UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

SISTEMA DE ADMINISTRACIÓN DE BASES DE DATOS PARA LA DIRECCIÓN GENERAL DE ADMINISTRACIÓN ESCOLAR

TESIS

QUE PARA OBTENER EL TÍTULO DE

INGENIERO EN COMPUTACIÓN

P R E S E N T A

JAVIER CORTÉS
ELSA MIRIAM GUERRA NIEVES
ARMANDO VEGA ALVARADO

DIRECTOR

ING. ANGÉLICA NAVA JUÁREZ

COODIRECTOR

ING LAURA SANDOVAL MONTAÑO

México, D.F. Noviembre del 2002

INDICE

Prefacio

Bases de datos es uno de los términos más populares dentro de la informática. La mayoría de las personas interactúan con una computadora y casi con toda seguridad han interactuado con algún producto de software para que el manejo de las bases de datos y su administración sea de forma sencilla.

Las bases de datos son una herramienta la cual desde desarrolladores de sistemas hasta usuarios finales han utilizado. Se han desarrollado productos de software los cuales permiten realizar una comunicación lógica y sencilla, ocultando la complejidad teórica, conceptual y física en la cual se basa el producto.

A pesar de todo, es cierto que estos productos no cubren con todas las características que se requieren para un sistema, la UNAM cuenta con distintas escuelas y facultades así como escuelas preparatorias y CCHs, de tal forma que la administración de las bases de datos con las que se cuenta por cada plantel se ha hecho compleja, es por tal razón que se implemento el Sistema de Administración de Bases de Datos (SABAD). Este sistema se adapta a las particularidades que se requieren para la administración de las bases de datos del SIAE y que ningún producto de software para la administración ofrece.

Objetivo

Realizar un sistema que permita la administración de los servidores de bases de datos Sybase de la Dirección General de Administración Escolar (DGAE), para facilitar el control, la configuración y el mantenimiento de los mismos, garantizando la seguridad e integridad de la información, apoyando así a las tareas de administración de los servidores de bases de datos.

Contenido

Capítulo 1. Dentro de este capítulo se encuentran los antecedentes de la Dirección General de Administración Escolar (DGAE) su objetivo, su misión y sus funciones así como su estructura interna.

Se explica lo qué es el Sistema Integral de Administración Escolar (SIAE) el cual permite la simplificación, agilización y descentralización de los trámites para todos los alumnos que se encuentran inscritos ya sea en escuelas, facultades, escuelas preparatorias y CCHs, esta información debe ser de carácter seguro y confiable.

Por otro lado se explica cómo se encuentran estructuradas las bases de datos del SIAE, en las cuales se almacenan todos los registros de los alumnos de la UNAM.

Capítulo 2. En este capítulo se explica lo que es la administración de una base de datos, se describen los componentes para las bases de datos, también se explican las responsabilidades que debe tener un administrador las cuales son importantes para el desarrollo e implementación de las mismas.

Se explican las características de las bases de datos del SIAE y cómo es que se encuentran conformadas apegándose a los conceptos generales para una base de datos.

Además se hace la descripción del problema en donde se detallan los procedimientos actuales, las necesidades actuales y los resultados esperados para la DGAE.

Capítulo 3. Dentro de este capítulo se hace el análisis y diseño del Sistema de Administración de Bases de Datos (SABAD) en el cual se propone la solución así como las tareas que se deben tomar en cuenta, describiendo lo que contendrá cada uno de los módulos propuestos en el capítulo 2.

Se explican los procedimientos, las herramientas y los lenguajes que se emplearán para el desarrollo del SABAD.

Capítulo 4. En este capítulo se explican los estándares para las bibliotecas de programación, las bibliotecas que se emplearán para el desarrollo de la interfaz gráfica además de las bibliotecas que permiten realizar la conexión al servidor de bases de datos. También se describen las características del hardware utilizado para el desarrollo del SABAD.

Se muestran las ventanas más representativas del SABAD además del código que genera estas ventanas.

Por otro lado se muestran los distintos reportes generados por el SABAD los cuales permiten verificar la consistencia de las bases de datos así como su integridad con respecto a la información contenida.

1. ANTECEDENTES

1.1 ¿Qué es la Dirección General de Administración Escolar (DGAE)?

La Dirección General de Administración Escolar (DGAE) es una dependencia normativa y de dirección dependiente de la Secretaría General de la UNAM, es el instrumento administrativo capaz de traducir los ordenamientos contenidos en la legislación universitaria y las disposiciones emitidas por la autoridad en esta materia, a planos operativos de la administración escolar.

1.1.1 Objetivo

Contribuir en el cumplimiento de las funciones sustantivas de la UNAM, para realizar la validación y ratificación de los resultados de las evaluaciones del proceso enseñanza-aprendizaje, en sus diferentes niveles y modalidades, desde el ingreso hasta la titulación de los alumnos de la UNAM, acorde con las necesidades y evolución académica de la población estudiantil y de la institución.

1.1.2 Misión

La misión de la Dirección General de Administración Escolar (DGAE) es dar validez a los resultados del proceso enseñanza - aprendizaje durante la vida académica de los alumnos en la institución, desde su ingreso hasta la conclusión de sus estudios.

1.1.3 Funciones

- Dirigir, coordinar y realizar las actividades de administración escolar en la Institución, con base en la normatividad y a los requerimientos de servicio.

- Planear, organizar, dirigir y controlar los eventos de primer ingreso a la UNAM.

- Ejercer las facultades que le confiere la Legislación Universitaria en relación a la administración escolar.

- Dar seguimiento y controlar la permanencia de los estudiantes, durante las diversas etapas de su preparación.

- Registrar y aplicar los planes y programas de estudio, para la certificación de los alumnos.

- Definir e implementar los programas para el cumplimiento de los servicios de administración escolar.

- Legalizar los estudios de los alumnos de la UNAM.

- Resguardar y conservar los distintos documentos que avalan la trayectoria académica de los alumnos y exalumnos de la UNAM.

- Emitir diplomas, títulos, certificados y grados de los alumnos.

- Planear, organizar y controlar los sistemas de cómputo requeridos por la DGAE.

- Controlar la emisión de documentación computarizada referente al registro escolar de los alumnos.

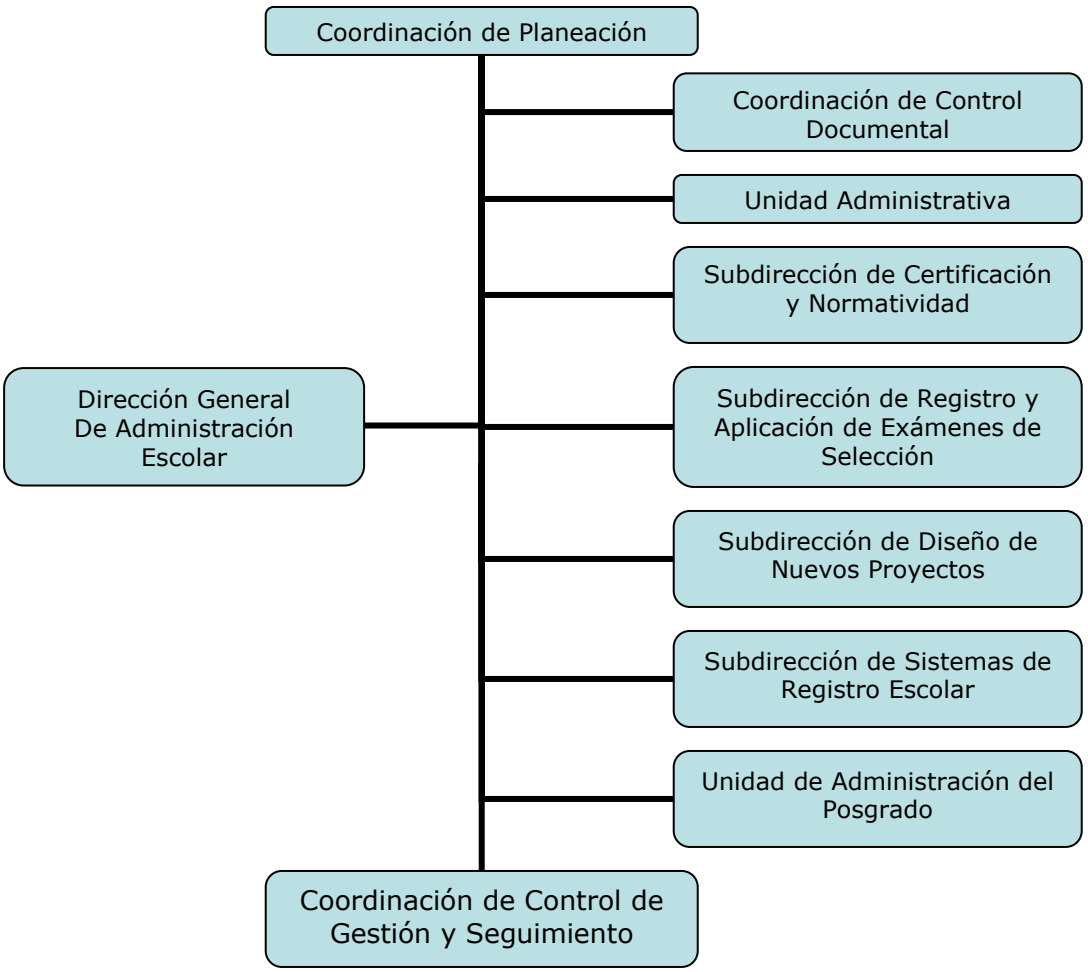
- Difundir la información que la UNAM emita sobre la administración escolar.

- Representar a la UNAM ante diversos órganos y comisiones.

1.1.4 Organigrama

La DGAE depende jerárquicamente de la Secretaría General de la UNAM, la cual a su vez depende directamente de la Rectoría. De la Secretaria General, la DGAE recibe las políticas generales para su funcionamiento.

La DGAE está integrada por cuatro subdirecciones, tres coordinaciones y dos unidades administrativas.



Coordinación de Planeación

Asesora a la Dirección General y al personal directivo de la DGAE en materia de administración, planeación, organización, dirección, control y evaluación de programas y subprogramas.

Coordinación de Control Documental

Planear, coordinar, organizar, controlar y dar fe que los documentos académicos y administrativos que estén presentes conformando el expediente escolar de los alumnos, de acuerdo a los normas establecidas en la Legislación Universitaria, con el fin de autorizar los Exámenes Profesionales y de Grado, expedir Certificados de Estudios, Títulos Profesionales, Grado y Diplomas. Establecer los vínculos necesarios con las diferentes dependencias de la UNAM o con las correspondientes de la Secretaría de Educación Pública para garantizar la eficiencia y simplicidad de los procedimientos.

Unidad Administrativa

Organizar, controlar, evaluar y tramitar los asuntos administrativos relacionados con el ejercicio del presupuesto, el control del personal y el suministro de los bienes y servicios, con la finalidad de racionalizar recursos y coadyuvar al cumplimiento de las atribuciones conferidas a la Dirección General de Administración Escolar.

Subdirección de Certificación y Normatividad

Coordinar y supervisar la certificación de estudios de los alumnos en las Oficinas de Servicios Escolares de las Escuelas y Facultades de la UNAM Elaborar las normas y reglamentos necesarios que apoyen los servicios escolares de acuerdo a los ordenamientos jurídicos vigentes de la legislación de la UNAM

Subdirección de Registro y Aplicación de Exámenes de Selección

Planear, desarrollar y efectuar el registro, la aplicación y la evaluación del examen de selección para aspirantes de primer ingreso a la UNAM, previniendo los recursos correspondientes, para brindar una atención apropiada a los aspirantes.

Subdirección de Diseño de Nuevos Proyectos

Diseñar, desarrollar y supervisar la implementación de nuevos sistemas de cómputo que intervengan en la actualización de las actividades de la administración escolar y de otras dependencias de la UNAM Así mismo coordinar, planear, organizar, vigilar y controlar el evento de primer ingreso de los niveles técnico, bachillerato y licenciatura por concurso de selección, así como de pase reglamentado a la licenciatura de los egresados de la Escuela Nacional Preparatoria y del Colegio de Ciencias y Humanidades.

Subdirección de Sistemas de Registro Escolar

Dirigir, supervisar, coordinar, resguardar y evaluar la sistematización y emisión de la documentación computarizada relacionada con el registro escolar y el historial académico de los alumnos de la UNAM, niveles bachillerato, técnico y licenciatura, bajo el marco legal que rige la Legislación Universitaria en este ámbito.

Unidad de Administración del Posgrado

Planear, organizar, integrar, dirigir y controlar el proceso de registro, seguimiento y egreso académico de los alumnos del posgrado, mediante un servicio eficaz y eficiente.

Coordinación de Control de Gestión y Seguimiento

Realizar el seguimiento de los acuerdos que se realizan en la dependencia y de esta forma brindar apoyo al cumplimiento de los planes y programas propuestos.

1.2 ¿Qué es el Sistema Integral de Administración Escolar (SIAE)?

La Dirección General de Administración Escolar, a través de la Subdirección de Sistemas de Registro Escolar (SSRE), desarrolló el Sistema Integral de Administración Escolar (SIAE), para otorgar un servicio de calidad y eficiencia a cada miembro que integra nuestra máxima casa de estudios, en todo lo referente al registro y seguimiento académico de la trayectoria escolar de los alumnos de la UNAM

Tiene como objetivo lograr dentro de la Administración Escolar de la UNAM, la simplificación, agilización y descentralización de los trámites académico-administrativos de los alumnos en todas las escuelas y facultades, integrando información confiable y consistente como el producto de la coordinación de todos los elementos de la Institución.

1.2.1 Historia

En la Subdirección de Sistemas de Registro Escolar (SSRE) se han marcado diversas etapas de transformación, en cuanto al manejo de sistemas enfocados a atender el registro y seguimiento académico de los alumnos de las escuelas y facultades de la UNAM.

Sus inicios se ubican en 1967 cuando se comienzan a desarrollar servicios computarizados de apoyo a la administración universitaria.

En 1969 orientan los recursos de cómputo a la solución de los problemas administrativos de la universidad.

En 1971 se diseña y desarrolla el Sistema Automatizado de Registro y Control Escolar (SARCE), que entra en funcionamiento en 1972.

En 1978 el procedimiento de tarjetas perforadas cambia a formas ópticas, creándose un subsistema de control de actas emitidas por plantel-semestre.

En 1980 se emprende la reprogramación del sistema y se modifica la estructura del registro de las historias académicas, lo que se traduce en la disminución de tiempos de proceso.

En 1983 se crea el sistema de exalumnos para alumnos inactivos en registro escolar. Aunado a esto se crea el sistema de Dictámenes que modifica y controla la transferencia de alumnos al sistema de exalumnos y viceversa.

En 1986 se cambia la estructura de acceso directo al registro del alumno vía número de cuenta, por una tabla relativa simplificada reduciendo así el tiempo de proceso de máquina.

En 1987 se instrumenta en el sistema el control del artículo 19 del Reglamento General de Inscripciones (RGI), y comienza a controlarse la autorización para presentar más de dos extraordinarios por período escolar. Así mismo, se instrumenta el artículo 27 del RGI con el fin de impedir la tercera inscripción a un ordinario.

En este mismo año se concentraron esfuerzos en el diseño de un ciclo de planeación que aseguró la captación y entrega oportuna de información, organizando fechas y actividades para los eventos de reinscripción, exámenes extraordinarios y emisión de historias académicas.

En apoyo a algunos planteles, la SSRE. trabaja en el desarrollo de sistemas de cómputo para la administración escolar local, los cuales, comienzan a enviar información a través de medios magnéticos, para ser procesados en el SARCE.

En los años 1991-1993 las políticas de ingreso y los procedimientos, se modificaron. A los planteles que cuentan con sistemas de cómputo en licenciatura se les envía información, para apoyar el evento de reinscripción local, tales como archivos de historial académico, directorio de alumnos, resumen de historial académico y relación de asignaturas.

La forma de envío y recepción de información es a través de formatos de captura, hojas de lectura óptica, discos magnéticos y transferencia de archivos vía línea directa al SARCE ó red UNAM

En los años 1994 a 1998 los planteles cuentan con sistemas locales en los planteles de bachillerato, técnico y licenciatura, la manera de envío es a través de red UNAM en archivos; esto trajo como consecuencia la minimización en el tiempo de entrega.

Actualmente, el SIAE se encarga del registro y seguimiento escolar de los alumnos de todas las Escuelas y Facultades de la UNAM, verificando el cumplimiento de los reglamentos escolares de la institución contenidos en la Legislación Universitaria.

Mantiene la información actualizada de las asignaturas, planes de estudio y carreras que ofrece la UNAM en sus diferentes planteles, de acuerdo a las autorizaciones hechas por Consejo Universitario, a través del Departamento de Programas y Planes de Estudio.

Cuenta con medios de monitoreo en la entrega de documentos oficiales para anexarlos al expediente del alumno. Genera información para los sistemas locales de los planteles, para que en el momento que lo requieran, puedan hacer uso de ella.

Da acceso a la información en línea, por medio de módulos automatizados para el servicio de consulta para las secretarías académicas del plantel, oficinas y departamentos involucrados en movimientos académicos dentro de la Dirección General de Administración Escolar y finalmente para ser manejado por el propio alumno.

1.2.1 Servicios que ofrece la SSRE a través del SIAE

Actualización

De carreras, planes de estudio, asignaturas y planteles que se ofrecen en la UNAM En este apartado, es el Departamento de Planes y Programas de Estudio y con base en la autorización del Consejo Universitario a un plan de estudios, el que asigna claves a las nuevas carreras y asignaturas, en respuesta a un requerimiento de alguna facultad o escuela.

Consulta

A planes de estudio, proporciona acceso a escuelas, facultades, alumnos y público en general para consulta sobre asignaturas, créditos, seriación, equivalencia de asignaturas entre diferentes planes de estudio de una misma carrera, así como consulta de los requisitos de ingreso y titulación a la carrera.

Acceso al Ciclo de planeación

En este apartado se puede consultar las fechas compromiso, entre la Subdirección de Sistemas y el plantel para el procesamiento de información relativa a actualización de historias académicas, reinscripción, reactivación de alumnos afectados por Art. 20, cambios de carrera y de unidad y cierre de periodo escolar, entre otros.

Registro

De alumnos de primer ingreso al ciclo (secundaria, bachillerato, técnico y licenciatura)

Solicitud y autorización de trámites

El plantel tiene acceso para la actualización del registro del alumno, a través de los siguientes trámites: cambios de unidad, y cambios internos de carrera. Asimismo, el alumno puede solicitar la corrección de datos personales como nombre, fecha de nacimiento, nacionalidad, domicilio y teléfono en su plantel y tiene acceso al seguimiento de su solicitud en forma automatizada (vía Internet); así como solicitud de segunda carrera y simultánea.

Registro y auditoría

De la reinscripción e inscripción a cursos ordinarios, exámenes extraordinarios, que se efectúen en los sistemas locales del plantel.

- *Registro de Información* desde el plantel hacia el SIAE correspondiente a profesores, grupos, alumnos asignaturas, reinscripción, inscripción a extraordinarios. El registro garantizará el cumplimiento de los Reglamentos generales de Inscripciones y Exámenes

- *Validación* de la reinscripción contra el plan de estudios en el cual el alumno se encuentra registrado, por lo que al momento de que el alumno la solicita, se verifica: seriación, acreditación anterior de asignaturas, doble inscripción en ordinario y límite de tiempo para cursar en ordinario.

- *Autorización de extraordinarios*: para la inscripción a exámenes extraordinarios, la respuesta es inmediata para conocer si el alumno ya acreditó anteriormente la asignatura o la autorización para inscribir más de dos exámenes.

Consulta a la trayectoria académica del alumno

El SIAE cuenta con información de alumnos en relación a todos los movimientos académicos dentro de la UNAM, como es el tipo de ingreso al ciclo, los cambios de plantel, sistema (escolarizado o abierto) carrera o plan de estudios que el alumno realiza, así como su situación actual y su ubicación en el plan de estudios de la carrera que cursa, ya sea como primera o segunda carrera o carrera simultánea dando el acceso a las escuelas y facultades y al propio alumno.

Consulta de la historia académica del alumno

El plantel y el alumno con un número de identificación personal, cuenta con el acceso para conocer el avance académico mediante la consulta de su historia académica, así como la información referente a su inscripción y a sus datos personales. (Vía Internet).

Emisión e impresión de directorio de alumnos

Por tipo de ingreso, sexo, nacionalidad, tipo de egreso o afectación.

Obtención de estadísticas

El acceso a información actualizada y en forma expedita es una de las principales características que ofrece la Subdirección de Sistemas a través del SIAE por lo que el plantel tiene la posibilidad de contar con estadísticas sobre los diferentes eventos que se lleven a cabo; reinscripción. Inscripción a extraordinarios, control de cupos y grupos, estadísticas de aprobación y reprobación, por asignatura, etc.

1.3 Las bases de datos del SIAE

El procesamiento de la información es esencial para la administración de los gobiernos, los negocios y la educación, y aún para las actividades de entretenimiento y ocio. En nuestra sociedad es vital para una organización o empresa proporcionar información correcta y puntual para apoyar la toma de decisiones y otras actividades gerenciales. Como resultado del crecimiento económico y avances tecnológicos, muchas organizaciones han crecido tanto en el tamaño como en la sofisticación de sus funciones administrativas. Mientras el volumen de procesamiento de datos crece a una rapidez sin precedentes, también crece la demanda de medios eficientes para manejarlos.

En los sistemas de información convencionales, las aplicaciones individuales se desarrollaban independientemente, y cada programa de aplicación procesaba sus propios archivos privados. Un banco de registros no computarizados podría duplicar toda la información de los clientes en el departamento de préstamos así como el de ahorros. Como resultado, algunas actividades se duplicaban y la información redundante se almacenaba para usarla en distintas operaciones. Al final de los años 60 surgió el sistema de bases de datos para superar los problemas asociados con los sistemas de información tradicionales. Archivos individuales se integraban en una sola base de datos para ser compartidos por todos los usuarios de una empresa. En vista de la centralización de los datos por medio de un sistema de bases de datos, los requerimientos de todos los usuarios se podían coordinar de una manera efectiva para alcanzar la mejor utilidad general para la organización.

El SIAE cuenta con bases de datos para almacenar la información de toda la comunidad universitaria, cuenta con un servidor central Sun Sparc Enterprise 3500 de 4 procesadores, memoria de 1 Gb, y un disco duro de 120 Gb, este servidor contiene dos servidores de bases de datos uno denominado DGAE y otro denominado SSRE.

El servidor de bases de datos DGAE almacena todos los registros de Historias Académicas (Calificaciones de alumnos) y alumnos de la UNAM, el servidor SSRE almacena una base de datos por cada plantel de la UNAM, la estructura de cada base de datos es la misma. Estas bases almacenan información exclusiva de cada plantel como lo es la inscripción de los alumnos.

Para crear los elementos dentro de la base de datos se emplean *scripts*, los cuales crean las tablas de usuario, constrains, índices, procedimientos almacenados, triggers, reglas, etc. que se detallaran en el siguiente capítulo.

La DGAE a través de las recomendaciones del consejo asesor de cómputo de la UNAM adquirió las licencias del SQL Server de la compañía Sybase.

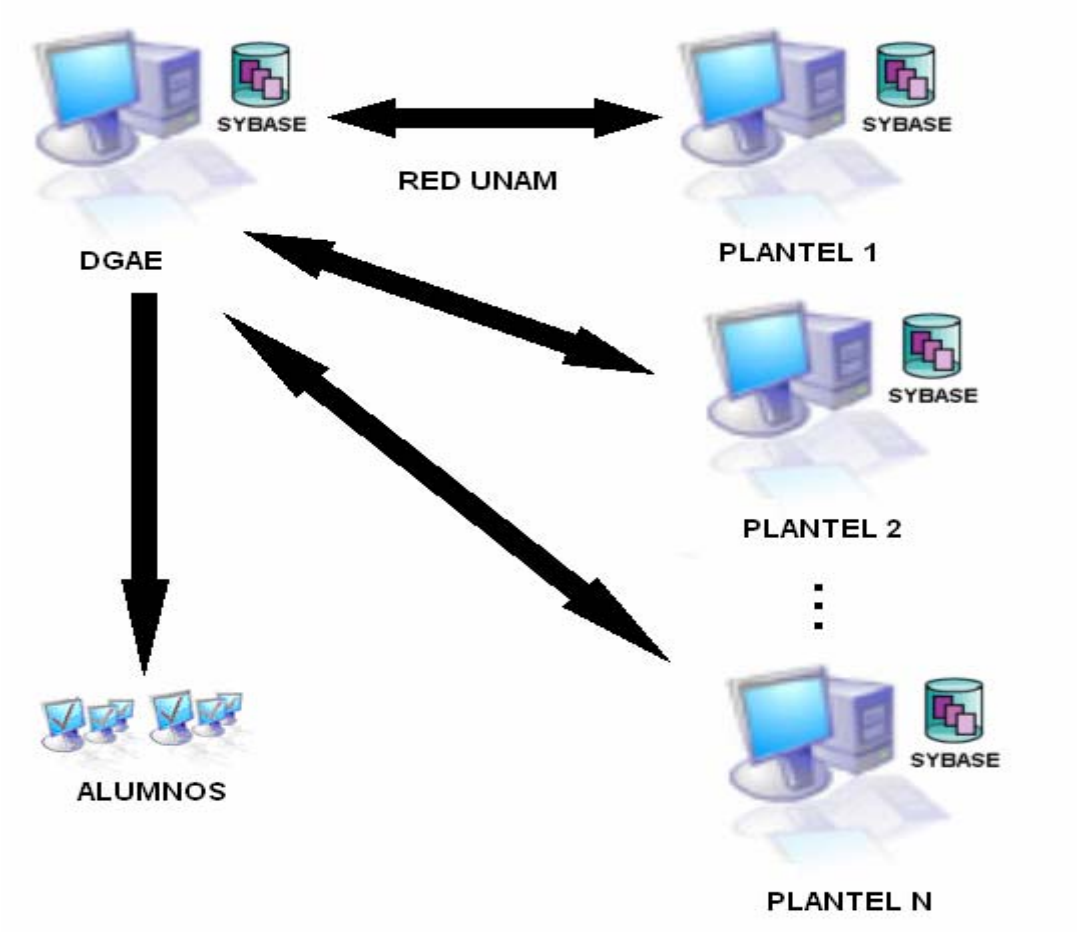
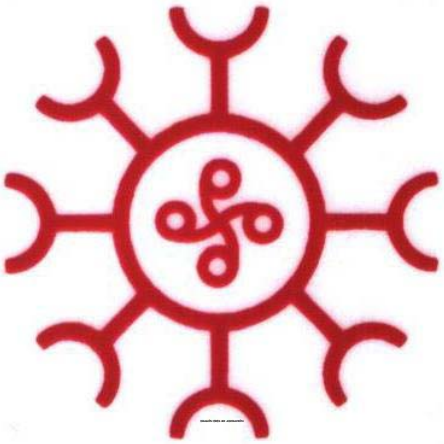


Figura 1.1 Información DGAE - Plantel



2. ADMINISTRACIÓN DE BASES DE DATOS DEL SIAE

2.1 Qué es la administración de una Base de Datos

Un sistema de bases de datos es un sistema computarizado de información para el manejo de datos por medio de paquetes de software llamados sistemas de manejo de bases de datos (DBMS). Los tres componentes principales de un sistema de bases de datos son el hardware, el software DBMS y los datos por manejar.

2.1.1 Definición de base de datos

Una base de datos es una colección de archivos interrelacionados creados con un DBMS. El contenido de una base se obtiene combinando datos de todas las diferentes fuentes en una organización, de tal manera que los datos estén disponibles para todos los usuarios, y los datos redundantes puedan eliminarse, o al menos minimizarse. La figura 2.1 muestra la base de datos como un recipiente de datos a ser compartidos por varios programas. El usuario podrá recobrar datos de varias partes de la base ya que los archivos ahí almacenados, están conectados directa o indirectamente.

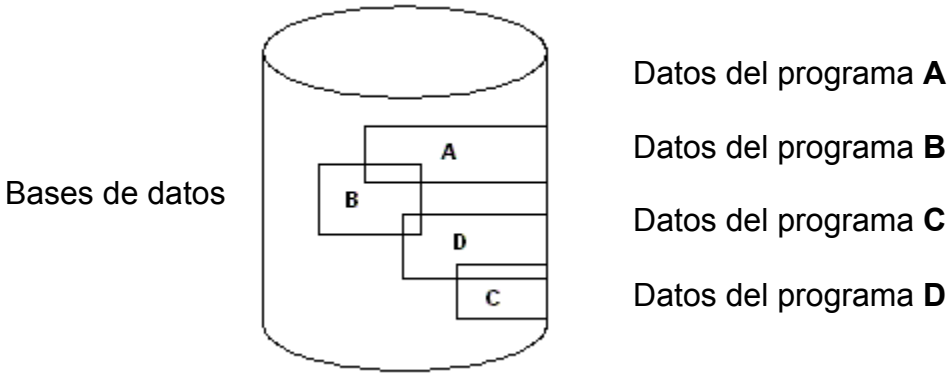


Figura 2.1 Vista esquemática de los datos almacenados.
Los datos se almacenan físicamente en una disposición distinta a la de la perspectiva lógica. Todos los usuarios pueden tener acceso a los datos

En el mundo actual, la cantidad de datos almacenados en una empresa crece casi en progresión geométrica. El proceso administrativo de toma de decisiones depende de la calidad y cantidad de información. La información que se puede extraer de la base de datos es uno de los recursos más valiosos de la empresa. La base de datos se debe diseñar, procesar y mantener adecuadamente para proporcionar información correcta en el momento oportuno a las personas autorizadas; estas responsabilidades corresponden al administrador de la base de datos (DBA Database Administrator) y a su personal. La función de administración de la base de datos está compuesta de personas y procedimientos. La expresión "DBA" se usa indistintamente para referirse tanto a la persona como a la función.

