DETALLE. Define una expresión para decidir si la línea de detalle se escribe o no.

12. ARCHIVO ENTRADA. Describe los campos del archivo de entrada:
   -nombre,
   -tipo (int, unsigned, long, float, double, char),
   -longitud,
   -número de decimales,
   -indicador para leerse del archivo (campo activo).

13. EVALUACIÓN DE REGISTROS DE ENTRADA/SALIDA. Define la expresión para decidir si el registro se procesa o no.

14. CAMPOS DE CONTROL. Lista los campos que intervienen en el control de cortes.

CONREP tiene programadas las teclas de función desde F1 hasta F10 similares a CONPAN.

II.5.4. CONELC

El desarrollo de herramientas para bases de datos incluye un lenguaje de consulta que por lo general es un lenguaje de consulta estructurado conocido por sus siglas del inglés SQL (Structured Query Language). ADSI no utiliza exactamente un SQL sino que cuenta con un lenguaje de consulta semejante a un SQL estándar al cual denominamos LCA (Lenguaje de Consulta del Ambiente). Las expresiones que se elaboran para invocar al lenguaje de consulta se diseñan con la ayuda de un editor de textos.

Las palabras claves del lenguaje de consulta se utilizan en inglés. Algunas diferencias consisten básicamente en lo siguiente:

1. En SQL se inicia la proyección de variables con el comando SELECT y los archivos se enumeran con el comando FROM. En ADSI primero se enuncian los archivos a involucrar (FROM) y luego se proyecta con el comando PROJECT. No se usa SELECT ya que la selección se genera en el contenido de la cláusula WHERE, en una función agregada y en la selección de grupos de registros.

2. En ADSI se permite que al acceder un archivo (FROM) se pueda declarar la llave a usar, el valor de la llave a leer o cuando se trata de llave compuesta indicar únicamente algunos campos que forman dicha llave (los campos de más a la izquierda).

3. En ADSI la invocación a subconsultas (subqueries) se hace en la forma de llamados a funciones como en el lenguaje 'C'.

4. Los enunciados del lenguaje de consulta pueden contener parámetros que se sustituyen con los valores actuales de la pantalla que los invoca.

La explicación detallada del funcionamiento del lenguaje de consulta LCA se expresa en un manual dedicado a ello y la fuente se tomó en la referencia bibliográfica [12] y [8].

Las invocaciones a los enunciados del LCA se ligan en el diseño de pantallas y su salida se ve en una terminal o consola. El proceso EJEELC sustituye los parámetros declarados en los enunciados.

En el capítulo VII se describe un caso práctico y ahí se muestran algunos enunciados de SQL estándar y su equivalencia a los enunciados de LCA de ADSI.

En términos generales el lenguaje de consulta LCA de ADSI está provisto de las siguientes capacidades:

1. Recupera información desde un archivo mencionando los campos a proyectar o colocando un asterisco para proyectar todos.

2. Proyecta registros evitando la duplicidad de valores de salida con el uso del argumento DISTINCT.

3. Recurre a la cláusula de selección WHERE con el fin de proyectar solo registros con valores evaluados. Algunos operadores usados:

    3.1 Operadores relacionales usados en el lenguaje C (== , <, <=, >, >=, !=).

    3.2. Operadores booleanos usados en el lenguaje C (&&, ||, !).

    3.3. Operadores IN, BETWEEN y LIKE. La negación de una evaluación obtenida con estos operadores es con el operador (!).

4. Proyecta registros con el uso de funciones agregadas (COUNT, SUM, AVG, MAX, MIN).

5. Usa la cláusula GROUP_BY para definir un subconjunto de valores en un campo particular en términos de otro campo y aplicar una función agregada a dicho subconjunto.

6. Emplea la cláusula HAVING para definir criterios para seleccionar grupos de registros, tal como WHERE se emplea para seleccionar registros individuales.

7. Refina la salida proyectada mediante el uso de cadenas constantes, cálculos numéricos y expresiones escalares.

8. Ordena los registros proyectados mediante el comando ORDER_BY para ordenar en cualquier orden y mezcla de ordenamientos, las columnas emitidas.

9. Implementa la junta, la junta a través de integridad referencial, equijunta y la junta de más de dos archivos.

10. Recurre al uso de alias de archivos para distinguir la calificación de campos de archivos repetidos.

11. Coloca consultas dentro de otras consultas en modo anidado en la cláusula WHERE preguntando por las respuestas obtenidas en las subconsultas por medio de los operadores mencionados anteriormente. Las subconsultas pueden emplear funciones agregadas y DISTINCT.

12. Soporta el uso de subconsultas en la cláusula HAVING.

13. Tiene la capacidad de usar los operadores EXISTS, ANY, SOME, ALL e IN.

14. Evalúa consultas con las cláusulas UNION, UNION_ALL, MINUS.

15. Responde de manera modesta a las operaciones del lenguaje de manipulación de datos LMD de INSERT, UPDATE y DELETE, ya que se implementaron de manera sencilla.

16. Guarda en disco salidas producidas por medio del comando SAVE_AS.

17. Liga archivos de enunciados LCA por medio del comando CALL_MQL.

II.6. EXPLOTADORES

Los explotadores son procesos que sirven para obtener beneficios de los componentes diseñados por los constructores. Estos procesos no se ejecutan permanentemente en ADSI, como sucede con los administradores, y se usan en todas las fases del desarrollo de un sistema de información: prueba, implementación y puesta a punto.

II.6.1. EXPPAN

El explotador de pantallas es un proceso que toma las pantallas diseñadas por el constructor de pantallas (CONPAN) y actúa de acuerdo a la aplicación que tiene definida. Cada aplicación implica una conducta particular para el explotador. Por medio de EXPPAN el usuario puede enlazarse con el ambiente y acceder a todos los recursos declarados en las hipermedias definidas en las pantallas. EXPPAN está documentado de forma tal que en cualquier momento el usuario puede ser asistido. La principal ayuda se proporciona al oprimir simultáneamente las teclas CTRL/Y. Las teclas de función están programadas de la siguiente manera:

F1. Muesta la ayuda asociada al campo.

F2. Llena el campo con el caracter de limpieza declarado.

F3. Llena el campo con el valor por omisión.

F4. Despliega nuevamente el contenido de la pantalla.

F5. Cambia la aplicación de la pantalla a alguna aplicación permitida.

F6. Muestra las hipermedias asociadas y se prepara para ejecutarlas.

F7. Activa/Desactiva la captura en notación científica. Únicamente para campos con punto y se deplaza dentro del campo sin atender la máscara numérica definida.

F8. Entra al menú de manejo de memoria común. Esta memoria mantiene información mientras el usuario se encuentra en sesión. Cierta información es depositada en esta área como resultado de invocación a ciertas hipermedias ejecutadas.

F9. Habilita/Deshabilita la posibilidad de conservar el contenido de los campos en la captura (esto ocurre cuando aparece un asterisco enseguida de 'EXPPAN' en la parte superior central de la pantalla). Cuando no aparece el asterisco, indica que se borra el contenido de los campos, cada vez que se pide el primer campo de la pantalla (esto se hace de modo por omisión).

II.6.2. EXPREP

El explotador de reportes es un proceso que toma los informes diseñados por el constructor de reportes (CONREP). Cada informe es interpretado por EXPREP de acuerdo a la programación de líneas, cortes y accesos a los archivos del Ambiente. La invocación de EXPREP se hace generalmente a través de las hipermedias definidas en las pantallas por medio de EXPPAN y se asocia a una pantalla de control de ejecución. Al solicitar la generación del informe, EXPPAN pregunta dónde se desea mostrar la salida a obtener (en pantalla o en impresora), y una vez que el informe se genera, EXPREP precisa saber a qué impresora se enviará, mostrando si así se desea, la lista de impresoras contenidas en el archivo LPT.INFO. Ver anexo M.

II.6.3. EXPELC

El explotador de enunciados (macros) del lenguaje de consulta del ambiente LCA, es un proceso que toma como entrada a los archivos de texto que contienen las macros LCA, sustituye los parámetros declarados, interpreta las macros y las ejecuta produciendo una salida, que bien puede aparecer formateada en pantalla, o depositada en algún archivo para usarse como entrada en algún informe. La invocación de EXPELC se hace generalmente a través de las hipermedias definidas en las pantallas por medio de EXPPAN y se asocia a una pantalla de control de ejecución. El proceso que se invoca se llama EJEELC y este a su vez llama a EJEPRM para que sustituya parámetros y a EXPELC para que procese las macros.

II.6.4. EXPPAU

El explotador de programas de aplicación es todo programa elaborado por los desarrolladores del sistema de información y que no pertenece al ambiente como administrador ADMPAA. En estos programas se puede definir e invocar toda clase de funciones y recursos del ambiente, tal como sucede con los ADMPAAS, pero que no necesitan estar permanentemente en operación como ocurre con los administradores. Las rutinas comunes que se describen en el anexo K, indican la forma de hacer uso de los recursos de ADSI. La invocación a un proceso EXPPAU, se hace generalmente a través de las hipermedias definidas en las pantallas por medio de EXPPAN y se asocia a una pantalla de control de ejecución.

II.7. UTILERÍAS

Las utilerías son recursos que sirven para apoyar al diseño y explotación de los elementos del sistema. Estos procesos operan mientras se invocan.

II.7.1. UTIORD Y UTIMORD

Los recursos denominados UTIORD y UTIMORD son programas óptimos para ordenar archivos. Estos procesos fueron desarrollados debido a que en la mayoría de los SO las

utilerías de ordenamiento no son capaces de combinar tipos de ordenamientos, o sea, algunos campos ascendentes y otros descendentes. Estos procesos usan el algoritmo quicksort. Para análisis del algoritmo quicksort, ver referencia bibliográfica [13].

La invocación de UTIORD y UTIMORD se hace generalmente a través de EXPELC, el cual se invoca en las hipermedias definidas en las pantallas por medio de EXPPAN y se asocia a una pantalla de control de ejecución.

II.7.2. UTIEDIT Y UTIEDIC

Los recursos denominados UTIEDIT y UTIEDIC son programas utilizados para mostrar salidas obtenidas en consultas, informes y ayudas. La idea original de estos productos surge de la necesidad de contar con editores de texto capaces de mostrar archivos que contienen grandes volúmenes de información, permitiendo editar y buscar cadenas. La utilería comercial que acompaña a los SO para esta actividad se le conoce como MORE.

La utilería UTIEDIT se utiliza cuando la salida que se presenta está en modo texto.

La utilería UTIEDIC se utiliza cuando la salida que se presenta está en modo binario, y requiere una interpretación de valores numéricos a texto, facilitando la implementación de máscaras de salida adecuadas.

La invocación de UTIEDIT y UTIEDIC se hace generalmente a través de EXPPAN y EXPELC.

II.7.3. RUTINAS COMUNES

Las rutinas comunes son un conjunto de rutinas programadas de modo que el usuario puede considerarlas dentro de su código al elaborar programas de aplicación para:

1. Establecer comunicación con ADSI.
2. Acceder archivos.
3. Usar recursos de administradores ADMPAA.
4. Ordenar archivos.
5. Enviar informes a impresión.

Una vez que se elabora un programa se compila y se puede generar el tipo de programa deseado con la utilería UTIGEN. Para mayor información sobre los parámetros de las rutinas comunes, existe en ADSI documentación adecuada; una lista de las rutinas aparece en el anexo K.

II.7.4. UTIGEN

La utilería UTIGEN es un proceso que encadena los programas objeto de una aplicación. Cada proceso de ADSI es código ejecutable que se forma de varios archivos de programas objetos previamente compilados en lenguaje C. La forma en que se ligan se define en UTIGEN y para generar algún proceso EXPPAU y EXPPAA de ADSI basta con invocar esta utilería.

II.7.5. INSTALA

La utilería INSTALA es un proceso que se emplea para instalar en forma permanente en disco duro el ambiente ADSI y viene grabado en diskettes en el formato estándar de la utilería del SO "tar" Para efectuar la instalación se deben seguir 3 pasos:

1. Recuperar del diskette el archivo INSTALA y salvarlo en algún directorio en disco duro.

2. Ejecutar el archivo INSTALA recuperado en el paso anterior.

3. Eliminar el archivo INSTALA del directorio empleado en el paso 1.

Esta es la oportunidad para mencionar la forma en que ADSI se desplaza dentro de los directorios del SO. Como se comentó en la introducción, ADSI funciona en sistemas operativos multiusuario y multitarea, donde a su vez puede instalarse una red e incluso el multiproceso. Para ello, ADSI usa el formato de direccionamiento a directorios de la manera siguiente:

a) Dos diagonales seguidas "//"

b) El número de nodo de la red. Cuando existe una sola computadora se emplea el número de nodo 1.

c) Una diagonal "/".

d) Una cadena de dieciocho caracteres:

d.1) La clave del producto en cinco caracteres.

d.2) Un guión (underline) "_".

d.3) El número de serie en doce caracteres.

e) Una diagonal "/"

Al directorio descrito anteriormente se le denomina en ADSI el directorio raíz (por ejemplo //1/01011_012010000001/). El directorio raíz está formado por cuatro directorios a saber y utilizando el ejemplo:

1. //1/01011_012010000001/dir_dat/. Se usa para mantener archivos:

    1.1. Datos (*.dat).

    1.2. Índices (*.pri, *.sec, *.aut).

    1.3. Mensajes de error y advertencia (MSG_ERR.ERR).

    1.4. Bitácoras (*.LOG) y (*.ADM).

    1.5. Configuración de impresoras (LPT.INFO).

    1.6. Configuración de parámetros (*.SYS).

    1.7. Diccionario de datos en texto (*.DD).

    1.8. Diccionario de datos compilado (*.DIC).

2. //1/01011_012010000001/dir_frm/. Se usa para mantener archivos:

    2.1. Diseños de pantallas (*.FRM).

    2.2. Diseños de reportes (*.REP).

    2.3. Macros de enunciados de consulta LCA (*.MQL).

    2.4. Ayudas (*.MHELP).

    2.5. Programas compilados del ambiente (*.o).

3. //1/01011_012010000001/dir_exe/. Se usa para mantener archivos de los programas ejecutables de ADSI. Todos los programas ejecutables están en letras MAYÚSCULAS.

4. //1/01011_012010000001/dir_wor/. Se usa para mantener archivos de paso o de uso de ADSI. Aquí se conservan temporalmente archivos producidos en informes; archivos para ordenar y mezclar; archivos producidos en consultas LCA. Cuando ADSI arranca la operación del ambiente, elimina todos los archivos de este directorio.

II.7.6. RESPALDA

La utilería RESPALDA es un proceso que se emplea para cumplir con la tarea cotidiana de salvar/recuperar archivos en/desde diskettes. El directorio de datos (dir_dat) es el único que se involucra en este proceso.

II.8. CONTROL

Existen básicamente dos procesos que se utilizan para controlar la operación de ADSI: iniciar y detener el ambiente. También se usa un archivo para sintonizar los parámetros; otro para declarar las impresoras; un archivo que conserva los mensajes de advertencia y errores que ocurren duante la explotación del sistema.

II.8.1. CONFIGURACIÓN

Existe un archivo que contiene algunos parámetros que permiten que ADSI optimize el uso de recursos disponibles en el SO. En el anexo L se muestra un ejemplo y se acompaña de comentarios que explican cada uno de los parámetros ahí señalados. El nombre de este archivo se forma con la clave del producto (cinco caracteres) y se termina con un punto y SYS (por ejemplo 01011.SYS). Este archivo se ubica en el directorio de datos (dir_dat).

Existe un archivo de configuración para las impresoras disponibles en la computadora o red de computadoras. El anexo M explica como elaborar un archivo de este tipo. El nombre de este archivo es constante LPT.INFO. Este archivo se ubica en el directorio de datos (dir_dat).

II.8.2. ARRANCA

Este proceso cumple con la función de iniciar la operación del ambiente ADSI. Para ello, es necesario cubrir los requisitos siguientes:

1. Entrar al SO con la cuenta de superusuario (login: root).

2. Colocar el candado de seguridad en el puerto paralelo deseado.

3. Cambiarse al directorio donde se encuentran los archivos de configuración del ambiente de acuerdo al producto y número de serie que se va a iniciar (por ejemplo //1/01011_012010000001/dir_dat/) y modificar si es necesario el número de puerto paralelo donde se asoció el candado de seguridad. En las computadoras personales PC, internamente los puertos paralelos tienen direcciones. De acuerdo a la dirección se anota en el archivo de configuración *.SYS el número de puerto paralelo: 0x278 es 1, 0x378 es 2 y 0x3BC es 3.

4. Cambiarse al directorio donde se encuentran los programas ejecutables del ambiente de acuerdo al producto y número de serie que se va a iniciar (por ejemplo //1/01011_012010000001/dir_exe/) e invocar el proceso //1/01011_012010000001/dir_exe/ARRANCA.

5. Puede abandonar la cuenta de superusuario si se desea ya que ADSI permanece operando mientras no se le ordene terminar su operación.

Los pasos que ARRANCA ejecuta son los siguientes:

1. Analiza el contenido del archivo de configuración del ambiente (*.SYS) para conocer el número de administradores ADMARC declarados.

2. Analiza el contenido del archivo de configuración de impresoras (LPT.INFO).

3. Analiza el contenido del archivo del diccionario de datos compilado (*.DIC) para conocer el contexto de archivos, campos y administradores ADMPAA declarados.

4. Investiga con el SO si el ambiente está en operación para este sistema. En caso de ya estar operando el sistema, aborta ARRANCA.

5. Envía a ejecutar al administrador ADMCEN.

6. Envía a ejecutar al administrador ADMDBD.

7. Envía a ejecutar al/los administrador/es ADMARC.

8. Envía a ejecutar al/los administrador/es ADMPAA.

9. Espera que los administradores se interconecten.

10. Termina exitosamente ARRANCA dejando en operación permanente el ambiente ADSI para el sistema en cuestión.

II.8.3. CONTROLA

El proceso CONTROLA permite que eventualmente el usuario responsable del sistema, pueda conocer información de la operación de ADSI o tomar algunas acciones importantes para el mejor desempeño del mismo.

Es necesario que el responsable del sistema entre al SO con la cuenta de superusuario e invoque el proceso (usando el ejemplo mencionado en pasos anteriores) //1/01011_012010000001/dir_exe/CONTROLA y de este modo CONTROLA se pone en contacto con ADMCEN del ambiente del sistema correspondiente.

Los comandos que el responsable del sistema puede invocar desde CONTROLA son los siguientes:

a) usuarios. Muestra una lista de los procesos que están incorporados al ambiente.

b) procesos. Similar a a).

c) salida. Abandona CONTROLA pero el ambiente ADSI continúa en operación.

d) info. Muestra la clave del producto, número de serie, puerto paralelo donde se encuentra el candado de seguridad del sistema, número máximo de licencias corporativas que se pueden usar, el número de licencia corporativa que está operando, el número de año en operación. También señala cuando es necesario ejecutar 'rehacer' para concluir el cierre de archivos.

e) rehacer. Cuando la sesión anterior terminó sin el comando 'termina_ambiente', seguramente los archivos no fueron cerrados correctamente o las últimas operaciones no fueron concluidas con el punto de verificación (checkpoint). Para concluir la terminación de la sesión anterior es necesario ejecutar el presente comando. ADSI conoce cuáles archivos del ambiente no se cerraron correctamente, por lo tanto, lleva a cabo una recuperación desde los archivos de datos dañados, regenerando todos los índices correspondientes e inmediatamente procesa todas las transacciones que aparecen después del último punto de verificación (checkpoint) del archivo bitácora. Una explicación más adecuada aparece posteriormente en el capítulo referente a manejo de transacciones.

f) optimiza. Permite que el usuario mejore la eficiencia de todos los archivos de datos e índices del ambiente. Este proceso se lleva a cabo en tres fases: a) a partir de los archivos de índices de llaves primarias se generan archivos de datos secuenciales de paso; b) se borran los archivos de datos e índices del ambiente; c) se generan los archivos de datos e índices a partir de los archivos secuenciales de paso. Mientras esto sucede, ningún usuario puede estar conectado a ADSI.

h). termina_ambiente. La operación del ambiente ADSI debe terminar. El único requisito es que no existan usuarios conectados al ambiente en el momento actual.

II.8.4. MENSAJES DE ERROR

Existen alrededor de 850 mensajes almacenados en el archivo MSG_ERR.ERR del directorio de datos (dir_dat). Estos mensajes se emplean para advertir, aconsejar o justificar ciertas acciones que ADSI detecta durante el arranque o terminación del ambiente, el uso de constructores o explotadores, en la prueba, implementación, puesta a punto o liberación de un sistema de información.

III. DEFINICIÓN DE DATOS

III.1. ESQUEMA GENERAL

Cuando se inició el diseño de ADSI se pensó en una herramienta de trabajo que permitiera desarrollar sistemas de información para capturar, almacenar y recuperar datos de modo sencillo y práctico, donde se elaboraran la menor cantidad de programas posibles y se obtuviera información proporcionando los mínimos datos posibles. Por supuesto que no se tenía la menor idea de cómo hacerlo, aunque el objetivo fundamental era hacer algo novedoso, ingenioso y de ejecución muy rápida basado en estructuras y datos.

De aquí surge la necesidad de contar con un modelo que describa los datos y las estructuras que conforman a ADSI, lo cual, a su vez, es indispensable para el desarrollo de toda aplicación, cuyas etapas se desglosan a lo largo de este capítulo.

III.2. ANÁLISIS Y DISEÑO DE UN SISTEMA DE INFORMACIÓN

En el inicio del desarrollo de un sistema de información se efectúa un trabajo de investigación acerca del ámbito relacionado. Se cuestiona sobre el planteamiento del problema, las metas y objetivos que se persiguen. El conocimiento de la situación actual es importante para el descubrimiento de información constante y variable, la influencia que se recibe del medio y el impacto que produce en las entidades que circundan. En conclusión, el analista debe ser capaz de identificar cada componente del problema y clasificarlo de acuerdo con un criterio que facilite el diseño de una solución. Por lo general se espera que el analista de sistemas sea capaz de dominar los términos y vocablos del problema; entender a detalle todas las situaciones y condiciones que se presentan en el contexto de la realidad; manifestar soluciones y sugerencias ante cuestionamientos prácticos como una demostración del dominio que se tiene acerca del problema. A pesar de lo anterior, la experiencia señala que no es obvio obtener un buen diseño que resuelva los planteamientos de un problema y un asunto fundamental que se presenta ante esta situación: dado un conjunto de datos que se va a representar en las bases de datos, ¿Cómo se opta por una estructura lógica adecuada para los mismos? En otras palabras, ¿Cómo se decide qué relaciones se necesitan y qué atributos deben tener? He ahí el problema del diseño de un sistema de información. Lo aconsejable es atender a la intuición y evitar la redundancia en tanto sea posible.

III.2.1. CONCEPTUALIZACIÓN GENERAL

Una vez obtenido el diseño del sistema es necesario apoyarse en herramientas que reflejen en un lenguaje sencillo y claro, cada componente del mismo.

El modelo *entidad-vínculo E-V* es una herramienta adecuada para cumplir con esta tarea (ver [1] y [7]). El propósito de E-V es permitir que se escriba la descripción del esquema conceptual de una empresa sin poner atención a la eficiencia ni al diseño de las bases de datos físicas. En el modelo E-V se usa una terminología que permite identificar los componentes del sistema. *Entidad* es cualquier cosa que existe y que se puede distinguir entre las cosas de su misma especie. Un grupo de entidades similares se le llama *conjunto de entidades*. Las características de los miembros de un conjunto de entidades se les conoce como *atributos*. El conjunto de valores que un atributo puede tomar se le llama *dominio*. El atributo o conjunto de atributos cuyos valores identifican individualmente a cada entidad dentro de un conjunto de entidades se le llama *llave*. El uso de jerarquías entre conjuntos de entidades es importante para la *generalización y herencia de atributos*. La relación entre conjunto de entidades es necesaria para determinar la *funcionalidad entre las entidades* (muchos-uno y muchos-muchos).

La manera más recomendada para conceptualizar un sistema, es utilizar la descripción tradicional de archivos (nombre del archivo y lista de atributos o campos) y apoyarse en el modelo E-V o en algún otro recurso que muestre gráficamente la interrelación entre archivos y datos.

III.2.2. ENTRADAS/SALIDAS

La determinación de las fuentes de información del sistema traen como consecuencia, el refinamiento de la interrelación de los datos hacia el interior del mismo. El flujo de la información a lo largo del sistema y su encausamiento hacia el exterior, permiten al analista de sistemas mejorar la visión de la conceptualización general.