La función del DBA auxilia a la empresa en el manejo y control del recurso de datos. Esta función corresponde sobre todo a la de un administrador que a la de un técnico. La existencia y la función del DBA identifican a los datos como un recurso; como tal, esta identificación comienza con la introducción de un sistema de manejo de la base de datos (DBMS, Data Base Management System), pero existe una diferencia entre DBMS y DBA. La mayoría de los medios el DBMS es un paquete que se compra, mientras que al DBA es una función. El DBA mantiene una visión general de la empresa, a la que llamamos modelo "conceptual". Este enfoque es el modelo de datos de la organización.

2.1.2 Sistemas de manejo de bases de datos

El DBMS es la porción del software más importante de un sistema de base de datos. Algunos DBMS disponibles comercialmente son: ORACLE, INFORMIX, SYBASE, etc.

Un DBMS es una colección de numerosos elementos de software interrelacionados, cada uno de los cuales es responsable de alguna tarea específica.

Las funciones principales de un DBMS son:

1) Crear y organizar la base de datos.
2) Establecer y mantener las trayectorias de acceso a las bases de datos, de tal manera que los datos en cualquier parte de la base se puedan acceder rápidamente.
3) Mantener los datos de acuerdo con las peticiones de los usuarios.
4) Mantener la integridad y seguridad de los datos.
5) Registrar el uso de las bases de datos.

2.2 Responsabilidades de un administrador

La primera tarea importante del administrador del servidor de bases de datos es resolver las diferencias entre varias funciones de la organización, con el fin de desarrollar una estructura conceptual y, más tarde, lógica del modelo de base de datos para la empresa.

En la fase inicial del diseño de la base de datos, el DBA se deberá concentrar en:

- La definición de los campos de datos y las entidades de la empresa.
- La determinación de los distintos nombres que se usarán para referirse a los mismos elementos.
- La definición de las relaciones entre los campos de datos.
- El establecimiento de la descripción textual de los campos de datos.
- El conocimiento de los departamentos o los usuarios que serán responsables de mantener la exactitud de los datos (por ejemplo, actualización, integridad de los datos).
- Determinación del uso de los campos de datos con propósitos de control y planeación, esto es la determinación de las personas autorizadas para efectuar ciertas operaciones.

La administración del sistema implica generalmente las siguientes responsabilidades:

- Crear y configurar servidores
- Crear bases de datos
- Crear dispositivos de disco para las bases de datos y sus objetos
- Crear logins de usuario
- Conceder acceso de usuarios a los datos que necesitan
- Controlar los respaldos de bases de datos
- Recuperar bases de datos
- Monitorear la actividad de las bases de datos
- Mantener un óptimo funcionamiento del servidor de bases de datos

2.3 Características de las bases de datos del SIAE

Debido a la gran cantidad de información que se procesa en el SIAE, se empleó el servidor de bases de datos Sybase el cual es uno de los manejadores relacionales más conocido y poderoso del mercado actual. Su capacidad de almacenamiento, ambiente multiusuario y excelente desempeño lo han convertido en uno de los DBMS (Sistema Manejador de Bases de Datos) más exitosos de la historia de la computación, ganándose una enorme reputación en menos de 10 años, gracias a sus innovaciones, confiabilidad y facilidad de uso.

Las bases de datos del SIAE se apegan a las características generales de una base de datos como pueden ser: Versatilidad para la representación, desempeño, mínima redundancia, capacidad de acceso, simplicidad, integridad, afinación, seguridad y privacidad de igual forma estas bases de datos están en un esquema de base de datos distribuidas.

2.3.1 Versatilidad para la representación de la información

Si se considera que un procedimiento es capaz de "ver" la información que maneja como un registro, la organización de la información en la base de datos debe permitir que diferentes procedimientos puedan construir diferentes registros a partir de la información existente en la base de datos. Estos registros (lógicos) estarán formados por registros de datos que forman parte del dominio del problema y que son derivados del conjunto de registros de datos existentes en ese problema y, además, estos registros lógicos construidos por el procedimiento deben ser independientes de los registros físicos existentes en la base de datos para almacenar la información.

En SIAE el control de la gran mayoría de actualizaciones y de consultas, es por medio de procedimientos almacenados y de programas en lenguaje C, los cuales limitan de manera enfática el acceso a la información esto además de las posibilidades que nos da la creación y configuración de vistas en las cuales podemos limitar el subconjunto de las representaciones de la información ya sea de manera vertical u horizontal.

2.3.2 Desempeño

Las bases de datos deben asegurar un tiempo de respuesta adecuado en la comunicación hombre-máquina, permitiendo el acceso simultáneo al mismo o distinto conjunto de registros de datos por el mismo o distinto procedimiento o canal de comunicación.

En el SIAE se cuenta con diferentes clientes (programas) que requieren cualquier cantidad de información de manera simultánea, lo mismo una consulta vía WEB, las cuales son de gran tamaño, que un programa de inscripción o de normalización, por lo tanto es una preocupación constante por parte de los administradores de base de datos los tiempos de respuesta y para esto se cuida mucho la configuración y se deben implementar mecanismos que nos permita verificar de manera oportuna el desempeño. En su mejor momento las bases de datos dan tiempos de respuesta, en horas pico, del orden de 19 milisegundos para una consulta.

2.3.3 Mínima redundancia

La existencia de redundancia es nefasta debido a la posibilidad de inconsistencia en la información almacenada en la base de datos. La redundancia implica la existencia de varias copias de un mismo registro las cuales pueden en un momento dado, tener distintos valores, por lo tanto si existen varias copias de un mismo registro debemos implementar mecanismos que nos permitan garantizar la consistencia de la información cuando estas copias sean utilizadas por varias aplicaciones o clientes (programas).

Por otro lado, y en contra parte, si solo existe una copia de un registro debemos establecer los mecanismos para garantizar el acceso a esta copia por varios procedimientos o programas para garantizar un desempeño aceptable.

En SIAE como se cuenta con un esquema de base de datos distribuido se vió en la necesidad de aceptar una redundancia mínima con entidades como pueden ser *pde* (planes de estudios), *crr* (carreras), etc. Estas entidades (tablas) son comunes para toda la universidad pero no podíamos concentrar en un solo lugar esta información así que cada base de datos cuenta con su copia, para que de este modo puedan los diferentes alumnos y autoridades consultar de manera local al momento que lo requieran la información correspondiente a cualquier plan de estudios o cualquier carrera, por esto se tendrá que desarrollar

aplicaciones que puedan verificar la consistencia de la información para poder garantizarla.

2.3.4 Integridad

La integridad de una base de datos hace referencia a la veracidad de los datos almacenados con respecto a la información existente. Como los datos de la base de datos son manejados por una gran cantidad de usuarios haciendo uso de muchos procedimientos almacenados que tratan los datos de diferentes formas, es necesario garantizar que estos datos no sean destruidos ni modificados de forma anómala (naturalmente, este control debe tenerse en cuenta tanto con el valor de los registros de los datos como de las relaciones existentes entre ellos).

Durante el procesamiento se pueden producir fallos de muy diversa naturaleza: errores del sistema operativo, del hardware, software, etc. Así los procedimientos que manejan la información (en inserción, borrado y actualización) deben asegurar que el sistema pueda garantizar la integridad de la información a pesar de los errores que se puedan producir, a causa de los fallos.

Además de garantizar la integridad de la base de datos con respecto a este tipo de fallos, esta integridad debe garantizarse con respecto a la veracidad de los datos y sus relaciones.

En SIAE se requiere implementar varios programas que sirvan de auditores de la integridad de la información además de que verifiquen la autenticidad de cada registro con respecto a normas establecidas, de tal manera que podamos ser capaces de detectar faltantes de registros, modificaciones no autorizadas, inserciones no permitidas o incluso incongruencias en las diferentes bases de datos y sus replicaciones o en sus diferentes distribuciones.

2.3.5 Seguridad y privacidad

La seguridad de una base de datos hace referencia a la capacidad de ésta para proteger los datos contra pérdida total o parcial por fallos del sistema o por accesos accidentales o intencionados a los mismos. Mientras que la privacidad de una base de datos hace referencia a la reserva de la información de la misma a personas no autorizadas.

La información de la base de datos del SIAE es de vital importancia para una gran parte de la comunidad universitaria de la UNAM, por lo tanto se puso especial interés en ese rubro al momento del diseño. Se requirió

implementar cualquier cantidad de medidas que nos garanticen por todos los medios que la información no será vista por personas no autorizadas, esto tomando en cuenta los diferentes tipos de usuarios y los diferentes tipos de clientes (programas) que se han desarrollado en el SIAE; por ejemplo los clientes en PHP(Lenguaje de programación orientado al diseño de páginas de Internet) que en esencia son los usuarios vía WEB son de un gran tamaño en consultas al día, los usuarios del tipo "privilegiados" los cuales pueden ver un poco más que el usuario promedio, y los usuarios del tipo "producción" los cuales sólo entran a ciertas horas del día, y sólo a ciertas bases de datos. Es por todo lo anterior que la seguridad y control de los usuarios es de vital importancia en el mantenimiento y control de *passwords* y *logins* de acceso al sistema y por lo cual se requiere una parte importante para la actualización de esta información.

2.3.6 Bases de datos distribuidas

Ésta es la característica más importante de las bases de datos que componen el SIAE, recordando que: Una Base de Datos Distribuida es, una base de datos construida sobre una red computacional y no por el contrario en una máquina aislada. La información que constituye la base de datos esta almacenada en diferentes sitios en la red, y las aplicaciones que se ejecutan acceden a datos en distintos sitios.

Una Base de Datos Distribuida entonces es una colección de datos que pertenecen lógicamente a un sólo sistema, pero se encuentra físicamente esparcida en varios "sitios" de la red.

Un sistema de base de datos distribuidos se compone de un conjunto de sitios, conectados entre sí mediante algún tipo de red de comunicaciones, en el cual:

1. Cada sitio es un sistema de base de datos en sí mismo, pero,
2. Los sitios han convenido en trabajar juntos (si es necesario) con el fin de que un usuario de cualquier sitio pueda obtener acceso a los datos de cualquier punto de la red tal como si todos los datos estuvieran almacenados en el sitio propio del usuario.

Cada sitio tiene sus propias bases de datos "reales" locales, sus propios usuarios locales, sus propios DBMS y programas para administración de transacciones. La diferencia principal entre los sistemas de bases de datos centralizados y los distribuidos es que en los primeros, los datos residen en una sola localidad, mientras que, en los últimos, se encuentran en varias localidades. Cada localidad (plantel) del SIAE

puede procesar transacciones locales (P.E. inscripción, consulta de alumnos, etc.), es decir, aquellas que sólo acceden a datos que residen en esa localidad. Además, una localidad puede participar en la ejecución de transacciones globales (estadísticas de rendimiento por nivel bachillerato, licenciatura), es decir, aquellas que acceden a datos de varias localidades, ésta requiere comunicación entre las localidades.

La justificación del porqué se realizó esta distribución se basa en la estructura misma de la UNAM la cual está distribuida tanto administrativamente como físicamente por toda el área metropolitana, lo mismo tememos un CCH sur, que un campo 4 en Cuautitlán, esto nos obliga a dividir de la misma manera la información para que el sistema en sí pueda trabajar de forma independiente, garantizando siempre la disponibilidad de la información mínima para que puedan trabajar, pero a su vez con la posibilidad de consultar información a nivel global.

Con esta característica el SIAE cumple las siguientes reglas:

Autonomía Local.
Todos los sitios de un sistema distribuido deben ser autónomos, esto se cumple por la autonomía que cuentan los sitios de cada plantel, y/o facultad en cada sitio se cuenta con su propia base de datos autónoma en operación de tareas locales como podría ser una inscripción.

No dependencia de un sitio central.
La autonomía local implica que todos los sitios deben tratarse igual; no debe haber dependencia de un sitio central "maestro" para su operación pero sí debe de existir los mecanismos de verificación y autentificación de la información.

Operación continua.
En un sistema distribuido, lo mismo que en uno no distribuido, idealmente, nunca debería haber necesidad de apagar a propósito el sistema; en el SIAE sólo que ocurra un evento no planeado se apagan los sistemas primarios, pero sólo para dar paso a los sistemas espejo de consulta; esto garantiza que la información está siempre disponible para una consulta.

Independencia con respecto a la localización.
La idea básica de la independencia con respecto a la localización (también conocida como transparencia de localización) es simple : no debe ser necesario que los usuarios sepan dónde están almacenados físicamente los datos, sino que más bien deben poder comportarse - al menos desde un punto de vista lógico - como si todos los datos

estuvieran almacenados en su propio sitio local. En el SIAE los usuarios y cuentas de cada área cuenta con los mecanismos para ver la información sin que necesariamente éste sepa de dónde está sacando los datos.

Independencia con respecto a la fragmentación.
Un sistema maneja fragmentación de los datos si es posible dividir una relación en partes o "fragmentos" para propósitos de almacenamiento físico. La fragmentación es deseable por razones de desempeño: los datos pueden almacenarse en la localidad donde se utilizan con mayor frecuencia, de manera que la mayor parte de las operaciones sean sólo locales y se reduzca al tráfico en la red. Por ejemplo cada sitio-plantel cuenta con solo los alumnos de su plantel para fines de tareas o transacciones locales.

Existen en esencia dos clases de fragmentación, horizontal y vertical, correspondientes a las operaciones relacionales de restricción y proyección; respectivamente. En términos más generales, un fragmento puede ser cualquier subrelación arbitraria que pueda derivarse de la relación original mediante operaciones de restricción y proyección (excepto que, en el caso de la proyección es obvio que las proyecciones deben conservar la clave primaria de la relación original).

La reconstrucción de la relación original a partir de los fragmentos se hace mediante operaciones de reunión y unión apropiadas (reunión en el caso de fragmentación vertical, y la unión en casos de fragmentación horizontal).

Este punto lo cumple SIAE cuando para cada plantel se tienen sólo los planes de estudios de cada plantel.

Independencia con respecto al equipo.
En realidad, no hay mucho que decir acerca de este tema, el título lo dice todo. Las instalaciones de cómputo en el mundo real por lo regular incluyen varias máquinas diferentes -máquinas IBM, DEC, HP, UNISYS, PC etc. y existe una verdadera necesidad de poder integrar los datos en todos esos sistemas y presentar al usuario "una sola imagen del sistema". Por tanto conviene ejecutar el mismo DBMS en diferentes equipos, y además lograr que esos diferentes equipos participen como socios iguales en un sistema distribuido.

Independencia con respecto a la red.
Si el sistema ha de poder manejar múltiples sitios diferentes, con equipo distinto y diferentes sistemas operativos, resulta obvia la conveniencia de poder manejar también varias redes de comunicación distintas.

Independencia con respecto al DBMS
Bajo este título consideramos las implicaciones de relajar la suposición de homogeneidad estricta. Puede alegarse que esa suposición es quizá demasiado rígida. En realidad, no se requiere sino que los DBMS en los diferentes sitios manejen todos la misma interfaz; no necesitan ser por fuerza copias del mismo sistema.

Estas ocho reglas no son todas independientes entre sí, ni son por fuerza exhaustivas, ni tienen todas la misma importancia (diferentes usuarios darán diferentes grados de importancia a diferentes reglas en diferentes ambientes). Sin embargo, esta misma característica complica en mucho lo que se refiere a la administración de los diferentes servidores de base de datos, de los diferentes sitios ya que es responsabilidad de la SSRE de esta disponibilidad y transparencia de la forma de trabajo.

Existen ventajas del procesamiento de bases de datos distribuidas del SIAE.

1. Puede dar como resultado un mejor rendimiento que el que se obtiene por un procesamiento centralizado.
2. Los datos pueden colocarse cerca del punto de su utilización, de forma que el tiempo de comunicación sea más corto e independiente de la REDUNAM, incluso si ocurriera un fallo en la misma.
3. Los datos duplicados aumentan su confiabilidad. Cuando falla una computadora, se pueden obtener los datos extraídos de otras computadoras. Los usuarios no dependen de la disponibilidad de una sola fuente para sus datos.
4. Otra ventaja, es que los sistemas distribuidos pueden variar su tamaño de un modo más sencillo. Se pueden agregar computadoras adicionales a la red conforme aumentan el número de planteles, carreras y/o su carga de procesamiento.

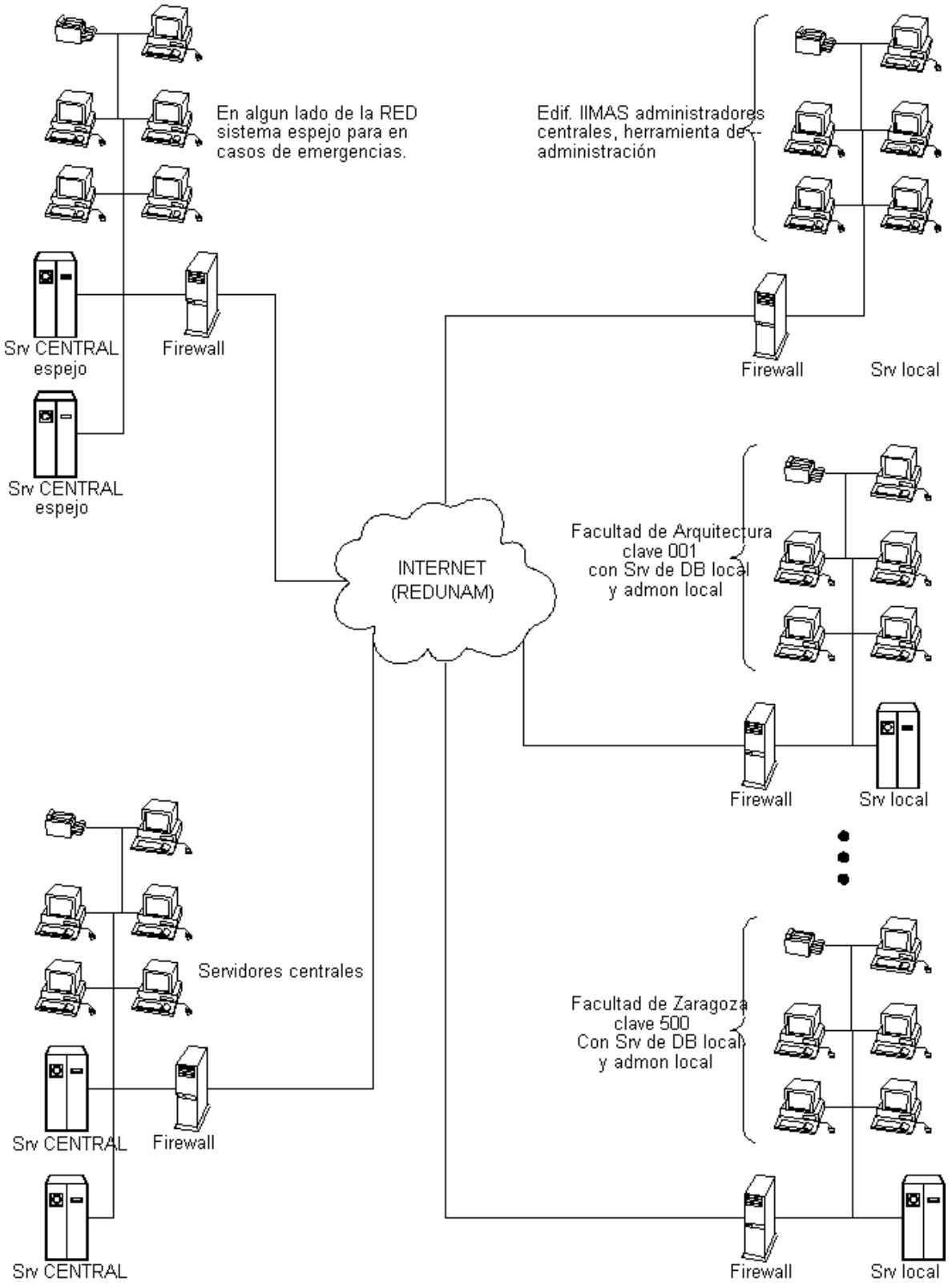


Figura 2.2 Características de las bases de datos del SIAE

2.4 Descripción del problema

Dentro de la Dirección General de Administración Escolar, a través de la Subdirección de Sistemas de Registro Escolar (SSRE), desarrolló el SIAE (Sistema Integral de Administración Escolar), para la simplificación, agilización y descentralización de los trámites académico-administrativos de los alumnos en todas las escuelas y facultades de la UNAM.

El SIAE procesa aproximadamente la inscripción de 235,000 alumnos, emite 250,000 actas de examen, 550,000 historias académicas, actualiza 3,000,000 de calificaciones anualmente, resguarda casi 1,800,000 registros de alumnos y 76,000,000 de calificaciones. Para el almacenamiento de esta información cuenta con un servidor Sun Sparc Enterprise de 4 procesadores, memoria de 1 Gb, y un disco duro de 120 Gb, además cuenta con dos servidores Sybase, donde se encuentran almacenados los registros de los alumnos; debido a la gran cantidad de información es indispensable resguardarla y mantenerla.

Para llevar a cabo estas actividades, se requiere de una correcta administración de los servidores de bases de datos, por la cantidad de información almacenada; las tareas de administración se han hecho complejas y en muchos casos repetitivas.

2.4.1 Procedimientos actuales

Los procedimientos para la administración que actualmente se realizan se pueden clasificar en cuatro módulos, los cuales se describen a continuación:

Usuarios

Podemos definir a los usuarios como toda persona que tenga cualquier tipo de contacto con el sistema de base de datos desde que éste se diseña, elabora, termina y se usa.

Una base de datos sirve a una comunidad más amplia de usuarios que los sistemas tradicionales. Los sistemas relacionales con lenguaje de consulta de cuarta generación permiten que los empleados no especializados en cómputo tengan acceso a grandes bases de datos. Además, entre los usuarios se debe incluir a los especialistas entrenados en el cómputo.

Los usuarios que acceden a una base de datos pueden clasificarse como:

-Programadores de aplicaciones.

Los profesionales en computación que interactúan con el sistema por medio de llamadas en DML (Lenguaje de Manipulación de Datos), las cuales están incorporadas en un programa escrito en un lenguaje de programación (Por ejemplo, Power Builder, C, etc.)

-Usuarios sofisticados.

Los usuarios sofisticados interactúan con el sistema sin escribir programas. En cambio escriben sus preguntas en un lenguaje de consultas de base de datos.

-Usuarios especializados.

Algunos usuarios sofisticados escriben aplicaciones de base de datos especializadas que no encajan en el marco tradicional de procesamiento de datos.

-Usuarios ingenuos.

Los usuarios no sofisticados interactúan con el sistema invocando a uno de los programas de aplicación permanentes que se han escrito anteriormente en el sistema de base de datos, podemos mencionar al usuario ingenuo como el usuario final que utiliza el sistema de base de datos sin saber nada del diseño interno del mismo, por ejemplo: un alumno.

El Administrador del servidor de bases de datos debe permitir a los usuarios el acceso a los datos. Dentro de un servidor se tiene un esquema con varias capas para permitir al usuario el acceso a los datos.

- El usuario debe tener permiso para entrar al servidor.
- El usuario debe tener permisos para acceder a una base de datos
- Finalmente, el usuario final debe tener el permiso para usar un objeto dentro de una base de datos

A continuación se muestra un esquema de las diferentes capas para el acceso al servidor sybase

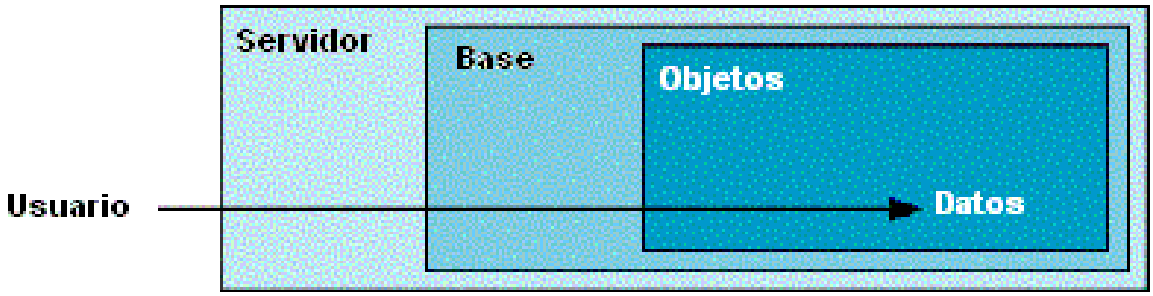


Figura 2.3 Capas del acceso al servidor Sybase

Para que las diferentes capas de acceso se cumplan, el usuario debe cumplir con lo siguiente:

- Tener un "login" válido en el servidor
- Ser un "user" válido de la base de datos
- Tener "permisos" para utilizar un objeto
- El acceso se puede restringir en cada nivel

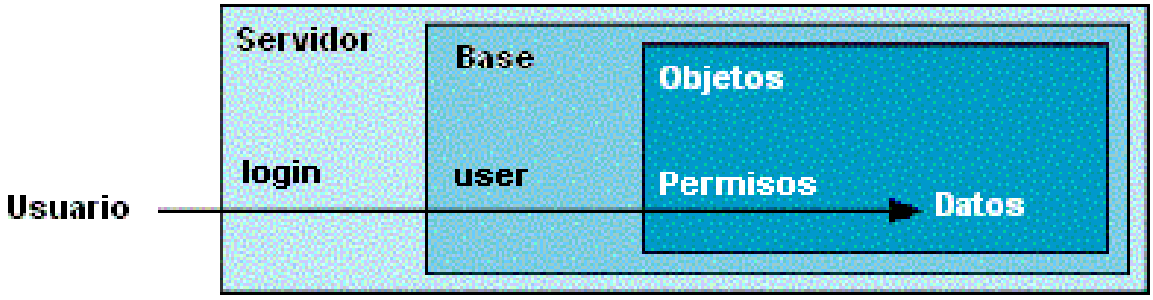


Figura 2.4 Acceso al servidor Sybase

En el SIAE se han implementado diferentes cuentas de usuario para tener control sobre el acceso a los datos ya que la información contenida en los servidores es delicada.

Con base en las funciones que realizan los usuarios, dentro de los servidores de bases de datos se crearon los siguientes grupos:

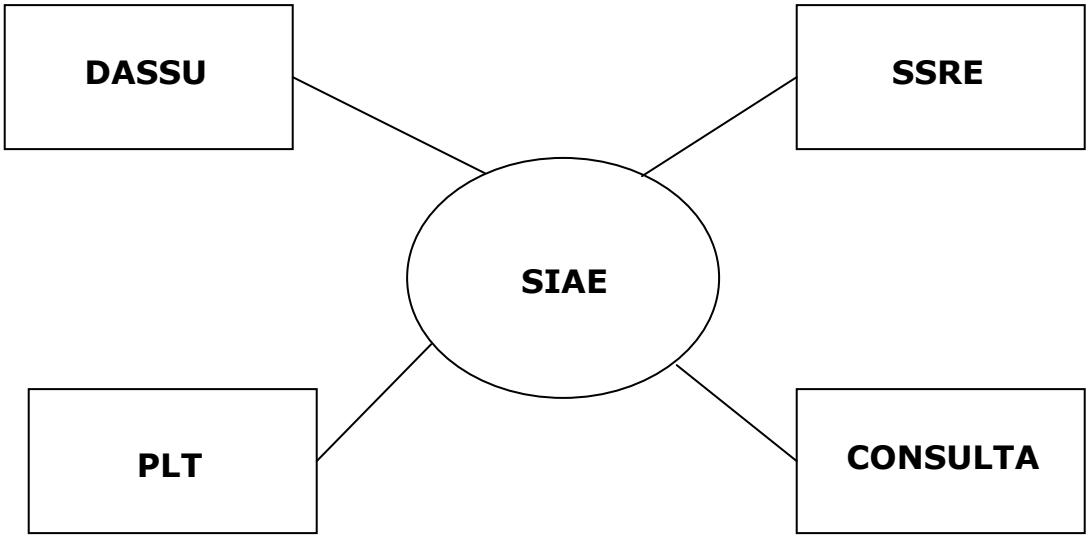


Figura 2.5 Grupos de usuarios dentro del SIAE

DASSU

Dentro de este grupo se encuentran los usuarios avanzados, en este caso los que administran el servidor de bases de datos, y el nombre esta formado por las siglas del departamento al que pertenecen cuyo nombre es Departamento de Administración de Servidores Sybase y UNIX (DASSU), ellos tienen diferentes roles que son privilegios asignados a un "login"; los roles son:

sa (System Administrator) el cual tiene a su cargo:

- Instalación del Servidor
- Asignación de recursos de disco
- Manejo del almacenamiento en disco
- Crear usuarios de bases de datos y otorgar permisos a ellos
- Privilegios de accesos, modificar, borrar y bloquear "logins"
- Crear grupos
- Auditar la integridad de la base de datos y de información

SSO (System Security Officer) el cual tiene a su cargo:

- Crear "logins" y asignar passwords
- Modificar "logins"
- Cambiar passwords
- Manejo del sistema de auditoría
- Bloquear y desbloquear "logins"

OPER (Server Operator) el cual tiene a su cargo:

- Respaldo y recuperación de bases de datos

SSRE

En este grupo se encuentran aquellos usuarios que pueden hacer modificaciones a todas las bases de datos cuyas acciones son inserción de nuevos registros, actualización y consulta de los mismos, el nombre del grupo se formo por las siglas de la Subdirección de Sistemas de Registro Escolar (SSRE), los usuarios del grupo SSRE pertenecen a esta subdirección.

Para poder realizar estos procedimientos se requiere del uso de aplicaciones, que interactúan con la base de datos.

PLT

Los usuarios que se encuentran dentro de este grupo, son aquellos que pueden realizar modificaciones a los datos de la base que pertenecen a su plantel (PLT).

Este grupo también requiere del uso de aplicaciones, que interactúan con la base de datos.

CONSULTA

Este grupo de usuarios sólo puede consultar información dentro de la base de datos.

Los procedimientos que se siguen, para dar de alta a los usuarios dentro de los servidores son:

- Generar los passwords cifrados que se usarán para tener acceso al servidor, se generan diferentes claves para cada servidor.

- Se agrega al usuario al servidor, proporcionando un "login", su nombre, la base de datos por default y la clave que se generó.

- Se agrega el "login" del usuario, como usuario (user) válido dentro de la base de datos.

- Cuando es usuario valido dentro de la base de datos, se agrega al grupo al cual va a pertenecer.

- Los permisos para el uso de objetos dentro de la base de datos ya están definidos para cada grupo, con base en las funciones del usuario.

Dentro de las políticas que se siguen en el SIAE, es que los usuarios dados de alta en los servidores, no se pueden dar de baja, porque tienen registros asociados a su login como lo son las firmas digitales, que se usan para saber los registros que modificó.

Los procedimientos que se siguen, para modificar datos y claves a los usuarios dentro de los servidores son:

- Cambio de datos personales.
- Cambio de passwords para cada servidor.
- Cambio de grupo.

Bases de Datos

Los sistemas de bases de datos se diseñan para manejar grandes cantidades de información, la manipulación de los datos involucra tanto la definición de estructuras para el almacenamiento de la información como la provisión de mecanismos para la manipulación de la información, además un sistema de base de datos debe de tener implementados mecanismos de seguridad que garanticen la integridad de la información, a pesar de caídas del sistema o intentos de accesos no autorizados.

Un objetivo principal de un sistema de base de datos es proporcionar a los usuarios finales una visión abstracta de los datos, esto se logra escondiendo ciertos detalles de como se almacenan y mantienen los datos.

Una Base de Datos es un conjunto exhaustivo, con redundancia controlada de datos estructurados, organizados independientemente de

su utilización y su implementación en máquinas accesibles y compatibles con usuarios concurrentes con necesidad de información diferente.

Componentes de la base de datos

Dentro de las bases de datos Sybase se encuentran los siguientes componentes:

Tablas. Es una unidad donde se almacena el conjunto de datos para la base de datos. Estos datos estarán ordenados en columnas verticales, a las cuales se les denomina campos.

Procedimientos Almacenados. Un procedimiento almacenado es un conjunto de sentencias SQL, que han sido almacenados en una base de datos que pueden ser ejecutados por su nombre.

Los procedimientos almacenados pueden aceptar y regresar parámetros, pueden correr más rápidamente que un archivo batch, reducen el tráfico de la red, refuerzan consistencia e integridad en la base, mejoran significativamente el desempeño de los queries de SQL.

Vistas. Una vista es una ventana con datos de una o más ventanas internas, a través de la cual se puede ver o cambiar información de una tabla muy grande. Es una tabla virtual, se ve y se usa como una tabla pero no existe, sus datos se derivan de tablas sin embargo no son copiados. Las vistas dan un mecanismo de seguridad a las bases de datos, ya que dichas vistas no contienen datos, son como un espejo de los datos.

Triggers. Los triggers son un tipo especial de procedimientos almacenados y son de gran utilidad en las tablas ya que se activan automáticamente por el SQL server al insertar, borrar o actualizar (insert, delete y update) en una tabla.

Los triggers no aceptan parámetros ni tampoco pueden ser llamados por su nombre.

Reglas. Las reglas pueden especificar una máscara para una columna o un tipo definido por un usuario.

Una regla puede ser una lista de valores, un rango o un patrón; una regla puede ser aplicada para varias columnas de una o distintas tablas.

Defaults. Los defaults especifican los datos que deben estar dentro de una columna. Una columna puede tener un solo default.

Las bases de datos se pueden crear, borrar y ampliar, en el SIAE se implementaron esquemas de trabajo con bases de datos y son las siguientes:

Creación de una base de datos

En el SIAE se crearon las bases de datos para cada uno de los planteles, estas bases tienen la misma estructura, para la creación se tomó en cuenta lo siguiente:

- El tamaño que debe de tener la base de datos
- La localización y el espacio necesario en un dispositivo
- Separación de datos y bitácoras (logs)

Una vez creada la base de datos el dueño de ésta es el administrador el cual debe de otorgar los permisos necesarios para su uso, además debe de crear los componentes dentro de la base de datos.

El procedimiento que se sigue para la creación de los componentes es:

- Crear todas las tablas que contendrán los registros de los alumnos.
- Crear todas las reglas y vistas para las tablas
- Crear los procedimientos almacenados.
- Crear los grupos de trabajo para los usuarios.
- Conceder los permisos sobre los componentes a los grupos.

Modificación de una base de datos

Las bases de datos del SIAE están en constate crecimiento es por ello que se les debe dar mantenimiento, pues semestre con semestre el número de registros crece, de tal forma que el espacio en el dispositivo resulta insuficiente y es necesario crecer la base de datos en tamaño. Se lleva un control de todas las bases de datos y cómo se encuentran distribuidas en los dispositivos.

Cada base de datos en el servidor pertenece a un plantel, cuando esta base de datos requiere ubicarse en un dispositivo con mayor tamaño se realiza manualmente lo siguiente:

- Respaldar la base de datos original
- Recrear la base de datos en el dispositivo con mayor espacio

- Recrear los grupos de trabajo
- Recrear la estructura de tablas
- Recrear los procedimientos almacenados, las reglas y las vistas para las tablas.
- Copiar los registros a la nueva base
- Borrar la base de datos original
- Renombrar la nueva base con el nombre de la original
- Otorgar los permisos sobre los componentes a los grupos

Cuando se requiere crear un nuevo componente o modificar uno existente se realiza lo siguiente:

- Crear el componente o modificarlo
- Otorgar permisos sobre el componente

Respaldos de las bases de datos

Uno de los desafíos más grandes y difíciles de manejar para los administradores de sistemas a la hora de respaldar o recuperar datos, es precisamente cómo crear ambientes de respaldo confiables y seguros y que dispongan de mecanismos para la rápida ejecución de tareas y eventos en casos de pérdida de datos .

Cuando una empresa se decide a utilizar un sistema de base de datos, se vuelve dependiente en grado sumo del funcionamiento correcto de ese sistema. En caso de que sufra daño cualquier porción de la base de datos – por causa de un error humano, digamos, o una falla en el equipo o en el sistema que lo apoya – resulta esencial poder reparar los datos implicados con un mínimo de retraso y afectando lo menos posible el resto del sistema. En teoría, por ejemplo la disponibilidad de los datos no dañados no debería verse afectada. El DBA debe definir y poner en práctica un plan de recuperación adecuado que incluya, por ejemplo una descarga o "vaciado" periódico de la base de datos en un medio de almacenamiento de respaldo, y procedimientos para cargar otra vez la base de datos a partir del vaciado más reciente cuando sea necesario.

Existen diversos métodos de respaldos, y cada uno debe ajustarse a las necesidades particulares de cada organización, de tal forma que los analistas de sistemas o los administradores de información, deben identificar el mecanismo de respaldo que responda a las necesidades reales de los usuarios.

Algunos aspectos básicos que deben tomarse en consideración a la hora de respaldar son los siguientes:

1. Debe asignarse a una persona, debidamente capacitada, quien será el responsable de realizar periódicamente los respaldos y el momento oportuno para efectuar los respaldos, situación que implica el análisis del rol de actividades de producción.

2. Establecer un procedimiento por escrito, que detalle claramente la secuencia de pasos que han de ejecutarse para realizar el respaldo y su restauración.

3. Disponer de un lugar adecuado para guardar las copias de respaldo, por ejemplo: una caja fuerte, un lugar geográfico diferente y alejado del centro de cómputo, entre otros.

4. Por último, debe disponerse de todos los recursos tecnológicos para soportar el mecanismo de respaldo, esto es: hardware, software y comunicaciones.

Copias de Información (Backups)

Estos respaldos son sólo duplicados de archivos que se guardan en "Tape Drives" de alta capacidad (24 GB aprox). Los dos tipos de respaldos comúnmente utilizados son:

Respaldo Completo ("Full"). Guarda todos los archivos que sean especificados al tiempo de ejecutarse el respaldo. Como es el caso de respaldar una base de datos completa.

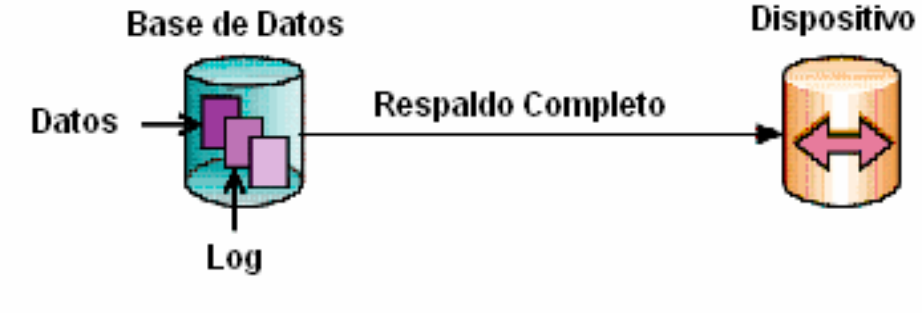


Figura 2.6 Respaldo completo de una base de datos

Respaldo de Incremento ("Incremental"). Cuando se lleva acabo un Respaldo de Incremento, solo se respalda el log (bitácora) de la base de datos.

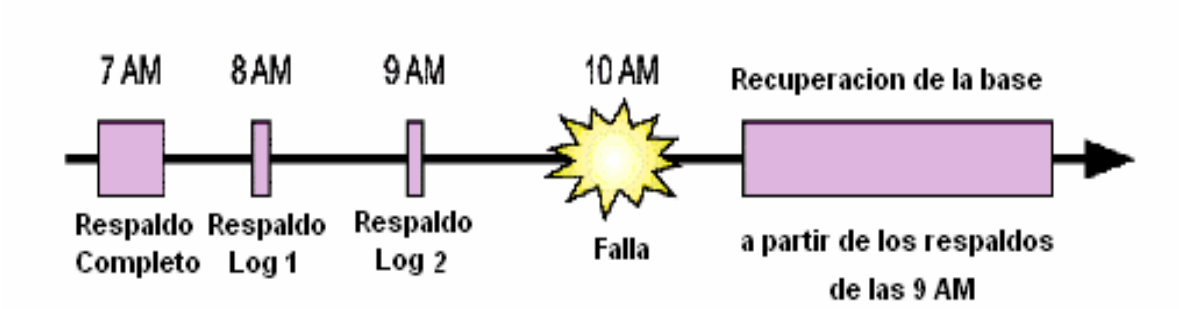


Figura 2.7 Respaldo incremental y recuperación de una base de datos

El proceso de respaldo que se sigue en el SIAE es:

- Seleccionar la o las bases de datos a respaldar.
- Elegir el tipo de respaldo (completo o incremental).
- Elegir si el respaldo se almacenara en cinta o archivo.
- Guardar el registro del día, hora y minuto en que se realiza el respaldo.
- Ejecutar el respaldo.
- Verificar que el respaldo se haya hecho correctamente, si ocurrió un error se vuelve a procesar.

Los respaldos completos se realizan cada 15 días, los respaldos incrementales se realizan diariamente para todos los planteles, si el respaldo se hace en una cinta, ésta tiene un número secuencial el cual permite tener el control de los respaldos de las bases, esta cinta se almacenada por un periodo de un año aproximadamente.

Si el respaldo se hace en forma de archivo éste se almacena en una maquina UNIX, posteriormente se hace una copia de dicho respaldo a un servidor de alta seguridad.

Auditorías

Se deben realizar periódicamente, y según un calendario y un procedimiento predeterminado, auditorías internas de las actividades de los servidores para comprobar que siguen cumpliendo los requisitos del sistema.

El programa de auditorías debe cubrir todos los elementos del sistema. Las auditorías se deben realizar por el administrador del sistema.

Si los resultados de la auditoría ponen en duda la validez de los resultados de los registros auditados, el administrador debe adoptar inmediatamente las acciones correctivas oportunas.

El administrador debe mantener un registro de las bases de datos auditadas, de los resultados de la auditoría y de las acciones correctivas que se hayan derivado de la misma.

El administrador debe documentar en procedimientos sus políticas, sistemas, programas, e instrucciones en la medida en que sea necesario.

En el SIAE se han realizado algunos procedimientos que permiten realizar la auditoría de las bases de datos, para verificar la integridad de los registros.

Estos procedimientos están semiautomatizados y se requiere de la supervisión manual de ellos. Los procedimientos que se realizan son verificar los registros de los alumnos uno a uno para validar que su historia académica se encuentra correctamente. Además se verifica que las calificaciones asentadas en el acta sean las correspondientes a las calificaciones contenidas en la base de datos.

Por otra parte se realiza la auditoría de las actas de los planteles, de las asignaturas así como su seriación, además se realiza la auditoría de los planes de estudio para verificar diferencias entre ellos si es que existen.

2.4.2 Necesidades actuales

Actualmente en el SIAE los procedimientos antes mencionados se llevan a cabo de manera eficiente pero éstos requieren de mucho tiempo para realizarlas correctamente, además el personal requiere de experiencia y un nivel alto de capacitación.

Por lo tanto se necesitan automatizar los procedimientos actuales de tal forma que para el usuario el manejo del sistema sea de forma rápida e intuitiva.

Al automatizar los procedimientos se creará un conjunto de técnicas con el fin de reducir la intervención del usuario en la administración del servidor de bases de datos así como su funcionamiento.

El diseño del sistema será de forma gráfica en la que se contemplan los siguientes módulos

Módulo de usuarios

Dentro de este módulo se realizaran los siguientes procedimientos:

Creación de un usuario

- Se debe proporcionar un "login" y un password.
- El password se debe de encriptar generando un password diferente para cada uno de los servidores.
- El login se debe agregar al servidor.
- El login debe ser usuario de cada una de las bases

Dependiendo del tipo de usuario, éste tendrá los permisos necesarios en la base de datos del grupo que se le asigne. Como se mencionó anteriormente existen cuatro grupos cada uno de los cuales tienen permisos asignados para la consulta o modificación de datos dentro del servidor.

Para la creación de un nuevo usuario se inserta un registro en las tablas internas del sistema de la siguiente forma:

- Insertar el registro del "login" en la tabla syslogins
- Insertar el registro del usuario en la tabla sysusers

La siguiente gráfica muestra las tablas de sistema del servidor

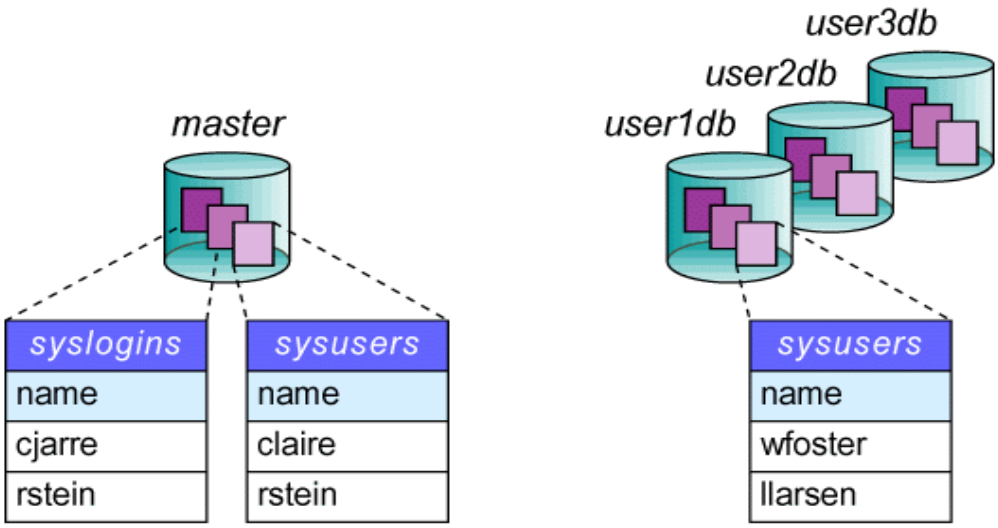


Figura 2.8 Tablas internas del servidor Sybase

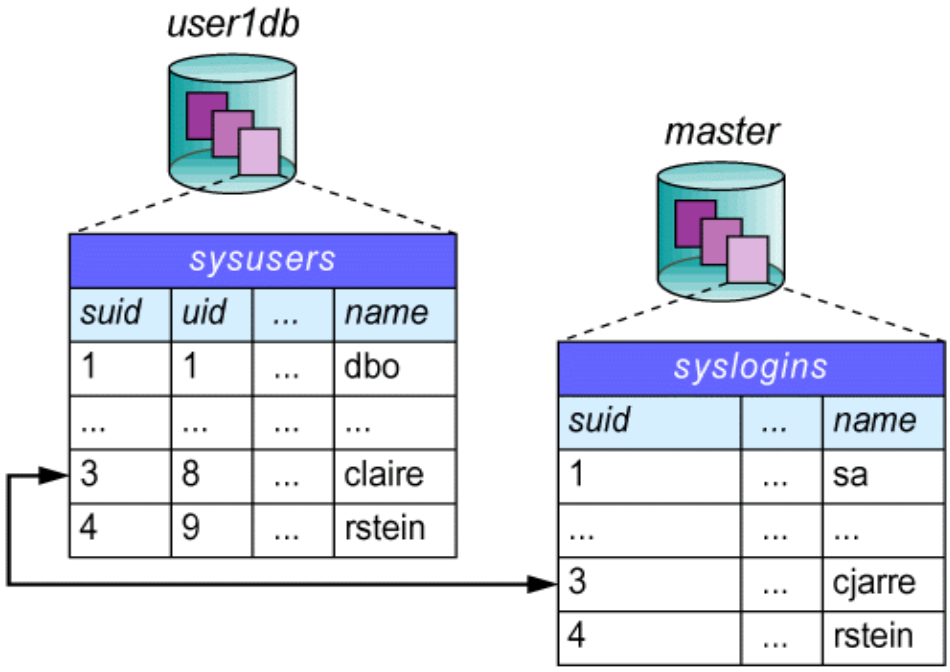


Figura 2.9 Tablas de sistema relacionado a los usuarios de la base de datos

Modificación de un usuario

La modificación al usuario se refiere al cambio de su password, ya que en el SIAE se implementó la política de cambio de password mensualmente, debido a la gran cantidad de usuarios del servidor. Esta tarea sea ha hecho compleja, además se puede modificar el grupo al que pertenece.

Dentro de las tablas internas del servidor Sybase se actualiza el password de la tabla syslogins, y se realiza un encriptamiento interno.

Las opciones que tendrá la modificación de un usuario son las siguientes:

- Modificar el password de un usuario
  - ❖ Obtener el nuevo password
  - ❖ Generar los passwords encriptados para cada servidor
  - ❖ Realizar la modificación de todos los passwords en todos los servidores para tener una sincronización.
- Modificar el password de todos los usuarios
  - ❖ Obtener la lista de los nuevos passwords
  - ❖ Generar todos los passwords de los usuarios para cada servidor

  - ❖ Realizar la modificación de todos los passwords de todos los usuarios en todos los servidores.
- Modificar el grupo al que pertenece el usuario

Otra política que se ha implementado en el SIAE es que los logins que se crean en el servidor no se pueden borrar, porque se guardan los registros de los movimientos que han realizado, de tal forma que se procede a realizar el bloqueo del usuario para que no tenga acceso al servidor.

Módulo Base de datos

Para este módulo se requiere de los siguientes pasos:

-Creación de una base de datos

Cuando se crea una base de datos el servidor internamente inserta registros en las siguientes tablas de sistema:

- sysdatabases
- sysusages
- sysdevices

Las cuales contienen información acerca del nombre de la base, el tamaño y los dispositivos con los que cuenta el servidor.

Los procedimientos que se deben de seguir para la creación de una base de datos son:

- Asignar un nombre a la base de datos nueva.
- Asignar el tamaño de los datos y del log.
- Elegir entre los diferentes dispositivos con los que se cuenta.
- Otorgar los permisos para que los usuarios puedan utilizar la base de datos.
- Crear todos los componentes que contendrá la base de datos
  - ❖ Crear tablas
  - ❖ Crear reglas y vistas para las tablas
  - ❖ Crear los procedimientos almacenados
  - ❖ Crear los grupos de trabajo
  - ❖ Conceder los permisos sobre los componentes a los grupos

-Modificación de una base de datos

Los procedimientos a seguir para la modificación de una base de datos son:

- Respaldar la base de datos original
- Recrear la base de datos con un estructura definida, dependiendo de cada plantel
- Recrear los grupos de trabajo
- Recrear la tablas de usuario
- Recrear los procedimientos almacenados, las reglas y las vistas para las tablas.
- Copiar los registros a la nueva base
- Borrar la base de datos original
- Renombrar la nueva base con el nombre de la original
- Otorgar los permisos sobre los componentes a los grupos

Una opción que estará disponible es la de crear nuevos componentes o modificar los existentes para la base de datos, además de otorgar los permisos necesarios a los grupos.

Los pasos antes mencionados se realizarán de forma automática.

Módulo respaldos de las bases de datos

Para este módulo se requiere de automatizar los respaldos de tal forma que exista la posibilidad de que todos los días a determinada hora se realice el respaldo de log de transacciones. Debe de contar con la opción de cambio de hora y debe trasladar los respaldos a los servidores alternos, además de que se deben actualizar los datos en dichos servidores.

Los respaldos de las bases de datos se almacenan en archivos y los procedimientos a seguir son:

- Especificar la hora en la que se realizaran los respaldos, se realizara todos los días.
- Seleccionar las bases de datos a respaldar.
- Elegir tipo de respaldo log de transacciones o base de datos completa.
- Elegir cómo se realizará el respaldo en cinta o archivo
- Enviar los archivos respaldados a los servidores alternos
- Guardar el registro del día, hora y minuto en que se realiza el respaldo.

- Actualizar en los servidores alternos las bases de datos.
- Verificar que los respaldos se hayan hecho correctamente y la actualización de las bases de datos de los servidores alternas no hayan tenido errores.

Módulo de auditorías

En el módulo de auditorías se llevarán a cabo los siguientes procedimientos:

Realizar auditorías diarias de todos los usuarios que accedan al servidor, horarios de acceso, tipo de procesos realizados (consultas, actualizaciones, inserciones, o borrados de datos), accesos exitosos y fallidos.

Realizar auditorías programadas de tablas de las bases de datos que requieran un análisis detallado de los registros que contienen, como por ejemplo, auditar las historias académicas de los alumnos, las calificaciones y su respectiva correspondencia con las actas en que se encuentran asentadas, auditar asignaturas y seriaciones así como planes de estudio.

Realizar las auditorías de los *logs* del sistema para saber los movimientos que se han hecho y si existen problemas, para tomar una decisión correcta al momento de solucionarlos.

Realizar el monitoreo del espacio de todas las bases de datos que se encuentran dentro del servidor para saber cuándo se requiere aumentar de tamaño a una base de datos determinada.

Realizar el monitoreo de la tabla interna del sistema llamada *sysaudits*, para conocer el tipo de *query* que se está ejecutando, de esta forma podemos saber si el usuario realizó una inserción, actualización o consulta.

2.4.3 Resultados esperados

Con base en los procedimientos y necesidades actuales que se describieron anteriormente, los resultados que se esperan obtener son:

Verificar que toda la información que se encuentra almacenada en los servidores de bases de datos, sea la correcta y que todos los procedimientos que se siguen para su explotación, cumplan con las políticas establecidas para el SIAE. Pues la información almacenada es delicada ya que se tiene el registro de todas las historias académicas de la comunidad universitaria.

Mejorar las labores de la administración de los servidores de bases de datos, creando un ambiente gráfico de fácil manejo e intuitivo para optimizar su funcionamiento.

Reducir los tiempos y procedimientos que se emplean para la correcta administración de la información contenida en los servidores, usuarios, dispositivos, auditorías y respaldos.

Agrupar todas las herramientas que existen en el SIAE de tal forma que cumplan con los estándares establecidos dentro de ésta, para que la administración de los servidores sea sencilla.

Crear nuevas aplicaciones para complementar las tareas de administración.

Con este sistema se pretenden cubrir las necesidades actuales y a futuro para lograr que la administración escolar cumpla con sus objetivos.

3. ANÁLISIS Y DISEÑO DEL SISTEMA DE ADMINISTRACIÓN DE BASES DE DATOS

3.1 Análisis del sistema

Como se explicó en la sección 2.3 Características de las bases de datos del SIAE, la principal característica es que cuenta con una estructura de bases de datos distribuidas. Esta estructura tiene ventajas y desventajas como ya se mencionó, la principal desventaja, es que la administración de los recursos y las tareas inherentes se complica en mucho; para el mejor entendimiento de estas complicaciones se hace una relación de las tareas y se trata de explicar las complicaciones que surgen de éstas.

En la actualidad la administración de las bases de datos del SIAE se compone de tareas repetitivas y de gran complejidad, por el gran número de pasos que llevan y porque requieren un orden específico, lo que implica que la persona que realiza estas tareas tenga un nivel técnico especializado muy alto.

Estas tareas, que implican a su vez subtareas, dependiendo de las características y condiciones en las que la tarea en sí es ejecutada puede implicar o no, más o menos pasos, esto hay que agregarle, a que son tareas, por lo general, calendarizadas o que se repiten con cierto periodo, y éstas requieren un cierto tiempo, que en los mejores casos son horas de trabajo, horas que bien podríamos dedicarlo a tareas de investigación o de desarrollo.

Es por todo lo anterior que se propone la siguiente solución para ayudar a agilizar el problema que es realizar tareas simples pero que implican inversión de tiempo y personal.

El Sistema de Administración de base de datos del SIAE (SABAD-SIAE) deberá de contemplar las siguientes tareas:

Administrar cuentas (*logins*) y claves de acceso (passwords) de los usuarios que ingresan a los diferentes servidores de bases de datos.

Administrar y controlar las tareas de auditoría e integridad de la información en los diferentes servidores de bases de datos en todas las bases que contengan al SIAE.

Controlar y ejecutar los respaldos de los diferentes sitios y las particularidades de cada servidor de bases de datos dependiendo de las mismas características del sitio (plantel), incluyendo en esta tarea los respaldos que se realizan en los servidores principales (centrales).

Crear, configurar y darle afinación a cada base de datos en cada servidor, además de poder Administrar los recursos asignados a los diferentes servidores, como es el espacio y la memoria asignada a cada base de datos y cada servidor.

Estas tareas deben tomar en cuenta estas limitantes:

1. *Administración de cuentas*

   1.1 Un usuario de plantel no puede tener más de una cuenta de acceso al servidor de bases de datos
   1.2 Los usuarios de plantel sólo pueden pertenecer al grupo "PLT" y estarán sujetos a los permisos que se le asigne a su grupo de trabajo.
   1.3 El usuario de plantel solo podrá conectarse a su servidor local y a sus bases locales y con las correspondientes medidas a los servidores centrales.
   1.4 Nunca una cuenta de usuario desaparece, sólo se inactiva o se bloquea.
   1.5 Un usuario de la SSRE puede tener más de una cuenta pero con passwords y cuentas diferentes.
   1.6 Dependiendo de la solicitud se deberá especificar a qué bases de datos tendrá acceso esta cuenta, así como los permisos que en cada una de ellas tendrá.
   1.7 Un usuario de la SSRE debe pertenecer al grupo *"SSRE"* y se le puede dar permisos adicionales a los objetos de una base de datos, siempre y cuando sea solicitado explícitamente y con permiso del responsable del área.
   1.8 Las cuentas especiales serán contempladas y será responsabilidad del DBA decidir el password y la dificultad del mismo, así como el mecanismo de seguridad que se decida para esta cuenta.
   1.9 El sistema deberá cambiar el password de cualquier cuenta y en cualquier momento tomando a consideración las características de la misma cuenta.
   1.10 El sistema también deberá cambiar el password a grupos de usuarios tomando en consideración las características de cada una de las cuentas que componen ese grupo.
   1.11 La sección en el programa que maneje estas tareas estará protegido por una clave de acceso adicional que sea cambiante y es responsabilidad del DBA implementar el mecanismo que sea más conveniente para que cumpla esta característica.
   1.12 Una cuenta de la SSRE deberá ser dada de alta a los diferentes servidores de bases de datos y con el password correspondiente

para cada servidor, implementando el mecanismo correspondiente.

1.13 El sistema deberá cambiar el password de una cuenta cuantas veces sea necesario dependiendo de las características de la misma cuenta, (pertenece a plantel o a la SSRE)

1.14 Los passwords de las cuentas deberán ser siempre de 8 caracteres exactos.

1.15 Los passwords del grupo de trabajo *"SSRE"* deberán contener un carácter especial en cualquiera de las 8 posiciones que componen el password.

1.16 Dependiendo del grupo y las tareas que realice la cuenta, se deberán asignar los mecanismos que permitan garantizar la veracidad de la información que esta cuenta genere.

2. *Administración de auditoría e integridad de la información*

2.1 La auditoría deberá estar dividida entre lo que serían tareas de auditoría y lo que serían tareas de verificación de integridad de la información

2.2 Los mecanismos de auditoría de los servidores de bases de datos deberán ser instaladas y configuradas en los mismos términos de los diferentes servidores, para que registren lo mismo en cada servidor.