III.2.3. TIPOS DE DATOS

Una vez obtenida la conceptualización general del sistema y habiendo elaborado esquemas que representen gráficamente la interrelación entre archivos y campos, es necesario asignar tipos de datos. Los tipos de datos son la forma interna en que ADSI trata los contenidos de cada dato del sistema y estos son equivalentes a los tipos de datos del lenguaje C. ADSI cuenta con seis tipos de datos internos:

a) Números enteros con signo (1 = int).

b) Números enteros sin signo (2 = unsigned).

c) Números enteros largos con signo (3 = long).

d) Números flotantes con signo (4 = float).

e) Números dobles con signo (5 = double).

f) Caracter (6 = char).

La capacidad de representación de cada tipo de dato está en función de la computadora.

III.2.4. TIPOS DE LLAVES

Se presentan los dos tipos de llaves convencionales que sirven para identificar registros en los archivos. Se agrega el terminador que se emplea para nombrar los archivos en el directorio de datos (dir_dat):

a) Llave primaria. Llave principal que identifica individualmente un registro y puede estar formada por un campo o varios campos (*.pri).

b) Llave secundaria. Llave alterna que accede a un registro y puede estar formada por un campo o varios campos (*.sec).

Se ofrece el tipo de llave no convencional denominada automática, que permite llevar un conteo automático de registros que tienen el mismo valor en su llave, y su terminador para nombrar los archivos en el directorio de datos es (*.aut).

III.2.5. FORMATOS

Los formatos de un sistema de información se emplean para recopilar datos directamente de las fuentes, en la captura de datos, informes a emitir, consultas por terminal o consultas impresas así como el intercambio de información con otros sistemas.

III.2.6. OPTIMIZACIÓN DEL DISEÑO (FORMAS NORMALES)

Para facilitar el diseño de sistemas de información existen modelos conocidos. Quizá el modelo más utilizado por los sistemas clásicos de bases de datos es el modelo relacional. Bajo este modelo se han definido restricciones para las relaciones participantes denominadas *formas normales*. Ver [1], [7] y [8].

Se debe optimizar el diseño de un sistema cuando se representan múltiples entidades y relaciones y entre los atributos se crean dependencias, como la *dependencia funcional* (DF), ocasionando que una modificación (inserción, actualización, borrado) realizada en un punto de las bases de datos, llegue a afectar alguna área que no se deseaba tocar.

Algunos de los principales problemas que se presentan son:

a) Inserción anómala. Este problema surge cuando al insertar el valor de un atributo es necesario insertar el valor de otros atributos debido a que en el esquema son dependientes, aunque en la realidad no sean necesarios.

b) Eliminación anómala. En este caso al borrar un dato se borran otros datos dependientes aunque quizá se requieran después.

c) Actualización anómala. Si un atributo aparece en forma innecesaria en varios puntos del esquema, al modificar alguno de sus valores, es necesario modificarlo en todos los puntos donde aparece, con el costo y posibilidad de error que esto implica. Esto forma parte del problema de minimizar la redundancia como uno de los objetivos del diseño lógico.

El proceso de *Normalización* permite disminuir las anomalías anteriores y busca reducir a su forma más simple el esquema de las bases de datos.

El proceso de Normalización se basa en dos operaciones básicas sobre las tablas relacionales: Proyección y Unión, mediante las cuales se puede cambiar la estructura de los esquemas.

a) Proyección. Mediante la proyección se pueden obtener nuevas tablas a partir de una tabla dada, extrayendo las columnas que se requieren y eliminando los renglones repetidos.

b) Unión. Mediante la unión natural se toman dos tablas relacionales y a partir de ellas se obtiene una nueva tabla relacional. Para poder realizar la unión se requiere que existan dos atributos que tengan los mismos valores posibles (mismo dominio), y que uno de los atributos corresponda a una tabla y el otro atributo a la otra tabla. Entonces la unión se realiza haciendo que cuando en los dos atributos se tiene el mismo valor, se concatenen los dos renglones.

A continuación se presentan los pasos del proceso de normalización:

1. Primera Forma Normal (1FN). La normalización en su primera etapa busca que en cada tabla no existan 2 registros con la misma información, que no exista un atributo con un valor constante en todos sus registros y que no existan grupos de repetición (conjunto de atributos que se repiten para la misma entidad). Un esquema que cumple con lo anterior se dice que está en 1FN y a cada tabla se le denomina tabla atómica.

   Para conducir una tabla a 1FN se efectúan los siguientes pasos:

   a) Si se tienen dos registros con la misma información, se elimina uno.

   b) Si se tiene un atributo con valor constante se elimina y se almacena en forma independiente.

   c) Si se tienen grupos de repetición se proyecta la llave del registro y los atributos del grupo de repetición.

Tabla original

Num-Emp	Nombre	Función1	Función2	Función3	Teléfono
1	PEDRO	B DE D	SISTEMAS	OPERACIÓN	375
2	JUAN	B DE D	OPERACIÓN		375
3	LUIS	ADMIN			375
1	PEDRO	B DE D	SISTEMAS	OPERACIÓN	375
4	MARIO	OPERACIÓN			375

El registro repetido es el número 4.
Los atributos repetidos son función1, función2 y función3.
El atributo constante es el teléfono.

Tabla sin registros repetidos y sin columna constante

Num-Emp	Nombre	Función1	Función2	Función3
1	PEDRO	B DE D	SISTEMAS	OPERACIÓN
2	JUAN	B DE D	OPERACIÓN	
3	LUIS	ADMIN		
4	MARIO	OPERACIÓN		

Tabla en 1FN

Num-Emp	Nombre
1	PEDRO
2	JUAN
3	LUIS
4	MARIO

Tabla en 1FN

Num-Emp	Función
1	B DE D
1	SISTEMAS
1	OPERACIÓN
2	B DE D
2	OPERACIÓN
3	ADMIN
4	OPERACIÓN

Constante

Teléfono
375

2. Segunda Forma Normal (2FN). Se dice que un atributo B es dependiente funcional completo de otro atributo A si B es dependiente funcionalmente del total de A pero no de ninguna subparte de A. Una tabla está en 2FN si está en 1FN y cualquier atributo no-primo es dependiente funcional completo de una llave candidata.

Dada una tabla que está en 1FN pero no en 2FN, se puede pasar a 2FN observando cual es el conjunto de atributos primos de los cuales depende el atributo problema y se proyecta por un lado la tabla sin el atributo problema y por otro lado una tabla con el atributo problema y los atributos de los cuales depende.

Tabla original

Num-Profr	Salón	Horario	Nombre
1	301	10	LUIS
1	302	11	LUIS
2	301	12	JUAN
3	302	10	PEDRO

Num-Prof + Salón → Horario.
Num-Prof → Nombre.

Tabla en 2FN

Num-Profr	Salón	Horario
1	301	10
1	302	11
2	301	12
3	302	10

Tabla en 2FN

Num-Profr	Nombre
1	LUIS
1	LUIS
2	JUAN
3	PEDRO

3. Tercera Forma Normal (3FN). Un atributo C depende transitivamente de un atributo A, si:

   a) A→B, B→C y además,

   b) B≠A, B no determina A o C no determina B para algún B.

Una tabla está en 3FN si está en 2FN y ningún atributo depende transitivamente de otro. Si algún atributo depende transitivamente de otro, nuevamente se ejecuta una proyección. Si A→B, B→C entonces se obtienen dos tablas, una donde esté A y B y otra con B y C.

Tabla original

Num-Profr	Nombre	Clasif	Sueldo
1	LUIS	E1	500
2	JUAN	E2	600
3	PEDRO	E1	500
4	MARIO	E3	700

Num-Prof → Nombre.
Num-Prof → Clasif.
Num-Prof → Sueldo.
Clasif → Sueldo.
La transición existente: Num-Prof → Clasif → Sueldo.

Tabla en 3FN

Num-Profr	Nombre	Clasif
1	LUIS	E1
2	JUAN	E2
3	PEDRO	E1
4	MARIO	E3

Tabla en 3FN

Clasif	Sueldo
E1	500
E2	600
E3	700

4. Forma Normal de Boyce-Codd (FNBC). Se dice que una tabla está en la forma FNBC si cualquier determinante es una llave candidata. Esto significa que si el esquema no está en FNBC se debe a que existen tablas que tienen deteminantes que no son llaves candidatas.

   Para pasar una tabla a la FNBC se tiene que encontrar el determinante problema (que no es llave) y los atributos que dependen funcionalemente de el y entonces por un lado se proyecta una tabla sin los atributos que dependen del determinante problema y por otro lado se proyecta una tabla con el determinante problema y sus atributos dependientes.

Tabla original

Num-Profr	Nombre	Materia	Depto
1	LUIS	FIS1	FIS
2	JUAN	MAT1	MAT
3	PEDRO	FIS2	FIS
4	MARIO	FIS1	FIS

Num-Prof → Nombre.
Materia → Depto.
Materia es un determinante pero no es llave candidata.

Tabla en FNBC

Num-Profr	Nombre	Materia
1	LUIS	FIS1
2	JUAN	MAT1
3	PEDRO	FIS2
4	MARIO	FIS1

Tabla en FNBC

Materia	Depto
FIS1	FIS
MAT1	MAT
FIS2	FIS

5. Cuarta Forma Normal (4FN). Se dice que un atributo A multidetermina a un atributo B si para cada valor de A se obtiene un conjunto de posibles valores de B. También se dice que existe una dependencia multivaluada de B en A.

   Una tabla relacional está en 4FN si para cualquier dependencia multivaluada A →→ B entonces A es una llave candidata, o sea que todos los atributos dependen (funcionalmente o multivaluadamente) de A. Si una tabla no está en 4FN quiere decir que dado que A→→B existe algún atributo C al que A no determina C. En este caso se proyecta buscando que desaparezca el problema.

Tabla original

Num-Profr	Materia	Salón
1	FIS1	301
1	FIS1	301
1	FIS2	302
2	MAT1	303
2	FIS2	302
3	MAT1	303

Num-Prof →→ Materia.
Num-Prof no determina Salón.
Materia → Salón.

Tabla en 4FN

Num-Profr	Materia
1	FIS1
1	FIS1
1	FIS2
2	MAT1
2	FIS2
3	MAT1

Tabla en 4FN

Materia	Salón
FIS1	301
FIS2	302
MAT1	303

6. Quinta Forma Normal (5FN). Recordemos que una tabla T se normaliza óptimamente si se puede proyectar en un conjunto de tablas, tales que, al unirse se pueda obtener T; se dice que el conjunto de tablas tienen una dependencia de unión entre sí.

   Un esquema está en 5FN, si cualquier dependencia de unión es implicada por las llaves candidatas de la relación R.

III.3. ELABORACIÓN DEL DICCIONARIO DE DATOS

Una vez que se cuenta con el esquema conceptual del sistema, es necesario codificar el esquema en un formato fácil de interpretar por ADSI.

La información que aparece en el diccionario de datos es:

a) Lista de atributos (indicando nombre, tipo y longitud).

b) Lista de archivos (nombre del archivo y lista de atributos que lo integran, así como los índices asociados).

c) Lista de administradores de programas de aplicación del Ambiente ADMPAA.

La forma en que se elabora el diccionario de datos se menciona en el proceso CONDIC del captítulo II. En el anexo N se muestra un ejemplo completo.

III.4. INTEGRIDAD DE DATOS

La integridad es todo aquello referente a la conservación auténtica e inviolabilidad de los datos.

III.4.1. RANGOS Y VALIDACIONES

Ningún componente del valor de una llave debe ser nulo. Es necesario que las entidades o eventos de un sistema sean plenamente identificados, y las llaves primarias cumplen con la función de identificación única en una BD. Sin embargo, no es suficiente con poner atención a los campos llaves de un archivo, también se requiere la asignación de rangos de valores que puede tomar un dato cualquiera. En ADSI se permite que se usen hipermedias para la evaluación de rangos de valores en el momento de la captura de información. Otras validaciones pueden hacerse en los programas de aplicación ADMPAA y ADMPAU.

III.4.2. INTEGRIDAD REFERENCIAL

Un dominio específico puede designarse como primario sí y solo sí existe alguna llave primaria de un único atributo definida sobre ese dominio. Cualquier relación que incluya un atributo que se defina sobre un dominio primario debe obedecer a la restricción siguiente.

Sea D un dominio primario, y sea R_1 una relación con un atributo A que se define sobre D. Entonces, en cualquier instante dado, cada valor de A en R_1 debe ser o bien (a) nulo, o bien (b) igual a V, por ejemplo, donde V es el valor de la llave primaria de alguna tupla de alguna relación R_2 (R_1 y R_2 no son necesariamente distintas) con llave primaria definida sobre D. (Nótese que R_2 debe existir, por definición de dominio primario. Obsérvese también que la restricción se satisface de modo trivial si A es la llave primaria de R_1.)

Se llama *llave foránea* a un atributo como A. Las llaves primarias y foráneas proporcionan un medio para representar asociaciones entre tuplas, nótese sin embargo, que tales atributos de "asociación" no siempre son llaves. El acceso a un registro de un archivo no debería restringirse a llaves primarias, sin embargo, se facilita enormemente su manipulación.

En ADSI existe la implementación de cuidar la integridad de valores no nulos y la integridad referencial. Por medio del diseño de pantallas CONPAN se pueden definir restricciones de integridad a los valores de los datos. Con el constructor de reportes CONREP se pueden definir restricciones para datos a mostrar. El administrador ADMDBD por sí mismo tiene implementada la integridad referencial y cuando no es posible resolverla, existe la posibilidad de expresarla en CONPAN usando la hipermedia de acceso a archivos.

III.5. INTEGRACIÓN DEL DICCIONARIO DE DATOS AL AMBIENTE

La elaboración del diccionario de datos es la consumación del trabajo de análisis y diseño de un sistema (abstracción). Su creación se lleva a cabo por medio de un editor de textos para producir un archivo con extensión (*.DD) el cual es entendible al usuario. Sin embargo, ADSI necesita de un diccionario de datos revisado y compilado. Por eso, el proceso CONDIC hace esta función produciendo un archivo con terminación (*.DIC). Cuando se inicia el ambiente por medio del proceso ARRANCA, se toma el diccionario de datos (*.DIC) y a partir de el se elaboran las tablas de estructuras internas del sistema y se disponen criterios definidos en ADMDBD principalmente para la operación de ADSI.

IV. ORGANIZACIÓN FÍSICA DE LOS DATOS

A lo largo del escrito se mencionan operaciones sobre los archivos de ADSI, con la finalidad de consultar, insertar, borrar y actualizar información, pero hasta ahora no se ha comentado nada acerca de la organización del almacenamiento.

Se inicia con la concepción de que existen organizaciones fisicas de datos basadas en llaves o estructuras de *índices primarios*, por medio de las cuales se pueden localizar registros rápidamente al proporcionar valores para un conjunto de campos que constituyen las llaves. También se habla de cómo estas estructuras se modifican para localizar registros por medio de llaves secundarias.

Dentro del ambiente ADSI, son los administradores de archivos (ADMARC) quienes efectúan las tareas relacionadas con la organización fisica de los datos.

IV.1. CAMPOS, REGISTROS Y ARCHIVOS

La BD fisica es una colección almacenada de registros, donde cada registro consiste de uno o más campos. Los valores de los campos son de un tipo elemental definido por ADSI (capítulo III). Internamente se incluye un tipo de dato llamado apuntador el cual es una referencia a un registro. Los registros se usan para almacenar fisicamente cada uno de los objetos básicos de datos de acuerdo al modelo. Por ejemplo:

1. Una tupla puede ser almacenada como un registro; cada componente de la tupla es almacenada como un campo.

2. Un registro lógico, usado en el modelo de red y jerárquico, puede ser almacenado como un registro.

Ahora bien, normalmente se trata con colecciones de registros que tienen el mismo número de campos, y cuyos correspondientes campos tienen el mismo tipo de datos, nombre de campo, e incluso un significado intuitivo. Por ejemplo, los registros que representan las tuplas de una relación tienen un campo para cada atributo de esa relación, y el campo para cada atributo tiene el tipo de dato asociado con ese atributo. Se le denomina a la lista de nombres de campos y sus correspondientes tipos de dato como el *formato de un registro*.

Se denota el término *archivo* para una colección de registros del mismo formato.

La memoria es un medio de almacenamiento rápido pero volátil que no puede soportar el almacenamiento de grandes o inmensos volúmenes de información. El almacenamiento en disco se considera ilimitado ya que los discos pueden agregarse en la medida que aumenten las necesidades, sin embargo, su acceso es lento. La combinación adecuada de estos dos medios de almacenamiento a través del uso de técnicas y algoritmos, es la solución para conjugar capacidad de almacenamiento y velocidad de acceso.

IV.2. BLOQUES, COSTO DE ACCESO Y APUNTADORES

Un factor que influye en la manera de calcular el costo de acceso a los datos en disco es el uso de bloques. Los *bloques* son una partición del almacenamiento en un número sustancial de caracteres, por decir de 512 hasta 4096, y la transferencia de datos entre disco y memoria se hace por medio de bloques completos. Para que los bloques sean un factor que beneficie al desempeño de acceso a los datos, es recomendable que los registros sean de longitud pequeña para que el mayor número de registros se almacenen en un bloque.

La unidad de costo para operaciones sobre bloques fisicos se le denomina *acceso a bloques*, lo cual implica la lectura de un bloque desde disco o la escritura de un bloque hacia disco. Se supone que los cálculos que se efectúan sobre un bloque (en memoria) no requieren tanto tiempo como la transferencia de un bloque entre memoria principal y memoria secundaria. En realidad, no siempre que se requiera leer o escribir sobre un bloque, implica la transferencia fisica entre memoria y disco. Esto se debe a que el SO o ADSI almacena en bloques de memoria (buffers) cierto número de bloques de la BD, manteniendo dichos bloques (buffers) en alguna parte de la memoria por algún tiempo estratégico, dadas las consideraciones del espacio y el recuerdo que pueda tener de ellos. Sin embargo, a menudo no se puede predecir si algún bloque que se requiere está almacenado en memoria, ya que esto depende de factores que van más allá de nuestro control, tal como lo que otros procesos están haciendo en el sistema al mismo tiempo o lo que otras operaciones de la BD están ejecutando. Por otro lado, el tiempo para acceder un bloque particular en un disco depende del lugar donde se hizo el último acceso sobre ese disco, debido a que se requiere de un tiempo estimado para mover las cabezas desde el cilindro donde estaba el último acceso, al cilindro donde se requiere posicionar.

En resumen, se supone que existe una probabilidad fija para que al momento de usar un bloque que se requiera, implique una transferencia entre memoria y disco. También se supone que el costo de un acceso no depende de las operaciones hechas anteriormente. Con estas suposiciones, asumimos que el acceso a un bloque cuesta lo mismo que el acceso a cualquier otro y así se justifica el acceso a bloques como la medida de tiempo de operación.

Con respecto a los apuntadores, se dice que en esencia, un apuntador p a un registro r es dato suficiente para localizar r "rápidamente" Hay dos clases de apuntadores que emplea ADSI: direcciones absolutas para la memoria, y para el acceso a disco, el sistema de direcciones al principio de un registro. Posteriormente se muestran algunas estructuras empleadas por ADMARC para organizar los datos en memoria y en disco.

IV.3. ARCHIVOS DE DATOS

Cuando el ambiente ADSI inicia su operación (proceso ARRANCA), debe conocer toda la información referente a los archivos del ambiente. Para ello, toma el archivo que contiene el diccionario de datos (*.DIC en el capítulo III.3), e investiga lo siguiente:

a) Número de archivos a controlar.

b) Nombre de cada archivo.

c) Formato de los registros de cada archivo.

d) Información de llaves por cada archivo (número, tipo y longitud de cada campo de las llaves).

También se analiza el contenido del archivo de configuración (*.SYS) (Ver capítulo II.8.2), para conocer el número de administradores de archivos (ADMARCs) que debe existir en el ambiente y la distribución de carga de trabajo (archivos) entre los ADMARCs. Por cada archivo que aparece en el diccionario, se abre un archivo de datos con extensión (*.dat).

IV.3.1. ESTRUCTURA

Al inicio de cada archivo de datos se inserta una estructura de encabezado que contiene la siguiente información:

1. Nombre del producto.
2. Número de archivo que le corresponde dentro de ADSI.
3. Nombre del archivo.
4. Longitud de registro (número de caracteres).
5. Número máximo de registros a almacenar.
6. Número actual de registros activos (registros que no se han dado de baja).
7. Número de licencia corporativa (por cada sistema puede haber 1-65,000 licencias corporativas).
8. Número de año al que pertenecen los datos.
9. Número de serie del sistema.

IV.3.2. ALMACENAMIENTO

Los archivos de datos se abren en modo lectura/escritura. Si no existen, se crean. Si ya existen, se abren para actualización. En seguida de la estructura de encabezado se agregan los registros en modo secuencial conforme van llegando. Por cada registro de datos que se incluye, los ADMARC agregan un campo de control (CAMPOCON) de 16 bits (short int) que contiene lo siguiente:

a) el bit 15 (ACTIVO) indica si el registro está activo o se ha dado de baja.

b) En los bits 0-14 (LONGREG) se señala la longitud del registro (0-32767).

Cuando se da de alta un registro, el ADMARC correspondiente carga el bit ACTIVO en 1. Cuando se da de baja un registro, el registro de datos se rellena a ceros incluyendo el bit ACTIVO. Debido a que los registros de datos se van agregando conforme llegan, no existe un orden de estos registros con respecto al valor actual de sus llaves. En los archivos de índices se mantiene el orden de los registros de acuerdo al valor de sus llaves y si se desea que los registros de los archivos de datos se mantengan ordenados de acuerdo al valor de las llaves primarias, puede recurrirse a la optimización de archivos, con la debida aclaración que aparece en el punto IV.4.

Una vez transferidos los registros de datos de disco a memoria, las operaciones pueden referirse a nivel de campo. ADSI cuenta con una operación de lectura por bloques, que mejora el tiempo de respuesta gracias a la lectura específica de algunos campos.

IV.4. ARCHIVOS DE ÍNDICES (ÁRBOLES B)

Como ya se explica en la introducción del punto IV.3, cada administrador de archivos (ADMARC) toma la información de los archivos que le corresponde controlar. En el punto anterior se explica lo relacionado a los archivos de datos y en este punto se habla de los archivos de índices.

Cada archivo definido en el diccionario de datos trae consigo la declaración de campos que se consideran llaves. Algunas pueden ser llaves primarias, llaves secundarias y otras llaves automáticas y cada llave puede estar formada de uno o varios campos. Las llaves primarias identifican individualmente a cada registro y las llaves secundarias pueden ser duplicadas. La organización interna que emplea ADSI para acceder rápidamente los datos, es por medio de Árboles Balanceados (B-tree). Ya que existe mucha documentación acerca de esta estructura de datos, no es necesario distraerse en los algoritmos que sustentan a los árboles-B, más bien, concentrarse en la manera de implementarlos, haciendo observaciones sobre la estructura y almacenamiento que permite que ADSI ofrezca un rendimiento atractivo y simple de entender. Ver [8] y [13].

Para entender la implementación de los archivos de índices, es necesario establecer que un *árbol* es una gráfica acíclica dirigida y conexa que satisface las propiedades siguientes:

1. Hay exactamente un vértice, llamado raíz, al cual no le llegan arcos.

2. A cada vértice excepto la raíz, le llega exactamente un arco.

3. Hay una trayectoria (única) desde la raíz a cada vértice.

A una colección de árboles se le denomina *bosque*.

Sea F = (V,E) una gráfica la cual es un bosque. Si (v, w) está en E, entonces v se llama el padre de w, y w es un hijo de v. Si hay una trayectoria de v a w, entonces v es un ancestro de w y w es un descendiente de v. Además, si v es diferente de w, entonces v es un ancestro propio de w, y w es un descendiente propio de v. Un vértice sin descendientes propios se llama hoja. Un vértice v y todos sus descendientes se llaman un subárbol de F. El vértice v se llama la raíz de ese subárbol.

La profundidad de un vértice v en un árbol es la longitud de la trayectoria desde la raíz a v. La altura de un vértice v en un árbol es la longitud de la trayectoria más larga desde v a una hoja. La altura de un árbol es la altura de la raíz. El nivel de un vértice v en un árbol es la altura del árbol menos la profundidad de v.

Un árbol ordenado es un árbol en el cual los hijos de cada vértice están ordenados. Cuando se dibuja un árbol ordenado, se asume que los hijos de cada vértice están ordenados de izquierda a derecha. Un árbol binario es un árbol ordenado tal que:

1. Cada hijo de un vértice se distingue como un hijo izquierdo o un hijo derecho (el hijo izquierdo tiene un valor menor al valor del vértice (padre) y el hijo derecho tiene un valor mayor al vértice (padre)) , y

2. Ningún vértice tiene más de un hijo izquierdo ni más de un hijo derecho.

A partir de las definiciones anteriores donde formalmente se describe un árbol, la implementación hecha en ADSI aplica una modificación. En lugar de utilizar un vértice el cual representa solamente el valor de una llave y contando a lo más con dos hijos (el vértice + 1), se emplea un bloque con capacidad para almacenar varios vértices m (m llaves) al que se le denomina *página*, y el número de hijos consiste en a lo más (m + 1).