2.3 La auditoría siempre deberá estar activada por los diferentes servidores de base de datos, por lo que el sistema deberá contemplar las opciones del encendido y apagado de la misma. Todos los *logins* del servidor se auditan, cada movimiento que se realiza se registra en la tabla de auditoría, de tal forma que un *login* puede realizar muchos movimientos, cuando se excede la capacidad de almacenamiento de la tabla, el servidor queda bloqueado y es por esta razón que se debe de apagar la auditoría para así solucionar el problema.

2.4 La auditoría deberá contemplar generar reportes de diferentes tipos.

2.5 Será responsabilidad del DBA programar la frecuencia y periodicidad de la auditoría.

2.6 La integridad se refiere a la verificación de la veracidad y consistencia de la información por lo cual deberá contener mecanismos que verifiquen y garanticen este objetivo.

2.7 La integridad de la información hará referencia a patrones que deberán poder ser configurados.

2.8 La veracidad de la información deberá ser sobre objetos (tablas alu, rsa, hsa, try, etc.) que contengan información de vital importancia.

2.9 El anterior punto se hará sobre tablas que contengan el campo que permita guardar esta información.
2.10 Cuando la tarea de integridad implique comparaciones entre servidores, éstos deberán permitir el acceso a esta tarea.

3. *Administración de respaldos*

3.1 Los respaldos son de diferentes tipos y con diferentes características.
3.2 Se pueden realizar respaldos completos o solo de *log*.
3.3 Si son respaldos completos siempre son a cinta.
3.4 Si son respaldos de *log* siempre son hacia archivos del sistema operativo.
3.5 Los respaldos completos de los diferentes planteles pueden ser hacia archivos o hacia una cinta que controla el administrador local.
3.6 Los respaldos que se realicen hacia archivos deben ser compactados con las utilerías que nos proporciona el sistema operativo garantizando la compatibilidad de las diferentes versiones de UNIX.
3.7 Los respaldos ya compactados se deberán transferirse al servidor de seguridad fuera de la máquina en la cual se generaron.
3.8 Los archivos de respaldo deberán respetar la notación de los nombres de archivos de respaldo que se implemente y en el formato que se indique.
3.9 El control del respaldo deberá estar contenida dentro del servidor de bases de datos.

4. *Crear y configurar las bases de datos*

4.1 Deberá poder configurar el tamaño de la base de datos tanto la parte de datos y la parte de *log*, esto con reglas preestablecidas dependiendo del sitio (plantel) y la actividad que se tenga en la misma.
4.2 Se podrá configurar los objetos (tablas, reglas, vistas, etc.) que contendrá esta base de datos por medio de un *script* general para cada tipo de base.
4.3 Deberá poder modificar, configurar las características particulares de cada base de datos, como podría ser: read only, select into, dbo only etc.
4.4 Podrá configurar las características por objeto, como podría ser particiones o asignación a algún cache de memoria o algún segmento.

4.5 Se podrá configurar y programar aplicaciones sobre límites de espacio (thresh hold)
4.6 Deberá poder verificar la consistencia de la base de datos así como la integridad a nivel página de la información.
4.7 Se podrá crear bases de datos con características en tamaño de datos y tamaño de *log* acorde a las necesidades de cada sitio-plantel.
4.8 Tendrá la opción de crear los objetos (tablas, reglas, índices, vistas, etc.) necesarios para que funcione adecuadamente el SIAE.
4.9 Podrá actualizar y crear procedimientos particulares requeridos para el buen funcionamiento del SIAE.

Estas tareas dependiendo de cada opción y limitante deberán poder general los siguientes reportes.

1. *Administración de cuentas*

1.1 Cuando se dé de alta un usuario generar el reporte de cúal fue el resultado del alta y si se llevo a cabo en cada uno de los servidores que involucra la cuenta o solicitud.
1.2 Cuando se cambie el password a un usuario o grupo de usuarios se generará un reporte para poder llevar un registro de este cambio.
1.3 Se reportará también las características de la cuenta, permisos, privilegios o roles asignados y su status.

2. *Administración de la auditoría e integridad de la información*

2.1 Para esta opción ver la sección de reportes del capítulo 4.

3. *Administración de respaldos*

3.1 Deberá reportar el tipo y qué base de datos ven el respaldo.
3.2 Deberá generar reporte de la ejecución de respaldos conteniendo el nombre del archivo con el cual dejó el respaldo y en dónde lo dejó, para el caso de archivos.
3.3 Para el caso de respaldos completos deberá contener el nombre o etiqueta de la cinta así como que número de la misma.
3.4 Reportará cuando la compactación se haya terminado cuál es el nombre del compactado que está dejando, dónde lo está dejando y qué porcentaje de compactación genera esta tarea.

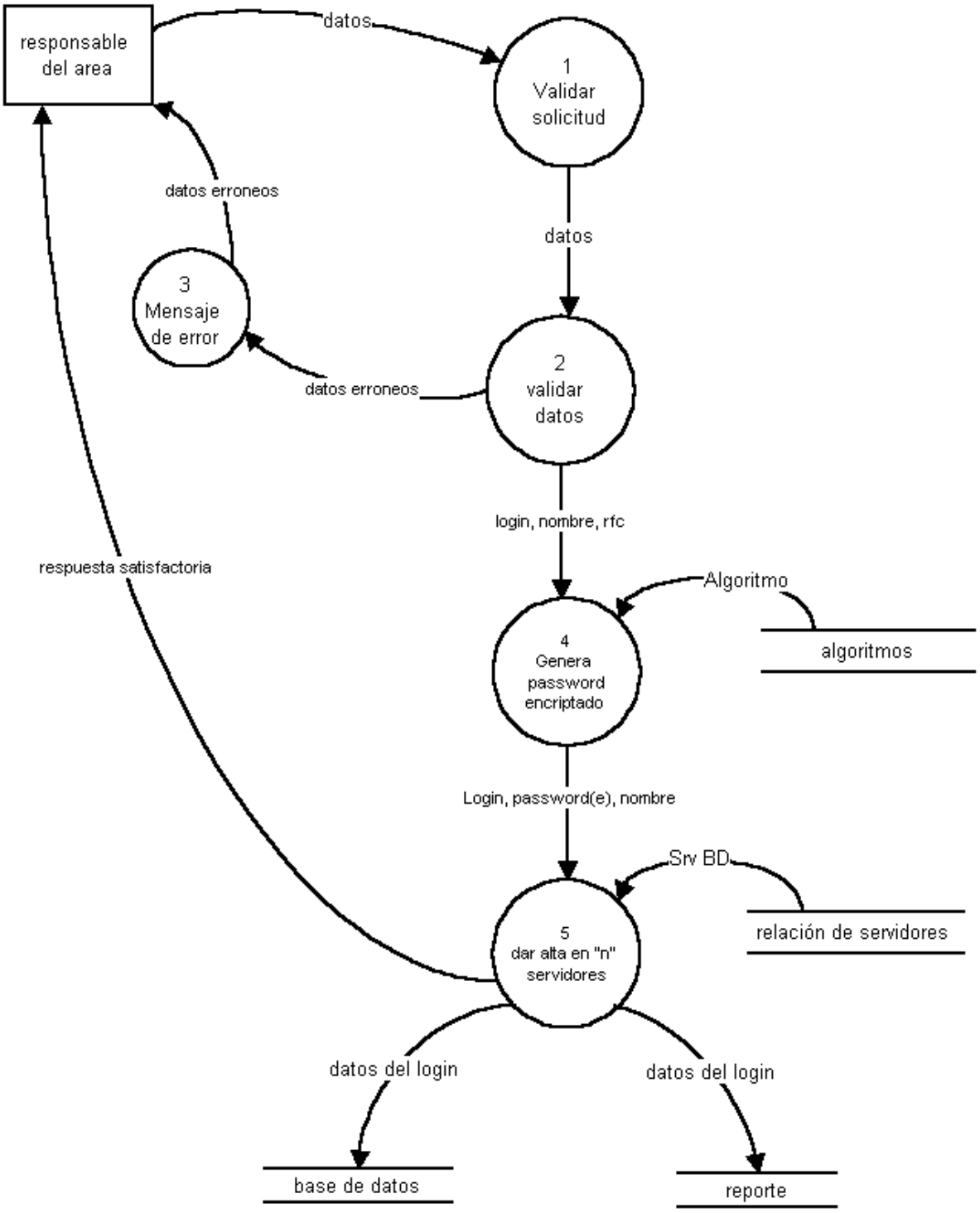
3.5 Cuando se realice la tarea de transferencia hacia el servidor seguro, deberá contener hora de inicio y fin de la transferencia así como también el tiempo de duración de la misma.

4. *Crear y configurar las bases de datos*

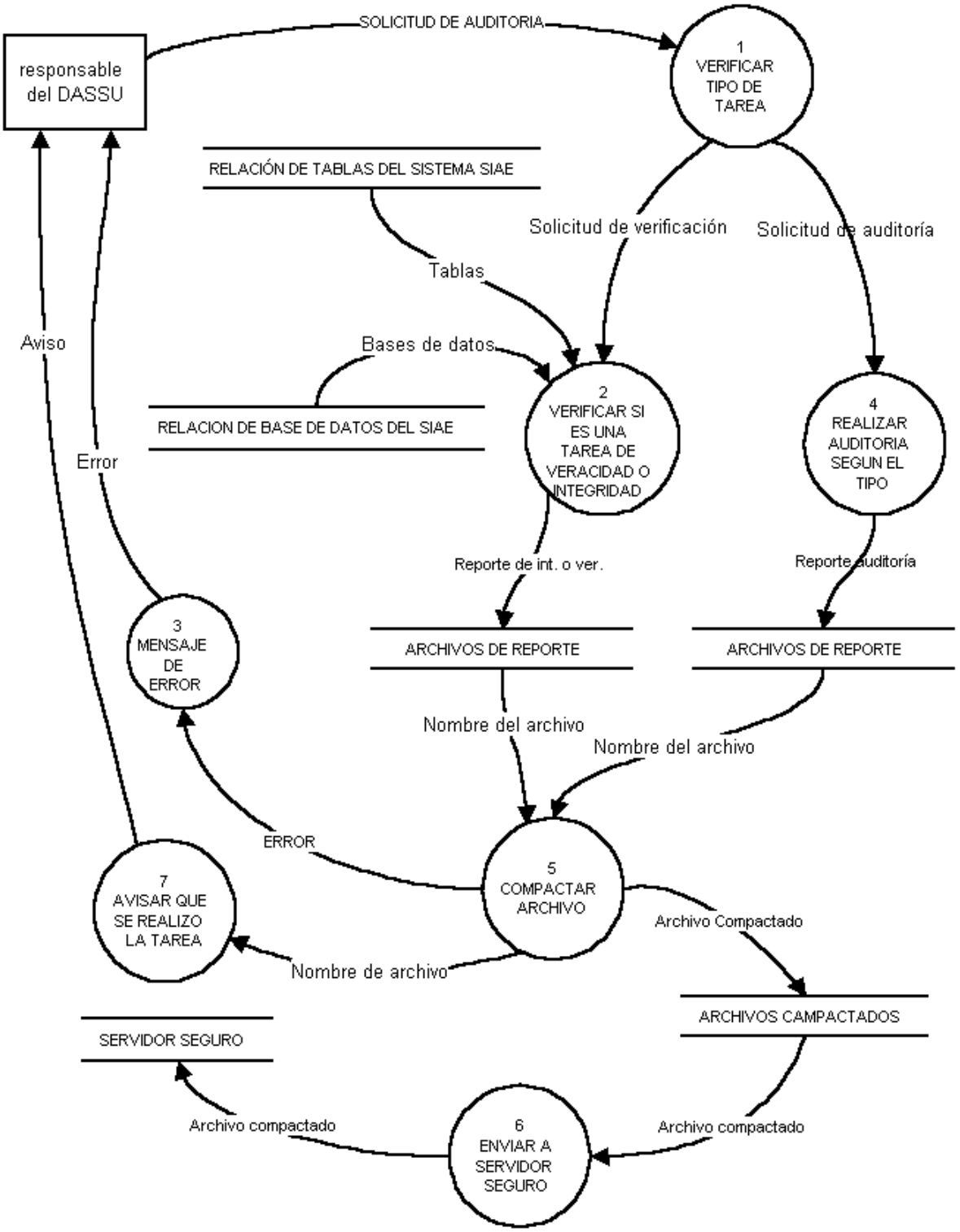
4.1 Al momento de crear la base de datos deberá de reportar las características en tamaño (tanto en datos como en *log*) y parámetros configurados, con los que fue creada.
4.2 Reporte de usuarios y tipos de permisos dependiendo del grupo al que pertenecen.
4.3 Dispositivos en los cuales esta contenida la base de datos.
4.4 Al momento de configurar la base de datos deberá reportar los parámetros modificados y cual era el valor antes y después de realizar el cambio.

A continuación se muestran los diagramas de flujo de datos respectivos a cada módulo:

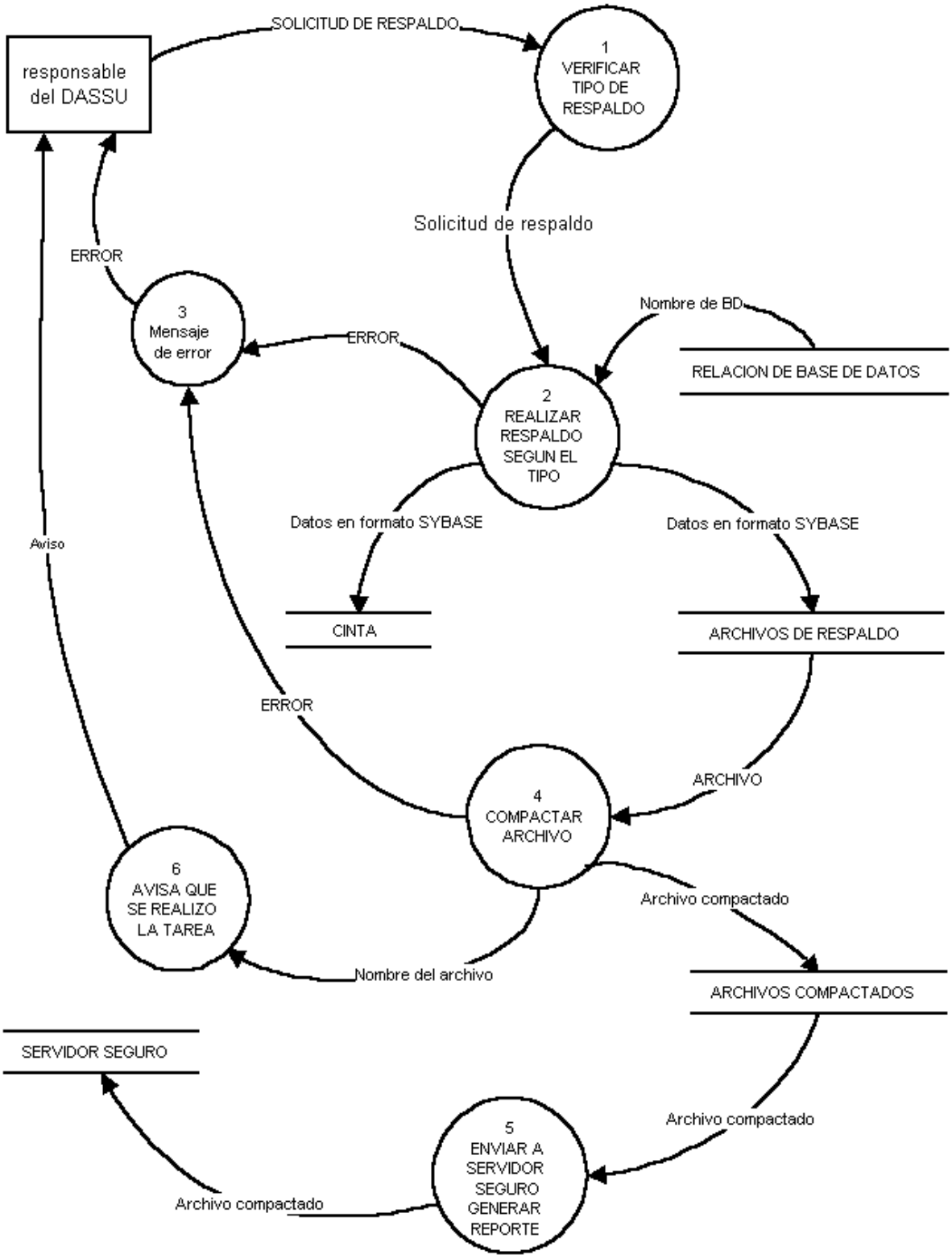
Administración de cuentas (A)



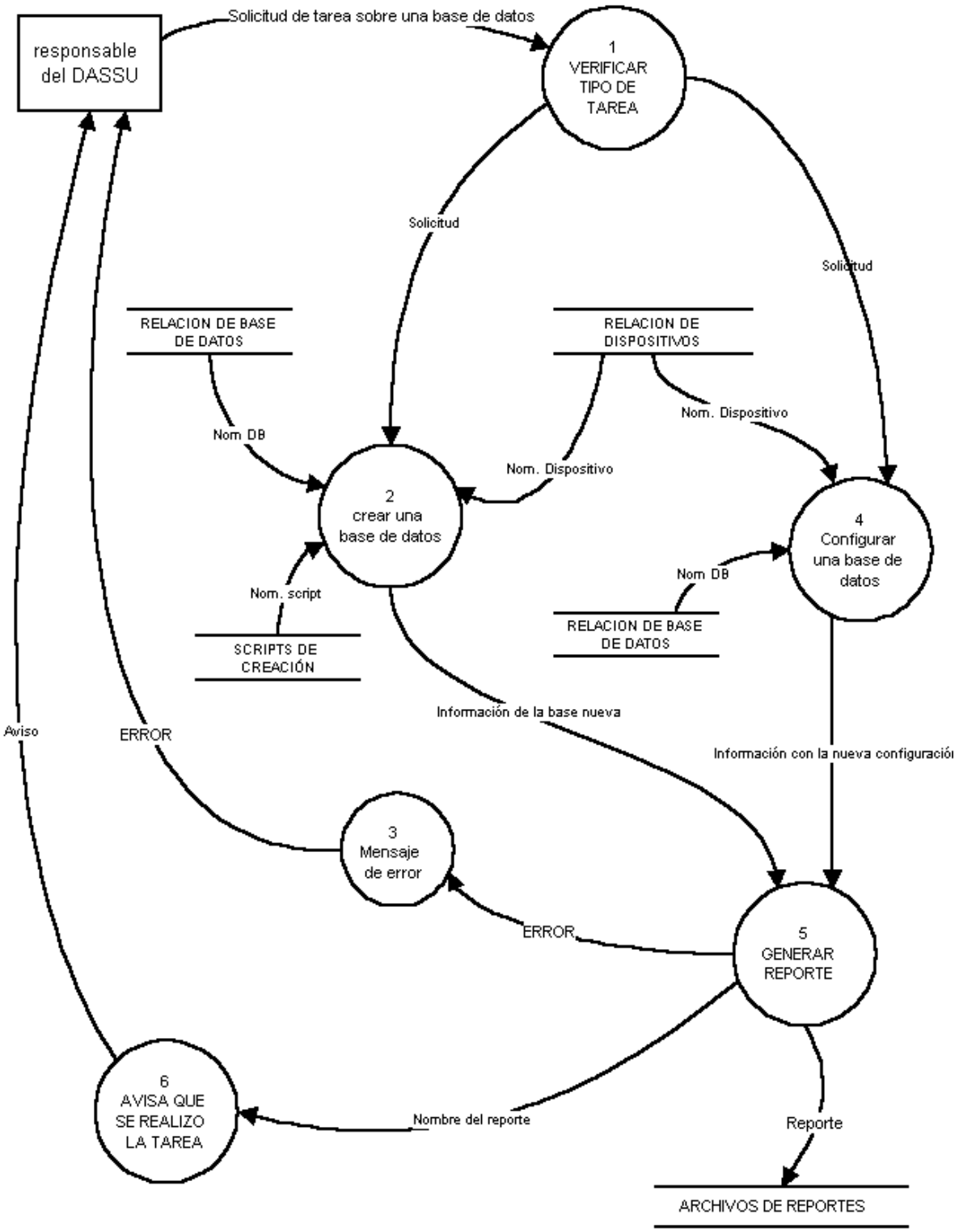
Administración de auditoría e integridad de la información (B)



Administración de respaldos (C)



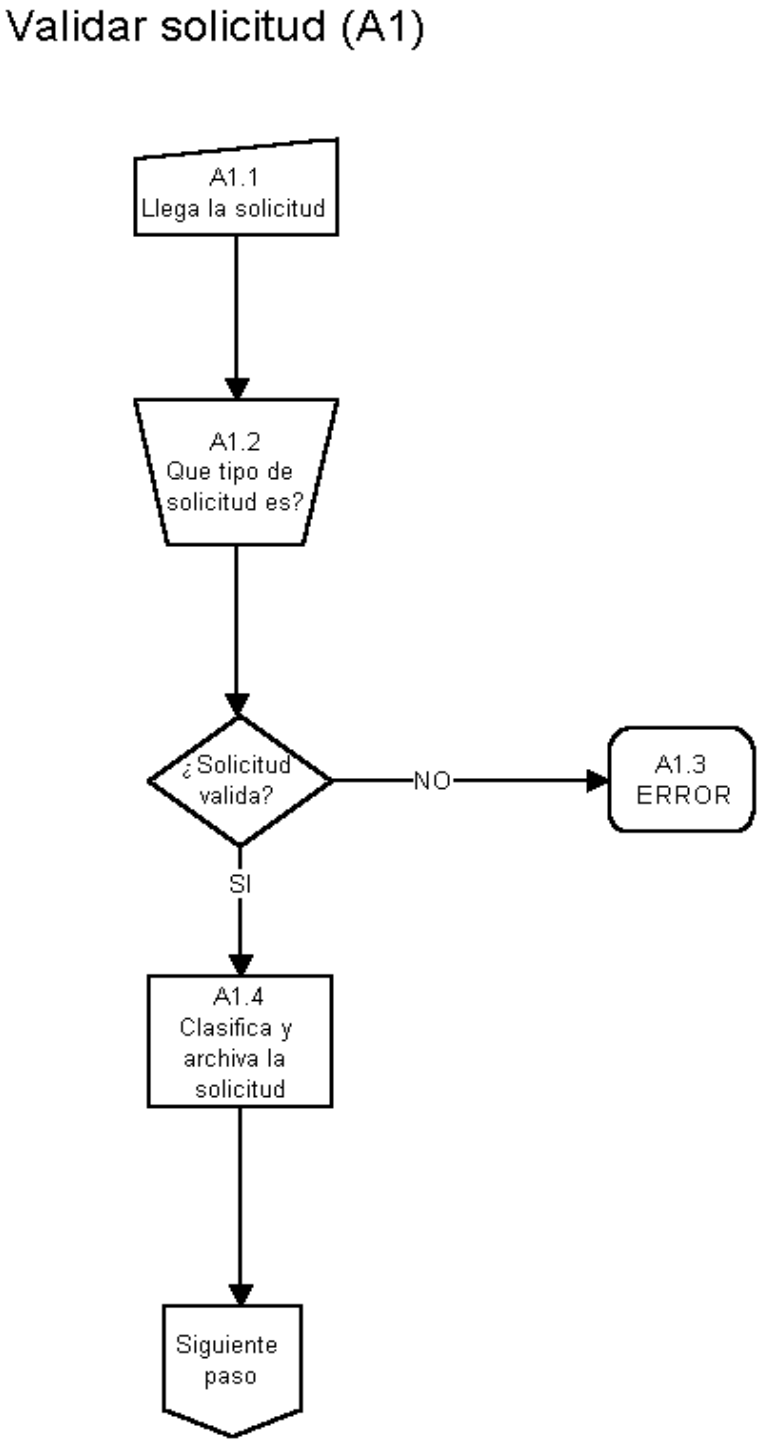
Crear y configurar las bases de datos (D)

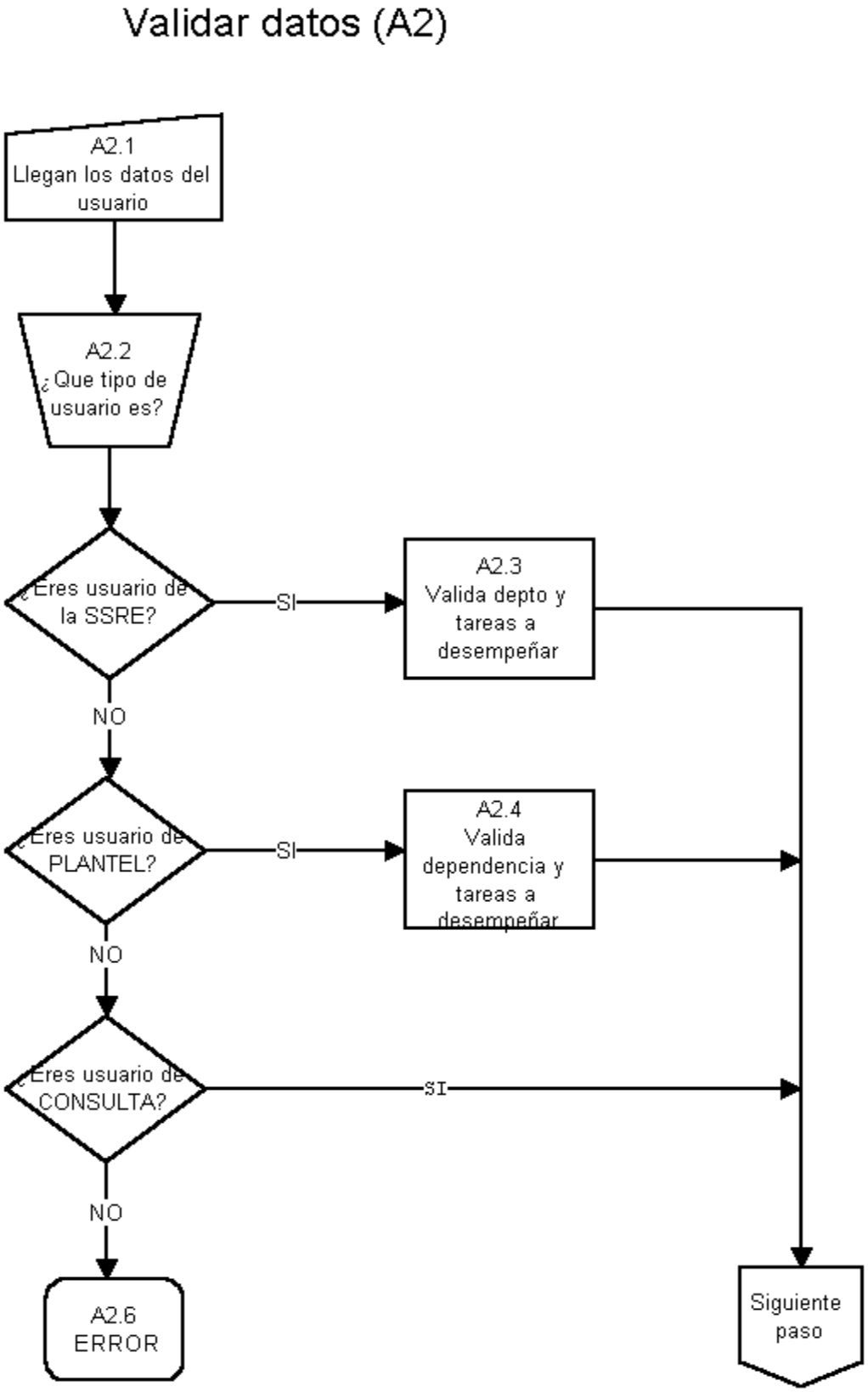


3.2 Diseño del SABAD

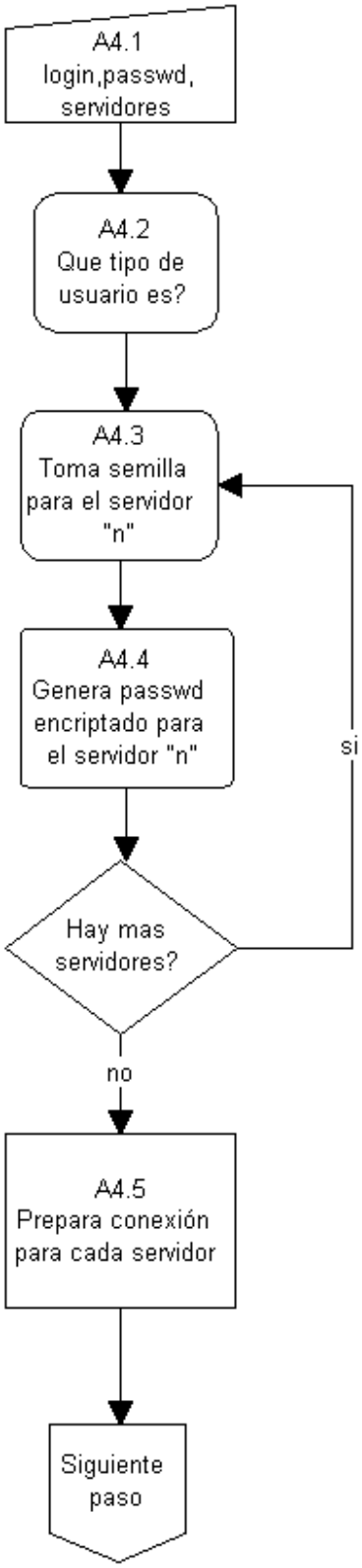
De acuerdo al análisis anterior el sistema se desprende el siguiente diseño para sus respectivos módulos:

Módulo Administración de cuentas (A)

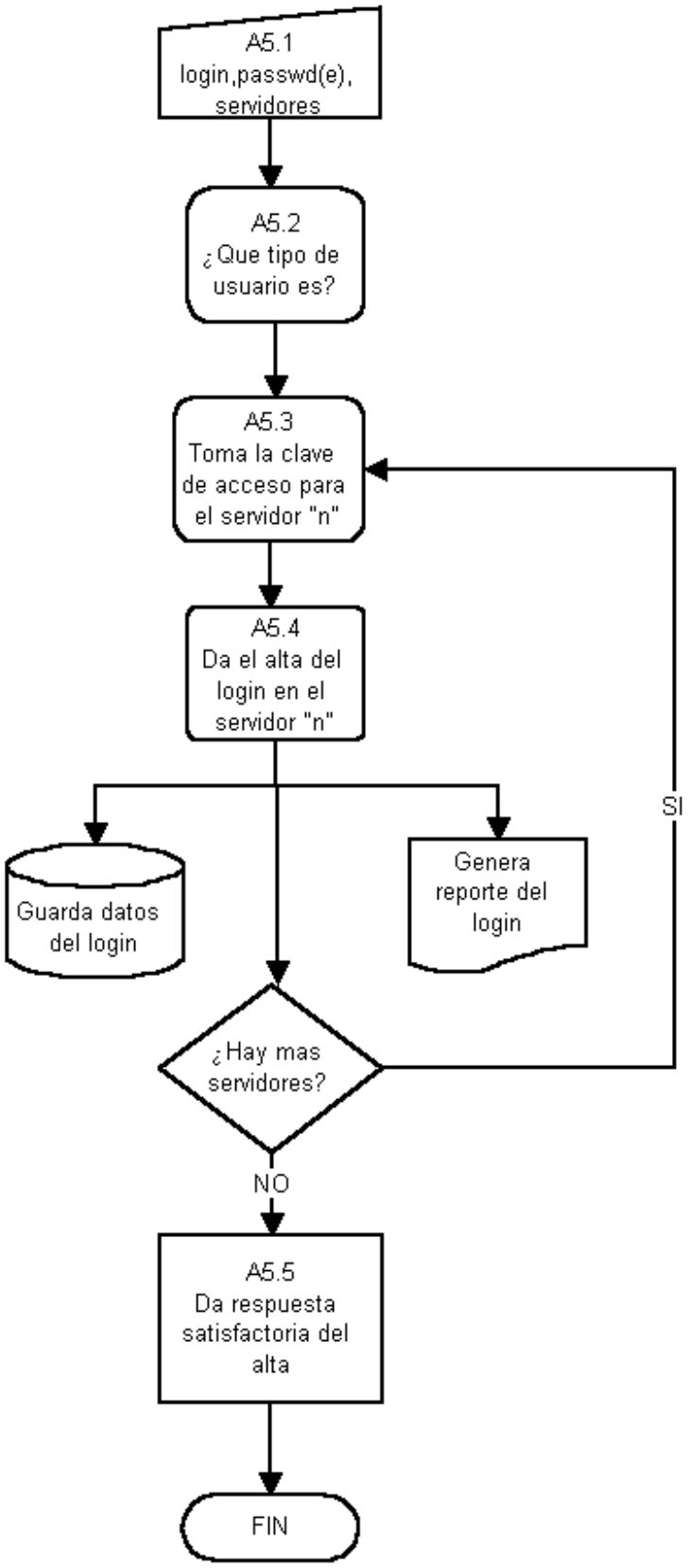



Validar datos (A2)
A2.1
Llegan los datos del usuario
A2.2
¿Que tipo de usuario es?
¿Eres usuario de la SSRE?
SI
A2.3
Valida depto y tareas a desempeñar
NO
¿Eres usuario de PLANTEL?
SI
A2.4
Valida dependencia y tareas a desempeñar
NO
¿Eres usuario de CONSULTA?
SI
NO
A2.6
ERROR
Siguiente paso

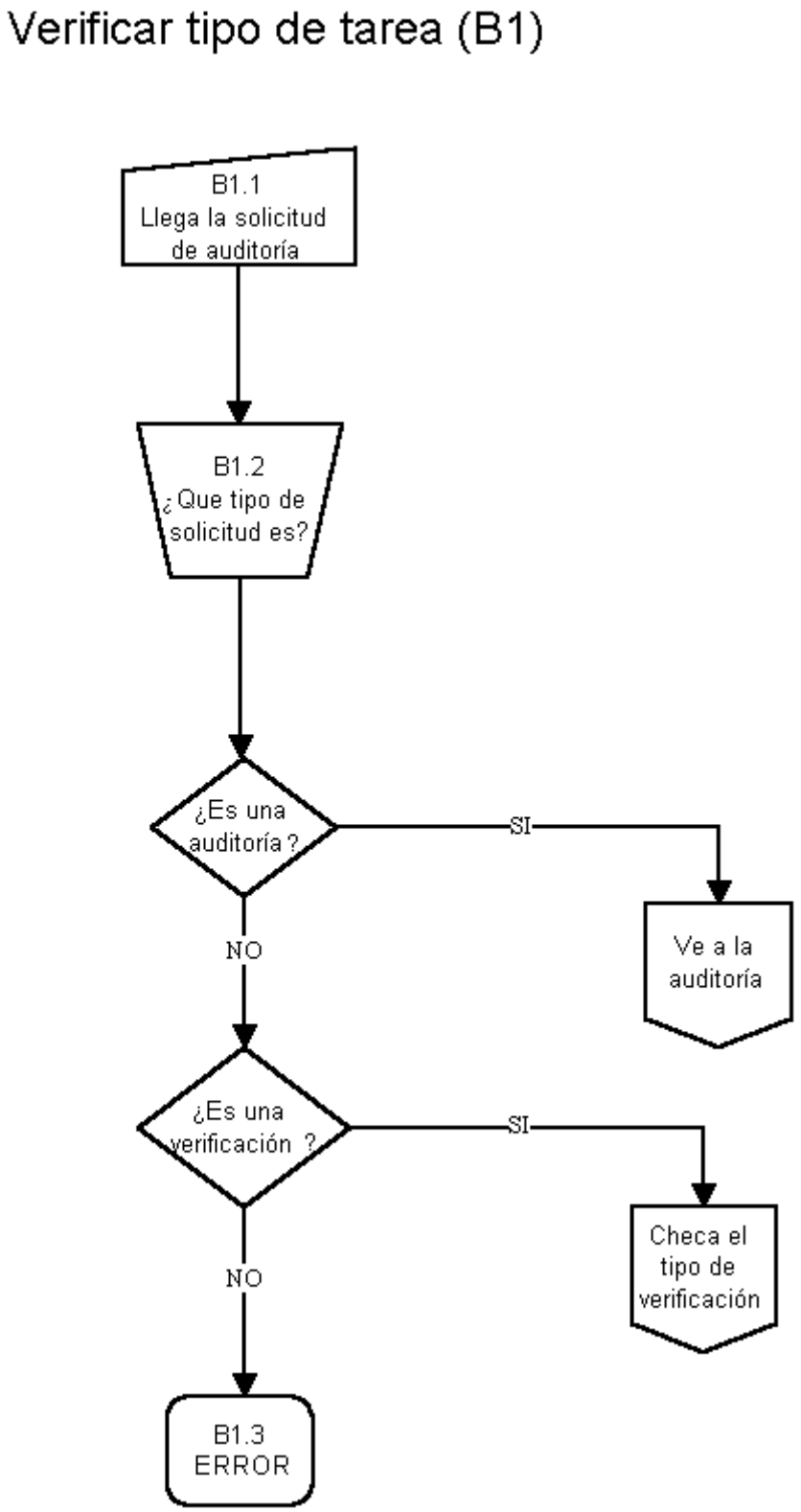
Generar el passwd encriptado (A4)

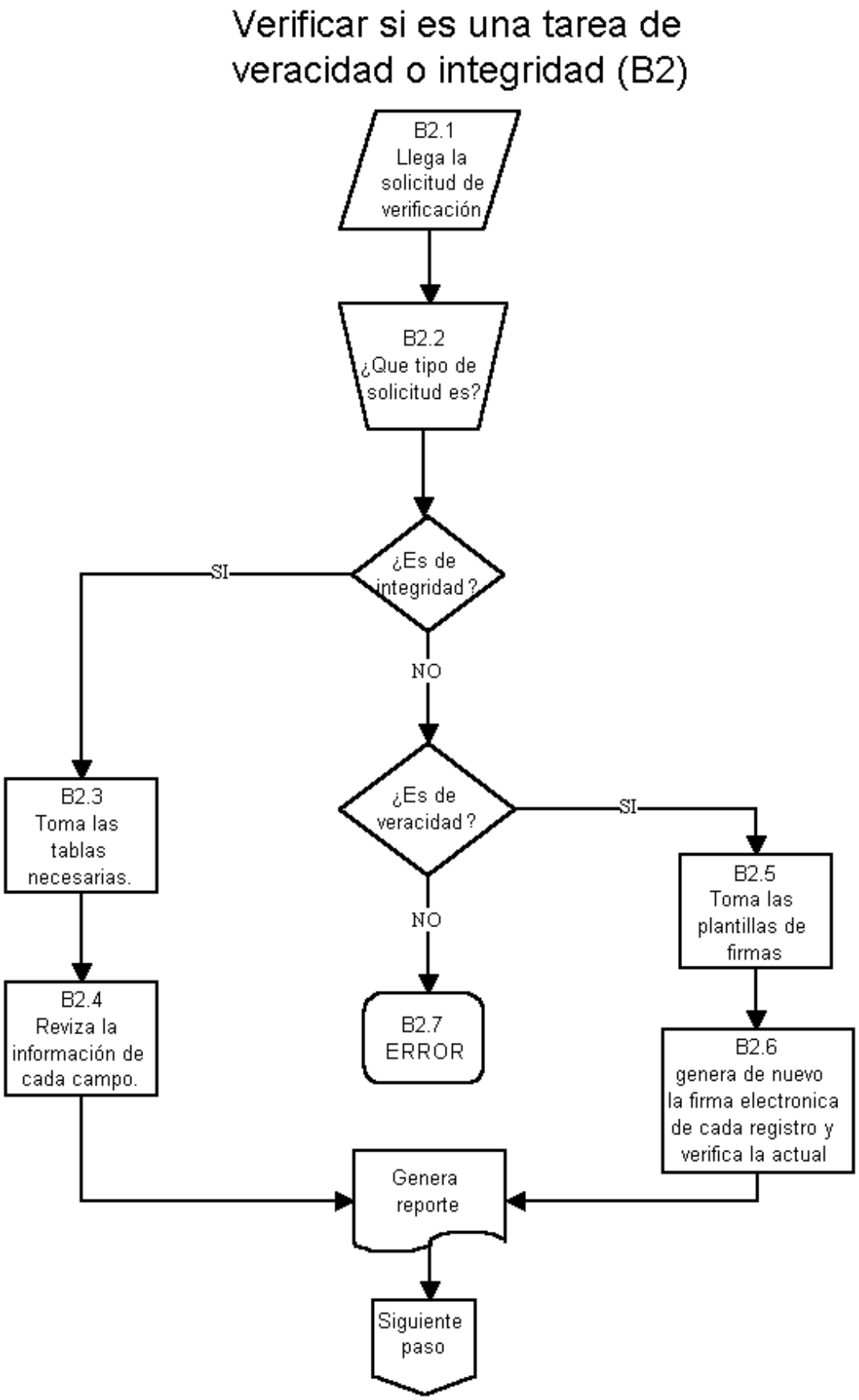


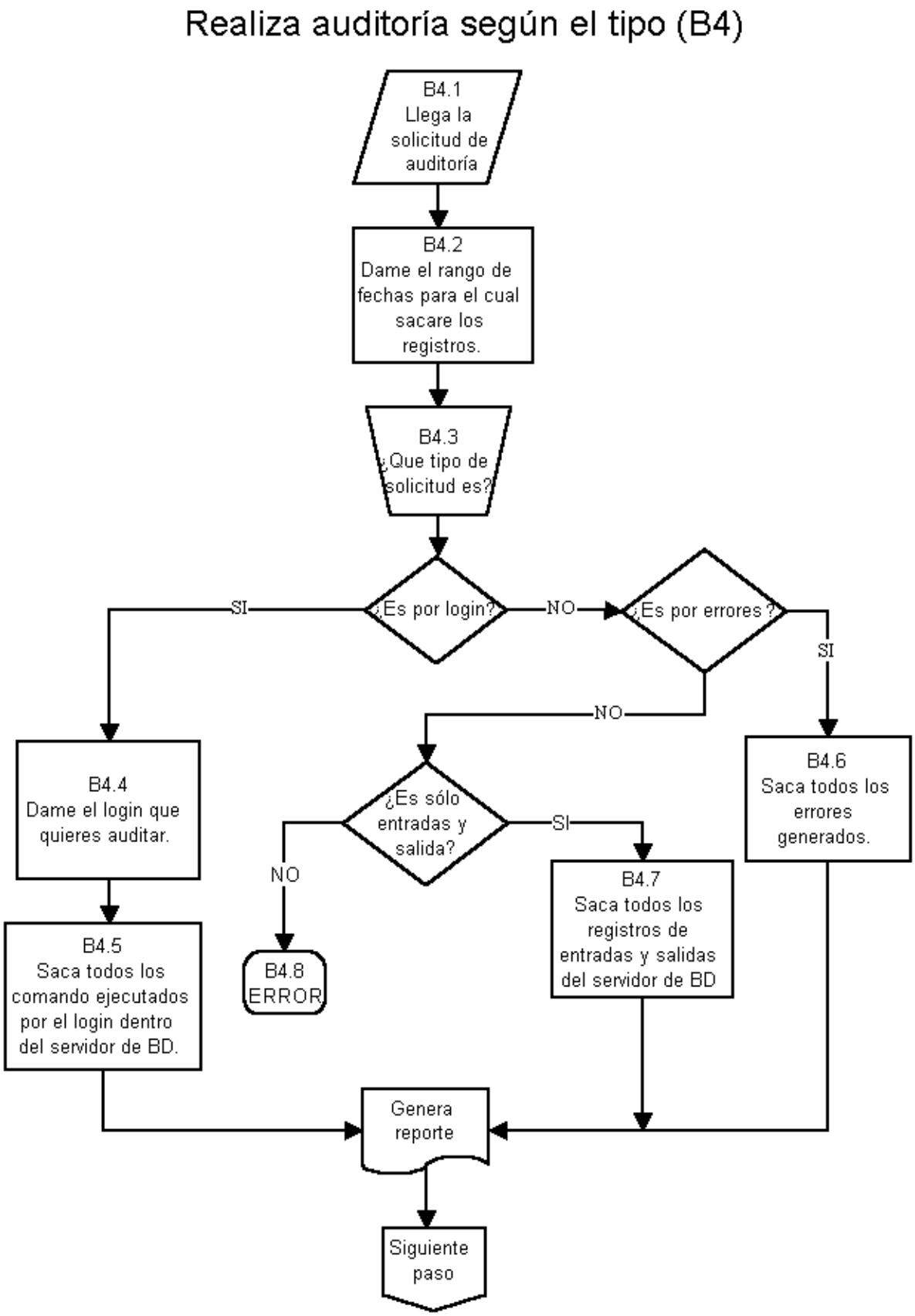
Dar de alta en "n" servidores (A5)

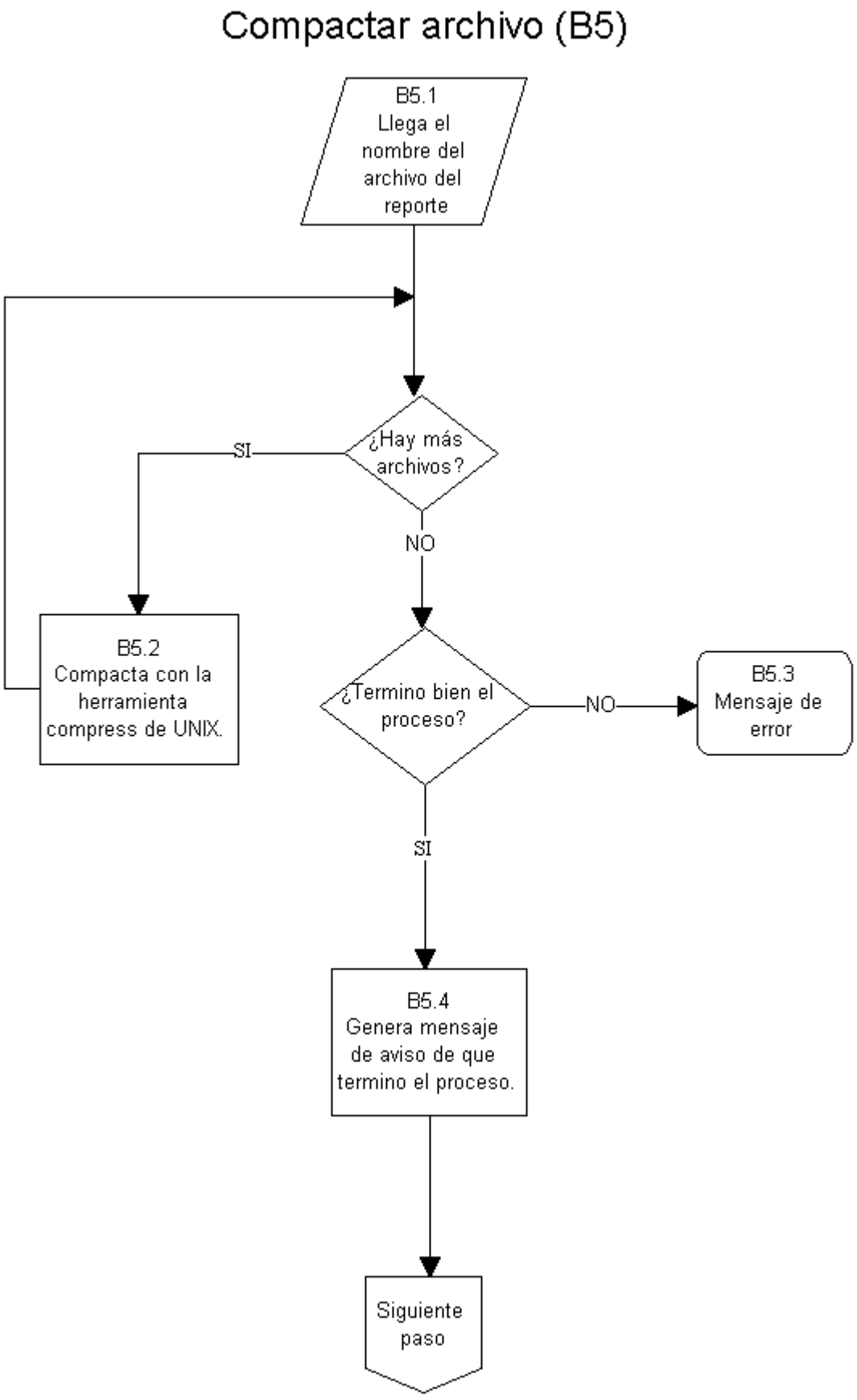


Módulo de auditoría e integridad de la información (B)

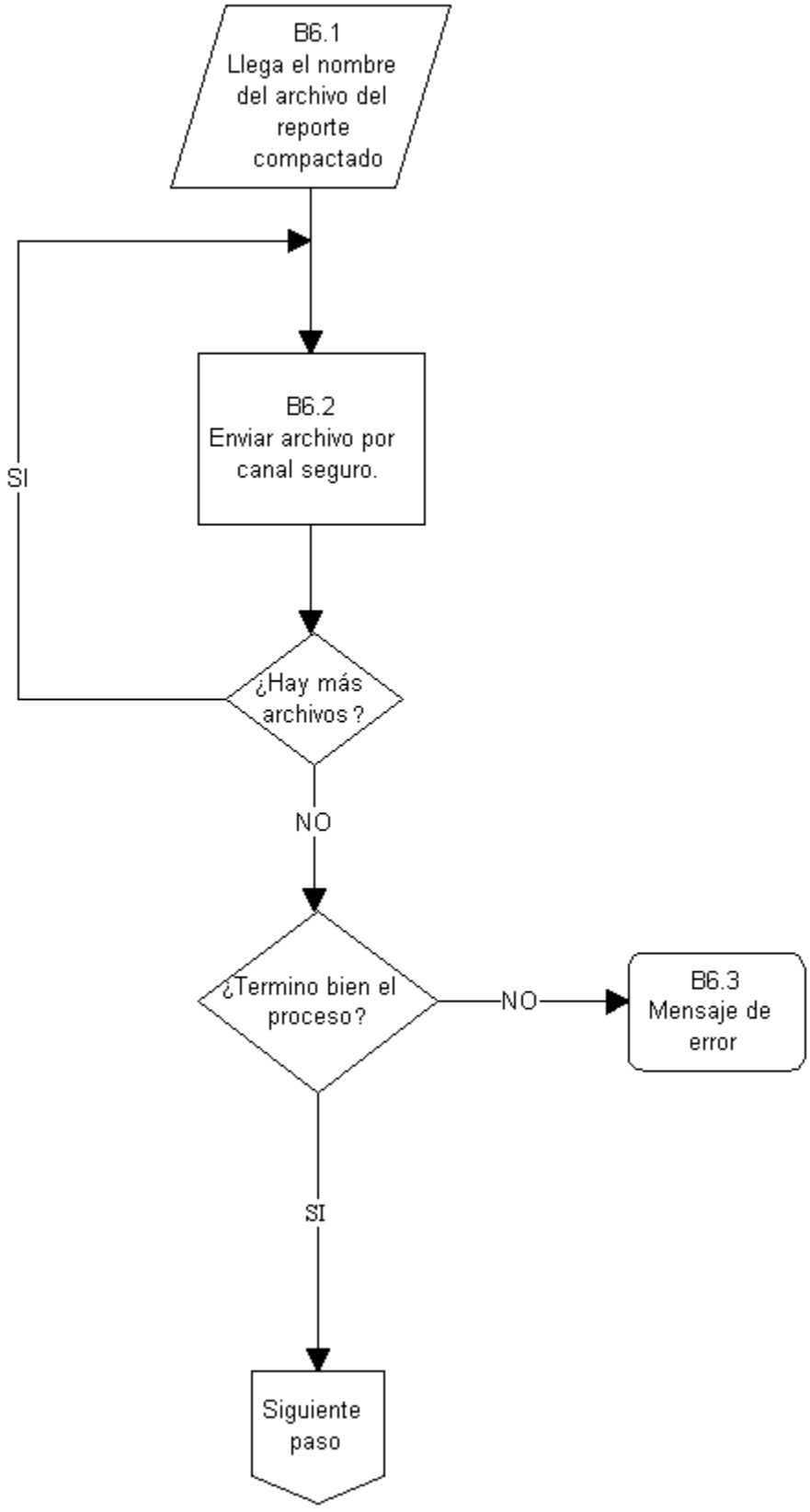



Verificar si es una tarea de veracidad o integridad (B2)
B2.1 Llega la solicitud de verificación
B2.2 ¿Que tipo de solicitud es?
¿Es de integridad ?
SI
NO
B2.3 Toma las tablas necesarias.
B2.4 Reviza la información de cada campo.
¿Es de veracidad ?
SI
NO
B2.5 Toma las plantillas de firmas
B2.6 genera de nuevo la firma electronica de cada registro y verifica la actual
B2.7 ERROR
Genera reporte
Siguiente paso

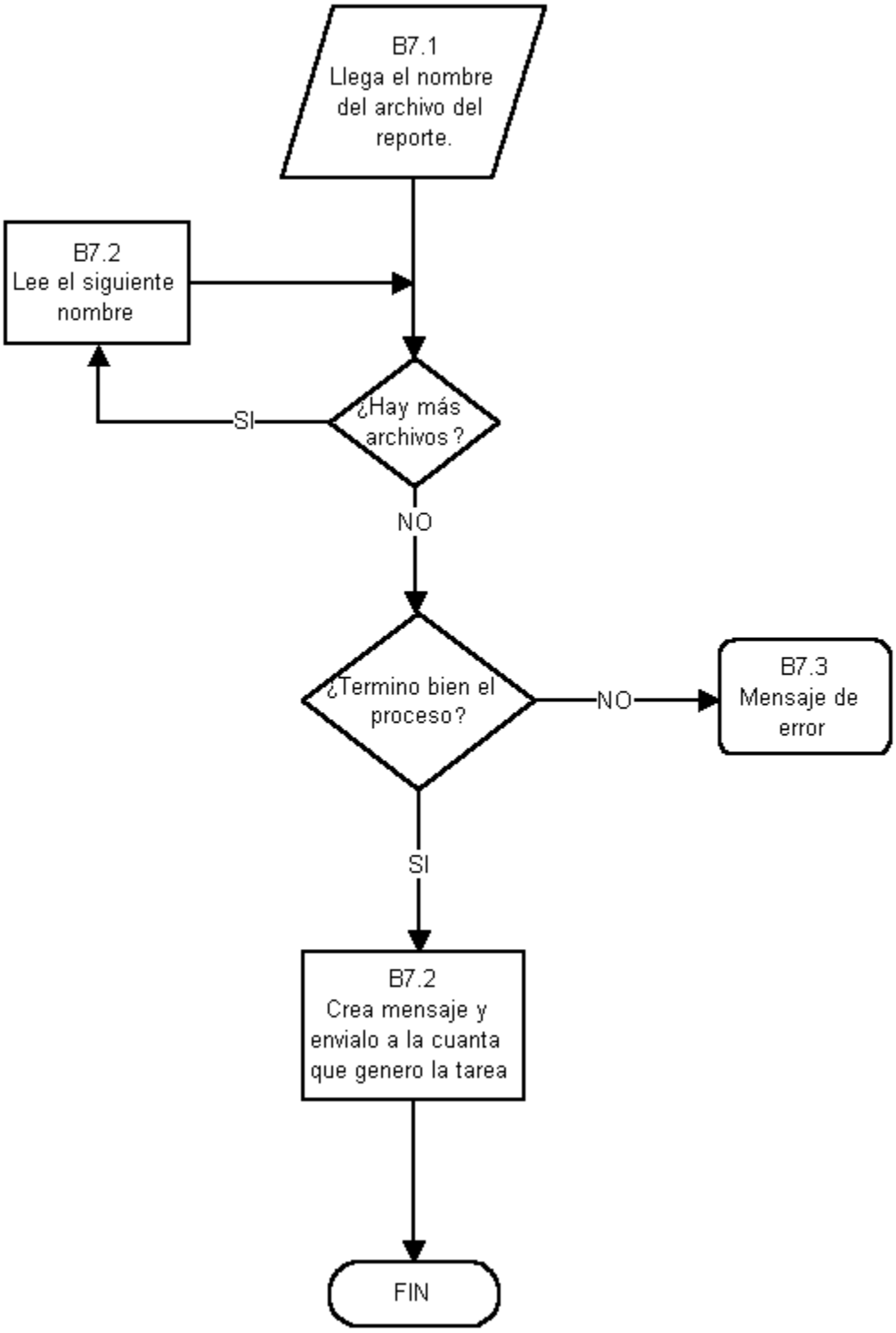

Realiza auditoría según el tipo (B4)
B4.1
Llega la solicitud de auditoría
B4.2
Dame el rango de fechas para el cual sacare los registros.
B4.3
¿Que tipo de solicitud es?
¿Es por login?
SI
NO
¿Es por errores ?
SI
NO
B4.4
Dame el login que quieres auditar.
B4.5
Saca todos los comando ejecutados por el login dentro del servidor de BD.
¿Es sólo entradas y salida?
NO
SI
B4.8
ERROR
B4.7
Saca todos los registros de entradas y salidas del servidor de BD
B4.6
Saca todos los errores generados.
Genera reporte
Siguiente paso


Compactar archivo (B5)
B5.1
Llega el nombre del archivo del reporte
¿Hay más archivos?
SI
NO
B5.2
Compacta con la herramienta compress de UNIX.
¿Termino bien el proceso?
NO
B5.3
Mensaje de error
SI
B5.4
Genera mensaje de aviso de que termino el proceso.
Siguiente paso

Enviar reporte a servidor seguro (B6)