Se espera que por implementación de los árboles-B, en cada página estén contenidos a lo menos n llaves (n = m / 2), a excepción de la página raíz que puede contener una sola llave. En ADSI se utiliza el valor de n = 2 y m = 4.

IV.4.1. ESTRUCTURA EN DISCO

Al inicio de cada archivo de índices se inserta una estructura de encabezado que contiene la siguiente información:

1. Nombre del producto.

2. Dirección de la página raíz (fisicamente en disco).

3. Longitud de la página en disco (número de caracteres).

4. Número de archivo que le corresponde dentro de ADSI.

5. Número de campo de la llave primaria que le corresponde dentro de ADSI.

6. Tipo de llave.

7. Longitud de la llave (número de caracteres).

8. Longitud de la página en memoria (número de caracteres).

9. Indicador de llave duplicada.

10. Longitud del registro de datos asociado (número de caracteres).

11. Número de serie del sistema.

12. Número de licencia corporativa (por cada sistema puede haber 1-65,000 licencias corporativas).

13. Número de año al que pertenecen los datos.

14. Número de campos que integran la llave asociada al archivo de índices.

15. Arreglo de campos que integran la llave.

    15.1. Número de campo.

    15.2. Tipo de campo.

    15.3. Posición del campo dentro del archivo de índices.

    15.4. Posición original del campo dentro del archivo de datos.

    15.5. Longitud del campo (número de caracteres).

IV.4.2. ESTRUCTURA EN MEMORIA

Las páginas en memoria tienen la siguiente estructura:

1. Posición de la página (fisicamente en disco).

2. Apuntador en memoria a la página con valores menores a la página actual (página izquierda).

3. Indicador de página modificada.

4. Número de llaves existentes en la página actual (n = 2, por lo tanto, debe haber al menos n llaves y máximo 2n = m = 4 llaves. La raíz puede tener 1 llave).

5. Un arreglo de apuntadores a valores actuales de las llaves existentes (elementos) en la página actual.

    5.1. Apuntador en memoria a la página con valores mayores al elemento de la página actual (página derecha del elemento).

    5.2. Posición física en disco del registro de datos.

    5.3. Apuntador en memoria del valor actual del elemento.

IV.4.3. ESTRUCTURA DE ÁRBOLES DENSOS

Para la optimización del manejo de páginas en memoria se implementó una estructura mediante la cual, se almacenan varias páginas dentro de un bloque de páginas. Las páginas que se almacenan representan a un subárbol con altura dos, esto es, la raíz del subárbol, los hijos de la raíz del subárbol y los nietos de la raíz del subárbol. A esta estructura se le denomina bloque de páginas o árbol denso. Esto permite que en la transferencia de un árbol denso entre disco y memoria, se aprovechen hasta treinta y un páginas con un máximo de ciento veinticuatro llaves, y en el recorrido del árbol se evite hacer a lo más dos lecturas.

Otra importante mejora para la implementación de árboles es que del archivo de configuración del sistema (*.SYS) se extrae el parámetro que señala el número de bloques de memoria por cada archivo de índices en ADMARC, el cual indica el número de árboles densos que se pueden soportar en memoria antes de mandar a escribir al disco en forma permanente los bloques modificados. En un proceso de alta masiva de registros (inserciones), por ejemplo, el nivel de respuesta es muy atractivo. Cuando se consultan todos los registros de un archivo gigante por medio de enunciados de consulta, la respuesta se obtiene casi sin esfuerzo.

La estructura en memoria usada para controlar los bloques de páginas se describe a continuación. Se reserva el espacio de memoria total para el número de bloques de páginas por cada archivo de índices.

1. Número de bloque dentro de la memoria total de bloques de páginas.

2. Longitud de la estructura del archivo de índices (número de caracteres).

3. Longitud de cada bloque de páginas incluido dentro del total de bloques (número de caracteres).

4. Apuntador a la memoria que contiene el total del bloque de páginas.

5. Arreglo de información de cada bloque dentro del total de bloques.

    5.1. Desplazamiento en caracteres del bloque de páginas actual.

    5.2. Posición física en disco del bloque de páginas actual.

    5.3. Indicador de bloque modificado (si se modifica, se escribe o reescribe).

5.4. Posición física en disco de la página raíz del bloque de páginas (raíz del subárbol contenido).

5.5. Arreglo de información interna de cada página dentro del bloque de páginas actual (esta información se usa para recorrer páginas desocupadas, controlar las páginas subocupadas, aprovechar al máximo el espacio disponible)

5.5.1. Número interno de cada llave dentro de cada página.

5.5.2. Número de llaves ocupadas por página.

5.5.3. Arreglo de números internos de llaves dentro de páginas.

IV.4.4. ALMACENAMIENTO

Los archivos de índices se abren en modo lectura/escritura. Si no existen, se crean. Si ya existen, se abren para actualización. Después de la estructura de encabezado que se inserta al inicio de cada archivo de índices, se agregan los bloques de páginas en modo secuencial conforme van llegando. Como se explicó en el punto IV.4.3, cada bloque de páginas trae consigo la información necesaria para que los ADMARC desempeñen su función adecuadamente. Debido a que los registros de datos se agregan conforme van llegando, las estructuras internas de control de los ADMARC se modifican, manteniendo los valores de las llaves en orden ascendente, tal como se requiere en un árbol ordenado. Sin embargo, como es de esperarse, al agregar o eliminar registros, los archivos de índices se van ajustando y los bloques de páginas se van recorriendo o desplazando para asegurar que se agrupen bloques de índices, dando a las hojas la prioridad de agrupación más que a la raíz.

Si se desea que los archivos de datos se mantengan ordenados de acuerdo al valor de las llaves primarias, puede recurrirse a la optimización de archivos (CONTROLA), pero debe considerar lo siguiente:

1. Los registros de datos se graban secuencialmente conforme se van insertando.

2. El valor de la llave primaria de cada registro, proporciona un orden dentro del archivo de índices correspondiente.

3. En caso de haber llaves secundarias, cada archivo de índices tendrá un orden particular de los registros en función del valor de las llaves.

ADSI ejecuta las siguientes acciones cuando se da de alta un registro:

1. Investiga que no exista el valor de la llave primaria.

2. Agrega el registro en el archivo de datos y recupera la posición física del mismo.

3. Agrega los valores de las llaves en cada uno de los archivos de índices involucrados.

4. Convierte internamente las llaves duplicadas en llaves únicas, agregando al valor de la llave la posición del registro de datos.

5. El campo de control CAMPOCON se inicializa (ACTIVO y LONGREG).

Cuando se da de baja un registro, ADSI hace lo siguiente:

1. Investiga que exista el valor de la llave primaria.

2. Lee el registro de datos y recupera la posición física del mismo.

3. Elimina los valores de las llaves declaradas en cada uno de los archivos de índices involucrados.

4. Rellena a ceros la posición física del registro de datos (incluyendo el bit ACTIVO).

La modificación a un registro se lleva a cabo considerando lo siguiente:

1. Investiga que exista el valor de la llave primaria.

2. Lee el registro de datos y recupera la posición física del mismo.

3. Reemplaza en memoria los campos que se señalan a modificarse (la modificación se hace por campo y nunca permite modificar campos llaves).

4. Reescribe todo el registro de datos en la posición física correspondiente.

V. MANIPULACIÓN DE DATOS

ADSI tiene dispuestas funciones que manipulan los datos e información almacenada dentro de los archivos del ambiente. Primeramente se listan estas funciones a las cuales, por lo general, se les denomina operaciones sobre archivos, y luego se comenta brevemente la forma en que los explotadores y el usuario en general pueden accederlas. Los nombres de las operaciones se presentan en el idioma inglés, debido a que en computación, estos términos son universalmente conocidos en la literatura y aparecen como instrucciones en la mayoría de los lenguajes de programación.

V.1. OPERACIONES SOBRE ARCHIVOS

Las operaciones que ADSI reconoce para manipular datos aparecen explicadas en el anexo E y aquí se mencionan brevemente. La solicitud de operaciones para manipular datos viene dentro del contexto del paso de mensajes, recurso que se emplea para la comunicación entre procesos. El proceso ADMCEN recibe las operaciones, las decodifica, las interpreta y las envía al proceso ADMARC que corresponda a la solicitud.

Las operaciones sobre los archivos solicitan las acciones siguientes:

a) MSG_ADMARC_DELETE. Eliminar un registro previamente leído.

b) MSG_ADMARC_FIND. Localizar un registro dado el valor de una llave.

c) MSG_ADMARC_ISFILEEMPTY. Preguntar al ADMARC si un determinado archivo está vacío.

d) MSG_ADMARC_READDK. Leer un registro dado el valor de la llave primaria.

e) MSG_ADMARC_READMK. Leer un registro dado el valor de alguna llave por la parte mayor, esto es, los primeros caracteres de una llave alfanumérica o los primeros campos de una llave compuesta.

f) MSG_ADMARC_READVK. Leer un registro dado el valor de algunos campos señalados de una llave compuesta.

g) MSG_ADMARC_READNEXT. Leer un bloque de registros dado el valor de una llave. Puede leer desde el principio del archivo, desde algún valor particular de la llave o leer el siguiente bloque.

h) MSG_ADMARC_READONE. Leer un registro dado el valor de una llave. Puede leer desde el principio del archivo o desde algún valor particular de la llave.

i) MSG_ADMARC_REWRITE. Modificar un registro previamente leído.

j) MSG_ADMARC_WRITE. Agregar un registro.

V.2. EXPLOTADOR DE PANTALLAS

Como se explicó en el capítulo II.4.2 y II.6.1, el proceso EXPPAN interpreta la información contenida en cada pantalla, y envía al ADMDBD esta información para que le devuelva la trayectoria conteniendo las acciones a tomar. Dentro de estas acciones se encuentran las operaciones para manipular datos. La invocación de estas operaciones están inmersas dentro del proceso EXPPAN y aparecen en el anexo E.

V.3. EXPLOTADOR DE REPORTES

En el capítulo II.6.2, se explicó la manera en que EXPREP interpreta los diseños de reportes. La invocación a las operaciones de manipulación de datos está programada dentro de EXPREP y se refieren a lectura.

V.4. EXPLOTADOR DE ENUNCIADOS DEL LENGUAJE DE CONSULTA

Anteriormente, en el capítulo II.5.4 se introdujo la idea de cómo se elaboran enunciados del lenguaje de consulta LCA. En el capítulo II.6.3 se expresa la forma cómo se explotan los enunciados construidos. Sin embargo, ADSI no dedica toda la atención al uso del lenguaje LCA como el lenguaje fundamental de manipulación de datos LMD, ya que el proceso más poderoso que utiliza es EXPPAN. Sin embargo, se reconoce que por medio de EXPELC se invocan operaciones de manipulación de datos, y en algunas macros de LCA, esto se ejecuta de modo complicado pero eficiente.

V.5. PROGRAMAS DE APLICACIÓN DEL AMBIENTE

En el capítulo II.4.3 se describe la función de los administradores ADMPAA, mediante los cuales se pueden invocar operaciones de manipulación de datos y hacer uso de los recursos del ambiente. Ver capítulo II.7.3 sobre rutinas comunes.

V.6. PROGRAMAS DE APLICACIÓN DE USUARIO

En el capítulo II.6.4 se describe la función de los programas que desarrolla el usuario EXPPAU, mediante los cuales se pueden invocar operaciones de manipulación de datos y el uso de cualquier recurso del ambiente. Ver capítulo II.7.3 sobre rutinas comunes.

VI. MANEJO DE TRANSACCIONES

En este capítulo se describen e involucran conceptos que pasan desapercibidos por el usuario final, y que muchas veces, quizá son ignorados por los desarrolladores.

Pero, ¿qué es y en qué consiste una transacción? Para responder a este cuestionamiento, es necesario reconocer que un programa de computadora de ADSI es un conjunto de instrucciones ordenadas con criterios lógicos, controladas por una secuencia de acciones semánticas que conducen a alcanzar un objetivo determinado dentro del sistema de información. En un sistema administrador de bases de datos, una *transacción* es el conjunto de instrucciones necesarias para acceder y manipular datos de los archivos del ambiente con un fin específico.

VI.1. ATOMICIDAD, DURABILIDAD, SERIALIZABILIDAD Y AISLAMIENTO

Una transacción consiste en operaciones que acceden registros de los archivos de ADSI. Debido a que más de una transacción opera simultáneamente, el ambiente debe garantizar que sus operaciones se efectúen como si no existiera ninguna otra transacción.

Una transacción debe ser *atómica*, es decir, todas o ninguna de sus operaciones son ejecutadas. La *atomicidad* requiere que si una transacción se interrumpe por alguna falla, sus resultados parciales no deben afectar la información.

Hay dos razones por las que una transacción no se completa: la transacción aborta o el sistema falla. La actividad para asegurar la atomicidad cuando aborta una transacción se llama *recuperación de la transacción*, y la actividad para asegurar la atomicidad cuando falla el sistema se llama *recuperación del sistema*.

La conclusión de una transacción se llama *aceptación* (commitment). Cada transacción comienza con una operación primitiva de inicio y concluye con una operación primitiva de terminación. En el caso de ADSI, existen dos maneras de iniciar y terminar transacciones, ver anexo I:

a) MSG_ADMTRA_OPENT. Inicia transacción solicitando la liberación de registros bloqueados, asociados al proceso llamador.

b) MSG_ADMTRA_BEGINT. Inicia transacción solicitando que no se liberen los registros bloqueados, asociados al proceso llamador.

c) MSG_ADMTRA_CLOSET. Termina transacción (aceptación) solicitando la liberación de registros bloqueados, asociados al proceso llamador.

d) MSG_ADMTRA_ENDT. Termina transacción (aceptación) solicitando que no se liberen los registros bloqueados, asociados al proceso llamador.

Una vez que la transacción se haya aceptado, el sistema garantiza que el resultado de su operación nunca se perderá, esto se llama *durabilidad de la transacción*, independientemente de las fallas subsecuentes. Los resultados de una transacción son resguardados por el sistema en sus archivos y la actividad de proporcionar la durabilidad de una transacción se llama *recuperación de archivos*.

Si varias transacciones se ejecutan concurrentemente, el resultado debe ser igual al que ellas obtienen, si se ejecutasen serialmente en algún orden. La actividad de garantizar la *serializabilidad* de las transacciones se llama *control de concurrencia*. Debido a que ADSI proporciona este control, los usuarios pueden escribir programas que emplean primitivas de transacciones como si se ejecutasen solas en el ambiente.

Debido a que una transacción incompleta no puede revelar sus resultados a otras transacciones antes de ser aceptada, debe mantenerse *aislada*. Esta propiedad se emplea para evitar el problema de *efecto dominó* (aborto por cascada).

VI.2. CONTROL DE CONCURRENCIA

El control de concurrencia está implementado en ADSI por medio de candados. Hay básicamente dos maneras de controlar la concurrencia: *estampas de tiempo y candados*. Las estampas de tiempo requieren menos recursos y funcionan óptimamente cuando abundan las solicitudes de lectura, se muestran deficientes cuando abundan las solicitudes de escritura. Por otro lado, los candados requieren más recursos y esfuerzos de programación pero son confiables al atender solicitudes de lectura y escritura y no afectan grandemente la carga de trabajo.

VI.2.1. GRANULARIDAD Y CANDADOS

Para manejar la concurrencia, los archivos se particionan en fragmentos que se reconocen como unidades de datos sobre las cuales se controla el acceso. Los fragmentos que están definidos en ADSI son: archivos completos o registros independientes. Al tamaño de estos fragmentos se le llama *granularidad*.

Un *candado* es un privilegio de acceso a un solo fragmento, que otorga o niega a las transacciones, el manejador de candados. Cuando se solicita un candado para un fragmento se dice que este queda bloqueado. Si existe un bloqueo para un archivo y se solicita bloquear un registro de ese archivo, se dice que el registro está bloqueado. Un candado por lo general, puede solicitarse para bloquear un archivo, para leer un registro o para escribir un registro.

La información que se proporciona al manejador de candados para solicitar un candado, usa la estructura siguiente:

<transacción> <tipo de candado> <fragmento>

donde la transacción es el identificador del proceso; el tipo de candado es: lectura, escritura; el fragmento es la dirección del registro o el nombre del archivo. Ver figura 6.1.

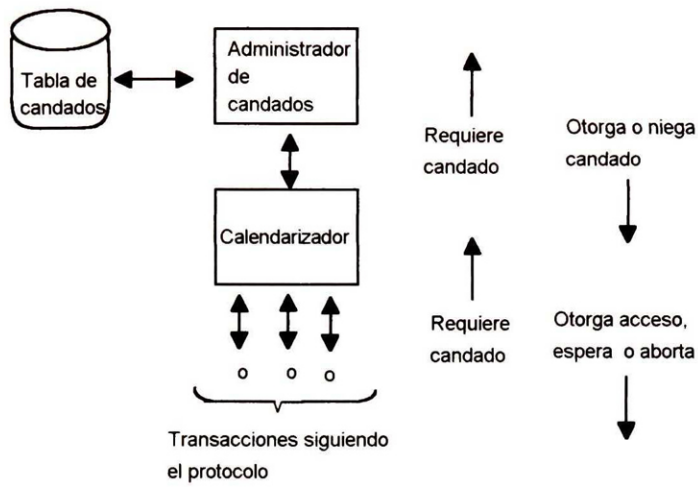


Fig. 6.1 Protocolo, calendarizador y administrador de candados

VI.2.2. CALENDARIZADOR

Debido a que las transacciones ocurren de manera aleatoria, puede suceder que al ejecutarse concurrentemente, den origen a problemas tales como candados-de-por-vida, candados-de-muerte y conductas no-serializables. Para eliminar estos problemas se tienen dos herramientas: el calendarizador y los protocolos. El *calendarizador* es una porción del sistema que funciona como un árbitro entre requerimientos conflictivos. Un calendarizador por lo general, emplea la técnica primera-operación-que-llega primera-operación-que-atiende para eliminar el candado-de-por-vida. Para resolver los problemas de candados-de-muerte puede tomar una de las dos decisiones siguientes: a) obliga a una transacción a esperar a que el candado que desea, esté liberado y b) le indica a la transacción que aborte y reinicie.

Sin embargo, la herramienta de calendarizador no se usa en ADSI ya que para evitar problemas de candados-de-por-vida se implementó un temporizador para cada solicitud de candado y se permite que cada proceso tome una acción de acuerdo al mensaje indicado por el manejador de transacciones. Para evitar conductas no-serializables y candados-de-muerte, el protocolo empleado garantiza la solución a ambas situaciones.

VI.2.3. PROTOCOLO DE BLOQUEO EN DOS FASES

La herramienta que emplea ADSI para manejar candados-de-muerte y serializabilidad es el protocolo. Un *protocolo* en su sentido más general, es una restricción sobre la secuencia de pasos atómicos (operaciones sobre archivos) que una transacción puede ejecutar. El protocolo usado por ADSI, denominado*protocolo en dos fases* requiere que todos los bloqueos precedan a todos los desbloqueos. Las transacciones obedecen a este protocolo y consiste en:

a) La primera fase se denomina *fase de bloqueo.*

b) La segunda fase se denomina *fase de desbloqueo.*

El protocolo en dos fases garantiza la serializabilidad ya que sencillamente otorga el candado al requerimiento de candado si es que este está disponible y si no está disponible hace esperar a la transacción o aborta. De acuerdo a la bibliografía usada, este protocolo es el mejor.

Existen dos clases de candados:

1. Candado de lectura (candado compartido). Una transacción T que desea únicamente leer un fragmento A, ejecuta la orden CANDADO-LECTURA A, la cual impide que cualquier otra transacción escriba un nuevo valor A mientras que T mantenga el candado asociado. Sin embargo, cualquier número de transacciones pueden asociar un candado de lectura sobre A simultáneamente.

2. Candado de escritura (candado exclusivo). Una transacción desea cambiar el valor de un fragmento A, y primero obtiene un candado de escritura sobre A, al ejecutar la orden CANDADO-ESCRITURA A. Cuando alguna transacción mantiene un candado de escritura sobre algún fragmento, ninguna otra transacción puede obtener ninguna clase de candado, esto es, ni candado de lectura ni candado de escritura.

VI.3. IMPLANTACIÓN

Lo referente al manejo de transacciones se implantó como un módulo del administrador central ADMCEN. Esto implica que todos los procesos que requieren un dato o un acceso sobre registros, se comunican con ADMCEN y este los dirige hacia el administrador de archivos correspondiente ADMARC. Sin embargo, con la implementación del manejo de transacciones, ADMCEN adquiere otras funciones fundamentales, entre ellas:

a) Crea una tabla de transacciones.

b) Crea una tabla de bloqueos (candados a fragmentos).

c) Conserva las operaciones de cada transacción en una lista de intenciones.

d) Registra las operaciones en una bitácora.

e) Controla la reconstrucción de archivos en caso de falla del sistema.

f) Maneja la optimización de archivos para aumentar la velocidad de acceso.

VI.3.1. LISTA DE INTENCIONES

Los procesos que se ejecutan en el ambiente, pueden realizar todas las operaciones e instrucciones que permite el lenguaje C, pero, cuando los procesos requieren información de los archivos de ADSI, envían mensajes que contienen operaciones de control sobre el protocolo de transacciones. Algunas operaciones de control sobre transacciones son: apertura, inicio, fin y terminación de transacciones. (Las operaciones se explican en el anexo I).

Cuando una transacción solicita su apertura o inicio, ADMCEN registra dentro de una tabla en memoria la información referente a las operaciones de alta, baja y modificación de registros para esa transacción. El registro de operaciones en la tabla se lleva a cabo en el orden como van sucediendo. A esta tabla se le denomina *lista de intenciones* y no se registran dentro de ella las operaciones de lectura. Mientras las operaciones están en la lista de intenciones, se garantiza el aislamiento.

En el archivo de configuración *.SYS existe el parámetro #NUM_OPER_BLOQ_TRANSACCION, el cual permite que el usuario anote el número máximo de operaciones que se deben guardar por cada transacción. Ver anexo L.

VI.3.2. ADMCEN

Como ya se señaló anteriormente, ADMCEN tiene la responsabilidad del módulo del manejo de transacciones y adquiere también funciones adicionales. Existen dos tablas fundamentales en memoria.

La tabla de transacciones contiene la siguiente información por cada transacción:

1. Identificador de proceso.
2. Contador de registros bloqueados actualmente.
3. Indicador de transacción iniciada.
4. Indicador de transacción candidata a no concluir correctamente (por repercusión de otra transacción terminada y aceptada).
5. Contador de operaciones almacenadas.
6. Arreglo de apuntadores a operaciones almacenadas (lista de intenciones que contiene altas, bajas y cambios).

La tabla de bloqueo de registros contiene la siguiente información por cada archivo:

1. Número interno asociado a cada archivo.
2. Identificador del proceso que mantiene bloqueado al archivo totalmente.

3. Contador de registros bloqueados dentro del archivo.

4. Estructura de control interna para almacenar en disco los registros bloqueados cuando no caben en memoria (se emplean árboles densos).

Es importante señalar que ADMCEN no es autosuficiente para llevar a cabo el manejo de transacciones por sí solo, para ello requiere del contacto estrecho con los ADMARC. Por ejemplo, internamente se asocian los candados a registros (bloqueos) utilizando la concatenación del identificador del proceso (pid) y la dirección física del registro, pero la dirección física del registro únicamente la conoce el archivo de índices (árbol denso asociado a la llave señalada).

Entonces, cuando una transacción termina, esto es, se compromete a que se lleven a cabo sus operaciones guardadas en la lista de intenciones, ADMCEN se dispone a aceptarla, y para esto, hace una verificación sobre las operaciones de dicha transacción. La verificación consiste en lo siguiente:

a) Que no exista la baja o cambio de un registro, el cual se pretende dar de baja previamente en la lista (transacción no aceptada).

b) Que no exista el alta de un registro, el cual se pretende dar de alta previamente en la lista (transacción no aceptada).

c) En caso de que el alta a un registro también aparezca en la lista de intenciones de otra transacción no aceptada aún, entonces, esta última transacción debe marcarse como candidata a no ser aceptada.

Estas verificaciones no aparecen en la literatura pero son indispensables.

Una vez realizada la verificación anterior y si no existe ningún inconveniente sobre la transacción en esta etapa de conclusión, ADMCEN escribe en una bitácora en disco duro y en forma permanente, las operaciones de la lista de intenciones y luego las manda ejecutar con los ADMARC correspondientes.

Para efectuar estos pasos, se emplea un esquema conocido en la literatura como *protocolo de bloqueo estricto en dos fases*, el cual señala:

1. Una transacción no puede escribir en los archivos del ambiente, hasta que dicha transacción haya alcanzado su punto de aceptación.