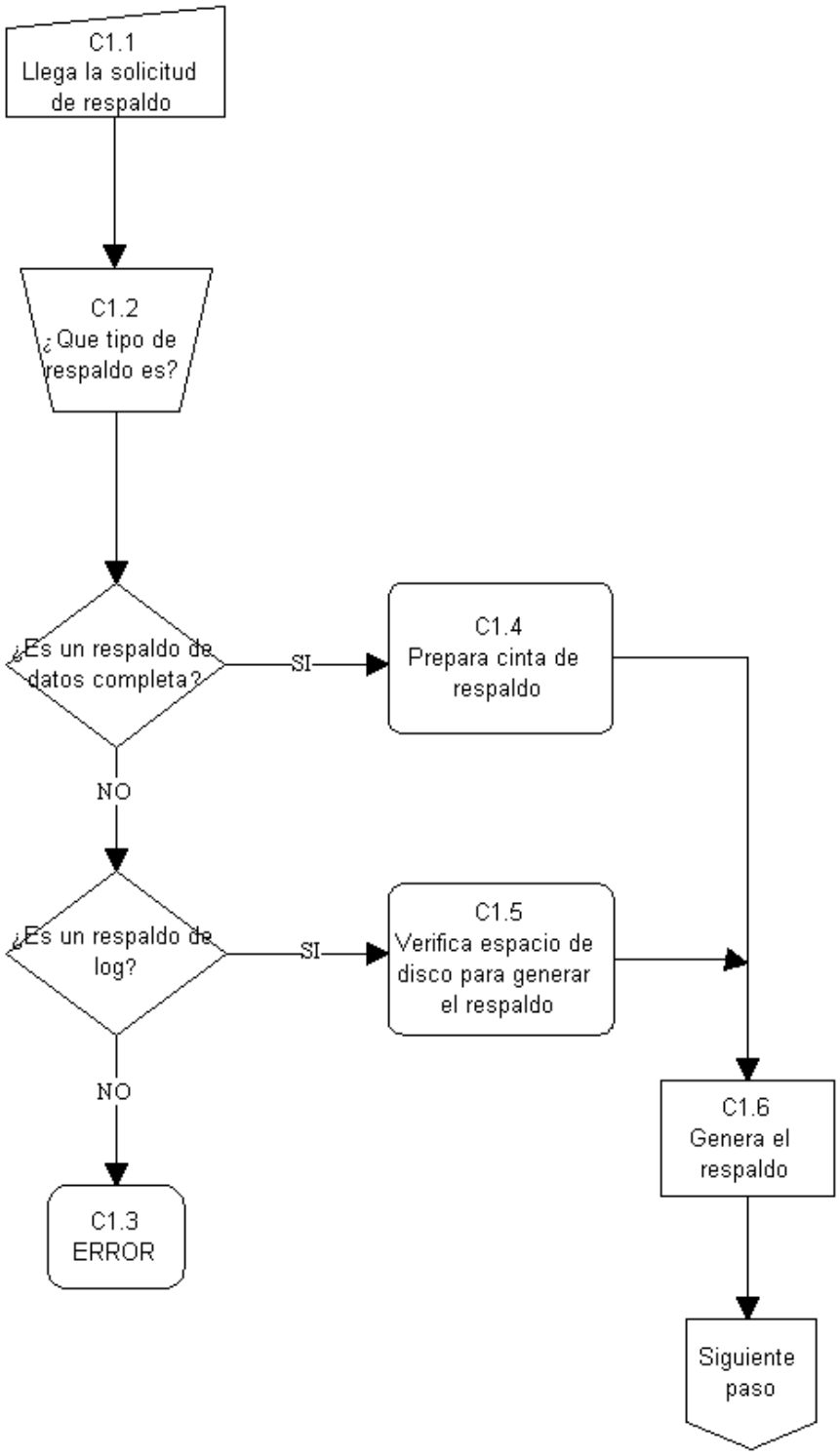


Avisar que se realizo la tarea (B7)

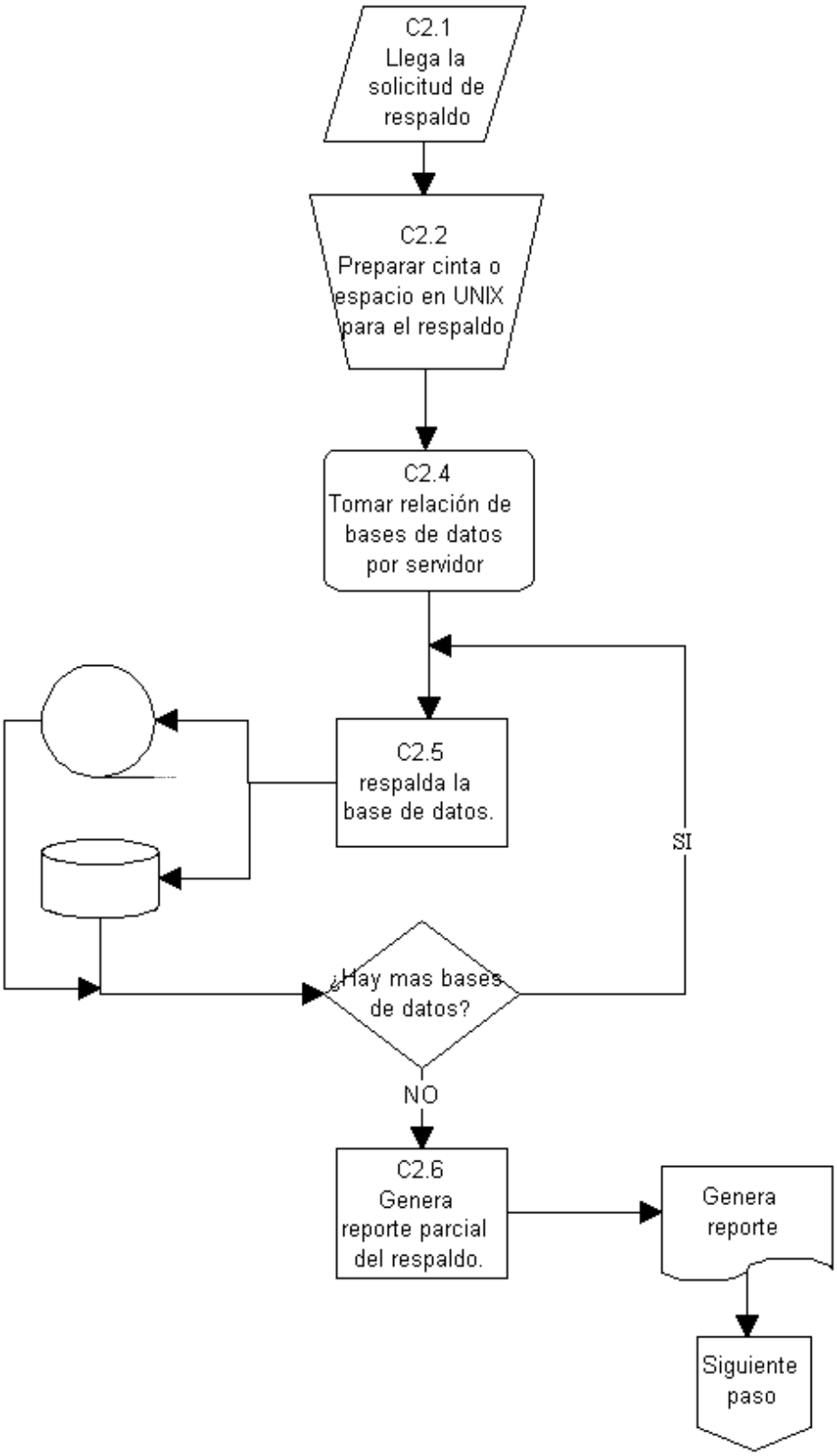


Módulo Administración de respaldos (C)

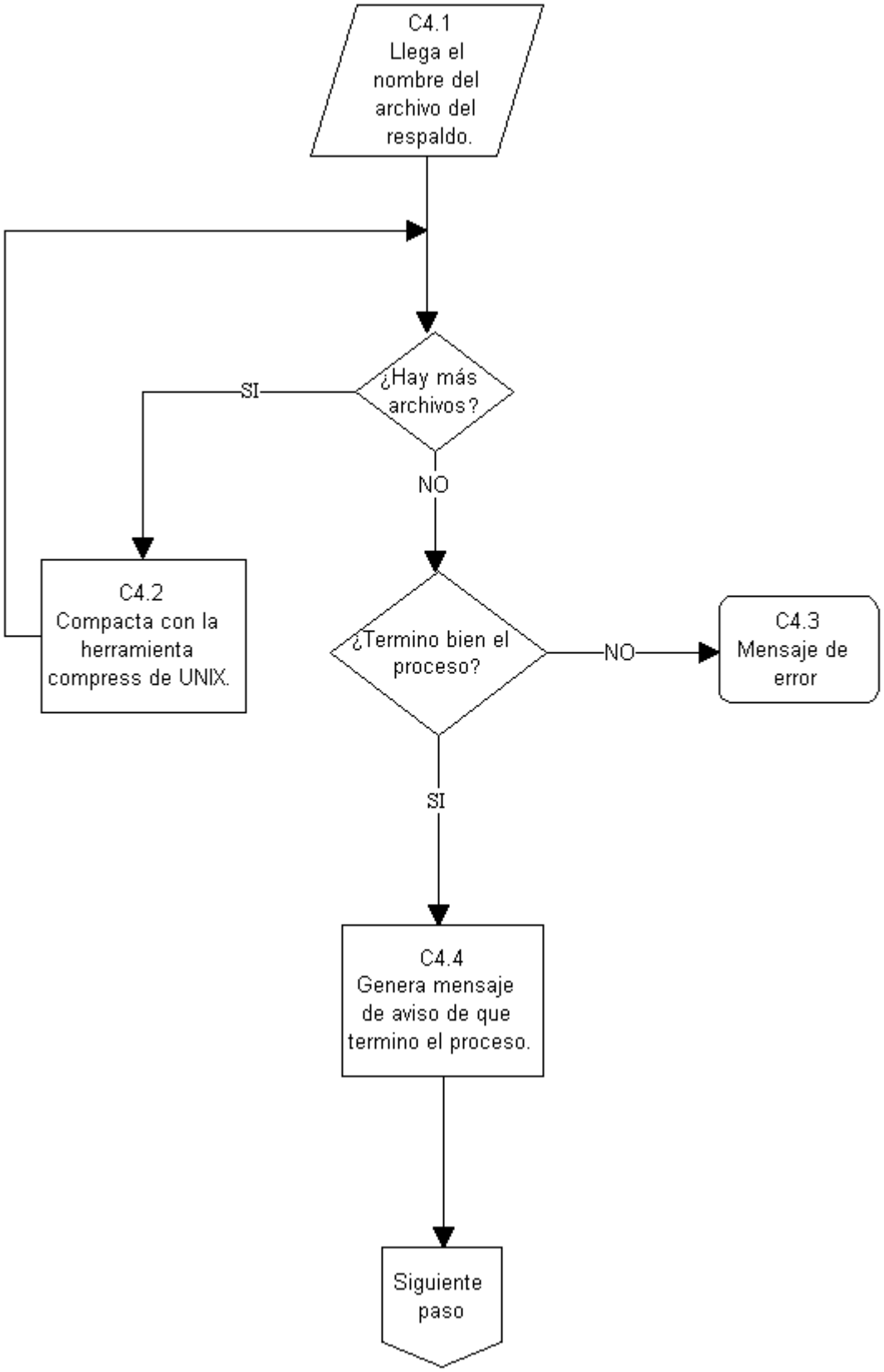
Verificar tipo de respaldo (C1)



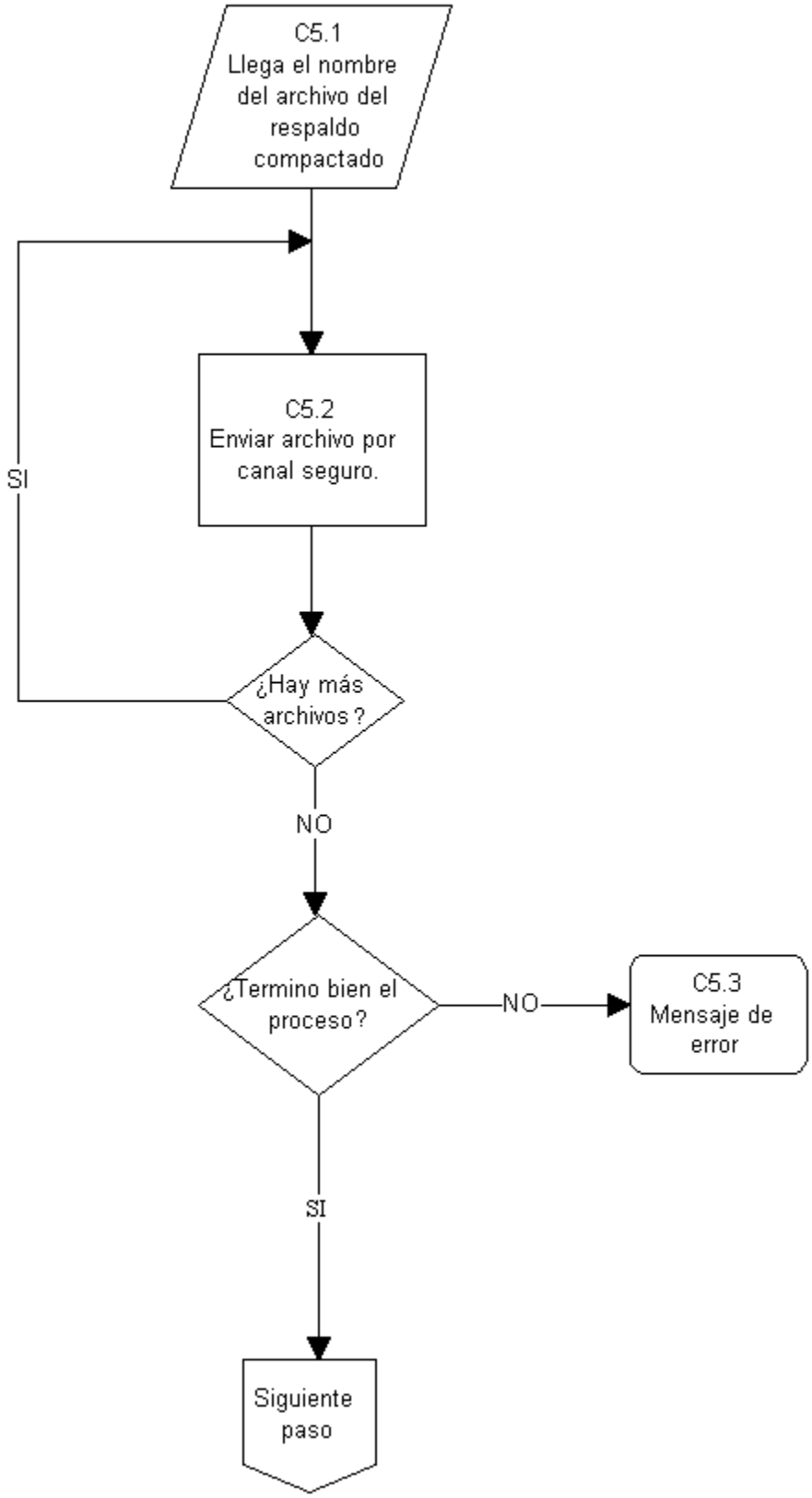
Realizar respaldo según el tipo (C2)



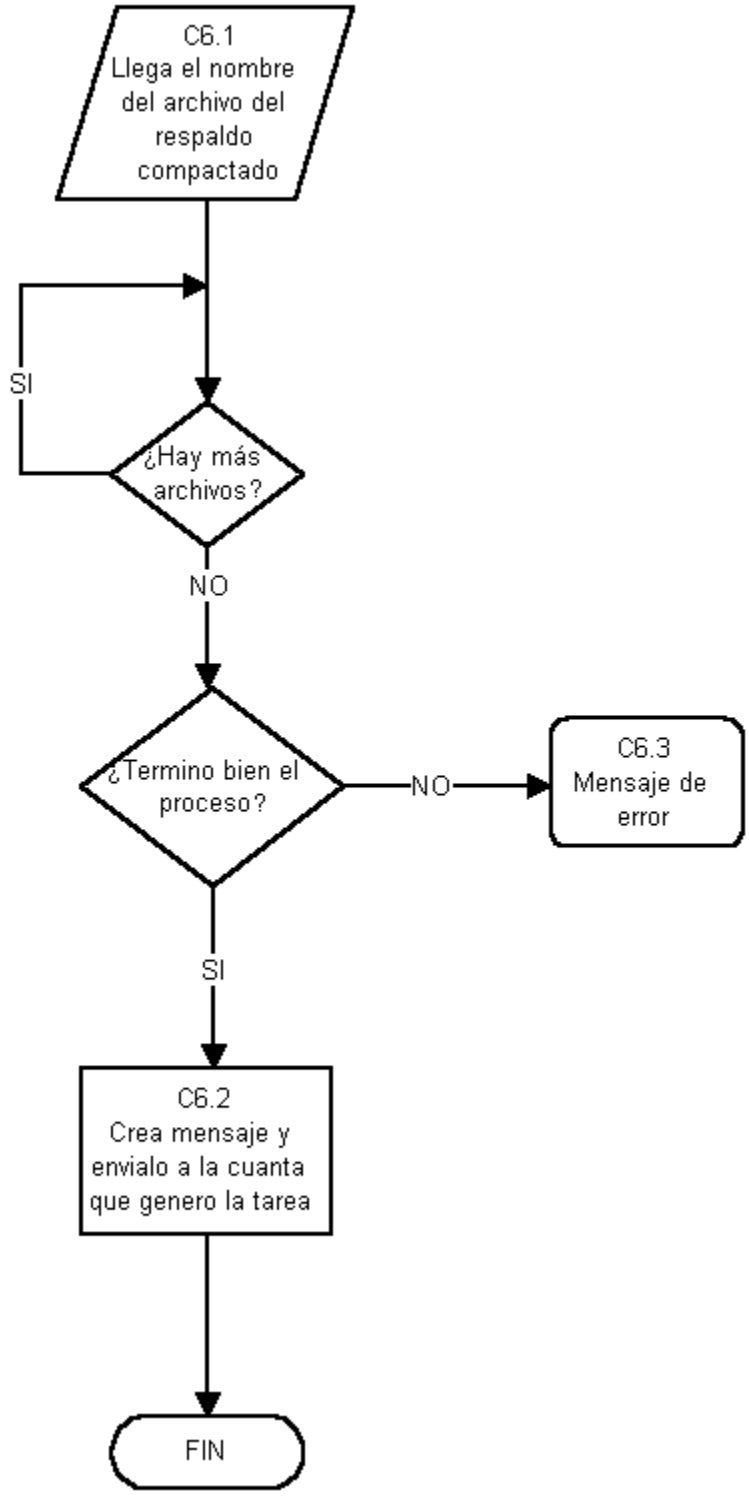
Compactar archivo (C4)



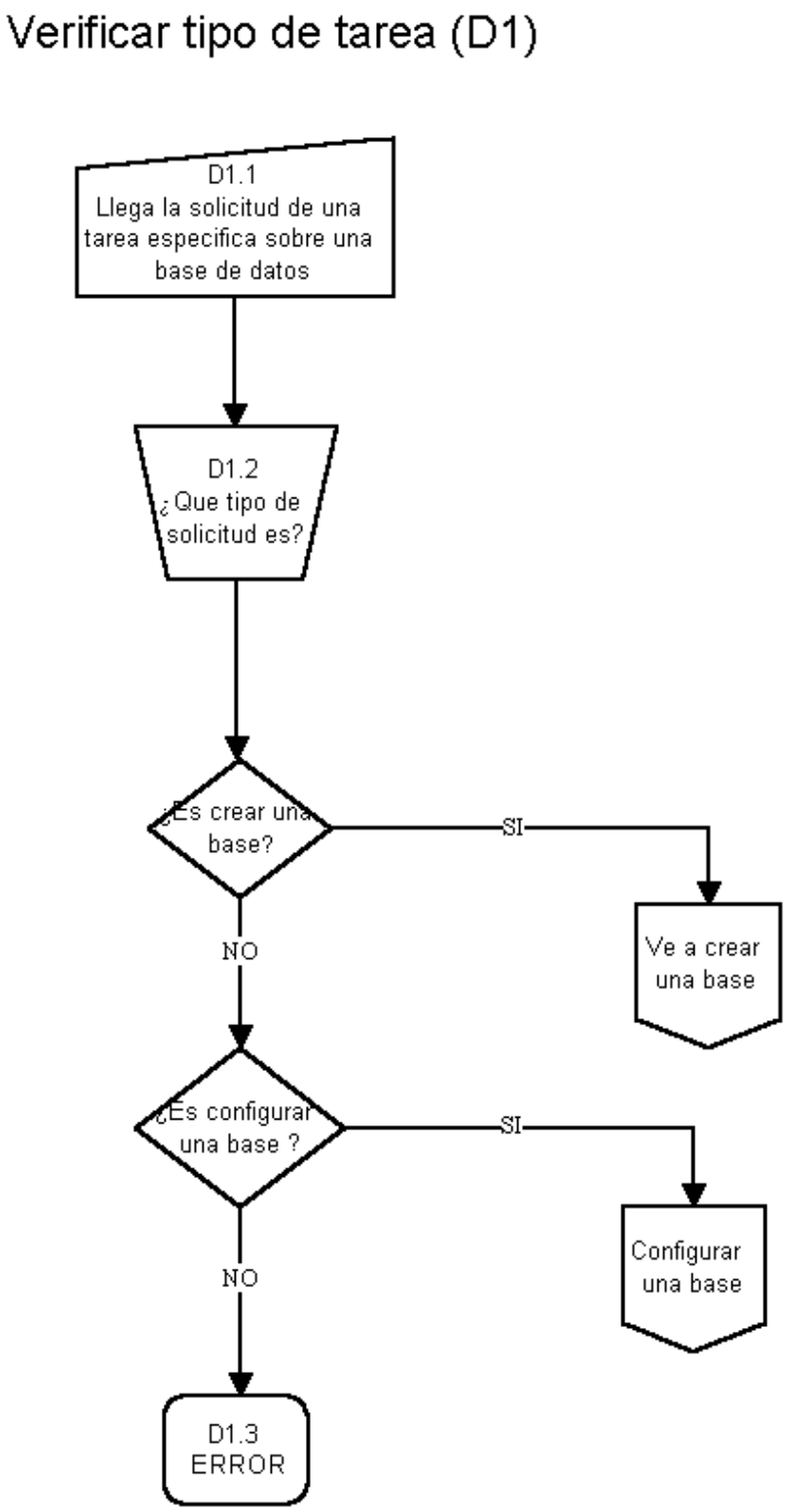
Enviar respaldo a servidor seguro (C5)

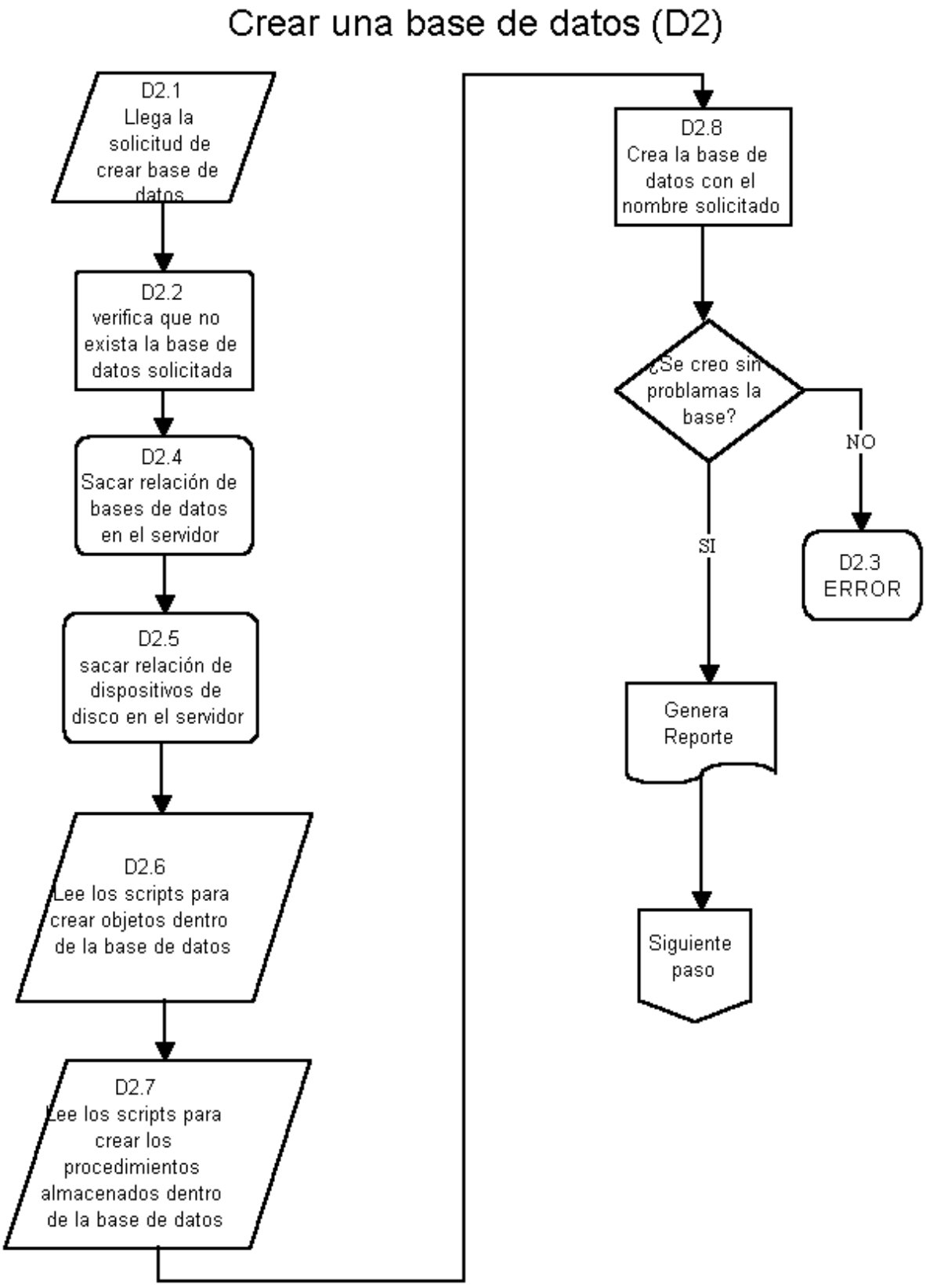


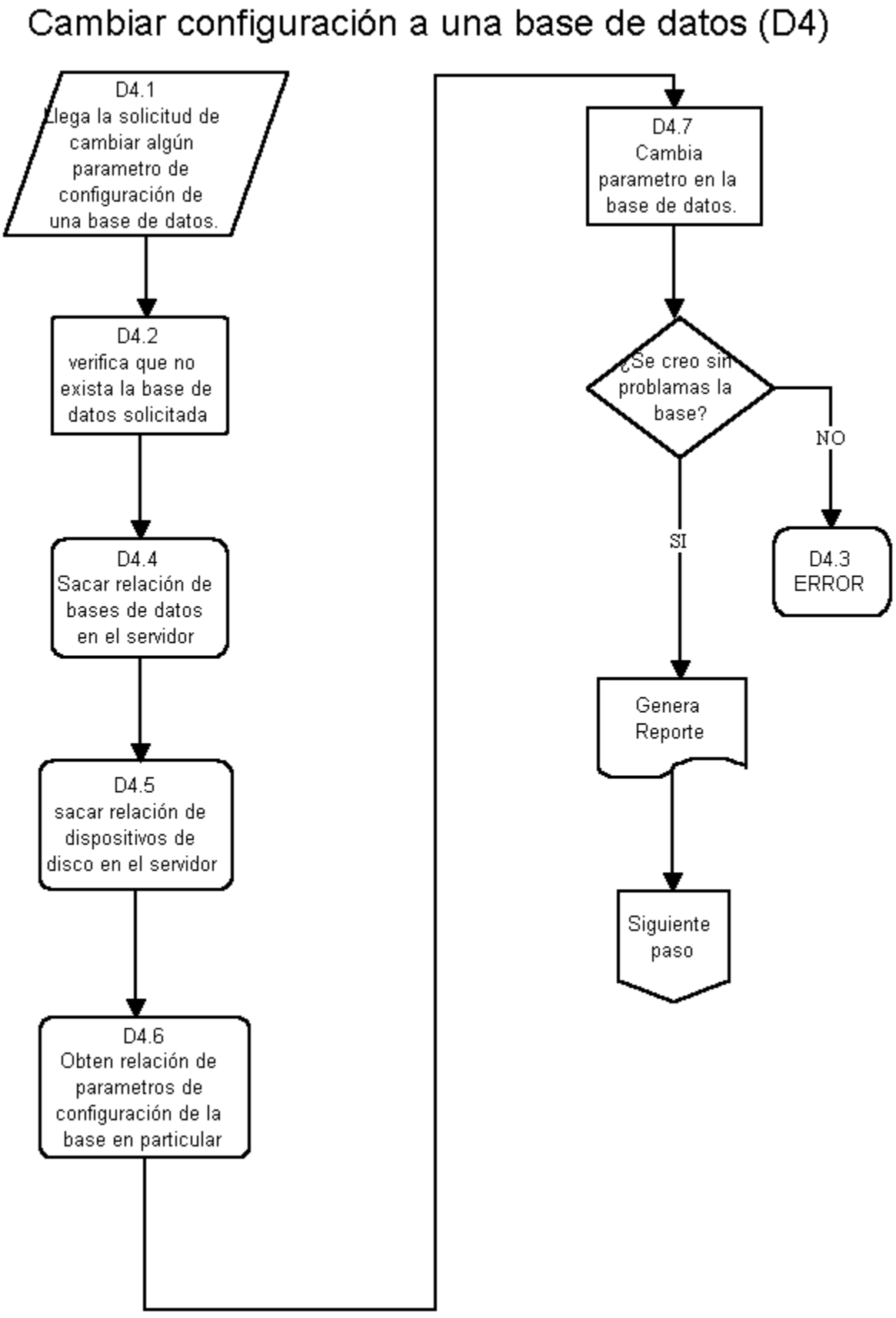
Avisar que se realizo la tarea (C6)