2. Una transacción no puede liberar ningún candado, hasta que dicha transacción haya terminado de escribir en los archivos del ambiente; por lo tanto, los candados no son liberados hasta después del punto de aceptación.

Por lo anterior, ADMCEN libera los bloqueos después de haber concluido la escritura en los archivos, aunque en el caso de ADSI, depende también del tipo de control de transacción: fin o terminación. Ver anexo L. ADSI libera la lista de intenciones de la transacción.

VI.3.3. ADMARC

Los administradores de archivos ADMARC no cumplen únicamente con las operaciones básicas de mantener los datos en almacenamiento secundario permanente, sino que además, cooperan con ADMCEN para satisfacer las necesidades referentes al manejo de transacciones.

Cuando una transacción desea leer un registro envía tres elementos:

a) El valor de los campos que forman la llave.

b) El número interno de la llave con la cual desea leer.

c) La lista de campos que desea obtener.

Cuando una transacción desea agregar un registro, debe enviar los tres elementos anteriores, con información completa para todos los campos del registro. Cuando una transacción desea borrar un registro, envía información similar a las altas.

Sin embargo, para bloquear un registro se requiere de la dirección física que corresponda al registro de datos. Esta dirección únicamente la conoce el archivo de índices que controla la llave del archivo involucrado. Por lo tanto, el ADMARC correspondiente debe hacer este trabajo. Cuando el ADMARC devuelve la dirección física de la operación solicitada a ADMCEN, también le devuelve la información de esta operación de una manera completa, ordenada y verificada de los componentes que ADMARC requiere para efectuar la operación sin temor a fracasar. Esto se refiere a las altas y bajas principalmente. Se anexan los mensajes de error que ADMARC y ADMCEN envían a los procesos que solicitan transacciones. Ver anexo N.

VI.4. PROTECCIÓN DE DATOS

Una de las capacidades que posee ADSI es la protección de los datos en caso de falla del sistema y esto sucede inesperadamente. Los algoritmos programados en ADSI para escribir en disco duro, sobre todo en los ADMARC, están orientados a almacenar en memoria la mayor cantidad de apuntadores a registros para proporcionar un mejor rendimiento en el tiempo de respuesta. En el archivo de configuración *.SYS existe el parámetro #NUM_BLOQUES_MEM_ARCHIVO, el cual señala el número de bloques de memoria por cada archivo de índices en ADMARC. Este mismo parámetro lo emplea ADMCEN para controlar el bloqueo de registros.

ADSI también escribe en el archivo bitácora, todas las operaciones de las transacciones, pero en este caso, la escritura se lleva a cabo permanentemente al momento de escribir.

VI.4.1. BITÁCORA

La bitácora es un archivo que contiene las operaciones de las transacciones que concluyen, esto es, cuando una transacción se compromete a llevar a cabo las operaciones sobre archivos incluidas dentro de las operaciones de control: inicio-final de transacción. Una vez que la transacción es aceptada, ADMCEN verifica las operaciones, las anota en la bitácora e inmediatamente las ejecuta por medio de los ADMARC. La escritura en la bitácora se hace permanente en forma inmediata con el fin de que si el sistema falla, por medio de la recuperación, se puedan actualizar los datos de las operaciones.

VI.4.2. RECUPERACIÓN CON PROTOCOLO-REHACER

Existe en las funciones de ADMCEN un procedimiento que permite rehacer las operaciones que están anotadas en la bitácora. Este es un procedimiento sencillo y lo único que se restringe es que no existan transacciones operando en el ambiente. El algoritmo para enfrentar fallas del sistema se conoce como algoritmo *rehacer* (redo) y se describe con los siguientes pasos:

1. Para cada registro A para el cual un valor nuevo, v, es escrito por la transacción T, agregar al archivo bitácora (T,A,v).

2. Agregar el registro al archivo bitácora (T, aceptación).

3. Escribir al almacenamiento estable el bloque o bloques del archivo bitácora que aún no se hayan escrito ahí. En este punto, T se considera aceptada y registrada.

4. Para cada registro A, escribir su nuevo valor v en el lugar donde A pertenece en el archivo correspondiente. Esta escritura puede ser concluida al traer el bloque A a la memoria principal y hacer la actualización ahí. Es opcional escribir el bloque A en el almacenamiento estable inmediatamente.

VI.4.3. PUNTO DE VERIFICACIÓN E IDEMPOTENCIA

Existe aún un detalle con respecto a la escritura permanente. Al enviar la instrucción de escritura al disco, no se garantiza que se escriba realmente en ese momento, esto se debe a que actualmente muchas controladoras de discos duros, son inteligentes, y hacen el trabajo físico de escribir, solamente cuando lo consideran muy necesario o cuando reciben una instrucción especial y directa para vaciar los "buffers" que tienen almacenados en memoria. En el lenguaje C, esta instrucción es "fflush".

Además de la estrategia que emplean los discos duros para escribir físicamente la información lo menos posible, existe la implantación de los algoritmos de ADSI que tratan de contener en memoria la mayor cantidad de información para ofrecer mejores tiempos de respuesta.

Debido a que es necesario garantizar la permanencia de la información en almacenamiento secundario estable, se ha implementado un punto de verificación en el cual se obliga a los ADMARC a enviar periódicamente instrucciones "fflush" a los discos duros y así, disminuir la cantidad de información que el ambiente debe reconstruir en caso de falla del sistema. Existe en el archivo de configuración un parámetro #NUM_TRANS_ESC_PERM, por medio del cual, el usuario sugiere el número de transacciones a acumular para escribir permanentemente en disco. Interiormente, el manejador de transacciones lleva un contador por cada transacción completa y al llegar al número proporcionado por el usuario, envía la orden de escribir la información que está en memoria al disco duro en forma permanente (solamente la información que haya sufrido cambio).

Con respecto a la idempotencia, el algoritmo que se emplea garantiza que cuando se recupera la información de la bitácora y se actualizan los archivos del ambiente, se puede repetir este proceso una y muchas veces y el resultado va a ser el mismo. Ver [8] y [9].

VI.4.4. TERMINA PROCESO NORMAL/ABORTA

Cuando un proceso termina de manera normal, envía un mensaje de terminación a ADSI o emite como última instrucción en lenguaje C "exit(0);", el manejador de transacciones libera todos los recursos que deja bloqueados y con ello, los hace disponibles para otras transacciones. Sin embargo, cuando un proceso termina anormalmente (aborta), el sistema operativo envía un mensaje a ADMCEN indicándole esta anomalía y le transfiere el identificador de proceso en cuestión para que ADMCEN lo busque en su tabla de procesos y tabla de transacciones y pueda con ello liberar los recursos bloqueados.

VII. DESARROLLO DE UNA APLICACIÓN

La aplicación que se desarrolla fue tomada de [4]. En este documento se agrega la conversión equivalente de las proposiciones escritas en el lenguaje estándar SQL a las macros interpretadas por LCA. En el documento original se anticipa que los ejercicios que se presentan en el tema son avanzados en el planteamiento de consultas a una base de datos relacional. También se menciona que el objetivo fundamental es presentar el formalismo del álgebra relacional y el cálculo de tuplas para plantear consultas. Se agregan además las expresiones en lenguaje SQL.

Los ejemplos se basan sobre una base de datos suficientemente simple para entender con facilidad el objetivo de las consultas; y lo suficientemente compleja para permitir consultas significativas. Se trata de una base de datos con información de cierto club gastronómico. Intervienen tres entidades cuyos esquemas de relación son los siguientes:

COMENSAL(nombre_com, dir_com, tel_com, peso_com). Almacena información sobre los socios del club y se registra para cada miembro su nombre, dirección, teléfono y peso.

PLATILLO(nombre_pla, hora_pla). Indica la lista de platillos que el club gastronómico pone a disposición de sus socios. En cada platillo se indica el nombre del platillo y la hora en que se recomienda servir (puede ser 'desayuno', 'comida' o 'cena').

INGREDIENTE(nombre_ing, temp_ing). Es una relación utilizada por los cocineros para registrar las listas de ingredientes que utilizan para preparar los platillos. Esta relación almacena el nombre de cada ingrediente, así como la temporada del año en que se puede conseguir ('primavera', 'verano', 'otoño' e 'invierno').

Los parentescos que se presentan entre estas entidades se almacenan en los esquemas de relación siguientes:

CONTIENE(nombre_pla, nombre_ing). Indica qué ingredientes lleva cada platillo. Es un parentesco muchos-muchos entre PLATILLO e INGREDIENTE.

GUSTA(nombre_com, nombre_pla). Indica qué platillos le gustan a cada miembro. Es un parentesco muchos-muchos entre COMENSAL y PLATILLO.

Una vez señalada la descripción de las entidades, se procede a elaborar el sistema de información que contiene procesos, diccionario de datos, pantallas, informes, etc. A continuación se listan los pasos que se siguen:

1. Encender la computadora y entrar al SO con cuenta de superusuario.

2. Ir a directorio raiz: "cd //1/"

3. Crear el directorio: "mkdir //1/ZZ131_012010000001/"

4. Ir al directorio: "cd //1/ZZ131_012010000001/"

5. Crear los directorios de trabajo: "mkdir dir_dat", "mkdir dir_exe", "mkdir dir_frm" y "mkdir dir_wor"

6. Ir al directorio de ejecutables: "cd dir_exe"

7. Cargar los programas ejecutables de ADSI al directorio "dir_exe"

8. Elaborar el diccionario de datos con el editor de textos: "vi //1/ZZ131_012010000001/dir_dat/ZZ131.DD" Ver N.

9. Compilar el diccionario de datos: "CONDIC". Se construye //1/ZZ131_012010000001/dir_dat/ZZ131.DIC".

10. Elaborar el archivo de configuración del ambiente: "vi //1/ZZ131_012010000001/dir_dat/ZZ131.SYS" Ver L.

11. Elaborar el archivo de configuración de impresoras: "vi //1/ZZ131_012010000001/dir_dat/LPT.INFO". Ver M.

12. Copiar el archivo de mensajes de error de ADSI al directorio "//1/ZZ131_012010000001/dir_dat/MSG_ERR.ERR"

13. Arrancar el ambiente ADSI: "ARRANCA"

14. Utilizar el constructor de formas para diseñar las pantallas: "CONPAN" Ver figura 7.1.

15. Utilizar el constructor de informes para diseñar los informes: "CONREP" Ver figura 7.2.

16. Elaborar los macros del lenguaje de consulta por medio del editor de textos: "vi //1/ZZ131_012010000001/dir_frm/*.MQL". Elegir los nombres deseados y posteriormente ligarlos en las pantallas.

17. Para probar y adecuar el desarrollo del sistema, se puede entrar en otra consola con alguna cuenta (no necesariamente de superusuario), e invocar el proceso "//1/ZZ131_012010000001/dir_exe/EXPPAN usuario MENU0"

    17.1. Las figuras 7.3, 7.4, 7.5, 7.6 y 7.7 muestran pantallas de captura de datos.

    17.2. La figura 7.8 muestra un informe visto en pantalla.

    17.3. La figura 7.9 muestra la macro del ELC que produjo una salida vista en pantalla.

18. Para salirse de probar y adecuar el desarrollo del sistema es necesario colocarse en la pantalla inicial y oprimir la tecla de escape.

19. Para terminar la operación del sistema se invoca el siguiente proceso desde la cuenta de superusuario: "//1/ZZ131_012010000001/dir_exe/CONTROLA"

20. Apagar la computadora.

A continuación se muestran algunas figuras que contienen pantallas del sistema de información referentes a la aplicación y posteriormente se presenta la lista de consultas que enfatizan el uso del lenguaje de consulta de ADSI. Por razones tipográficas, se utiliza el símbolo ⊗ para denotar las operaciones de junta (natural y selectiva).

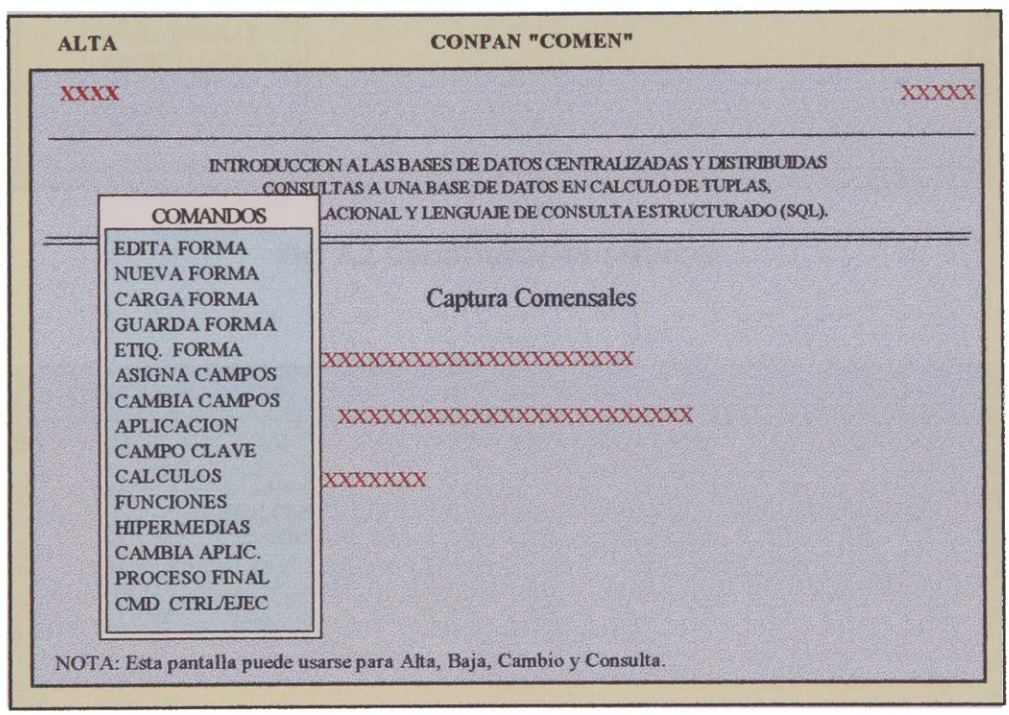


Fig. 7.1 Constructor de pantallas.

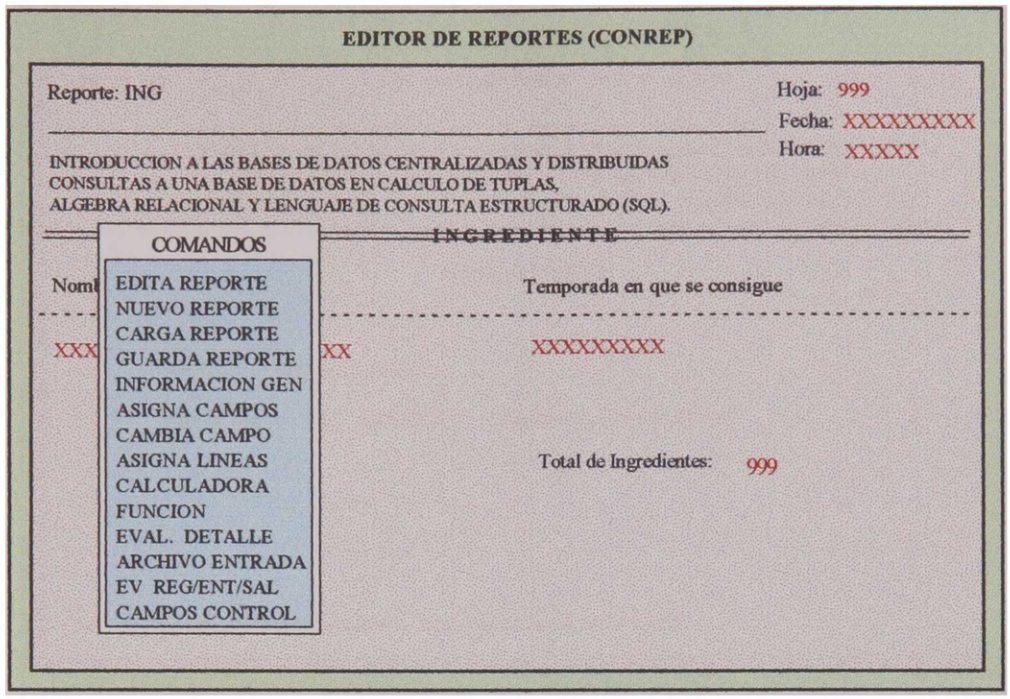


Fig. 7.2 Constructor de reportes.

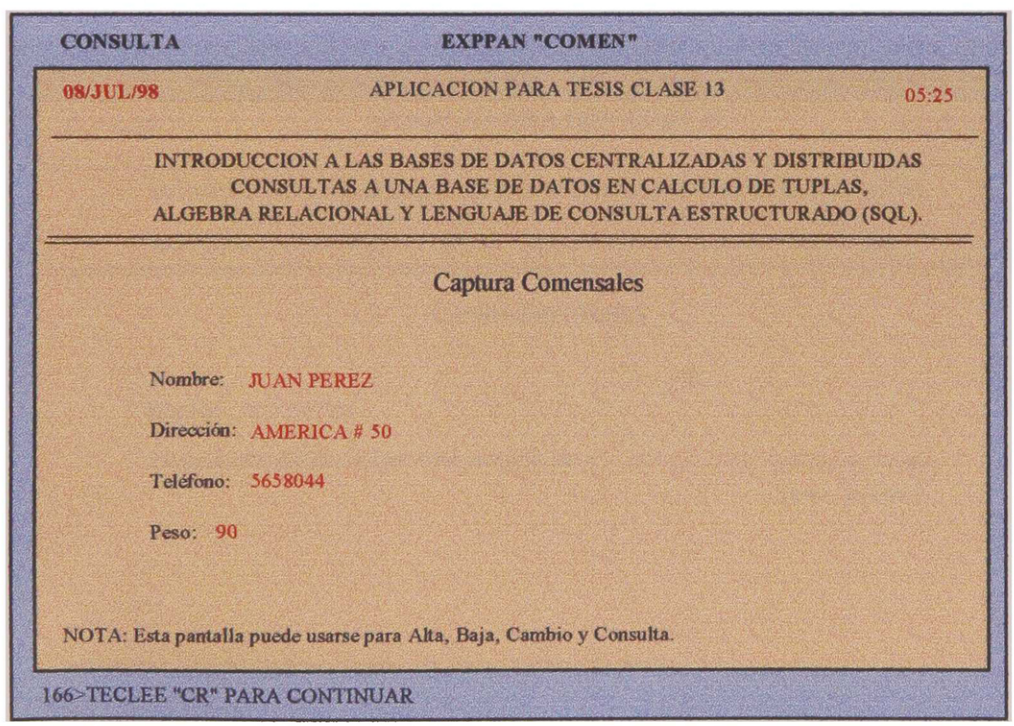


Fig. 7.3 Captura comensales.

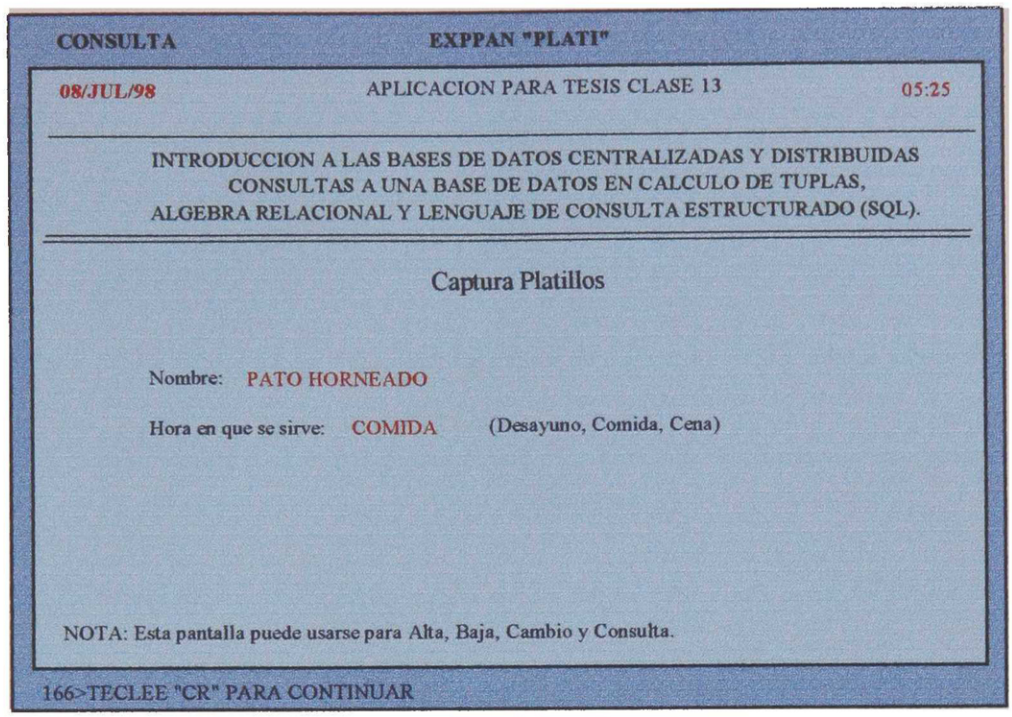


Fig. 7.4 Captura platillos.

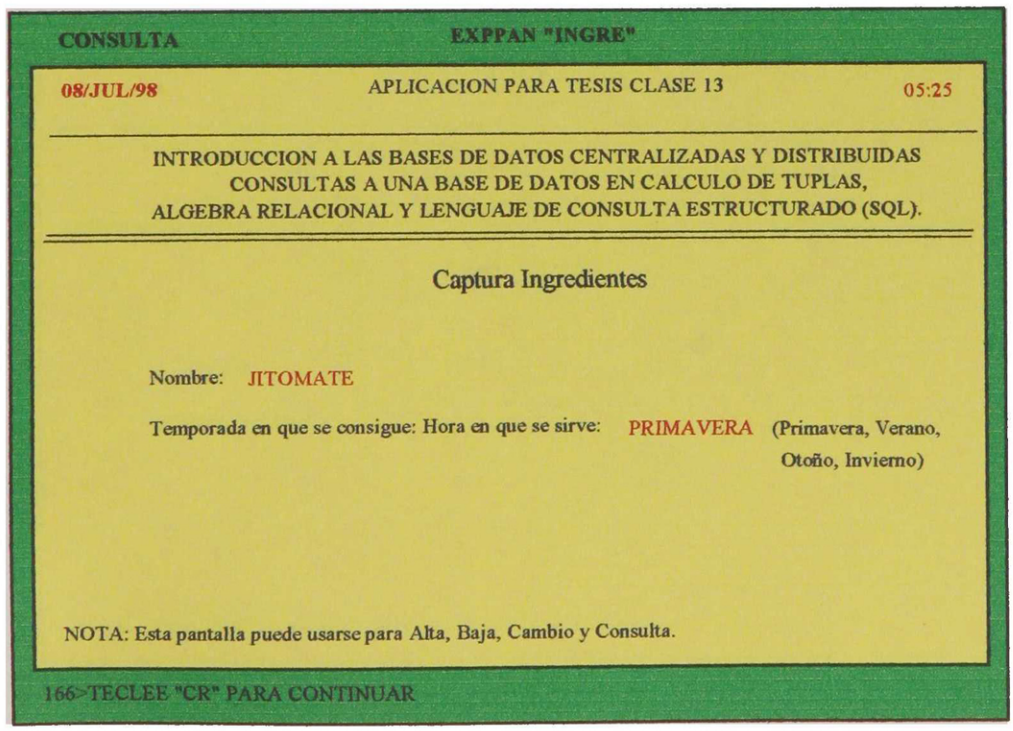


Fig. 7.5 Captura ingredientes.

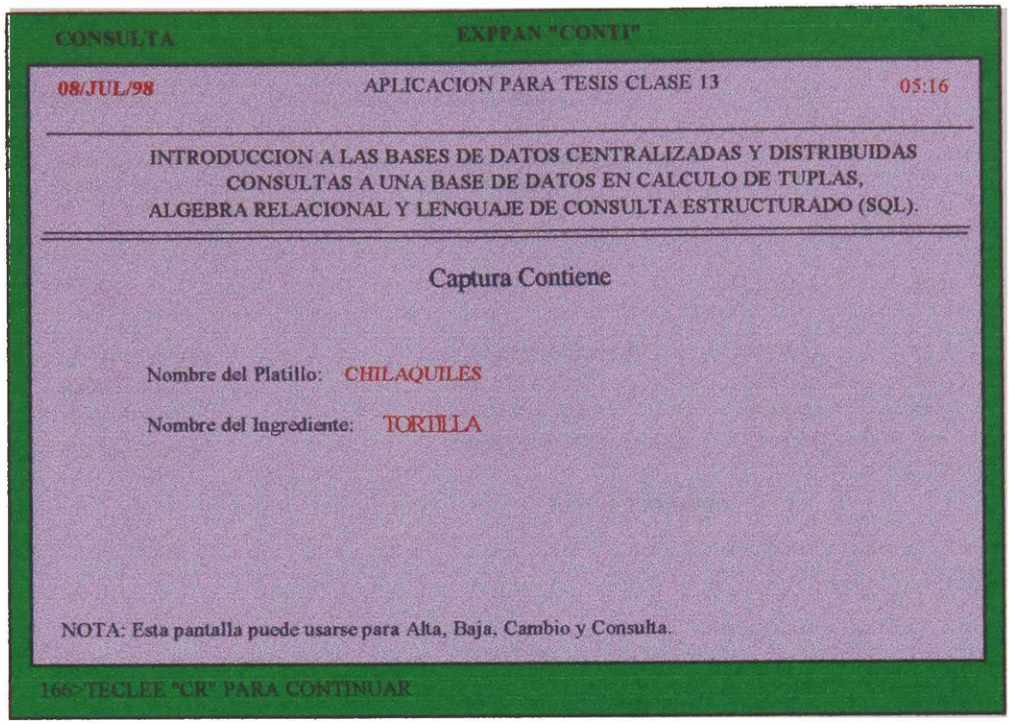


Fig. 7.6 Captura contiene.