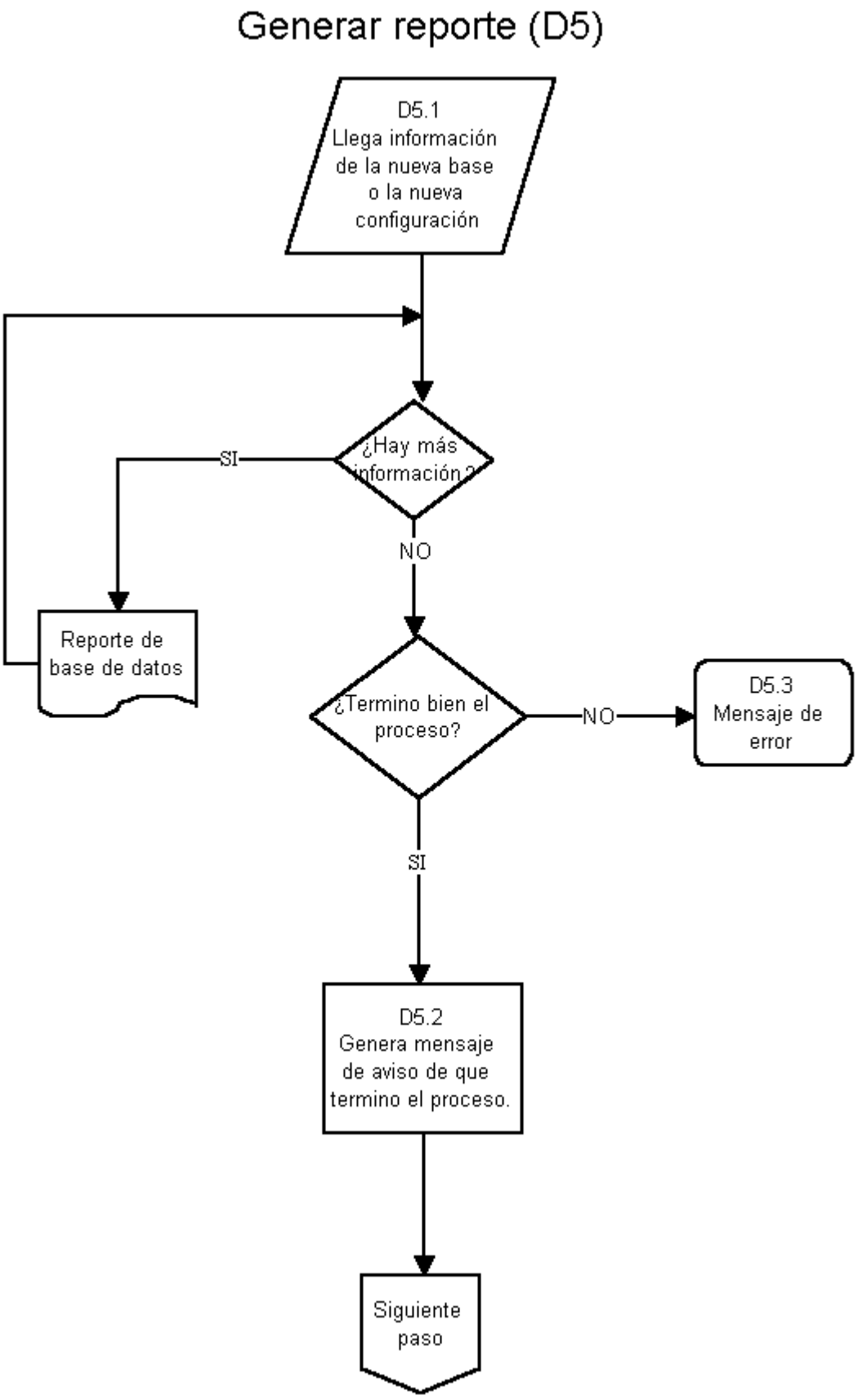


Crear y configurar las bases de datos

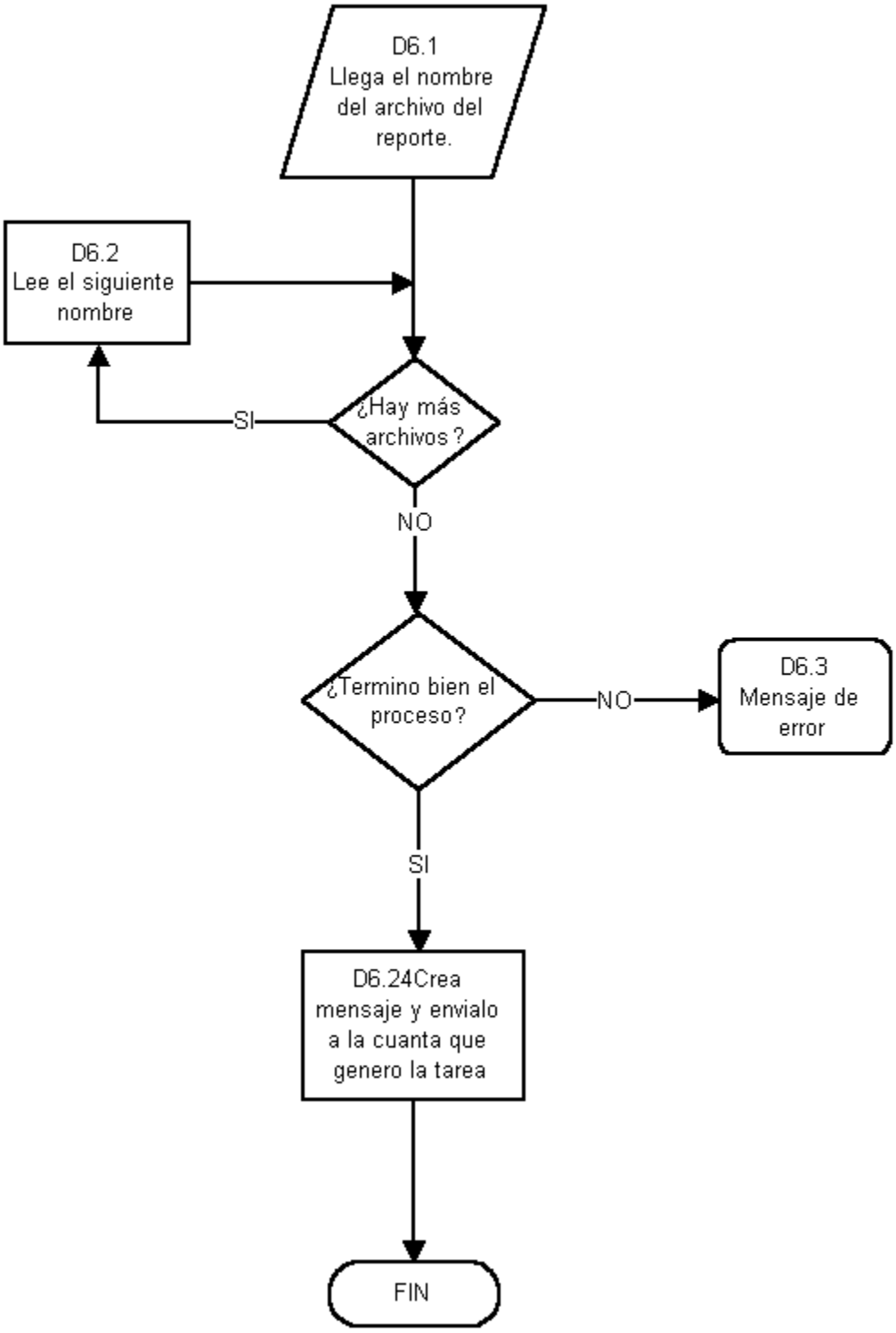



Crear una base de datos (D2)
D2.1
Llega la solicitud de crear base de datos
D2.2
verifica que no exista la base de datos solicitada
D2.4
Sacar relación de bases de datos en el servidor
D2.5
sacar relación de dispositivos de disco en el servidor
D2.6
Lee los scripts para crear objetos dentro de la base de datos
D2.7
Lee los scripts para crear los procedimientos almacenados dentro de la base de datos
D2.8
Crea la base de datos con el nombre solicitado
¿Se creo sin problamas la base?
NO
SI
D2.3
ERROR
Genera Reporte
Siguiente paso


Cambiar configuración a una base de datos (D4)
D4.1
Llega la solicitud de cambiar algún parametro de configuración de una base de datos.
D4.2
verifica que no exista la base de datos solicitada
D4.4
Sacar relación de bases de datos en el servidor
D4.5
sacar relación de dispositivos de disco en el servidor
D4.6
Obten relación de parametros de configuración de la base en particular
D4.7
Cambia parametro en la base de datos.
¿Se creo sin problamas la base?
NO
SI
D4.3
ERROR
Genera Reporte
Siguiente paso


Generar reporte (D5)
D5.1
Llega información
de la nueva base
o la nueva
configuración
¿Hay más
información?
SI
NO
Reporte de
base de datos
¿Termino bien el
proceso?
NO
D5.3
Mensaje de
error
SI
D5.2
Genera mensaje
de aviso de que
termino el proceso.
Siguiente
paso

Avisar que se realizo la tarea (D6)



3.3 Especificaciones de procedimientos

En esta sección estamos dando a detalle los pasos mínimos, que debería contener el sistema en sus diferentes módulos.

Módulo Administración de cuentas (A)

A. Este módulo administra y controla los *logins* y cuentas de usuarios, del mismo modo que todos los movimientos con respecto a passwords.

A.1. Validar solicitud. Este proceso de validación es un poco manual, ya que mucha de la información se verifica a partir del tipo de solicitud que nos haga llegar el interesado, jefe de departamento, jefe de área, secretario del plantel, subdirector, etc.

A.1.1. En este punto se hace llegar al Departamento de Servidores Sybase y UNIX (DASSU), por los diferentes medios las solicitudes que van desde un oficio acompañado con la solicitud A1 llena, hasta una simple solicitud verbal, pasando por el correo electrónico y/o llamada telefónica y es en virtud del tipo de solicitud y del tipo de solicitante que se atiende la misma.

A.1.2. Después de que llega la solicitud hay que verificar el tipo de la misma, así como los datos que se requieren para procesar el alta del *login* correspondiente.

A.1.3. Una vez que se verifica que la solicitud es candidata a ser procesada se pide el visto bueno del responsable del área, o plantel para verificar que esté enterado para posibles aclaraciones posteriores.

A.2. Validar datos. Este proceso también es de carácter manual en casi todos sus pasos pero es en este mismo que ya entra las particularidades del sistema SIAE.

A.2.1. En este paso se clasifica y verifica el tipo de clave que se está solicitando dar de alta ya que de esto depende los permisos y verificaciones que se deben de hacer con respecto a los datos de la solicitud.

A.2.2. Los posibles tipos de claves que se manejan en el SIAE son claves de consulta, actualización y/o inserción, a su vez estos tipos se subdividen en grupos dependiendo de la dependencia o plantel desde la cual se está haciendo la solicitud.

A.2.3. Si es una clave que solicite la subdirección de registro escolar por parte de alguno de sus departamentos se verifica qué tipo de tareas hace este usuario en las diferentes estructuras y tareas dependiendo del puesto así como los permisos particulares que se deben de dar sobre algunos objetos y los permisos sobre todos los servidores involucrados en las tareas del usuario.

A.2.3. Si es una clave de plantel o dependencia de igual manera se verifica lo anterior, pero los permisos solo se verán a nivel de base de datos local a su plantel y solo podrá tener estar tareas y permisos asignados dependiendo del criterio del responsable del área.

A.3. Mensaje de error. Este proceso contendrá los mensajes de error que hayan ocurrido al validar los datos.

A.4. Generar password encriptado. Este proceso ya lo ejecuta en su totalidad el sistema.

A.4.1. Como entrada tendremos el *login* y password asignado al *login* del usuario, así como la relación de los posibles servidores asignados para el *login* según el tipo y la clasificación.

A.4.2. Hay que verificar el tipo de usuario para saber si es del SSRE y PLT u otra dependencia que tenga acceso al SIAE.

A.4.3. Si están involucrados diferentes servidores se debe de tomar la semilla para los diferentes servidores esto dependiendo del tipo de *login* que se quiere dar de alta.

A.4.4. Se genera el password correspondiente para cada uno de los servidores que están involucrados en el proceso de alta.

A.4.5. Una vez que se han generado los diferentes passwords para cada uno de los servidores, se prepara el sistema para conectarse a los diferentes sitios y se verifica que no halla algún problema que pudiera entorpecer la tarea. Si algún sitio (máquina) no fuera posible conectarse se avisa al DBA tanto local como central para verificar porqué está ocurriendo.

A.5. Dar de alta en los "n" servidores.

A.5.1. Como entrada de este proceso se tiene el *login* asignado así como el password que se crea (encriptado) para cada uno de los servidores involucrados en el proceso de alta, en este momento ya se verificó que exista conexión hacia cada uno de los servidores así como la clave adecuada de acceso.

A.5.2. Como siempre, hay que verificar el tipo de usuario y los posible permisos adicionales que pudiera requerir la clave dependiendo de las tareas del usuario, así como el grupo al cual se va a unir la clave que se dé de alta.

A.5.3 Se toma la clave de acceso para cada servidor, la cual está almacenada en el sistema mismo, de igual manera se toman los datos adicionales como por ejemplo las base de datos a las cuales se dará el alta, y el nombre completo del usuario.

A.5.4. Damos el alta en cada servidor y base así como asignamos al grupo correspondiente al usuario y los permisos adicionales. En este momento y una vez que se ha realizado el alta se genera un registro del reporte correspondiente registrado el alta del *login*, así como se guarda un control dentro del sistema para posteriores

aclaraciones, el reporte no contendrá password que estén involucrados en el alta.

A.5.5. Por último se da respuesta al responsable del área de la correspondiente alta, este proceso corresponderá al tipo de solicitud y de cómo se nos hizo llegar la misma.

A.5.6. Cerramos todas las conexiones que se hayan realizado o abierto.

Módulo Administración de auditoría e integridad de la información (B)

B. En este módulo estarán las tareas de validación y verificación de la autenticidad de la información, esto se hará vía los mecanismos previamente establecidos como pueden ser: validación de historia académica, en esta tarea comparamos registro a registro los valores importantes de las tablas del servidor central contra las tablas del plantel, o por ejemplo una verificación de autenticidad en esta tarea en particular se verifica la información de una tabla por cada registro deberá existir un campo que firme o guarde información que garantice la veracidad de la misma, de igual manera es en esta opción que contendrá la opción para generar los reportes de la auditoría interna del servidor de base de datos.

B.1. Verificar tipo de tarea. Este proceso verifica el tipo de tareas que se pueden hacer dentro del sistema, es decir, se podrá elegir entre dos tipos de tarea: veracidad y autenticidad.

B.1.1. La llegada de la solicitud es por medio de un menú en el cual se contendrá sólo las opciones permitidas.

B.1.2. El sistema deberá ver el tipo de tarea que se esta solicitando y definirá la secuencia de los siguientes pasos: si es una auditoría vemos a el proceso adecuado y si es una verificación de igual manera el sistema deberá mandar la tarea a otro proceso.

B.1.3. Si la solicitud no esta dentro de los parámetros de programación se deberá de marcar como un error y se implementara una función para avisar al administrador o responsable de la administración del servidor de base de datos que ha generado un error la aplicación y/o que se esta tratando de forzar al sistema de hacer más de lo que fue programado.

B.2. Verificar si es una tarea de veracidad o integridad, esta tarea es especifica para el trabajo con los datos de la base de datos, es decir en este punto esta decidido que debemos trabajar con la información almacenada en tablas especificas y comparar ya se a campos entre tablas o un campo contra una firma generada por el sistema.

B.2.1. En este punto llega la solicitud de hacer una tarea de integridad o autenticidad.

B.2.2. Una vez filtrada la solicitud para saber si se quiere trabajar con la auditoría interna del servidor de base de datos o con tablas especificas del sistema SIAE, debemos elegir si es una tarea de integridad o autenticidad.

B.2.3. Si elegimos integridad debemos saber que tablas están involucradas en esta tarea, por lo tanto dependiendo de las tablas involucradas se tomaran las pertinentes para su validación.

B.2.4. Cuando ya tenemos las tablas involucradas debemos verificar que tanto la información que se contiene en las tablas del sitio-plantel y las tablas del servidor central coincida en todos sus campos tanto en la cantidad de información como en que la información que se contenga en el sitio-plantel este en el servidor central y viceversa, si se encontrara alguna diferencia se inserta en el archivo del reporte.

B.2.5. Si se optó por una tarea de veracidad debemos de tomar la información necesaria (plantilla) de usuario-*login* que ejecuta la tarea y tiene esta tarea asignada, por su puesto que ésta información estará condicionada a que el *login* que ejecuta esta tarea tenga asignada la información necesaria.

B.2.6. Teniendo la información necesario y la información del registro que se va ha validar debemos generar de nuevo la información que valida el registro y se compara con el campo adecuado para verificar que la información es valida, si se encontrara alguna diferencia se inserta en el archivo del reporte.

B.3. Mensaje de error. Este proceso contendrá una función que controle los mensajes de error que hayan ocurrido en cualquiera de los pasos anteriores y subsecuentes, así como los mecanismos para avisar al administrador de que ha ocurrido un problema con ésta tarea.

B.4. Realizar auditoría según el tipo: este proceso tiene englobados varias tareas, primero tiene que verificar la existencia del mecanismo auditoría de SYBASE así como identificar la tabla de la cual se sacara toda la información, una vez que se obtenga esta se tendrá que decidir que tipo de reporte se requiere por parte del usuario que esta ejecutando la tarea: reporte por usuario, por errores, etc.

B.4.1. En este momento de manera automática y por medio del sistema llega la solicitud de los reportes de la actividad del servidor de base de datos.

B.4.2. Tenemos que dar el rango de fechas para los cuales los reportes se generaran, esto podría ser incluso el reporte de la actividad del día actual hasta donde se halla trabajado.

B.4.3. Después debemos de saber el tipo de reporte que se requiere, por usuario, entradas y salidas del servidor de base de datos, por errores cometidos dentro del servidor de base de datos, o toda la información contenida en las tablas de la auditoria interna de SYBASE.

B.4.4. Si se requiere el reporte por *login*, debemos de dar el *login* que se requiere auditar o dar que saque la información por todos los *logins* que hallan tenido actividad en el rango de fechas antes capturadas.

B.4.5. Una vez que se tenga toda esta información se procede a sacar el reporte correspondiente para cada *login* un archivo por cada *login* que tenga actividad en el rango de fechas antes mencionadas.

B.4.6. Si el reporte que se requiere es sólo los errores cometidos dentro del servidor de base de datos entonces se procede a generar el reporte correspondiente.

B.4.7. Por último si lo que el usuario requiere es sólo las entradas y salidas del servidor de base de datos entonces se aplica la consulta correspondiente dentro de la tabla antes definida, que contenga la información que se requiere, y se procede a generar el reporte.

B.4.8. Si no es ninguno de los tipos antes mencionado y se requiere información entonces es un error.

B.5. Compactar Archivos: en este proceso lo único que se hará es una vez que tengan los reportes correspondientes y ya revisados entonces se compacta el archivo para que ocupe el menor espacio posible.

B.5.1. Llegada del nombre del archivo del reporte, el sistema estará esperando que llegue el nombre del archivo para poder trabajar la información.

B.5.2. Compacta el archivo con la herramienta estándar compress de UNIX que nos da rangos aceptables de compactación (al rededor de 36%) y nos garantiza que en cualquier otro lugar pueda descompactarlas, consultar y manejar la información, y así nos mantenemos hasta haber terminado con cada uno de los reportes que pudieran ser generados por las tareas anteriores.

B.5.3. Si la tarea en no termina bien por algún problema en la compactación manda error al administrador.

B.5.4. Cuando ya halla terminado adecuadamente la tarea se procede a generar un reporte de toda la tarea en su conjunto englobando todos los pasos anteriores.

B.6 Enviar reportes a un servidor seguro: cuando ya se tenga concluida la tarea de compactar todos los reportes generados, se procede a transferir los archivos compactados al servidor seguro.

B.6.1. Llega el nombre del archivo del reporte ya compactado y se abre un canal encriptado para su transferencia hacia un servidor previamente definido el cual garantiza que los reportes se tendrán en calidad de respaldo en otra maquina.

B.6.2. Se toma el nombre del archivo y se transfiere con un protocolo seguro y confiable, y se mantiene este paso hasta haber terminado con todos los archivos antes compactados.

B.6.3. Si no terminó adecuadamente la tarea genera un mensaje de error al administrador, y si termino bien termina la función

B.7. Avisar que se realizó la tarea: en este proceso lo único que se esta haciendo es una recopilación del estatus de la tarea para reportarlo al administrador.

B.7.1. Llega a este proceso los diferentes nombres de los archivos de los reportes que se generaron.

B.7.2. Se leen todos los nombres hasta que se terminen los mismos.

B.7.3. Si no terminó bien de leer todos los nombres de los archivos manda un mensaje de error al administrador.

B.7.4. Crea un mensaje que contenga la información antes mencionada y se envía ya sea vía correo electrónico o por medio del sistema en una pantalla

Módulo Administración de respaldos (C).

C. Este módulo deberá controlar y registrar todos los movimientos referentes a los respaldos de información de las bases de datos de cada servidor que integra el SIAE, estas tareas serán desde la creación del respaldo, compactación y transferencia hacia un servidor seguro.

C.1. Verificar el tipo de respaldo. En este proceso hay que decidir entre los diferentes tipos de respaldos, así como los diferentes medios para realizar el mismo.

C.1.1. El sistema recibe la solicitud de respaldar la información y se procesa y valida si esta en condiciones de realizar la tarea.

C.1.2. El sistema deberá dar las posibles opciones de los diferentes tipos de respaldos, como podrían ser: respaldos completos de datos y respaldos parciales de *log*.

C.1.3. Si el respaldo solicitado no está dentro de los tipos previamente definidos entonces mandar un mensaje de error al administrador.

C.1.4. Si el respaldo es completa de datos entonces hay que preparar el medio y verificar la existencia de cinta adecuada en tamaño para el tipo de respaldo.

C.1.5. Si el respaldo es parcial de *log* debemos verificar que el espacio de disco sea el adecuado para realizar las diferentes tareas que los procesos subsecuentes solicitan.

C.1.6. Una vez que se verifica lo anterior, se pasa a ver si el servidor de respaldo esta activo y hay comunicación entre este y el servidor de base de datos. Ya terminados todas las verificaciones anteriores se genera el respaldo desde el servidor de base de datos.

C.2. Realiza respaldo según el tipo. En este proceso ya definidos algunos puntos y verificado todo lo anterior se procede a realizar el respaldo.

C.2.1. Llega la solicitud ya definiendo y validando que el tipo solicitado es adecuado al sistema SIAE.

C.2.2. Se prepara la cinta o el espacio en disco para realizar el tipo de respaldo que se requiere.

C.2.3. Si hay algún problema con respecto a la cinta, al servidor de respaldos o al espacio en disco manda un error.

C.2.4. Si todo va bien identifica en que servidor esta las base de datos solicitada y hacer una relación para realizar los respaldos.

C.2.5. Respalda sólo páginas de datos y las manda al medio predefinido (disco o cinta), así hasta terminar con todas las bases de datos que se solicitaron.

C.2.6. Se empieza a generar un reporte que posteriormente se mandará al administrador.

C.3. Mensaje de error. Este proceso contendrá una función que controle los mensajes de error que hayan ocurrido en cualquiera de los pasos anteriores y subsecuentes, así como los mecanismos para avisar al administrador de que ha ocurrido un problema con esta tarea.

C.4. Compactar Archivos: en este proceso lo único que se hará es una vez que tengan los archivos del respaldo (si fue el caso) correspondientes entonces se compacta el archivo para que ocupe el menor espacio posible.

C.4.1. Llegada del nombre del archivo del reporte, el sistema estará esperando que llegue el nombre del archivo para poder compactar el archivo.

C.4.2. Compacta el archivo con la herramienta estándar compress de UNIX que nos da rangos aceptables de compactación (al rededor de 36%) y nos garantiza que en cualquier otro lugar pueda descompactarlas y manejar la información, y así nos mantenemos hasta haber terminado con cada uno de los archivos del respaldo que pudieran ser generados por las tareas anteriores.

C.4.3. Si la tarea no termina bien por algún problema en la compactación manda error al administrador.

C.4.4. Cuando ya haya terminado adecuadamente la tarea se procede a generar un reporte de toda la tarea en su conjunto englobando todos los pasos anteriores.

C.5. Enviar archivos de respaldos compactados a un servidor seguro: cuando ya se tenga concluida la tarea de compactar todos los archivos de respaldos generados, se procede a transferir los archivos compactados al servidor seguro.

C.5.1. Llega el nombre del archivo del archivo de respaldo ya campactado y se abre un canal encriptado para su transferencia hacia un servidor previamente definido el cual garantiza que los respaldos se tendrán en calidad de respaldo para su posterior transferencia al el servidor espejo en otra maquina.

C.5.2. Se toma el nombre del archivo y se transfiere con un protocolo seguro y confiable, y se mantiene este paso hasta haber terminado con todos los archivos antes compactados.

C.5.3. Si no termino adecuadamente la tarea genera un mensaje de error al administrador, y si termino bien termina la función

C.6. Avisar que se realizo la tarea: en éste proceso lo único que se esta haciendo es una recopilación del estatus de la tarea para reportarlo al administrador.

C.6.1. Llega a éste proceso los diferentes nombres de los archivos del respaldo que se generaron.

C.6.2. Se leen todos los nombres hasta que se terminen los mismos.

C.6.3. Si no terminó bien de leer todos los nombres de los archivos manda un mensaje de error al administrador.

C.6.4. Crea un mensaje que contenga la información antes mencionada y se envía ya sea vía correo electrónico o por medio del sistema en una pantalla

Módulo Crear y configurar las bases de datos (D).

D. Este módulo se controla la creación y configuración de bases de datos en los diferentes servidores, ya sea en su tamaño o en algún parámetro que se configure para su mejor rendimiento y desempeño.

D.1. Verificar el tipo de tarea: en éste proceso se valida el tipo de tarea que se puede realizar en este módulo, ya sea crear una base de datos o cambiar la configuración a una base de datos ya existente.

D.1.1. De manera automática llega la solicitud y por esta solicitud llega también el nombre de la base de datos con la cual se trabajará.

D.1.2. Preguntamos qué tipo de tarea es y definimos los pasos posteriores indicando ya sea una creación de base de datos o la configuración de la misma.

D.1.3. Si no es ninguna de las tareas antes mencionadas entonces manda un error al administrador.

D.2. Crear una base de datos. En este proceso es cuando se crean las bases de datos del SIAE con sus particularidades y estructuras internas, así como también los procedimientos almacenados, permisos, etc.

D.2.1. Llega la solicitud y es acompañada por el nombre de la base de datos que se va a crear, del mismo modo deben ser datos el tamaño en datos y en *log*.

D.2.2. Se verifica que no exista la base de datos en cuestión, así como que exista el espacio pertinente en los diferentes dispositivos que contenga el servidor de base de datos.

D.2.3. Si cualquier paso de los anteriores o subsecuentes genera un error entonces manda un mensaje de error al administrador.

D.2.4. Se obtiene una relación de las bases de datos existentes en el servidor en el cual se creará la base de datos.

D.2.5. También se requiere una relación detallada de los dispositivos con los que cuenta el servidor en cuestión y los espacios en los mismos para poder crear la base.

D.2.6. Se cargan a memoria el *script* que contiene toda la información de los objetos (tablas, vistas, índices, etc.) que se requieren el sistema SIAE.

D.2.7. También se leen los diferentes *scripts* que contengan la creación de los diferentes procedimientos almacenados que el SIAE requiere.

D.2.8. Una vez que se tiene toda esta información se crea la base de datos en el lugar y con las características necesarias para el SIAE, si ocurre algún error se manda un mensaje de error.

D.3. Mensaje de error. Este proceso contendrá una función que controle los mensajes de error que hayan ocurrido en cualquiera de los pasos anteriores y subsecuentes, así como los mecanismos para avisar al administrador de que ha ocurrido un problema con ésta tarea.

D.4. Configurar una base de datos. En este proceso se tiene como tarea cambiar algún parámetro de configuración que se requiera para el correcto funcionamiento o el mejor rendimiento en las tareas particulares del SIAE:

D.4.1. Llega la solicitud y es acompañada por el nombre de la base de datos que se va a modificar, del mismo modo deben llegar los parámetros que se van a cambiar.

D.4.2. Se verifica que no exista la base de datos en cuestión, así como que exista el espacio pertinente en los diferentes dispositivos que contenga el servidor de base de datos y/o exista el parámetro que se va a cambiar.

D.4.3. Si cualquier paso de los anteriores o subsecuentes genera un error entonces manda un mensaje de error al administrador.

D.4.4. Se obtiene una relación de las bases de datos existentes así como definiciones en tamaño y sus características dentro del servidor de base de datos.

D.4.5. También se requiere una relación detallada de los parámetros actuales con sus diferentes valores configurados actualmente.

D.4.6. Se procede a cambiar el o los parámetros solicitados, si se genera algún error se envía un mensaje al administrador, si todo sale bien se crea un reporte de lo que se realizó

D.5. Generar el reporte. En este proceso se recompila el estatus de las diferentes tareas y se genera un reporte conteniendo como se realizó dicha tarea.

D.5.1. Llega información referente a la nueva base de datos y/o la nueva configuración que se tiene para la base en cuestión.

D.5.2. Se recompila información de cada paso que se realizó en el módulo y se crea un mensaje que se enviará posteriormente.

D.5.3. Si ocurrió algún problema en este paso genera un mensaje de error y envíalo al administrador.

D.6. Avisar que se realizo la tarea: en éste proceso lo único que se esta haciendo es una recopilación del estatus de la tarea para reportarlo al administrador.

D.6.1. Llegan a este proceso los diferentes nombres de los archivos del respaldo que se generaron.

D.6.2. Se leen todos los nombres hasta que se terminen los mismos.

D.6.3. Si no terminó bien de leer todos los nombres de los archivos manda un mensaje de error al administrador.