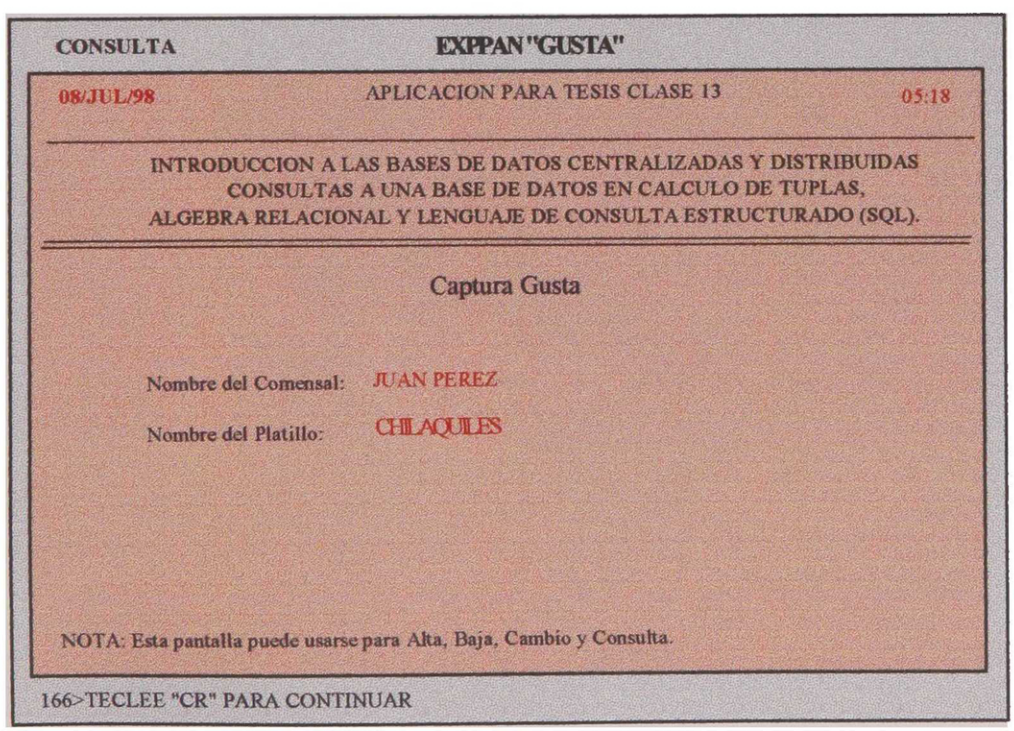


Fig. 7.7 Captura gusta.

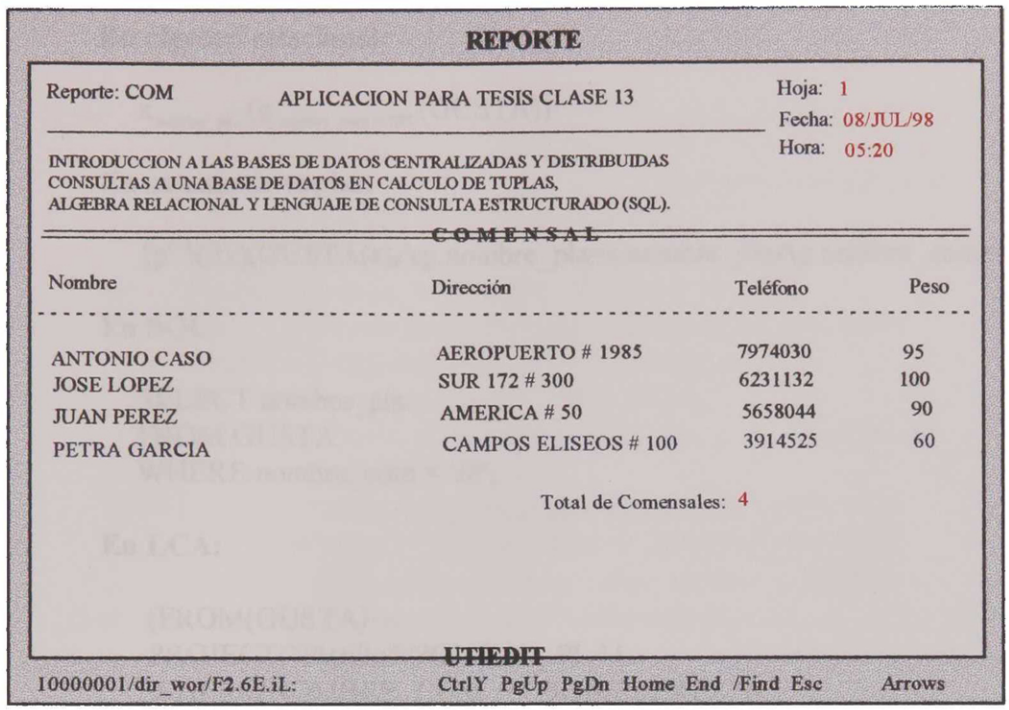


Fig. 7.8 Reporte visto en pantalla.

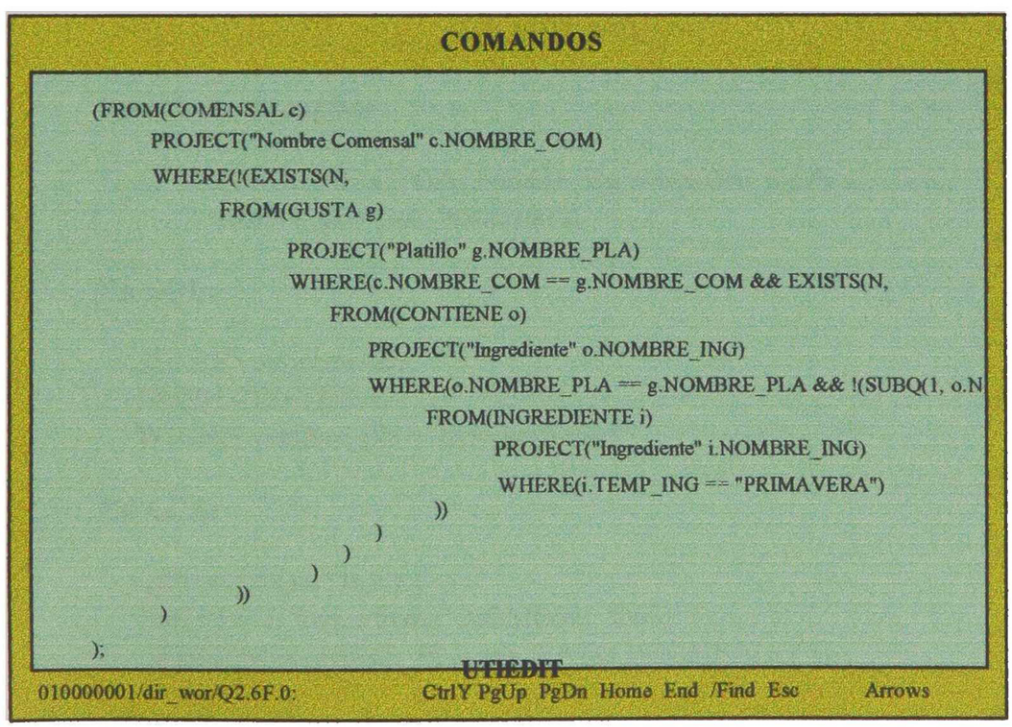


Fig. 7.9 Macros de ELC vistos en pantalla.

1. ¿Qué platillos le gustan a Juan Pérez?

En álgebra relacional:

$$\pi_{nombre\_pla}\,(\sigma_{nombre\_com\,=\,'JP'}\,(GUSTA))$$

En cálculo de tuplas:

$$\{p^{(1)} \mid (\exists s)(GUSTA(s) \wedge p.nombre\_pla = s.nombre\_pla \wedge s.nombre\_com = 'JP')\}$$

En SQL:

```
SELECT nombre_pla
FROM GUSTA
WHERE nombre_com = 'JP';
```

En LCA:

```
(FROM(GUSTA)
PROJECT("Platillo" NOMBRE_PLA)
WHERE(NOMBRE_COM == "JUAN PEREZ"));
```

2. ¿Cuáles son los ingredientes del pato horneado?

En álgebra relacional:

$$\pi_{nombre\_ing}\,(\sigma_{nombre\_pla\,=\,'pato\ horneado'}(CONTIENE))$$

En cálculo de tuplas:

$$\{i^{(1)} \mid (\exists t)(CONTIENE(t) \wedge i.nombre\_ing = t.nombre\_ing \wedge t.nombre\_pla = 'pato\ horneado')\}$$

En SQL:

```
SELECT nombre_ing
FROM CONTIENE
WHERE nombre_pla = 'pato horneado';
```

En LCA:

```
(FROM(CONTIENE)
PROJECT("Ingrediente" NOMBRE_ING)
WHERE(NOMBRE_PLA == "PATO HORNEADO"));
```

3. ¿A quiénes les gustan los chiles rellenos?

En álgebra relacional:

$$\pi_{\text{nombre\_com, dir\_com, tel\_com}} (\text{COMENSAL} \otimes \sigma_{\text{nombre\_pla = 'chiles rellenos'}}(\text{GUSTA}))$$

En cálculo de tuplas:

$\{b^{(3)} | (\exists t)(\text{COMENSAL}(t) \wedge b.\text{nombre\_com}=t.\text{nombre\_com} \wedge$
$b.\text{dir\_com}=t.\text{dir\_com} \wedge b.\text{tel\_com}=t.\text{tel\_com} \wedge$
$(\exists s)(\text{GUSTA}(s) s.\text{nombre\_pla}=\text{'chiles rellenos'} \wedge$
$s.\text{nombre\_com}=t.\text{nombre\_com}))\}$

En SQL:

```
SELECT p.nombre_pla, p.dir_com, p.tel_com
FROM COMENSAL p, GUSTA s
WHERE p.nombre_com = s.nombre_com AND
        s.nombre_pla = 'chiles rellenos'
```

En LCA:

```
(FROM(COMENSAL p, GUSTA s)
PROJECT("Nombre del Comensal" p.NOMBRE_COM,
        "Dir. Comensal" p.DIR_COM, "Tel. Comensal" p.TEL_COM)
WHERE(p.NOMBRE_COM == s.NOMBRE_COM &&
        s.NOMBRE_PLA == "CHILES RELLENOS"));
```

4. ¿Qué ingredientes no conseguimos en la temporada de primavera?

En álgebra relacional:

$$\pi_{\text{nombre\_ing}}(\text{INGREDIENTE}) - \pi_{\text{nombre\_ing}}(\sigma_{\text{temp\_ing = 'primavera'}}(\text{INGREDIENTE}))$$

En cálculo de tuplas:

$\{i^{(1)} | (\exists t)(\text{INGREDIENTE}(t) \wedge i.\text{nombre\_ing}=t.\text{nombre\_ing} \wedge$
$(\forall s)(\text{INGREDIENTE}(s) \wedge s.\text{nombre\_ing}=t.\text{nombre\_ing} \Rightarrow$
$s.\text{temp\_ing} \neq \text{'primavera'}))\}$

En SQL:

```
SELECT p.nombre_ing
FROM INGREDIENTE p
WHERE p.nombre_ing NOT IN
    (SELECT s.nombre_ing
    FROM INGREDIENTE s
    WHERE s.temp_ing = 'primavera');
```

Esta consulta se puede expresar de este modo

```
SELECT p.nombre_ing
FROM INGREDIENTE p
WHERE 'primavera' NOT IN
    (SELECT temp_ing
    FROM INGREDIENTE
    WHERE nombre_ing = p.nombre_ing);
```

En LCA:

```
(FROM(INGREDIENTE p)
PROJECT("Ingrediente" p.NOMBRE_ING)
WHERE(!(SUBQ(1, p.NOMBRE_ING, IN,
    FROM(INGREDIENTE s)
        PROJECT("Ingrediente" s.NOMBRE_ING)
        WHERE(s.TEMP_ING == "PRIMAVERA")))));
```

5. **¿Qué ingredientes no se consiguen en la temporada de primavera para preparar el pato horneado?**

En álgebra relacional:

$$[\pi_{nombre\_ing}(INGREDIENTE) - \pi_{nombre\_ing}(\sigma_{temp\_ing = 'primavera'}(INGREDIENTE))] \cap \pi_{nombre\_ing}(\sigma_{nombre\_pla = 'pato\ horneado'}(CONTIENE))$$

En cálculo de tuplas:

$$\{i^{(1)} | (\exists l)(CONTIENE(l) \wedge l.nombre\_pla='pato\ horneado' \wedge i.nombre\_ing=l.nombre\_ing \wedge (\forall s)(INGREDIENTE(s) \wedge s.nombre\_ing=l.nombre\_ing \Rightarrow s.temp\_ing \neq 'primavera'))\}$$

En SQL:

```
SELECT p.nombre_ing
FROM CONTIENE p
WHERE p.nombre_pla = 'pato horneado' AND p.nombre_ing NOT IN
    (SELECT s.nombre_ing
    FROM INGREDIENTE s
    WHERE s.temp_ing = 'primavera');
```

En LCA:

```
(FROM(CONTIENE p)
PROJECT("Ingrediente" p.NOMBRE_ING)
WHERE(p.NOMBRE_PLA == "PATO HORNEADO" &&
    !(SUBQ(1, p.NOMBRE_ING, IN,
            FROM(INGREDIENTE s)
            PROJECT("Ingrediente" s.NOMBRE_ING)
            WHERE(s.TEMP_ING == "PRIMAVERA")))));
```

6. ¿Quiénes son los comenzales que pesan más de 90 Kg. y les gustan los platillos que contienen tortilla?

En álgebra relacional:

$$B= \pi_{nombre\_com}[\pi_{nombre\_pla}(\sigma_{nombre\_ing = 'tortilla'}(CONTIENE))\otimes GUSTA]$$

$$\pi_{nombre\_com}(\sigma_{peso\_com > 90}(COMENSAL))\cap B$$

En cálculo de tuplas:

$$\{c^{(1)} | (\exists t)(COMENSAL(t) \wedge c.nombre\_com=t.nombre\_com \wedge t.peso\_com>90Kg) \wedge (\exists u)(GUSTA(u) \wedge u.nombre\_com=c.nombre\_com \wedge (\exists l)(CONTIENE(l) \wedge l.nombre\_pla=u.nombre\_pla \wedge l.nombre\_ing='tortillas'))\}$$

En SQL:

```
SELECT p.nombre_com
FROM COMENSAL p, GUSTA g
WHERE p.peso_com > 90 AND
            p.nombre_com = g.nombre_com AND g.nombre_pla IN
    (SELECT nombre_pla
    FROM CONTIENE
    WHERE nombre_ing = 'tortilla');
```

En LCA:

```
(FROM(COMENSAL p, GUSTA g)
PROJECT("Nombre del Comensal" p.NOMBRE_COM)
WHERE(p.PESO_COM > 90 &&
    p.NOMBRE_COM == g.NOMBRE_COM &&
    SUBQ(1, g.NOMBRE_PLA, IN,
            FROM(CONTIENE c)
            PROJECT("Platillo" c.NOMBRE_PLA)
            WHERE(c.NOMBRE_ING == "TORTILLA"))));
```

7. ¿Qué platillos de los que le gustan a Juan Pérez se pueden preparar para servir en el desayuno?

En álgebra relacional:

$A=\pi_{nombre\_pla}(\sigma_{nombre\_com = 'JP'}(GUSTA))\otimes CONTIENE$

A representa los platillos que le gustan a Juan Pérez junto con sus ingredientes.

$C=\pi_{nombre\_ing}(INGREDIENTE) - \pi_{nombre\_ing}(\sigma_{temp\_ing = 'primavera'}(INGREDIENTE))$

C representa los ingredientes que no se consiguen en primavera.

Ahora $B=\pi_{nombre\_pla}(A\otimes C)$ representa los platillos que le gustan a Juan Pérez y que no se pueden preparar.

Finalmente la respuesta a la consulta original es:

$(\pi_{nombre\_pla}(\sigma_{nombre\_com = 'JP'}(GUSTA)) - B)\cap$
$(\pi_{nombre\_pla}(\sigma_{hora\_pla = 'desayuno'}(PLATILLO)))$

En cálculo de tuplas:

$\{p^{(1)} | (\exists s)(GUSTA(s) \wedge s.nombre\_com='JP' \wedge (\exists a)(PLATILLO(a) \wedge$
$a.hora\_pla='desayuno' \wedge (a.nombre\_pla=s.nombre\_pla \wedge$
$p.nombre\_pla=s.nombre\_pla) \wedge (\forall t)(CONTIENE(t) \wedge$
$t.nombre\_pla=s.nombre\_pla \Rightarrow (\exists u)(INGREDIENTE(u) \wedge$
$u.nombre\_ing=t.nombre\_ing \wedge u.temp\_ing='primavera'))))\}$

En SQL:

Para responder a esta consulta se procede por partes en subconsultas.
¿Qué platillos no se pueden preparar?

```
SELECT nombre_pla
FROM CONTIENE
WHERE nombre_ing IN
    (SELECT nombre_ing
    FROM INGREDIENTE
    WHERE nombre_ing NOT IN
            (SELECT nombre_ing
            FROM INGREDIENTE
            WHERE temp_ing = 'primavera'))
INTO TEMP platillos_no_preparados;
```

¿Qué platillos se pueden preparar para el desayuno y le gustan a 'JP'?

```
SELECT u.nombre_pla
FROM GUSTA u, PLATILLO p
WHERE p.hora_pla = 'desayuno' AND u.nombre_pla = p.nombre_pla AND
        u.nombre_com = 'JP' AND
        p.nombre_pla NOT IN platillos_no_preparados;
DROP platillos_no_preparados;
```

En LCA:

Para responder a esta consulta se procede por partes en subconsultas.
¿Qué platillos no se pueden preparar?

```
CALL_MQL(CON07_2);
(FROM(CONTIENE c1)
PROJECT("Platillo" c1.NOMBRE_PLA)
WHERE(SUBQ(1, c1.NOMBRE_ING, IN,
    FROM(INGREDIENTE i1)
    PROJECT("Ingrediente" i1.NOMBRE_ING)
    WHERE(!(SUBQ(1, i1.NOMBRE_ING, IN,
        FROM(INGREDIENTE i2)
        PROJECT("Ingrediente" i2.NOMBRE_ING)
        WHERE(i2.TEMP_ING == "PRIMAVERA")))))));
```

¿Qué platillos se pueden preparar para el desayuno y le gustan a 'JP'?

```
(FROM(GUSTA u, PLATILLO p)
PROJECT("Platillo" u.NOMBRE_PLA)
WHERE(p.HORA_PLA == "DESAYUNO" &&
    u.NOMBRE_PLA == p.NOMBRE_PLA &&
    u.NOMBRE_COM == "JUAN PEREZ" &&
    !(SUBQ(1, p.NOMBRE_PLA, IN, REFERENCE(0, REFCERO)))));
```

8. ¿En la temporada de primavera se puede preparar pato horneado?

En álgebra relacional:

$$(\sigma_{nombre\_pla = 'pato\ horneado'}(CONTIENE) \otimes \pi_{nombre\_ing}(\sigma_{temp\_ing = 'primavera'}(INGREDIENTE))) / \pi_{nombre\_ing}(\sigma_{nombre\_pla = 'pato\ horneado'}(CONTIENE))$$

Esta expresión del álgebra relacional no responde a la consulta de manera afirmativa ó negativa, simplemente opera sobre relaciones para producir nuevas relaciones, y en este caso particular obtenemos la relación formada por la tupla 'pato horneado' si es que este platillo se puede preparar en primavera, de no ser así, la relación final es vacía.

En cálculo de tuplas:

$\{i^{(1)} | (\exists l)(PLATILLO(l) \wedge l.nombre\_pla = 'pato\ horneado' \wedge$
$i.nombre\_pla = l.nombre\_pla \wedge (\forall t)(CONTIENE(t) \wedge$
$t.nombre\_pla = 'pato\ horneado' \Rightarrow$
$(\exists s)(INGREDIENTE(s) \wedge s.nombre\_ing = t.nombre\_ing \wedge$
$s.temp\_ing = 'primavera'))\}$

En SQL:

```
SELECT c.nombre_pla
FROM CONTIENE c
WHERE c.nombre_pla  = 'pato horneado' AND NOT EXISTS
    (SELECT nombre_ing
    FROM CONTIENE
    WHERE nombre_plat = 'pato horneado' AND nombre_ing NOT IN
            (SELECT nombre_ing
            FROM INGREDIENTE
            WHERE temp_ing = 'primavera'));
```

En LCA:

```
(FROM(CONTIENE c1)
PROJECT("Platillo" c1.NOMBRE_PLA)
WHERE(c1.NOMBRE_PLA == "PATO HORNEADO" && !(EXISTS(1,
    FROM(CONTIENE c2)
    PROJECT("Ingrediente" c2.NOMBRE_ING)
    WHERE(c2.NOMBRE_PLA == "PATO HORNEADO" &&
            !(SUBQ(1, c2.NOMBRE_ING, IN,
                    FROM(INGREDIENTE i)
                    PROJECT("Ingrediente" i.NOMBRE_ING)
                    WHERE(i.TEMP_ING == "PRIMAVERA"))))))));
```

9. ¿En primavera, quiénes son los comenzales para los que se puede preparar alguno de los platillos que les gustan?

En álgebra relacional:

Para responder a esta consulta se procede por partes con subconsultas.

¿Qué platillos se pueden preparar en primavera?

$B = \pi_{nombre\_ing}(INGREDIENTE) - \pi_{nombre\_ing}(\sigma_{temp\_ing = 'primavera'}(INGREDIENTE))$

B contiene los ingredientes que se producen en primavera.

$A = \pi_{nombre\_pla}(PLATILLO) - \pi_{nombre\_pla}(CONTIENE \otimes B)$

A tiene los platillos que se pueden preparar.

Ahora, la respuesta a la consulta original es:

$\pi_{nombre\_com}(GUSTA \otimes A)$

En cálculo de tuplas:

$\{c^{(1)} | (\exists s)(GUSTA(s) \wedge c.nombre\_com=s.nombre\_com \wedge$
$(\forall t)(CONTIENE(t) \wedge t.nombre\_pla=s.nombre\_pla \Rightarrow$
$(\exists l)(INGREDIENTE(l) \wedge l.nombre\_ing=t.nombre\_ing \wedge$
$l.temp\_ing='primavera'))))\}$

En SQL:

```
SELECT nombre_com
FROM GUSTA g
WHERE NOT EXISTS
    (SELECT nombre_ing
    FROM CONTIENE c
    WHERE c.nombre_pla = g.nombre_pla AND nombre_ing NOT IN
            (SELECT nombre_ing
            FROM INGREDIENTE
            WHERE temp_ing = 'primavera'));
```

En LCA:

```
(FROM(GUSTA g)
PROJECT("Nombre Comensal" DISTINCT g.NOMBRE_COM)
WHERE(!(EXISTS(1,
    FROM(CONTIENE c)
    PROJECT("Ingrediente" c.NOMBRE_ING)
    WHERE(c.NOMBRE_PLA == g.NOMBRE_PLA &&
                !(SUBQ(1, c.NOMBRE_ING, IN,
            FROM(INGREDIENTE i)
            PROJECT("Ingrediente" i.NOMBRE_ING)
            WHERE(i.TEMP_ING == "PRIMAVERA")))))));
```

10. ¿En primavera, quiénes son los comenzales para los que se pueden preparar todos los platillos que les gustan?

En álgebra relacional:

$A=\pi_{nombre\_ing}(INGREDIENTE) - \pi_{nombre\_ing}(\sigma_{temp\_ing = 'primavera'}(INGREDIENTE))$

A contiene los ingredientes que no se producen en primavera.

$T=\pi_{nombre\_pla}(CONTIENE \otimes A)$

T representa los platillos que contienen algún ingrediente que no se consigue en primavera. La respuesta es:

$\pi_{nombre\_com}(COMENSAL) - \pi_{nombre\_com}(T \otimes GUSTA)$

En cálculo de tuplas:

```
{c(1)|(∃p)(COMENSAL(p)∧c.nombre_com=p.nombre_com∧
    (∀t)(GUSTA(t)∧t.nombre_com=p.nombre_com⇒
    (∀u)(CONTIENE(u)∧u.nombre_pla=t.nombre_pla⇒
    (∃s)(INGREDIENTE(s)∧s.nombre_ing=u.nombre_ing∧
        s.temp_ing='primavera')))))}
```

En SQL:

```
SELECT nombre_com
FROM COMENSAL c
WHERE NOT EXISTS
    (SELECT nombre_pla
    FROM GUSTA g
    WHERE c.nombre_com = g.nombre_com AND EXISTS
        (SELECT nombre_ing
        FROM CONTIENE o
        WHERE o.nombre_pla = g.nombre_pla AND
                o.nombre_ing NOT IN
            (SELECT nombre_ing
            FROM INGREDIENTE
            WHERE temp_ing = 'primavera')));
```

En LCA:

```
(FROM(COMENSAL c)
PROJECT("Nombre Comensal" c.NOMBRE_COM)
WHERE(!(EXISTS(N,
    FROM(GUSTA g)
    PROJECT("Platillo" g.NOMBRE_PLA)
    WHERE(c.NOMBRE_COM == g.NOMBRE_COM && EXISTS(N,
        FROM(CONTIENE o)
        PROJECT("Ingrediente" o.NOMBRE_ING)
        WHERE(o.NOMBRE_PLA == g.NOMBRE_PLA &&
                !(SUBQ(1, o.NOMBRE_ING, IN,
            FROM(INGREDIENTE i)
            PROJECT("Ingrediente" i.NOMBRE_ING)
            WHERE(i.TEMP_ING == "PRIMAVERA")))))))));
```

VIII. COMPARACIÓN CON OTRAS HERRAMIENTAS

Se muestran dos productos comerciales que tienen algunos años de experiencia en el mercado: Paradox 7 y Oracle 7. Ambos productos han evolucionado a partir de las exigencias del mercado y a prueba y error. Han adaptado las nuevas tecnologías que van surgiendo a estos productos conforme les permite el tiempo y la velocidad de actualización con que se desplaza la Informática en el mundo.

Consultando a desarrolladores y usuarios de estos productos, se nota que existe una satisfacción parcial, debido a que los tiempos de respuesta tienden a ser mayores conforme la carga de información aumenta.

En el caso de Paradox 7, se comporta muy rápido con pequeñas cantidades de datos y preferentemente con el menor número de tablas posible. Mientras más poderosa sea la computadora donde se instale, es mejor.

En el caso de Oracle 7 funciona muy rápido con pequeñas cantidades de datos, pero es lento conforme aumenta la carga de información y el número de usuarios. El consumo de recursos es muy grande y requiere computadoras muy bien equipadas.

En el caso de ADSI, la primera versión fue instalada en computadoras AT 286 y está elaborada para sistemas operativos multiusuario y de preferencia en verdaderos sistemas de tiempo real aunque esto último no es indispensable. ADSI tiene un rendimiento atractivo ya que consume pocos recursos y el tiempo de respuesta está influenciado por parámetros proporcionados por el usuario. El producto ADSI ocupa un espacio de 1.8MB con todos sus componentes, programas ejecutables y programas compilados para construir procesos de usuario.

VIII.1. PARADOX 7

Paradox 7 es considerado uno de los productos más avanzados y recientes de su empresa desarrolladora utilizando técnicas de SABD. Los nuevos desarrollos se harán utilizando nuevas herramientas de programación orientadas a objetos. La comparación que se hace, es en relación a las capacidades de PARADOX 7 y ADSI.

1. Tablas

1.1. Construcción de tablas.

- Ambos productos definen campos, tipos y tamaños.

- Ambos productos validan rangos y valores (indicador campo requerido, mínimo, máximo, valores de omisión, máscaras).

- Paradox7 construye tablas específicas de búsqueda para validación y ADSI valida contra archivos existentes.

  Paradox7 declara la integridad referencial explícitamente, ADSI la deduce.

- Paradox7 crea claves de acceso por campo y tabla, ADSI crea una estructura de árbol para encadenar pantallas permitidas a claves de acceso.

- Paradox7 reconfigura tablas temporalmente, ADSI no.

- Paradox7 permite datos en mayúsculas y minúsculas, ADSI únicamente en mayúsculas.

- Paradox7 permite que los componentes de una llave compuesta estén separados en un registro, ADSI también.

- Paradox7 permite tablas sin llaves, ADSI no.

1.2. Vistas de tablas. Las capacidades de Paradox7 sobre este aspecto son típicas de un producto que está orientado a un ambiente gráfico. ADSI no posee estas capacidades.

-Paradox7 modifica formatos de tablas, elimina campos, modifica formatos de salida, tamaños y colores de campos.

-Paradox7 muestra tablas en pantalla y puede posicionarse en un registro anterior, posterior, registro inicial, registro final.

- Paradox7 ordena tablas por algún campo pero no puede mezclar el ordenamiento de algunos campos ascendentes y otros descendentes. ADSI puede hacer esta mezcla.

- Paradox7 usa filtros para campos y tablas.

1.3. Actualización de datos. Paradox7 aprovecha la condición de estar orientado a un ambiente gráfico.

- Paradox7 puede ver y modificar datos sobre la lista de registros en pantalla (indicando al producto cada vez que se modifica algún campo), ADSI tiene pantallas definidas para cada movimiento y no muestra listas de registros para modificarse directamente de la lista.

  Paradox7 localiza registros por valor de llave o por número de registro, ADSI localiza registros por valor de llave.

- Paradox7 puede modificar un campo de todos los registros a la vez, ADSI lo hace por medio de LCA o por programa.

Paradox7 accesa gráficas y objetos OLE, ADSI no.

2. Formas. Paradox7 emplea su capacidad gráfica para diseñar tablas (una o varias a la vez). ADSI define tipos de formas de acuerdo a la aplicación o el movimiento (alta, baja, cambio, consulta) de registros.

3. Reportes. Paradox7 permite diseñar reportes complejos y mezcla datos con gráficas. ADSI soporta reportes tan complejos como sean necesarios.

4. Queries.

   - Paradox7 soporta la estructura QBE y la salida la deposita en tablas temporales (ANSWER). ADSI emite las salidas en archivo de edición después de invocar enunciados de LCA.

   - Paradox7 puede crear tablas LIVE que permiten modificar directamente las tablas originales. ADSI puede modificar directamente los archivos usando enunciados LCA.

   - Paradox7 requiere especificar explícitamente el orden de acceso de tablas para obtener consultas complicadas. ADSI puede obtener salidas de consultas tan complejas como las soporta el lenguaje estándar SQL, pero por medio del LCA.

5. Scripts. Paradox7 permite programar rutinas elaboradas con lenguaje PAL y asociarlas a campos y tablas. ADSI puede ligar cualquier tipo de hipermedia a campos, pantallas y ejecutar procesos globales de acuerdo al tipo de pantalla.

6. Bibliotecas. Paradox7 usa bibliotecas para soportar las rutinas PAL. ADSI no utiliza bibliotecas.

7. Archivos SQL. Paradox7 puede diseñar macros en el lenguaje estándar SQL. ADSI tiene su propio lenguaje equivalente.

8. Otras características.

   - Paradox7 puede bloquear/desbloquear registros y tablas manualmente definiendo tiempos de bloqueo y reintento cuando están bloqueadas. ADSI tiene un criterio de reintento constante y en las pantallas se hace automáticamente permitiendo que el usuario decida continuar intentando o abandonar el intento para el registro indicado.

   - Paradox7 puede importar/exportar datos, tablas, objetos e imágenes a otros productos, con la dificultad respectiva de acuerdo a lo que se desea hacer. ADSI puede llevar los datos a formato DOS.

   - Paradox7 requiere para formas y reportes la definición explícita de acceso a tablas. ADSI no lo requiere.

Paradox7 y ADSI permiten declarar campos de uso local en pantallas y reportes.

- Paradox7 y ADSI pueden ejecutar cálculos y lecturas a otras tablas en pantallas y reportes.

- Paradox7 puede mezclar gráficas e imágenes en pantallas y reportes. ADSI no.

- ADSI cuenta con manejo de transacciones, control de concurrencia, recuperación de datos, Paradox 7 no lo hace.

VIII.2. ORACLE 7

Oracle 7 es uno de los productos más avanzados con que cuenta su empresa desarrolladora. Actualmente también existe Oracle 8 aunque tiende a utilizar los productos complementarios que esta empresa introduce actualmente al mercado, tal como Developer/2000. Estos accesorios utilizan herramientas de programación orientadas a objetos y a un concepto que introducirán en el año 2000 denominado "computación en red". La comparación entre ORACLE 7 y ADSI se finca en las capacidades fundamentales de SABD, aunque ORACLE 7 se ocupa más en: aumentar accesorios que faciliten el desarrollo de sistemas; la integración con otros productos de mercado; tratando de superar el enorme problema en que están sumergidos los productos comerciales, en lo referente a comunicación entre procesos y entre computadoras (problema que por cierto, nunca ha existido en plataformas de tiempo real); y luchando por posicionarse en el mercado.

1. Formas.

   - Para Oracle7 son la manera principal de manipulación de datos. Inserta, actualiza, borra y ejecuta consultas SQL a las BD. ADSI también considera las formas como la manera principal de acceso a la información.

   - Oracle7 presenta datos usando texto, imágenes, gráficas y controles tipo Windows. ADSI muestra solamente texto y ASCII extendido.

   - Oracle7 presenta varias formas simultáneamente para ejecutar actualizaciones. ADSI solo permite una forma a la vez.

   - Oracle7 proporciona herramientas propias para ambiente gráfico. ADSI no.

   - Oracle7 permite 2 tipos de formas (acceso a tablas y menús). ADSI permite formas con fines específicos.

     Oracle7 requiere de información específica para accesar BD y mostrar datos en las formas (bloques). ADSI no requiere ninguna información adicional para ligar información a las formas.

- Oracle7 permite ligar pantallas maestro-detalle pero requiere de toda la información de archivos, llaves de liga y la prevención de integridad referencial. ADSI deduce y considera lo anterior de manera natural.

- Oracle7 puede integrar documentación a las formas y asociar un editor a cada campo de la forma. ADSI permite ligar a cada forma y campo, cualquier cosa deseada.

- Oracle7 puede asociar a un campo de la forma una lista de valores dada por el diseñador u obtenida de un tabla por medio de SQL. ADSI también.

2. Reportes.

   - Oracle7 facilita la creación de reportes preferentemente extrayendo datos de la BD por medio de SQL por medio del uso de modelos de datos. ADSI no requiere de modelos de datos.

   - Oracle7 permite obtener reportes sencillos. ADSI puede emitir reportes complejos.

3. Gráficas. Oracle7 puede crear gráficas de varios estilos preferentemente de datos de la BD por medio de modelos de datos. También puede producir gráficas a partir de varios archivos maestro-detalle. ADSI no obtiene gráficas.

4. Constructor de procedimientos. Propio de Oracle7 pués ADSI no posee esta característica.

   Puede generar varios procedimientos o proyectos comunes en los ambientes gráficos y ambientes Windows.

   - Existen constructores de módulos para formas, reportes y gráficas.

   - Existe un constructor de proyectos, consultas y esquemas de la BD.

   - El constructor de traducción permite que los contructores anteriores cuenten con los comandos en el idioma seleccionado.

5. Otros componentes de Oracle7.

   Puede acceder BD de otras plataformas permitidas.

   Utiliza procesos y técnicas para enlazarse con algunos productos comerciales por medio de internet.

   - Emplea el concepto ODBC y ACTIVEX.

   - Usa procesos "wizard" para apoyar el desarrollo de formas, reportes, gráficas, acceso a red, imágenes y otros recursos y productos comerciales.

IX. CONCLUSIONES

El Ambiente para el Desarrollo de Sistemas de Información está diseñado en forma modular y escalable, y se busca que los componentes que lo integran cumplan satisfactoriamente con su función, sea fácil de entender y se requiera poco tiempo para desarrollar una aplicación.

Se ha dedicado mucho esfuerzo al diseño de los componentes y a las funciones que se les atribuyeron, pero mayor esfuerzo y dedicación aún se le otorgó a su programación, cuidando el comportamiento de los elementos y su tiempo de respuesta.

Los componentes constructores producen formas, reportes, enunciados de consulta o diccionarios de datos, que son propuestos como modelos de solución a un problema planteado, donde dichos modelos se mejoran y se prueban mientras se desarrolla un sistema y, el modelo se convierte en la solución del sistema mismo.

En ADSI se contempla la integración de acciones relacionadas a la concurrencia, control de transacciones, manejo de procesos, protocolos de enlace, operaciones sobre archivos, distribución de carga de trabajo, disponibilidad de temporizadores, recuperación de información, optimización de datos y la respuesta rápida.

Además de que se busca incorporar las prescripciones indicadas en la literatura formal que describe a un sistema clásico de bases de datos y aplicar algunas adecuaciones que mejoran y facilitan el ambiente desarrollado, ADSI se presenta como una herramienta práctica y fácil de implementar.

La experiencia derivada de la evolución de la computación como rama de la Informática en los últimos años, manifiesta que por lo general, las herramientas de mercado cuidan primordialmente la penetración comercial y se presentan objetivamente como productos, exhibiendo todas sus características y bondades que las distinguen, pero ninguna de ellas ofrece explícitamente soluciones a problemas reales, ni tampoco se ocupan de minimizar recursos y garantizar resultados óptimos, pues ocurre que los usuarios que adquieren dichos productos, esperan constantemente la versión siguiente añorando superar con la nueva versión, la deficiencia que trajo consigo la versión actual.

La estructura del diseño base donde se cimentó ADSI permite crecer hacia las nuevas tecnologías tales como la orientación a objetos y los denominados sistemas de conocimiento. Esta tendencia se debe implementar sobre el contenido y significado de los datos almacenados en los archivos y no únicamente en accesorios de la herramienta. Importante es agregar características de ambiente gráfico, imágenes, voz y lenguaje natural para enriquecer la interrelación de un sistema con el medio que lo rodea.

ANEXOS

A. TÉRMINOS Y ACRÓNIMOS.

ADMARC	ADMinistrador de ARChivos.
ADMCEN	ADMinistrador CENtral.
ADMDBD	ADMinistrador de la Descripción de las Bases de Datos.
ADMPAA	ADMinistrador de Programas de Aplicación del Ambiente.
ADMPAU	ADMinistrador de Programas de Aplicación del Usuario.
ADSI	Ambiente para el Desarrollo de Sistemas de Información.
ARRANCA	ARRANCA la operación del ambiente.
BD	Bases de Datos.
CONDIC	CONstructor del DICcionario de datos.
CONELC	CONstructor de Enunciados de Lenguaje de Consulta.
CONFIG	CONFIGuración de parámetros del ambiente.
CONPAN	CONstructor de PANtallas.
CONREP	COnstructor de REPortes.
CONTROLA	CONTROLA y detiene la operación del ambiente.
DF	Dependencia Funcional.
E-V	Modelo Entidad-Vínculo.
EJEELC	EJEcutor de Enunciados de Lenguaje de Consulta.
EJEPRM	EJEcuta PaRáMetros a sustituir en los enunciados de LCA.
EXPELC	EXPlotador de Enunciados de Lenguaje de Consulta.
EXPPAN	EXPlotador de PANtallas.
EXPPAU	EXPlotador de Programas de Aplicación del Usuario.
EXPREP	EXPlotador de REPortes.
FN	Formas Normales.
INSTALA	INSTALA en Ambiente ADSI en disco duro.
LCA	Lenguaje de Consulta del Ambiente ADSI.
LDD	Lenguaje de Definición de Datos.
LMD	Lenguaje de Manipulación de Datos.
NUCLEO	Sinónimo de ADMCEN.
PC	Personal Computer (Computadora Personal).
RESPALDA	RESPALDA y recupera información en diskettes.

SABD	Sistema Administrador de Base de Datos.
SO	Sistema Operativo.
SQL	Structured Query Language (Lenguaje de Consulta Estructurado).
SR	Sistemas Relacionales.
UTIEDIC	UTIlería de EDIción de datos Comprimidos (archivos binarios).
UTIEDIT	UTIlería de EDIción de Textos (salida de reportes o consultas).
UTIGEN	UTIlería de GENeración de procesos del ambiente.
UTIMORD	UTIlería de Mezcla y ORDenamiento (sortmerge).
UTIORD	UTIlería de ORDenamiento (sort).

B. TODOS LOS TIPOS DE MENSAJES DE ENLACE DEL AMBIENTE.

MSG_ADMARC_ACCESS	El proceso solicita un acceso a algún ADMARC.
MSG_ADMARC_CHFINDREC	El proceso solicita a un ADMARC que verifique la existencia de un registro.
MSG_ADMARC_DELETE	El proceso solicita a un ADMARC eliminar un registro.
MSG_ADMARC_FIND	El proceso solicita a un ADMARC localizar un registro.
MSG_ADMARC_ISFILEEMPTY	El proceso pregunta a un ADMARC si un determinado archivo está vacio.
MSG_ADMARC_OPEN	Un ADMARC notifica al ADMCEN el inicio de sus operaciones.
MSG_ADMARC_PREVFIND	Mensaje interno del ADMDBD.
MSG_ADMARC_PREVREAD	Mensaje interno del ADMDBD.
MSG_ADMARC_READDK	El proceso solicita a algún ADMARC leer un registro por llave primaria (directa).
MSG_ADMARC_READMK	El proceso solicita a algún ADMARC leer un registro por llave mayor (los primeros caracteres de una llave alfanumérica o primeros campos de una llave compuesta).
MSG_ADMARC_READNEXT	El proceso solicita a algún ADMARC leer un bloque de registros (cero o mas). La lectura puede iniciar en el primer registro del archivo; en algún registro del archivo; continuar leyendo el siguiente bloque.
MSG_ADMARC_READONE	El proceso solicita a algún ADMARC leer un registro (cero o uno). La lectura puede iniciar en el primer registro del archivo; en algún registro del archivo.
MSG_ADMARC_READVK	El proceso solicita a algún ADMARC leer un registro cuyas llaves señaladas tengan los valores indicados.
MSG_ADMARC_REWRITE	El proceso solicita a un ADMARC modificar un registro.
MSG_ADMARC_WRITE	El proceso solicita a un ADMARC agregar un registro.
MSG_ADMCAL_ACCESS	Mensaje interno de un proceso para solicitar un cálculo o evaluación de una expresión con la "calculadora interna".
MSG_ADMCAL_CLC	Mensaje interno de un proceso para solicitar la evaluación de una expresión a la "calculadora interna".
MSG_ADMCAL_DVA	Mensaje interno de un proceso para solicitar la validación de una fecha a la "calculadora interna".
MSG_ADMCAL_FCT	Mensaje interno de un proceso para solicitar la ejecución de una función definida por la "calculadora interna".
MSG_ADMCAL_OPR	Mensaje interno de un proceso para solicitar el cálculo de una operación a la "calculadora interna".
MSG_ADMCAL_PREF	Mensaje interno de un proceso para solicitar la conversión de una expresión a forma inversa a la "calculadora interna".
MSG_ADMCEN_ADMFD	El proceso solicita a algún ADMPAA le proporcione el formato de campos definido.
MSG_ADMCEN_ALIVE	Un proceso que no ha recibido noticias del ADMCEN averigua si este último aún está en actividad.
MSG_ADMCEN_ASKMSG	El proceso solicita al ADMCEN la información que otro proceso dejó depositada.
MSG_ADMCEN_CLEARMSG	El proceso solicita al ADMCEN que destruya la información que tiene depositada para él.
MSG_ADMCEN_DEATH	El SO notifica al ADMCEN que un proceso dejó de operar.
MSG_ADMCEN_HOLDMSG	El proceso solicita al ADMCEN que conserve bajo depósito cierta información para otro proceso.
MSG_ADMCEN_LISTUSERS	El proceso solicita al ADMCEN una lista de todos los procesos en actividad dentro del ambiente.
MSG_ADMCEN_OPTIMIZE	El proceso solicita al ADMCEN la optimización de todos los archivos del ambiente.
MSG_ADMCEN_PRODINFO	El proceso solicita al ADMCEN que verifique su contexto para garantizar que pertenece al ambiente.
MSG_ADMCEN_REBUILD	El proceso solicita al ADMCEN la recuperación/reconstrucción de los archivos que no se cerraron correctamente en la sesión anterior o que están dañados.
MSG_ADMCEN_REDOEND	El proceso notifica al ADMCEN que el proceso de reconstrucción de archivos ha terminado.
MSG_ADMCEN_REDOSTART	El proceso notifica al ADMCEN que el proceso de reconstrucción de archivos debe comenzar.

MSG_ADMCEN_START	El ADMCEN le notifica a un administrador que ya puede iniciar a operar.
MSG_ADMCEN_TERM	El proceso "CONTROLA" notifica al ADMCEN que termine la operación del ambiente. El ADMCEN avisa a todos los procesos del ambiente que concluyan su operación.
MSG_ADMCEN_UNKMSG	El ADMCEN responde a un proceso que desconoce el mensaje recibido.
MSG_ADMDBD_ADDPATH	El proceso solicita al ADMDBD una trayectoria para agregar registros.
MSG_ADMDBD_ALL_VARFD	El proceso solicita al ADMDBD la descripción de todos los campos de un archivo.
MSG_ADMDBD_CREATEPATH	El proceso solicita al ADMDBD una trayectoria para crear registros.
MSG_ADMDBD_DELPATH	El proceso solicita al ADMDBD una trayectoria para eliminar registros.
MSG_ADMDBD_GETCKI	El proceso solicita al ADMDBD la descripción de todos los campos de una llave compuesta.
MSG_ADMDBD_GETFD	El proceso solicita al ADMDBD la descripción de todos los campos de un archivo.
MSG_ADMDBD_GETPATH	El proceso solicita al ADMDBD una trayectoria.
MSG_ADMDBD_GETVARINFO	El proceso solicita al ADMDBD información acerca de un campo.
MSG_ADMDBD_GETVARNAME	El proceso solicita al ADMDBD el nombre de un campo.
MSG_ADMDBD_KVINOTHERFILES	El proceso solicita al ADMDBD averigue si los campos llaves de un archivo son llave en otro archivo.
MSG_ADMDBD_LISTFILES	El proceso solicita al ADMDBD una lista de todos los archivos.
MSG_ADMDBD_LISTVARS	El proceso solicita al ADMDBD una lista de todos los campos.
MSG_ADMDBD_NOK	Mensaje interno del ADMDBD.
MSG_ADMDBD_NOPTHSZE	El ADMDBD responde al proceso que la trayectoria obtenida no cabe en el espacio del mensaje.
MSG_ADMDBD_OK	Mensaje interno del ADMDBD.
MSG_ADMDBD_OPEN	El proceso ADMDBD notifica al ADMCEN el inicio de sus operaciones.
MSG_ADMDBD_QRYPATH	El proceso solicita al ADMDBD una trayectoria para consultar registros.
MSG_ADMDBD_UPDPATH	El proceso solicita al ADMDBD una trayectoria para modificar registros.
MSG_ADMPAA_ACCESS	El proceso solicita un acceso a algún ADMPAA.
MSG_ADMPAA_ACTION	El proceso solicita a algún ADMPAA procesar la información enviada.
MSG_ADMPAA_GETPID	El proceso solicita al ADMCEN el identificador de proceso de algún ADMPAA.
MSG_ADMPAA_INCMPL	Un ADMPAA responde a un proceso que no pudo concluir correctamente la acción solicitada.
MSG_ADMPAA_NOREC	Un ADMPAA responde a un proceso que no existe el registro en el archivo señalado en la acción solicitada.
MSG_ADMPAA_OPEN	Un ADMPAA notifica al ADMCEN el inicio de sus operaciones.
MSG_ADMPAA_OUTSVC	El ADMCEN le responde a un proceso que un ADMPAA no está en operación o no reconoce la operación solicitada.
MSG_ADMTRA_BEGINT	El proceso notifica al ADMCEN (manejador de transacciones) el inicio de una transacción pidiendo que no se liberen registros asociados al proceso llamador.
MSG_ADMTRA_BLOCK1F	El proceso solicita al ADMCEN (manejador de transacciones) el bloqueo de un archivo.
MSG_ADMTRA_CHECKPOINT	El proceso solicita al ADMCEN (manejador de transacciones) que escriba en forma permanente en disco todos los cambios efectuados en los archivos de todos los ADMARC del ambiente.
MSG_ADMTRA_CLOSET	El proceso notifica al ADMCEN (manejador de transacciones) el fin de una transacción, esto es, que lleve a cabo en forma permanente las operaciones emitidas, pidiendo que se liberen registros asociados al proceso llamador excepto archivos bloqueados totalmente.
MSG_ADMTRA_ENDT	El proceso notifica al ADMCEN (manejador de transacciones) el fin de una transacción, esto es, que lleve a cabo en forma permanente las operaciones emitidas, pidiendo que no se liberen registros asociados al proceso llamador.
MSG_ADMTRA_OPENT	El proceso notifica al ADMCEN (manejador de transacciones) el inicio de una transacción pidiendo que se liberen registros asociados al proceso llamador excepto archivos bloqueados totalmente.

MSG_ADMTRA_OTWRCT	El proceso solicita al ADMCEN (manejador de transacciones) la ejecución de tres operaciones en un mismo evento: MSG_ADMTRA_OPENT, MSG_ADMARC_WRITE y MSG_ADMTRA_CLOSET.
MSG_ADMTRA_UNBLOCKLIST	El EXPREP o el EXPELC solicita al ADMCEN (manejador de transacciones) el desbloqueo de una lista de registros.
MSG_ADMTRA_UNBLOCKREC	El EXPPAN solicita al ADMCEN (manejador de transacciones) el desbloqueo de un registro.
MSG_CONPAN_OPEN	El CONPAN notifica al ADMCEN el inicio de sus operaciones.
MSG_CONREP_OPEN	El CONREP notifica al ADMCEN el inicio de sus operaciones.
MSG_EXPELC_OPEN	El EXPELC notifica al ADMCEN el inicio de sus operaciones.
MSG_EXPPAN_GETFIELD	Un mensaje interno del EXPPAN para obtener un campo desde pantalla.
MSG_EXPPAN_OPEN	El EXPPAN notifica al ADMCEN el inicio de sus operaciones.
MSG_EXPPAN_PUTFIELD	Un mensaje interno del EXPPAN para escribir un campo en la pantalla.
MSG_PROCESS_ASKHWL	El proceso "CONTROLA" solicita al ADMCEN información acerca del candado de seguridad del ambiente.
MSG_PROCESS_NOOPENED	El ADMCEN le responde a un proceso que sus operaciones no son reconocidas dentro del ambiente.
MSG_PROCESS_NOUSER	El ADMCEN le responde a un proceso que sus operaciones no son reconocidas dentro del ambiente.
MSG_PROCESS_OPEN	Un proceso de usuario notifica al ADMCEN el inicio de sus operaciones.
MSG_PROCESS_OPENED	El ADMCEN le responde a un proceso que ya puede iniciar operaciones puesto que está siendo reconocido dentro del ambiente.

C. RESPUESTAS GENERALES TRANSMITIDAS EN LOS MENSAJES DE ENLACE DEL AMBIENTE.

ANS_NOK	La respuesta de procesamiento indica que algo no se efectuó correctamente.
ANS_OK	La respuesta de procesamiento indica que todo se efectuó correctamente.
END_OF_FILE	Un ADMARC indica que detectó el fin del archivo señalado.

D. TIPOS DE MENSAJES DE ENLACE RELACIONADOS CON ADMCEN.

MSG_ADMARC_ACCESS	El proceso solicita un acceso a algún ADMARC.
MSG_ADMARC_OPEN	Un ADMARC notifica al ADMCEN el inicio de sus operaciones.
MSG_ADMCEN_ALIVE	Un proceso que no ha recibido noticias del ADMCEN averigua si este último aún está en actividad.
MSG_ADMCEN_ASKMSG	El proceso solicita al ADMCEN la información que otro proceso dejó depositada.
MSG_ADMCEN_CLEARMSG	El proceso solicita al ADMCEN que destruya la información que tiene depositada para él.
MSG_ADMCEN_DEATH	El SO notifica al ADMCEN que un proceso dejó de operar.
MSG_ADMCEN_HOLDMSG	El proceso solicita al ADMCEN que conserve bajo depósito cierta información para otro proceso.
MSG_ADMCEN_LISTUSERS	El proceso solicita al ADMCEN una lista de todos los procesos en actividad dentro del ambiente.
MSG_ADMCEN_OPTIMIZE	El proceso solicita al ADMCEN la optimización de todos los archivos del ambiente.
MSG_ADMCEN_PRODINFO	El proceso solicita al ADMCEN que verifique su contexto para garantizar que pertenece al ambiente.
MSG_ADMCEN_REBUILD	El proceso solicita al ADMCEN la recuperación/reconstrucción de los archivos que no se cerraron correctamente en la sesión anterior o que están dañados.
MSG_ADMCEN_REDOEND	El proceso notifica al ADMCEN que el proceso de reconstrucción de archivos ha terminado.
MSG_ADMCEN_REDOSTART	El proceso notifica al ADMCEN que el proceso de reconstrucción de archivos debe comenzar.
MSG_ADMCEN_START	El ADMCEN le notifica a un administrador que ya puede iniciar a operar.
MSG_ADMCEN_TERM	Eñ proceso "CONTROLA" notifica al ADMCEN que termine la operación del ambiente. El ADMCEN avisa a todos los procesos del ambiente que concluyan su operación.
MSG_ADMCEN_UNKMSG	El ADMCEN responde a un proceso que desconoce el mensaje recibido.
MSG_ADMDBD_ALL_VARFD	El proceso solicita al ADMDBD la descripción de todos los campos de un archivo.
MSG_ADMDBD_GETCKI	El proceso solicita al ADMDBD la descripción de todos los campos de una llave compuesta.
MSG_ADMDBD_GETFD	El proceso solicita al ADMDBD la descripción de todos los campos de un archivo.
MSG_ADMDBD_GETPATH	El proceso solicita al ADMDBD una trayectoria.
MSG_ADMDBD_GETVARINFO	El proceso solicita al ADMDBD información acerca de un campo.
MSG_ADMDBD_GETVARNAME	El proceso solicita al ADMDBD el nombre de un campo.
MSG_ADMDBD_KVINOTHERFILES	El proceso solicita al ADMDBD averigue si los campos llaves de un archivo es llave en otro archivo.
MSG_ADMDBD_LISTFILES	El proceso solicita al ADMDBD una lista de todos los archivos.
MSG_ADMDBD_LISTVARS	El proceso solicita al ADMDBD una lista de todos los campos.
MSG_ADMDBD_OPEN	El ADMDBD notifica al ADMCEN el inicio de sus operaciones.
MSG_ADMPAA_ACCESS	El proceso solicita un acceso a algún ADMPAA.
MSG_ADMPAA_GETPID	El proceso solicita al ADMCEN el identificador de proceso de algún ADMPAA.
MSG_ADMPAA_OPEN	Un ADMPAA notifica al ADMCEN el inicio de sus operaciones.
MSG_ADMPAA_OUTSVC	El ADMCEN le responde a un proceso que un ADMPAA no está en operación o no reconoce la operación solicitada.
MSG_ADMTRA_OTWRCT	El proceso solicita al ADMCEN (manejador de transacciones) la ejecución de tres operaciones en un mismo evento: MSG_ADMTRA_OPENT, MSG_ADMARC_WRITE y MSG_ADMTRA_CLOSET.
MSG_CONPAN_OPEN	El CONPAN notifica al ADMCEN el inicio de sus operaciones.
MSG_CONREP_OPEN	Un CONREP notifica al ADMCEN el inicio de sus operaciones.

MSG_EXPELC_OPEN	Un EXPELC notifica al ADMCEN el inicio de sus operaciones.
MSG_EXPPAN_OPEN	Un EXPPAN notifica al ADMCEN el inicio de sus operaciones.
MSG_PROCESS_ASKHWL	El proceso "CONTROLA" solicita al ADMCEN información acerca del candado de seguridad del ambiente.
MSG_PROCESS_NOOPENED	El ADMCEN le responde a un proceso que sus operaciones no son reconocidas dentro del ambiente.
MSG_PROCESS_NOUSER	El ADMCEN le responde a un proceso que sus operaciones no son reconocidas dentro del ambiente.
MSG_PROCESS_OPEN	Un proceso de usuario notifica al ADMCEN el inicio de sus operaciones.
MSG_PROCESS_OPENED	El ADMCEN le responde a un proceso que ya puede iniciar operaciones puesto que está siendo reconocido dentro del ambiente.

E. TIPOS DE MENSAJES DE ENLACE RELACIONADOS CON ADMARC.

MSG_ADMARC_ACCESS	El proceso solicita un acceso a algún ADMARC.
MSG_ADMARC_DELETE	El proceso solicita a un ADMARC eliminar un registro.
MSG_ADMARC_FIND	El proceso solicita a un ADMARC localizar un registro.
MSG_ADMARC_ISFILEEMPTY	El proceso pregunta a un ADMARC si un determinado archivo está vacio.
MSG_ADMARC_OPEN	Un ADMARC notifica al ADMCEN el inicio de sus operaciones.
MSG_ADMARC_READDK	El proceso solicita a algún ADMARC leer un registro por llave primaria (directa).
MSG_ADMARC_READMK	El proceso solicita a algún ADMARC leer un registro por llave mayor (los primeros caracteres de una llave alfanumérica o primeros campos de una llave compuesta).
MSG_ADMARC_READNEXT	El proceso solicita a algún ADMARC leer un bloque de registros (cero o mas). La lectura puede iniciar en el primer registro del archivo; en algún registro del archivo; continuar leyendo el siguiente bloque.
MSG_ADMARC_READONE	El proceso solicita a algún ADMARC leer un registro (cero o uno). La lectura puede iniciar en el primer registro del archivo; en algún registro del archivo.
MSG_ADMARC_READVK	El proceso solicita a algún ADMARC leer un registro cuyas llaves señaladas tengan los valores indicados.
MSG_ADMARC_REWRITE	El proceso solicita a un ADMARC modificar un registro.
MSG_ADMARC_WRITE	El proceso solicita a un ADMARC agregar un registro.
MSG_ADMCEN_OPTIMIZE	El proceso solicita al ADMCEN la optimización de todos los archivos del ambiente.
MSG_ADMCEN_REBUILD	El proceso solicita al ADMCEN la recuperación/reconstrucción de los archivos que no se cerraron correctamente en la sesión anterior o que están dañados.
MSG_ADMCEN_START	El ADMCEN le notifica a un ADMARC que ya puede iniciar a operar.
MSG_ADMCEN_TERM	El ADMCEN le notifica a un ADMARC que termine su operación.
MSG_ADMTRA_CHECKPOINT	El proceso solicita al ADMCEN (manejador de transacciones) que escriba en forma permanente en disco todos los cambios efectuados en los archivos de todos los ADMARC del ambiente.
MSG_PROCESS_OPENED	El ADMCEN le responde a un proceso que ya puede iniciar operaciones puesto que está siendo reconocido dentro del ambiente.

F. TIPOS DE MENSAJES DE ENLACE RELACIONADOS CON ADMDBD.

MSG_ADMARC_CHFINDREC	El proceso solicita a un ADMARC que verifique la existencia de un registro.
MSG_ADMARC_DELETE	El proceso solicita a un ADMARC eliminar un registro.
MSG_ADMARC_FIND	El proceso solicita a un ADMARC localizar un registro.
MSG_ADMARC_PREVFIND	Mensaje interno del ADMDBD.
MSG_ADMARC_PREVREAD	Mensaje interno del ADMDBD.
MSG_ADMARC_READDK	El proceso solicita a algún ADMARC leer un registro por llave primaria (directa).
MSG_ADMARC_REWRITE	El proceso solicita a un ADMARC modificar un registro.
MSG_ADMARC_WRITE	El proceso solicita a un ADMARC agregar un registro.
MSG_ADMCEN_START	El ADMCEN le notifica a un administrador que ya puede iniciar a operar.
MSG_ADMCEN_TERM	El ADMCEN le notifica a ADMDBD que termine su operación.
MSG_ADMDBD_ADDPATH	El proceso solicita al ADMDBD una trayectoria para agregar registros.
MSG_ADMDBD_ALL_VARFD	El proceso solicita al ADMDBD la descripción de todos los campos de un archivo.
MSG_ADMDBD_CREATEPATH	El proceso solicita al ADMDBD una trayectoria para crear registros.
MSG_ADMDBD_DELPATH	El proceso solicita al ADMDBD una trayectoria para eliminar registros.
MSG_ADMDBD_GETCKI	El proceso solicita al ADMDBD la descripción de todos los campos de una llave compuesta.
MSG_ADMDBD_GETFD	El proceso solicita al ADMDBD la descripción de todos los campos de un archivo.
MSG_ADMDBD_GETPATH	El proceso solicita al ADMDBD una trayectoria.
MSG_ADMDBD_GETVARINFO	El proceso solicita al ADMDBD información acerca de un campo.
MSG_ADMDBD_GETVARNAME	El proceso solicita al ADMDBD el nombre de un campo.
MSG_ADMDBD_KVINOTHERFILES	El proceso solicita al ADMDBD averigue si campos llaves de un archivo es llave en otro archivo.
MSG_ADMDBD_LISTFILES	El proceso solicita al ADMDBD una lista de todos los archivos.
MSG_ADMDBD_LISTVARS	El proceso solicita al ADMDBD una lista de todos los campos.
MSG_ADMDBD_NOK	Mensaje interno del ADMDBD.
MSG_ADMDBD_NOPTHSZE	El ADMDBD responde al proceso que la trayectoria obtenido no cabe en el espacio del mensaje.
MSG_ADMDBD_OK	Mensaje interno del ADMDBD.
MSG_ADMDBD_OPEN	El ADMDBD notifica al ADMCEN el inicio de sus operaciones.
MSG_ADMDBD_QRYPATH	El proceso solicita al ADMDBD una trayectoria para consultar registros.
MSG_ADMDBD_UPDPATH	El proceso solicita al ADMDBD una trayectoria para modificar registros.
MSG_EXPPAN_GETFIELD	Un mensaje interno del EXPPAN para obtener un campo desde pantalla.
MSG_EXPPAN_PUTFIELD	Un mensaje interno del EXPPAN para escribir un campo en la pantalla.
MSG_PROCESS_OPENED	El ADMCEN le responde a un proceso que ya puede iniciar operaciones puesto que está siendo reconocido dentro del ambiente.

G. TIPOS DE MENSAJES DE ENLACE RELACIONADOS CON ADMPAA.

MSG_ADMARC_ACCESS	El proceso solicita un acceso a algún ADMARC.
MSG_ADMARC_DELETE	El proceso solicita a un ADMARC eliminar un registro.
MSG_ADMARC_READDK	El proceso solicita a algún ADMARC leer un registro por llave primaria (directa).
MSG_ADMARC_READMK	El proceso solicita a algún ADMARC leer un registro por llave mayor (los primeros caracteres de una llave alfanumérica o primeros campos de una llave compuesta).
MSG_ADMARC_READNEXT	El proceso solicita a algún ADMARC leer un bloque de registros (cero o mas). La lectura puede iniciar en el primer registro del archivo; en algún registro del archivo; continuar leyendo el siguiente bloque.
MSG_ADMARC_READVK	El proceso solicita a algún ADMARC leer un registro cuyas llaves señaladas tengan los valores indicados.
MSG_ADMARC_REWRITE	El proceso solicita a un ADMARC modificar un registro.
MSG_ADMARC_WRITE	El proceso solicita a un ADMARC agregar un registro.
MSG_ADMCEN_ADMFD	El proceso solicita a algún ADMPAA le proporcione el formato de campos definido.
MSG_ADMCEN_START	El ADMCEN le notifica a un administrador que ya puede iniciar a operar.
MSG_ADMCEN_TERM	El ADMCEN le notifica a un ADMPAA que termine su operación.
MSG_ADMPAA_ACCESS	El proceso solicita un acceso a algún ADMPAA.
MSG_ADMPAA_ACTION	El proceso solicita a algún ADMPAA procesar la información enviada.
MSG_ADMPAA_INCMPL	Un ADMPAA responde a un proceso que no pudo concluir correctamente la acción solicitada.
MSG_ADMPAA_NOREC	Un ADMPAA responde a un proceso que no existe el registro en el archivo señalado en la acción solicitada.
MSG_ADMPAA_OPEN	Un ADMPAA notifica al ADMCEN el inicio de sus operaciones.
MSG_ADMPAA_OUTSVC	El ADMCEN le responde a un proceso que un ADMPAA no está en operación o no reconoce la operación solicitada.
MSG_PROCESS_OPENED	El ADMCEN le responde a un proceso que ya puede iniciar operaciones puesto que está siendo reconocido dentro del ambiente.

H. TIPOS DE MENSAJES DE ENLACE RELACIONADOS CON EXPPAU.

MSG_ADMARC_ACCESS	El proceso solicita un acceso a algún ADMARC.
MSG_ADMARC_DELETE	El proceso solicita a un ADMARC eliminar un registro.
MSG_ADMARC_READDK	El proceso solicita a algún ADMARC leer un registro por llave primaria (directa).
MSG_ADMARC_READMK	El proceso solicita a algún ADMARC leer un registro por llave mayor (los primeros caracteres de una llave alfanumérica o primeros campos de una llave compuesta).
MSG_ADMARC_READNEXT	El proceso solicita a algún ADMARC leer un bloque de registros (cero o mas). La lectura puede iniciar en el primer registro del archivo; en algún registro del archivo; continuar leyendo el siguiente bloque.
MSG_ADMARC_READVK	El proceso solicita a algún ADMARC leer un registro cuyas llaves señaladas tengan los valores indicados.
MSG_ADMARC_REWRITE	El proceso solicita a un ADMARC modificar un registro.
MSG_ADMARC_WRITE	El proceso solicita a un ADMARC agregar un registro.
MSG_ADMCEN_ADMFD	El proceso solicita a algún ADMPAA le proporcione el formato de campos definido.
MSG_ADMDBD_GETFD	El proceso solicita al ADMDBD la descripción de todos los campos de un archivo.
MSG_ADMDBD_GETVARNAME	El proceso solicita al ADMDBD el nombre de un campo.
MSG_PROCESS_OPEN	Un proceso de usuario notifica al ADMCEN el inicio de sus operaciones.
MSG_PROCESS_OPENED	El ADMCEN le responde a un proceso que ya puede iniciar operaciones puesto que está siendo reconocido dentro del ambiente.

I. TIPOS DE MENSAJES DE ENLACE RELACIONADOS CON EL MANEJADOR DE TRANSACCIONES (MÓDULO EN ADMCEN).

MSG_ADMARC_ACCESS	El proceso solicita un acceso a algún ADMARC.
MSG_ADMARC_DELETE	El proceso solicita a un ADMARC eliminar un registro.
MSG_ADMARC_FIND	El proceso solicita a un ADMARC localizar un registro.
MSG_ADMARC_ISFILEEMPTY	El proceso pregunta a un ADMARC si un determinado archivo está vacio.
MSG_ADMARC_READDK	El proceso solicita a algún ADMARC leer un registro por llave primaria (directa).
MSG_ADMARC_READMK	El proceso solicita a algún ADMARC leer un registro por llave mayor (los primeros caracteres de una llave alfanumérica o primeros campos de una llave compuesta).
MSG_ADMARC_READNEXT	El proceso solicita a algún ADMARC leer un bloque de registros (cero o mas). La lectura puede iniciar en el primer registro del archivo; en algún registro del archivo; continuar leyendo el siguiente bloque.
MSG_ADMARC_READONE	El proceso solicita a algún ADMARC leer un registro (cero o uno). La lectura puede iniciar en el primer registro del archivo; en algún registro del archivo.
MSG_ADMARC_READVK	El proceso solicita a algún ADMARC leer un registro cuyas llaves señaladas tengan los valores indicados.
MSG_ADMARC_REWRITE	El proceso solicita a un ADMARC modificar un registro.
MSG_ADMARC_WRITE	El proceso solicita a un ADMARC agregar un registro.
MSG_ADMCEN_DEATH	El SO notifica al ADMCEN que un proceso dejó de operar.
MSG_ADMCEN_OPTIMIZE	El proceso solicita al ADMCEN la optimización de todos los archivos del ambiente.
MSG_ADMCEN_REBUILD	El proceso solicita al ADMCEN la recuperación/reconstrucción de los archivos que no se cerraron correctamente en la sesión anterior o que están dañados.
MSG_ADMCEN_REDOEND	El proceso notifica al ADMCEN que el proceso de reconstrucción de archivos ha terminado.
MSG_ADMCEN_REDOSTART	El proceso notifica al ADMCEN que el proceso de reconstrucción de archivos debe comenzar.
MSG_ADMCEN_TERM	El proceso "CONTROLA" notifica al ADMCEN que termine la operación del ambiente. El ADMCEN avisa a todos los procesos del ambiente que concluyan su operación.
MSG_ADMTRA_BEGINT	El proceso notifica al ADMCEN (manejador de transacciones) el inicio de una transacción pidiendo que no se liberen registros asociados al proceso llamador.
MSG_ADMTRA_BLOCK1F	El proceso solicita al ADMCEN (manejador de transacciones) el bloqueo de un archivo.
MSG_ADMTRA_CHECKPOINT	El proceso solicita al ADMCEN (manejador de transacciones) que escriba en forma permanente en disco todos los cambios efectuados en los archivos de todos los administradores de archivos del ambiente.
MSG_ADMTRA_CLOSET	El proceso notifica al ADMCEN (manejador de transacciones) el fin de una transacción, esto es, que lleve a cabo en forma permanente las operaciones emitidas, pidiendo que se liberen registros asociados al proceso llamador excepto archivos bloqueados totalmente.
MSG_ADMTRA_ENDT	El proceso notifica al ADMCEN (manejador de transacciones) el fin de una transacción, esto es, que lleve a cabo en forma permanente las operaciones emitidas, pidiendo que no se liberen registros asociados al proceso llamador.
MSG_ADMTRA_OPENT	El proceso notifica al ADMCEN (manejador de transacciones) el inicio de una transacción pidiendo que se liberen registros asociados al proceso llamador excepto archivos bloqueados totalmente.
MSG_ADMTRA_OTWRCT	El proceso solicita al ADMCEN (manejador de transacciones) la ejecución de tres operaciones en un mismo evento: MSG_ADMTRA_OPENT, MSG_ADMARC_WRITE y MSG_ADMTRA_CLOSET.
MSG_ADMTRA_UNBLOCKLIST	El EXPREP o el EXPELC solicita al ADMCEN (manejador de transacciones) el desbloqueo de una lista de registros.
MSG_ADMTRA_UNBLOCKREC	El EXPPAN solicita al ADMCEN (manejador de transacciones) el desbloqueo de un registro.

J. FUNCIONES DISPONIBLES EN EL AMBIENTE PARA CONSTRUCTORES Y EXPLOTADORES DE PANTALLAS Y REPORTES.

FECHAD	<DDMMAA>	[Día Mes Año] 311293
FECHAM	<MMDDAA>	[Mes Día Año] 123193
FECHAA	<AAMMDD>	[Año Mes DíaA] 931231
MFECHAD	<DD/MM/AA>	[Día/Mes/Año] 31/12/93
MFECHAM	<MM/DD/AA>	[Mes/Día/Año] 12/31/93
MFECHAA	<AA/MM/DD>	[Año/Mes/Día] 93/12/31
FECHAMD	<DDMMMAA>	[Día Mes Año] 31DIC93
FECHAMM	<MMMDDAA>	[Mes Día Año] DIC3193
FECHAMA	<AAMMMDD>	[Año Mes Día] 93DIC31
MFECHAMD	<DD/MMM/AA>	[Día/Mes/Año] 31/DIC/93
MFECHAMM	<MMM/DD/AA>	[Mes/Día/Año] DIC/31/93
MFECHAMA	<AA/MMM/DD>	[Año/Mes/Día] 93/DIC/31
FECHAAD	<DDMMAAAA>	[Día Mes Año] 31121993
FECHAAM	<MMDDAAAA>	[Mes Día Año] 12311993
FECHAAA	<AAAAMMDD>	[Año Mes Día] 19931231
MFECHAAD	<DD/MM/AAAA>	[Día/Mes/Año] 31/12/1993
MFECHAAM	<MM/DD/AAAA>	[Mes/Día/Año] 12/31/1993
MFECHAAA	<AAAA/MM/DD>	[Año/Mes/Día] 1993/12/31
FECHAMAD	<DDMMMAAAA>	[Día Mes Año] 31DIC1993
FECHAMAM	<MMMDDAAAA>	[Mes Día Año] DIC311993
FECHAMAA	<AAAAMMMDD>	[Año Mes Día] 1993DIC31
MFECHAMAD	<DD/MMM/AAAA>	[Día/Mes/Año] 31/DIC/1993
MFECHAMAM	<MMM/DD/AAAA>	[Mes/Día/Año] DIC/31/1993
MFECHAMAA	<AAAA/MMM/DD>	[Año/Mes/Día] 1993/DIC/31
HM	<HHMM>	[Hora Min]1145
HMS	<HHMMSS>	[Hora Min Seg]114532
MHM	<HH:MM>	[Hora:Min]11:45
MHMS	<HH:MM:SS>	[Hora:Min:Seg]11:45:32
NOMBREDIA	<XXXXXXXXXX>	Obtiene el nombre del día: DOMINGO
NOMBREDIA3	<XXX>	Obtiene el nombre del día: DOM
NOMBREMES	<XXXXXXXXXX>	Obtiene el nombre del mes: DICIEMBRE
NOMBREMES3	<XXX>	Obtiene el nombre del mes: DIC
NUMDIAANO	<999>	Obtiene el número de día del año: 31 (para ENE/31/1993)
AÑO	<9999>	Obtiene el año: 1993
NUMALET	<VALOR O VARIABLE> <LETRERO1> <LETRERO2> <LETRERO3>	Traduce de números a letras
NPESOS	<VALOR O VARIABLE> <""> <""> <"">	Traduce de números a letras agregando "PESOS" y "M.N."

K. RUTINAS EXISTENTES EN EL AMBIENTE PARA LA ELABORACIÓN DE PROGRAMAS DE USUARIO Y ADMPAA.

int START_UP( ). Establece comunicación con el ambiente.

int GET_FD( ). Obtiene la descripción de un archivo.

int RESET_FILE( ). Inicializa una estructura interna asociada con algún archivo.

int GET_INFO( ). Obtiene la información general que requiere el usuario para accesar la información relacionada con un archivo.

int MOVE_KEY( ). Mueve el valor indicado de un campo llave de área de memoria local a una área reservada.

int GET_DATA( ). Mueve el valor de un campo leído del archivo indicado a una área de memoria local.

int GET_ALL_DATA( ). Mueve el valor de todos los campos leídos del archivo indicado a una área de memoria local.

int PUT_DATA( ). Mueve el valor indicado de un campo no llave de área de memoria local a una área reservada.

int READ_DBF( ). Ejecuta la operación de lectura a un archivo por medio del uso de una llave directa.

int READ_MK_DBF( ). Ejecuta la operación de lectura a un archivo por medio del uso de una llave mayor.

int READ_VK_DBF( ). Ejeccuta la operación de lectura a un archivo por medio del uso de varias llaves.

int REWRITE_DBF( ). Ejecuta la operación de actualización a un archivo por medio del uso de una llave directa.

int DELETE_DBF( ). Ejecuta la operación de eliminación de un registro a un archivo por medio del uso de una llave directa.

int WRITE_DBF( ). Ejecuta la operación de agregar un registro a un archivo por medio del uso de una llave directa.

int READ_SEQUENTIAL_DBF( ). Ejecuta la operación de lectura de bloques de registros de un archivo por medio de llave directa. La lectura puede comenzar al inicio del archivo; a partir de una llave sugerida; puede continuar leyendo el siguiente bloque. También se puede indicar que seleccione los registros que satisfagan ciertas características.

int CALL_SORTMRG( ). Solicita al ambiente la ordenación del archivo y los campos a ordenar.

void SNDPRT( ). Solicita al ambiente que el archivo señalado sea impreso o lo muestre en la pantalla.

int GET_ADMPAA_INFO( ). Obtiene la información general que requiere el usuario para accesar la información relacionada con un ADMPAA.

int USER_ROUTINE( ). Una rutina con este nombre debe ser realizada por el usuario cuando se trate de un ADMPAA. Las rutinas principales de un ADMPPA hacen el trabajo de enlace y acceso con el ambiente, pero invocan a esta rutina cuando el usuario le pide una acción y dicha acción se desarrolla aquí.

int OPEN_TRANSACTION( ). Inicia la transacción solicitando que se eliminen los recursos bloqueados asociados al proceso llamador.

int CLOSE_TRANSACTION( ). Termina la transacción solicitando que se eliminen los recursos bloqueados asociados al proceso llamador.

int BEGIN_TRANSACTION( ). Inicia la transacción solicitando que no se eliminen los recursos bloqueados asociados al proceso llamador.

int END_TRANSACTION( ). Termina la transacción solicitando que no se eliminen los recursos bloqueados asociados al proceso llamador.

int BLOCK_FILE( ). Solicita el bloqueo de todos los registros en el archivo señalado.

int WRITE_DBF_ONEPHASE( ). Equivale a tres operaciones en secuencia: OPEN_TRANSACTION( ), WRITE_DBF( ) y CLOSE_TRANSACTION( ).

El usuario puede disponer de cierta información global del ambiente:

(char *ROOTDIR). Directorio Raíz con formato //NODO/DIRECTORIO_RAIZ/.

(int MAXREC_TO_READ). El número máximo de registros que se pueden leer en READ_SEQUENTIAL_DBF( ).

(char *ARR_TIT[]). El nombre de la compañía y los títulos declarados en el archivo *.SYS del ambiente.

(int __N_L_C__). El número de licencia corporativa en operación.

(int __N_ANIOPROC__). El número de año en operación.

Estas variables globales están disponibles después de invocar la rutina START_UP( ).

L. UN ARCHIVO DE CONFIGURACIÓN DEL AMBIENTE *.SYS.

```
*ZZ131                          /*Inicia el archivo SYS con la clave del producto en 5 caracteres*/
#NUM_LICENCIA=1                 /*Número de licencia corporativa (1-65000)*/
#ANO_OPERACION=1998             /*Año de trabajo para abrir archivos*/
#CLAVE_ACCESO="123+456"         /*Clave de acceso para arrancar el ambiente*/
#NODO_RAIZ=1                    /*Número de nodo raíz del servidor*/
#PUERTO_PARAL_CANDADO=1         /*Puerto paralelo donde se encuentra el candado. Las direcciones de los puertos paralelos: 0x278 = 1,
                                  0x378 = 2. 0x3BC = 3*/
#MEMORIA_TRABAJO=256            /*Memoria de trabajo interno (256 caracteres ...)*/
#FILAS_CONPAN=30                /*Número de filas a diseñar en pantallas y reportes*/
#COLUMNAS_CONPAN=132            /*Ancho en caracteres a diseñar en pantallas y reportes*/
#MEMORIA_MENSAJES_KB=10         /*Número de KB de memoria para paso de mensajes*/
#NUM_DECIMALES=2                /*Número de decimales (2-6)*/
#TIEMPO_CONECTAR_RED=60         /*Tiempo en segundos para conectar la red al arrancar*/
#TIEMPO_ESPERA_MENSAJE=30       /*Tiempo en segundos a esperar respuesta de un mensaje*/
#TIEMPO_BLOQUEAR_RECURSO=10     /*Tiempo en segundos a bloquear recursos*/
#NUM_REG_BLOQUEAR_MEM=500       /*Número de registros a bloquear en memoria, antes de apoyarse en disco*/
#NUM_OPER_BLOQ_TRANSACCION=5            /*Número de operaciones a almacenar en una transacción*/
#NUM_TRANS_ESC_PERM=10                  /*Número de transacciones a acumular para escribir permanentemente en disco*/
#NUM_BLOQUES_MEM_ARCHIVO=6              /*Número de bloques de memoria por cada índice en ADMARC*/
#NUM_REG_DATOS_MEM=100                  /*Número de registros a guardar en memoria antes de escribir en disco*/
#NUM_ADMARC=2                           /*Número de ADMARC*/
#ADMARC[1]=COMENSAL,PLATILLO,INGREDIENTE        /*ADMARC[nodo]Lista de archivos*/
#ADMARC[1]=CONTIENE,GUSTA
#NOMBRE_EMPRESA="APLICACION PARA TESIS. CLASE 13"        /*Nombre de la compañía*/
#NUM_TITULOS=1                          /*Número de títulos usados en pantallas y reportes*/
#TITULO="PRESENTA: SERGIO RIVERA SANTIAGO"       /*Títulos*/
*FINZZ131                       /*Finaliza el archivo SYS con la clave del producto en 5 caracteres*/
```

M. UN ARCHIVO DE IMPRESORAS DEL AMBIENTE LPT.INFO.

0 //1/dev/par1 ESTA ES LA UNICA IMPRESORA DEL SISTEMA ACTUAL ;

Columna 1-3: Número de impresora que se usa para indicarle al ambiente donde se envía la impresión.

Columna 5-29: Dirección de la impresora reconocida por el SO.

Columna 30-78: Comentario.

Columna 79-79: Un punto y coma.

N. UN ARCHIVO DICCIONARIO DE DATOS *.DD.

Se documentan algunos comentarios para elaborar el diccionario de datos:

```
*ZZ131              /*Inicia el diccionario de datos del producto en 5 caracteres*/

+CAMPOS             /*Inicia la lista de campos*/
        /*Cada campo se forma de la siguiente manera:
                nombre de campo,
                tipo de dato (INT,UNSIGNED,LONG,FLOAT,DOUBLE,CHAR),
                longitud de campo, */
        .NOMBRE_COM,    CHAR,   30,
        .DIR_COM,       CHAR,   30,
        .TEL_COM,       CHAR,   8,
        .PESO_COM,      INT,    4,
        .NOMBRE_PLA,    CHAR,   30,
        .HORA_PLA,      CHAR,   9,
        .NOMBRE_ING,    CHAR,   25,
        .TEMP_ING,      CHAR,   10,
        .FIN            /*Termina la lista de campos*/

+ARCHIVOS           /*Inicia la lista de archivos*/
        -COMENSAL,      /*Nombre del archivo*/
                        /*Lista de campos*/
                .NOMBRE_COM,
                .DIR_COM,
                .TEL_COM,
                .PESO_COM,
                .FIN    /*Termina la lista de campos de un archivo*/
                >INDICES        /*nombre de la llave (lista de campos)(tipo de llave: P primaria, S secundaria, A automática)*/
                        .LLAVECOM(NOMBRE_COM)[P],
                        .FIN
        -PLATILLO,
                .NOMBRE_PLA,
                .HORA_PLA,
                .FIN
                >INDICES
                        .CPD1(NOMBRE_PLA, HORA_PLA)[P],
                        .FIN
        -INGREDIENTE,
                .NOMBRE_ING,
                .TEMP_ING,
                .FIN
                >INDICES
                        .CPD2(NOMBRE_ING, TEMP_ING)[P],
                        .FIN
        -CONTIENE,
                .NOMBRE_PLA,
                .NOMBRE_ING,
                .FIN
                >INDICES
                        .CPD3(NOMBRE_PLA, NOMBRE_ING)[P],
                        .FIN
        -GUSTA,
                .NOMBRE_COM,
                .NOMBRE_PLA,
                .FIN
                >INDICES
                        .CPD4(NOMBRE_COM, NOMBRE_PLA)[P],
                        .FIN
        -FIN    /*Termina la lista de archivos*/

+ADMPAAS            /*Inicia la lista de ADMPAA*/
        -FIN        /*Termina la lista de ADMPAA*/

*FINZZ131           /*Termina el diccionario de datos del producto en 5 caracteres*/
```

O. MENSAJES DE ERROR DEL ADMINISTRADOR DE TRANSACCIONES.

1 'ADMARC'. El archivo no existe.
2 'ADMARC'. Proporcione llaves para realizar la operación requerida.
3 'ADMARC'. No existe archivo de índices para la llave dada.
4 'ADMARC'. No se reconoce el tipo de operación.
5 'ADMARC'. La llave ya existe en el archivo.
6 'ADMARC'. Se insertó la llave, se detectó error, se intentó borrar, hubo error.
7 'ADMARC'. La llave no existe en el archivo.
8 'ADMARC'. No se pudieron dar de baja todas las llaves asociadas.
11 'ADMARC'. Error en la información de campos requeridos.
17 'ADMARC'. El número máximo de registros en el archivo fue alcanzado.
16 'EXPPAN'. No se pudo leer el registro de información.
9 'ADMTRA'. La transacción no ha sido abierta.
10 'ADMTRA'. La transacción ha saturado el espacio de operaciones. Aborte!!!
12 'ADMTRA'. Las operaciones de la transacción son rechazadas. Intente o aborte!!!
72 'ADMTRA'. No existe el registro a bloquear.
77 'ADMTRA'. Intenta desbloquear un registro pero está bloqueado todo el archivo.
82 'ADMTRA'. Intenta desbloquear el registro de un archivo que no tiene registros bloqueados.
89 'ADMTRA'. Intenta desbloquear registros de un archivo bloqueado por otra transacción.
91 'ADMTRA'. Esta transacción no tiene registros bloqueados en los archivos indicados.
92 'ADMTRA'. Intenta bloquear un archivo que ya tiene en propiedad (bloqueado).
95 'ADMTRA'. Intenta bloquear un archivo bloqueado por otra transacción.
108 'ADMTRA'. Intenta bloquear un archivo que está siendo usado.
109 'ADMTRA'. Intenta bloquear un registro de un archivo bloqueado.
136 'ADMTRA'. Se desbloquea información de una transacción que no tiene bloqueos.
138 'ADMTRA'. No se pudo bloquear el registro solicitado.
142 'ADMTRA'. Intenta bloquear/"lectura" un registro bloqueado por otra transacción.
151 'ADMTRA'. Intenta bloquear/"escritura" un registro bloqueado por otra transacción.
154 'ADMTRA'. Error en la operación sobre archivos de trabajo.
831 'ADMTRA'. La dirección del registro no corresponde al valor de la llave.

BIBLIOGRAFÍA

[1] Fernando Galindo Soria [1984]. *"Diseño y construcción de bases de datos. Vol. I."*. IPN-UPIICSA.

[2] Fernando Galindo Soria [1985]. *"Diseño y construcción de bases de datos. Vol. II."*. IPN-UPIICSA.

[3] Shamkant B. Navathe [1992]. *"Communication of the ACM."* Septiembre 1992. Vol. 35, Núm 9. Artículo: Evolution of Data Modeling for Databases, Pág. 112-123.

[4] Feliú Sagols Troncoso [1992]. *"Introducción a bases de datos centralizadas y distribuidas"*. IPN-CINVESTAV

[5] Sergio Chapa V. *"Herramientas para consulta y captura basadas en el descriptor de archivos"*. IPN-CINVESTAV.

[6] Raúl Hernández Stefanoni y Sergio Chapa V. [1990]. *"Normalización de bases de datos en C"*. IPN-CINVESTAV

[7] C. J. Date. *"Introducción a los sistemas de bases de datos"*. Addison Wesley Iberoamericana.

[8] Jeffrey D. Ullman. *"Principles of database and knowledge-base systems"*. Volumen1. Classical Database Systems. Stanford University. Computer Science Press.

[9] Ceri Stefano y Giuseppe Pelagatti. *"Distributed data base: principles & systems"*. Mc Graw Hill International Editions. Computer Science Series.

[10] Andrew S. Tanenbaum. *"Sistemas operativos modernos"*. Prentice Hall Hispanoamericana.

[11] Raúl Acosta Bermejo [1997]. *"Especificación formal de un sistema de archivos distribuido"*. Módulo de almacenamiento estable. Tesis para obtener el Grado de M. en C. IPN-CINVESTAV

[12] Martin Gruber. *"Understanding SQL"*. Sybex.

[13] Alfred V Aho, John E. Hopcroft y Jeffrey D. Ullman. *"The design and analysis of computer algorithms"* Addison-Wesley Publishing Company.

[14] *"Form Editor"*. Digital Equipment Corporation. Sistema operativo VAX 11/780.

[15] *"Cobol 4 Language. Report Writter"* Control Data Corporation. Sistema operativo NOS/BE y NOS.

[16] *"System Architecture"*. QNX Software Systems LTD. Sistema operativo QNX 2/4.

Ambiente para el Desarrollo de Sistemas de Información (ADSI)
Revisión del Dr. Feliú Davino Sagols Troncoso
México, D. F., a 1o. de junio de 1998.

Resulta muy grato el haber podido dirigir la construcción del sistema que aquí se presenta, sobre todo porque su autor, quien es un hábil programador con una gran intuición, logró conformar por sí mismo una herramienta con características sumamente interesantes, pero que en sus inicios reflejaba el desconocimiento de términos formales de algunos aspectos, que manejaba sutilmente de manera empírica.

Asimismo es notable la capacidad del autor para asimilar la formación que le hacía falta, al grado de hacer adecuaciones de gran valía científica a su sistema, así como realizar un escrito de gran calidad.

Considero que este es un magnífico trabajo cuyo único punto en contra ha sido el tiempo necesario para concluirse, sin embargo demuestra una gran capacidad del autor no solo para resolver problemas técnicos difíciles sino también para emprender el desarrollo de un trabajo científico sólido.

Ambiente para el Desarrollo de Sistemas de Información (ADSI)
Revisión del Dr. Guillermo Morales Luna
México, D. F., a 26 de junio de 1998.

1. Sobre la tesis

1.1. El trabajo consistió en la elaboración de un sistema para desarrollar bases de datos. El sistema preveé, en efecto, características importantes de manejadores de bases de datos, por lo cual, desde un punto de vista conceptual, lo considero satisfactorio y, desde el punto de vista técnico, lo aprecio con muchos méritos pues significó un trabajo considerable de diseño y de programación.

1.2. En el trabajo, luego de una adecuada introducción, el autor presenta a los módulos del sistema y posteriormente presenta las características generales de un sistema de información. Parece invertido el orden lógico, pero la secuencia elegida por el autor es motivante para la lectura. Esta segunda presentación ocupa cuatro capítulos de tamaño también adecuado. El autor presenta después consideraciones sobre el desarrollo de una aplicación en su sistema, una comparación de éste con otros convencionales y un último capítulo de conclusiones.

1.3. El sistema se desarrolló en C, con el sistema operativo UNIX, que el autor califica de tiempo real. La programación a un bajo nivel y el diseño del sistema lo hacen portátil a varias plataformas. Los elementos que componen al sistema son el núcleo mismo, los módulos administradores, los módulos constructores, los módulos explotadores, las utilerías y los procesos de control. El diseño propio del autor en este sistema hace que sea particularmente valioso su trabajo.

1.4. Considero pues que la tesis es aceptable tal como está y la valoraría yo en un nivel por encima del medio de los trabajos presentados en nuestro grupo de trabajo.

1.5. Sin embargo, algunos elementos que sería conveniente incluir en el trabajo son los siguientes:

- referencias a desarrollos similares que hayan servido de ejemplo e inspiración al autor,
- referencias a literatura técnica, tanto en publicaciones de investigación, como a manuales técnicos o de referencia,
- un ejemplo mayormente desarrollado,
- un "demo"

1.6. Es muy oportuna y objetiva la comparación que hace el autor de su trabajo con PARADOX y ORACLE.

...continúa

2. Sobre la presentación

2.1. En general está bien. Sin embargo hay algunos detallitos que necesariamente hay que corregir.

2.2. Hay que referirse a *bases de datos* así, en plural, y utilizar el singular sólo cuando se hable de *una* base de datos.

2.3. Hay que cuidar el uso de mayúsculas. Formas como Ambiente, Núcleo, Administradores, etc., son más bien impertinentes.

2.4. Cada vez que se pasa a modo itálico se introduce espacios en blanco que no deberían ir.

2.5. Hay faltas de ortografía, algunas muy graciosas. El pronombre *ellos* o *ellas* no llevan acento, la disyunción *o* tampoco, *aquello* siempre es pronombre, y por tanto no se acentúa, *resumen* es palabra grave que termina en "n" y por tanto no se acentúa, *comensal* va con "s" de Susi. Con "z" se ve muy chistosa.

2.6. El participio pasivo *acceso* corresponde al verbo *acceder* y *accesar* sólo existe como un horrendo seudotecnicismo.

2.7. Hablar de campos llaves o de un campo llave. Es decir, hay que cuidar la concordancia de plurales con plurales.

2.8. En la página 40 hay que quitarles los entrecomillados a las variables *v*, *w*, etc. pues tal como están parecen indicar otra cosa.

Ambiente para el Desarrollo de Sistemas de Información (ADSI)
Revisión del Dr. Hugo César Coyote Estrada
México, D. F., a 27 de julio de 1998.

Sobre la tesis

Desde mi punto de vista es un trabajo sobresaliente que cubre con creces los resultados que se esperan en una Tesis de Maestría.

Acerca del tema de tesis

Este trabajo se inserta en el área de Software de Sistema, el cual muy poco se ataca en México, trayendo como consecuencia una verdadera desgracia tecnológica para el país, pues somos 100% importadores de ese tipo de productos. Definitivamente el tema me parece muy interesante, de enorme complejidad y hago votos para que sirva de catalizador y dé como resultado que otros estudiantes brillantes se animen a trabajar en esa área del desarrollo de software.

Acerca del desarrollo de la tesis

Desarrollos de software de este tipo requieren de la aportación de mentes brillantes, críticas y perseverantes, aspectos a no dudar posee el sustentante por lo cual ha producido un trabajo de gran valor como Tesis de Maestría. De acuerdo a lo que he podido apreciar, el autor logró desarrollar todo lo que se propuso y con una gran calidad. Este hecho no debe hacer pensar que el producto pueda ser inmediatamente comercializable, pero sí podría ser usado en las escuelas de Informática para enseñar a los estudiantes a crear sistemas de información de manera rápida, con lo cual el autor podría retroalimentarse y realizar las mejoras necesarias para una eventual y futura comercialización.

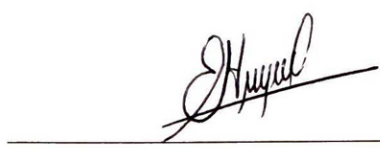
Acerca de la redacción y la presentación

El escrito original contenía una buena dotación de errores ortográficos y una presentación que por timidez no resaltaba todas las bondades que este trabajo posee. Estos aspectos simplemente corroboran el poco énfasis que la Educación Pública en México y en particular en el I.P.N. le brindan a esos enormes detalles que en muchos casos son la causa de que trabajos excepcionales simplemente pasen desapercibidos o bien tengan una buena acogida por parte del público.

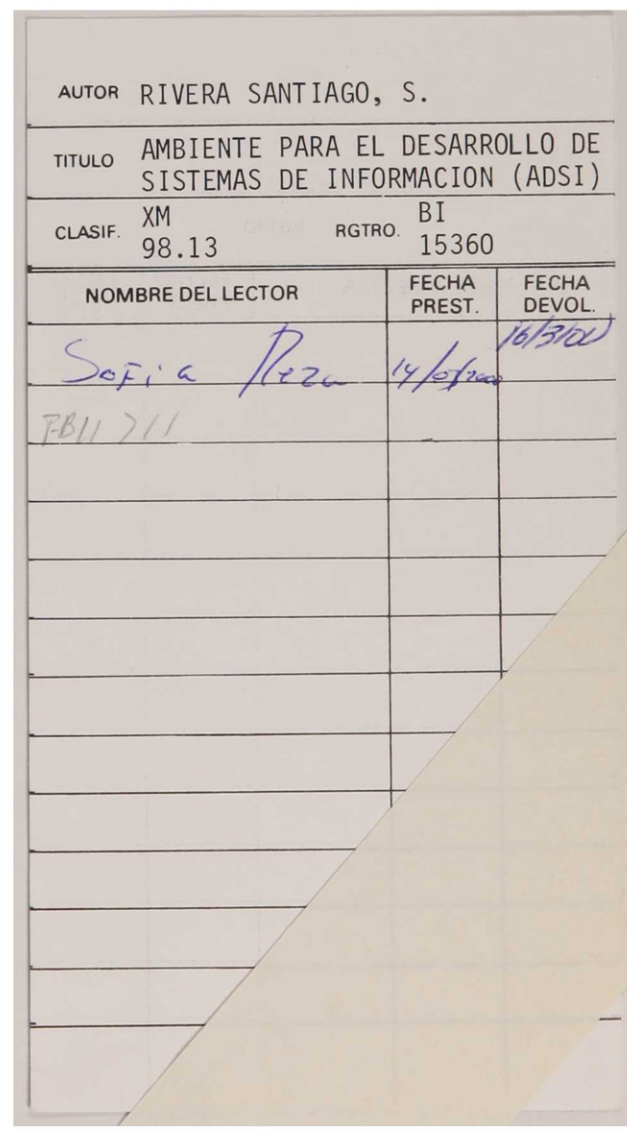
El jurado designado por la Sección de Computación del Departamento de Ingeniería Eléctrica del Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional, aprobó esta tesis el día 14 de agosto de 1998.

Dr. Feliú Davino Sagols Troncoso

Dr. Guillermo Morales Luna



Dr. Hugo César Coyote Estrada



AUTOR RIVERA SANTIAGO, S.

TITULO AMBIENTE PARA EL DESARROLLO DE SISTEMAS DE INFORMACION (ADSI)

CLASIF. XM 98.13 RGTRO. BI 15360

NOMBRE DEL LECTOR	FECHA PREST.	FECHA DEVOL.
Sofia Reza	14/ [illegible]	16/3/00
FBII 211		