D.6.4. Crea un mensaje que contenga la información antes mencionada y se envía ya sea vía correo electrónico o por medio del sistema en una pantalla

3.4 Lenguajes

La construcción del sistema puede ser escrita en varios lenguajes de programación. La selección depende de lo que se quiera realizar porque los diferentes lenguajes pueden ser especializados para diferentes propósitos.

Los lenguajes más primitivos fueron los lenguajes de máquina. Esto, ya que el hardware se desarrolló antes del software, y además cualquier software finalmente tiene que expresarse en el lenguaje que maneja el hardware.

La programación en esos momentos era sumamente tediosa, pues el programador tenía que "bajarse" al nivel de la máquina y decirle, paso a paso, cada punto de la tarea que tenía que realizar. Además, debía expresarlo en forma numérica; y por supuesto, este proceso era propenso a errores, con lo que la productividad del programador era muy limitada. Sin embargo, hay que recordar que en estos momentos, simplemente aún no existía alternativa.

Con el desarrollo en los 50s y 60s de algoritmos de más elevado nivel, y el aumento de poder del hardware, empezaron a entrar al uso de computadoras científicos de otras ramas; ellos conocían mucho de Física, Química y otras ramas similares, pero no de Computación, y por supuesto, les era sumamente complicado trabajar con lenguaje Ensamblador en vez de fórmulas. Así, nació el concepto de Lenguaje de Alto Nivel, con el primer compilador de FORTRAN (FORmula TRANslation), que, como su nombre indica, inició como un "simple" esfuerzo de traducir un lenguaje de fórmulas, al lenguaje ensamblador y por consiguiente al lenguaje de máquina. A partir de FORTRAN, se han desarrollado innumerables lenguajes, que siguen el mismo concepto: buscar la mayor abstracción posible, y facilitar la vida al programador, aumentando la productividad, encargándose los compiladores o intérpretes de traducir el lenguaje de alto nivel, al lenguaje de computadora.

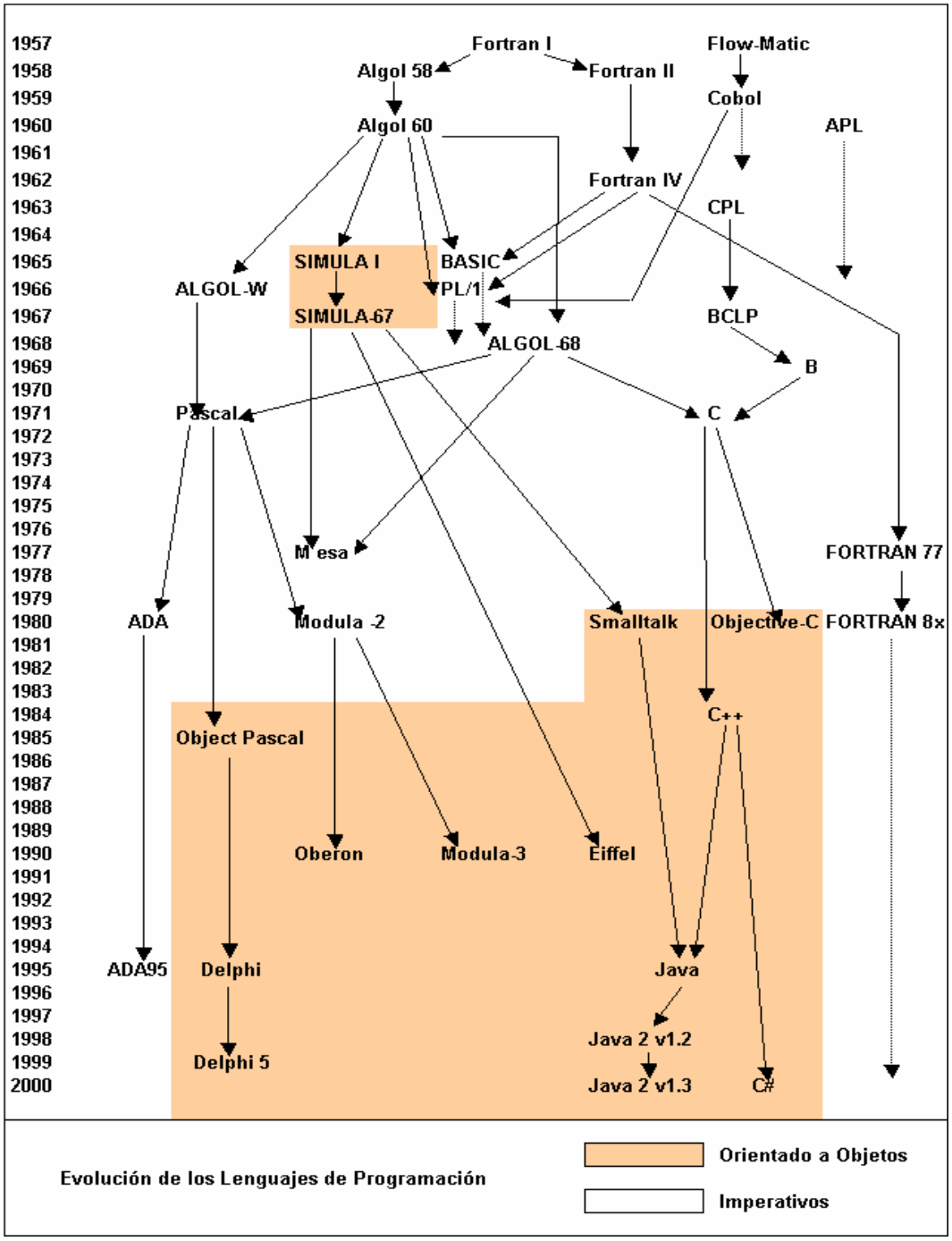


Figura 4.1 Evolución de los Lenguajes de

Algunos de los lenguajes de alto nivel actuales son:

Lenguaje C

El lenguaje C reúne características de programación intermedia entre los lenguajes ensambladores y los lenguajes de alto nivel; con gran poderío basado en sus operaciones a nivel de bits (propias de ensambladores) y la mayoría de los elementos de la programación estructurada de los lenguajes de alto nivel, por lo que resulta ser el lenguaje preferido para el desarrollo de software de sistemas y aplicaciones profesionales de la programación de computadoras.

En 1970 Ken Thompson de los laboratorios Bell se había propuesto desarrollar un compilador para el lenguaje Fortran que corría en la primera versión del sistema operativo UNIX tomando como referencia el lenguaje BCPL; el resultado fue el lenguaje B (orientado a palabras) que resultó adecuado para la programación de software de sistemas.

Este lenguaje tuvo la desventaja de producir programas relativamente lentos. En 1971 Dennis Ritchie, con base en el lenguaje B desarrollo NB que luego cambio su nombre por C; en un principio sirvió para mejorar el sistema UNIX por lo que se le considera su lenguaje nativo.

Su diseño incluyó una sintaxis simplificada, la aritmética de direcciones de memoria (permite al programador manipular bits, bytes y direcciones de memoria) y el concepto de apuntador; además, al ser diseñado para mejorar el software de sistemas, se buscó que generase códigos eficientes y una portabilidad total, es decir el que pudiese correr en cualquier máquina. Logrados los objetivos anteriores, C se convirtió en el lenguaje preferido de los programadores profesionales.

En 1980 Bjarne Stroustrup de los laboratorios Bell de Murray Hill, New Jersey, inspirado en el lenguaje Simula67 adicionó las características de la programación orientada a objetos (incluyendo la ventaja de una biblioteca de funciones orientada a objetos) y lo denomino C con clases.

Para 1983 dicha denominación cambio a la de C++. Con este nuevo enfoque surge la nueva metodología que aumenta las posibilidades de la programación bajo nuevos conceptos.

Java

El lenguaje de programación Java, fue diseñado por la compañía Sun Microsystems Inc, con el propósito de crear un lenguaje que pudiera funcionar en redes computacionales heterogéneas (redes de computadoras formadas por más de un tipo de computadora, ya sean PC, MAC's, estaciones de trabajo, etc.), y que fuera independiente de la plataforma en la que se vaya a ejecutar. Esto significa que un programa de Java puede ejecutarse en cualquier máquina o plataforma.

En cuanto al lenguaje en sí, es un descendiente de C++, aunque intenta corregir muchos de sus defectos. Está totalmente orientado a objetos e incluye numerosas bibliotecas estándar.

Pascal

Pascal es un lenguaje de programación de alto nivel de propósito general; esto es, se puede utilizar para escribir programas para fines científicos y comerciales.

El lenguaje de programación Pascal fue desarrollado por el profesor Niklaus (Nicolás) Wirth en Zurich, Suiza, al final de los años 60 y principios de los 70. Wirth diseñó este lenguaje para que fuese un buen primer lenguaje de programación para personas comenzando a aprender a programar. Pascal tiene un número relativamente pequeño de conceptos para aprender y dominar. Su diseño facilita escribir programas usando un estilo que está generalmente aceptado como práctica estándar de programación buena. Otra de las metas del diseño de Wirth era la implementación fácil. Él diseñó un lenguaje para el cual fuese fácil escribir un compilador para un nuevo tipo de computadora.

3.4.1 Selección del lenguaje de programación

De los lenguajes anteriormente descritos podemos observar distintas características particulares, el lenguaje que presenta mayores ventajas es el lenguaje C que se adapta a las necesidades que se requieren para el desarrollo del sistema.

El lenguaje C contiene estructuras de programación de alto nivel, y la facilidad de usar bibliotecas que también son características de alto nivel; Sybase diseñó bibliotecas llamadas DB-Library para dicho lenguaje, que es una colección de rutinas y macros que permiten a las aplicaciones desarrolladas en C interactuar con el servidor Sybase.

La ventaja que se tiene con las bibliotecas DB-Library es que se puede establecer uno o varios canales de comunicación entre el cliente y el servidor, para enviar peticiones y recibir respuestas.

Las peticiones que se envían son sentencias SQL como son select, insert, update y delete, además de ejecutar procedimientos almacenados. También contiene rutinas para el manejo de condiciones de error, mejorar la conversión de datos, y proveer una variedad de información acerca de la interacción de las aplicaciones con el servidor.

Las bibliotecas DB-Library para C contienen varios archivos de cabecera (include) que definen estructuras y valores usados por las rutinas.

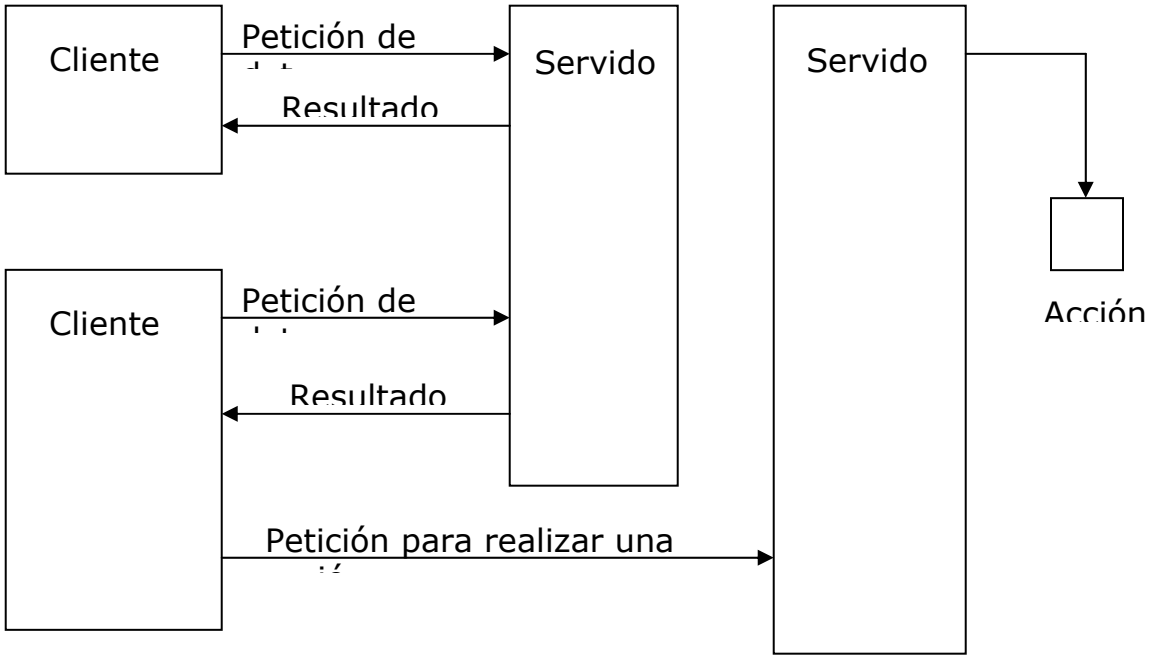


Figura 4.2 Comunicación cliente/servidor con DB-Library

Cuando se escribe un programa en C con las bibliotecas DB-Library, se realizan llamadas a las rutinas de las bibliotecas DB-Library, que permiten la conexión con el servidor o servidores, envío de comandos, procesamiento de resultados.

Cuando se programa con DB-Library usualmente se realizan los siguientes pasos básicos:

1. Conexión al servidor.
2. Envió de sentencias SQL al servidor.

3. Procesamiento de resultados, los resultados pueden ser manipulados dentro del programa.
4. Manejo de errores y mensajes del servidor.
5. Cerrar la conexión con el servidor.

Otras bibliotecas con características similares a las bibliotecas DB-Library son las la bibliotecas GTK+.

GTK+ significa GIMP Toolkit (conjunto de herramientas GIMP) y GIMP quiere decir Graphical Image Manipulation (Manipulación de Imágenes Gráficas). GIMP Toolkit es una biblioteca utilizada para desarrollar aplicaciones que tengan una interfaz gráfica de usuario (GUI, Graphical User Interface). Esta biblioteca se utiliza ampliamente hoy en día para desarrollar aplicaciones GUI para Linux. GIMP fue desarrollado con la biblioteca GTK+ y proporciona un ejemplo de aplicación con interfaz gráfica desarrollada profesionalmente. GIMP es una aplicación gratuita de código fuente de libre disposición que puede obtenerse en www.gimp.org. La biblioteca GTK+ puede también utilizarse para desarrollar aplicaciones en otras plataformas para las que se disponga de una versión de dicha biblioteca. GTK+ es una biblioteca orientada a objetos escrita en lenguaje C que puede utilizarse en aplicaciones escritas en diversos lenguajes. La lista de lenguajes permitidos incluye Ada95, C++, Eiffel, Objective C, Pascal, Perl, Python y muchos otros.

GTK es un API (Aplication Program Interface: es el conjunto de rutinas del sistema que se pueden usar en un programa para la gestión de entrada/salida, gestión de archivos, etc.) orientado a objetos. Aunque está completamente escrita en C, soporta la idea de clases y funciones de respuesta (es decir punteros a funciones).

Existe un componente más, llamado *glib* que proporciona un sustituto a algunas llamadas que podríamos denominar estándar.

La biblioteca *glib* es una colección de funciones comunes, ampliamente utilizadas por GTK+. Las listas enlazadas, los árboles, los mecanismos de tratamiento de errores, la gestión de memoria y los cronómetros son sólo una parte del contenido de esta biblioteca.

GTK+ es la más moderna biblioteca de interfaces gráficas de usuario, que utiliza el tipo de licencia para público en general; por tanto, es gratuita tanto para el desarrollo comercial como para el no comercial. GTK+ es, así mismo, un conjunto de herramientas multiplataforma que puede utilizarse en muchas máquinas UNIX, así como en Microsoft Windows. Con el soporte de algunas de las distribuciones más

extendidas de software, GTK+ ha evolucionado hasta transformarse en un conjunto de herramientas muy adecuado para el desarrollo de aplicaciones con interfaz gráfica de usuario.

Al igual que la mayor parte de las herramientas modernas para interfaces gráficas de usuario, GTK+ es una herramienta conducida por sucesos. La pantalla se construye mediante *widgets* (ventanas, cuadros combinados, cuadros de texto, botones) y se establecen una serie de *retrollamadas* para estos *widgets* con el fin de realizar el procesamiento basándose en señales, que usualmente son sucesos de teclado o de ratón. Cuando una *retrollamada* recibe la notificación de una señal, la aplicación responde a dichas señales con algún tipo de procesamiento. El concepto de señal es similar al de suceso en Java o Windows.

Además GTK+ utiliza GDK (GIMP Drawing Kit), que es un envoltorio alrededor de Xlib. GTK+ invoca GDK para todo lo que se relacione con la visualización de los *widgets*. GDK es una interfaz de programación de aplicaciones (Application Programming Interface, API) dependiente de la plataforma y que se sitúa por encima de la API gráfica nativa (Xlib, win32), proporcionando una API gráfica. Puesto que GDK está situado como un nivel entre Xlib y GTK+, puede portarse GTK+ a otros sistemas operativos con solo cambiar la biblioteca GDK. Tanto GTK+ como GDK utilizan ampliamente la biblioteca GLIB, que proporciona funciones para gestionar muchos tipos de datos comunes, como listas, árboles y cadenas, así como funciones de asignación de memoria y rutinas de tratamiento de errores.

Debido a las características que reúne esta biblioteca de programación es apto para desarrollar el ambiente gráfico del sistema, permite adaptar el código desarrollado en lenguaje C el cual establece la comunicación entre el cliente y el servidor.

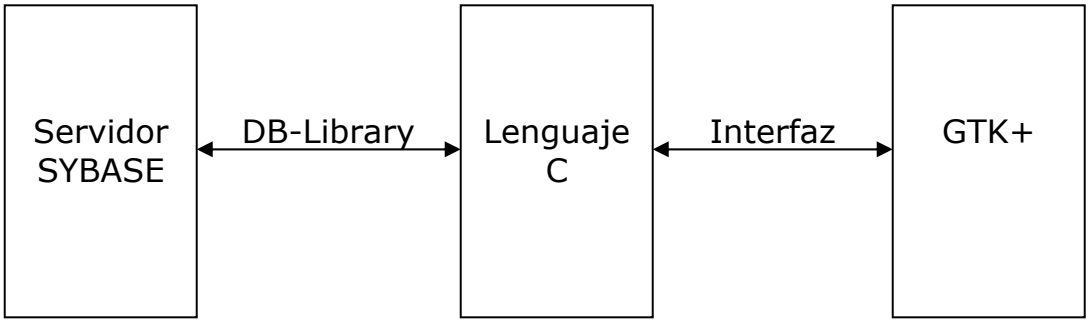


Figura 4.3 Relación de GTK+ con Lenguaje C y DB-Library

3.5 Herramientas

3.5.1 Sybase

En 1984 se fundó Sybase, en un mercado donde Oracle, Ingress, Informix y DB2 eran productos RDBMS. Sybase es un Sistema Manejador de Bases de Datos Relacional (RDBMS), Sybase fue el primero en incorporar la arquitectura cliente/servidor dentro de una base de datos relacional y el primero en promover el concepto de un nuevo camino a construir sistemas.

-Plataformas soportadas:

- Compaq Tru64
- HP/UX (32 y 64 bit)
- IBM AIX (32 y 64 bit)
- Microsoft Windows NT
- Microsoft Windows 98 (solo clientes)
- SGI IRIX (32 y 64 bit)
- Sun Solaris (32 y 64 bit)
- Linux

-Características:

	Número
Bases de datos en Sybase	32,767
Tablas en un query	50
Logins por servidor	2,147,516,416
Usuarios por base de datos	2,146,484,223
Grupos por base de datos	1,032,193
Columnas por tabla	1024
	Tamaño
Servidor	8 terabytes
Base de Datos	4 terabytes
Paginas	2K, 4K, 8K, 16K

- Operaciones dinámicas de respaldo continuo.
- Triggers o disparadores y procedimientos almacenados en la base de datos.
- Escalabilidad y flexibilidad.
- Ejecución de reglas del negocio e integridad referencial.
- Soporte a tipo de datos definidos por el usuario.
- Soporte a terceras empresas con productos front-end.

- Conectividad con diversos lenguajes de programación a través de librerías.
- Consultas en línea.
- Procesador de transacciones.
- Arquitectura cliente/servidor.
- Lenguaje: structured query language.

Sybase Transact-SQL ha sido extendido para minimizar la complejidad de programación de alguna tarea deseada. Transact-SQL contiene muchas versiones comerciales de SQL, y añade características como lenguaje de control de flujo, triggers o disparadores, reglas y defaults.

Lenguaje de control de flujo puede ser utilizado como parte de una sentencia SQL o un conjunto de instrucciones. Algunas sentencias son: begin… end, break, continue, declare, goto, if-else, return y while. Existen también técnicas especiales para el manejo de errores que están disponibles al programador de Transact-SQL.

3.5.2 ISQL

ISQL (Interactive SQL) es una interfaz interactiva a las bases de datos del gestor Sybase SQL. Permite ejecutar comandos del lenguaje de programación Transact-SQL, tales como consultas de bases de datos, comandos de mantenimiento y administración. ISQL es un programa que se encuentra en el cliente.

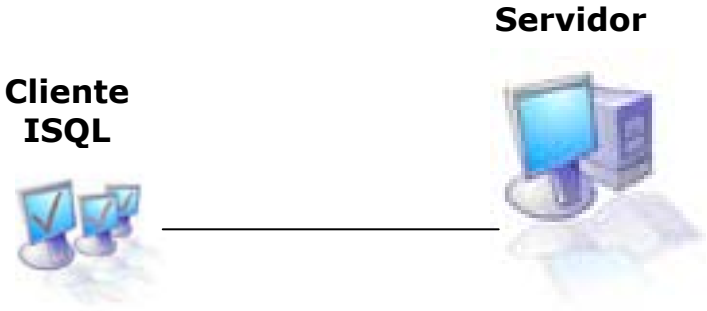


Figura 4.4 ISQL como cliente

3.5.3 SQSH

SQSH (SQL *Shel*l), es una herramienta que mejora el funcionamiento del programa ISQL provisto por Sybase. Este programa surge después de años de trabajo y de susceptibilidades en el programa ISQL, intenta hacer un trabajo mas rápido y eficiente.

SQSH es más que un bonito prompt, intenta proporcionar la funcionalidad de un buen *shell*, por ejemplo el uso de variables, redireccionamiento de información, control de trabajos, historial de comandos.

SQSH fue diseñado con portabilidad y se ha compilado con éxito en las principales plataformas UNIX soportadas por Sybase, por ejemplo Linux, FreeBSD, HP-UX, AIX, IRIX, SunOS, Solaris, Dynix, OSF/1, DEC Unix, SCO, NeXT, DG/UX y CP/M. SQSH también se ha compilado para plataformas de Microsoft Windows NT.

3.5.4 Compilador GCC

Las siglas GCC significan *GNU Compiler Collection* (Colección de compiladores GNU). Antes eran siglas de *GNU C Compiler* (Compilador C GNU). Como su nombre indica es una colección de compiladores y admite diversos lenguajes: C, C++, Objetive C, Chill, Fortran, y Java.

El compilador se distribuye bajo la licencia GPL (General Public License) lo que lo hace de libre distribución: se pueden hacer copias de él y regalarlas o venderlas siempre que se incluya el código fuente (o se indique cómo conseguirlo) y se mantenga la licencia.

Existen versiones para prácticamente todos los sistemas operativos. Viene incluido en la mayoría (si no en todas) las distribuciones de GNU/Linux. La versión DOS de este compilador es el DJGPP.

En el desarrollo de este compilador participan cientos de voluntarios de todo el mundo.

De las herramientas anteriormente descritas, empleamos a Sybase como manejador de base de datos, y el cliente predeterminado será SQSH por la versatilidad en las funciones que presenta, para compilar los programas hacemos uso del compilador GCC porque reúne características que permiten compilar programas desarrollados en lenguaje C y GTK+, además de que se distribuye de forma gratuita.

4. CONSTRUCCION DEL SISTEMA DE ADMINISTRACION DE BASES DE DATOS

Para el desarrollo contamos con una máquina SUNBlade 100 con un procesador de 500-MHz UltraSPARC-IIe, 20GB de disco duro y un sistema operativo SunOS 5.8 en cual se instaló el compilador GCC, las bibliotecas GTK+ y las bibliotecas DB-Library, el lenguaje C es, el lenguaje nativo de los sistemas UNIX por lo tanto ya viene incluido dentro del sistema operativo.

El servidor de bases de datos esta instalado en una máquina Sun Sparc Enterprise de 4 procesadores, memoria de 1 Gb, y un disco duro de 120 Gb que se encuentra en otro segmento de la red.

En la figura 4.1 se muestra como esta implementada la arquitectura general del sistema.

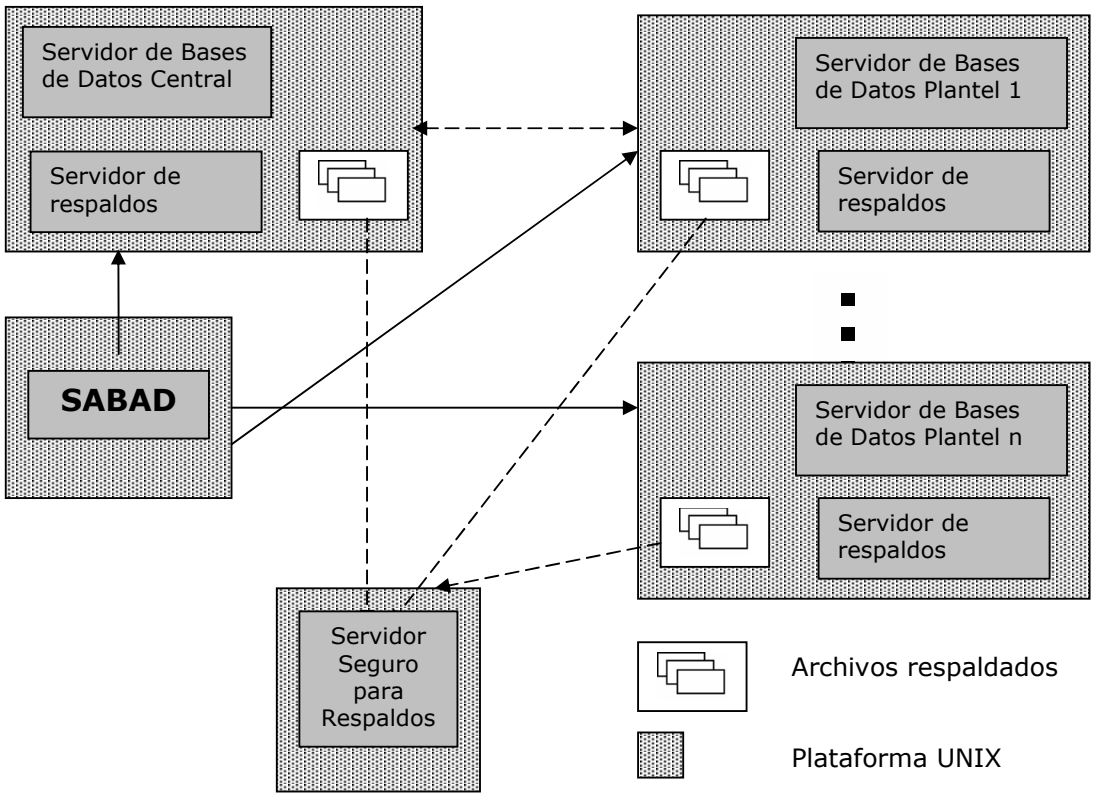


Figura 4.1 Arquitectura general del sistema.

El sistema de administración de bases de datos (SABAD) controlará desde una máquina remota todos los procesos relacionados a la administración de la base de datos central y cada una de las bases de datos de los planteles así como sus respaldos, los respaldos se realizan en cada máquina y se almacenan

como archivos, estos archivos se deben de compactar y se envían a un servidor de alta seguridad.

El proceso de programación consiste de traducir el diseño del sistema a código fuente.

4.1 Especificación de estándares de bibliotecas de programación

A continuación se describen las bibliotecas más usadas dentro de la programación del Sistema de Administración de Bases de Datos (SABAD)

4.1.1 Bibliotecas DB-Library

db12hour Determina si el idioma especificado usa el formato de 12 horas o 24 horas.

Sintaxis

```
DBBOOL db12hour(dbproc, language)
DBPROCESS *dbproc;
char *language;
```

dbadata Devuelve un apuntador a los datos para una columna compute.

Sintaxis

```
BYTE *dbadata(dbproc, computeid, column)
DBPROCESS *dbproc;
int computeid;
int column;
```

dbadlen Devuelve la longitud real de los datos para una columna compute.

Sintaxis

```
DBINT dbadlen(dbproc, computeid, column)
DBPROCESS *dbproc;
int computeid;
int column;
```

dbaltbind Asigna una columna compute a una variable del programa.

Sintaxis

```
RETCODE dbaltbind(dbproc, computeid, column, vartype, varlen,
varaddr)
DBPROCESS *dbproc;
int computeid;
int column;
int vartype;
DBINT varlen;
BYTE *varaddr;
```

dbaltbind_ps Asigna una columna compute a una variable del programa, con la precisión y escala que soporta para tipos de datos numeric y decimal.

Sintaxis

```
RETCODE dbaltbind_ps(dbproc, computeid, column, vartype, varlen,
varaddr, typeinfo)
DBPROCESS *dbproc;
```

```
int vartype;
DBINT varlen;
BYTE *varaddr;
DBTYPEINFO *typeinfo;
```

dbaltcolid Devuelve el id de la columna para una columna compute.

Sintaxis

```
int dbaltcolid(dbproc, computeid, column)
DBPROCESS *dbproc;
int computeid;
int column;
```

dbaltlen Devuelve la longitud máxima de los datos para una particular columna compute.

Sintaxis

```
DBINT dbaltlen(dbproc, computeid, column)
DBPROCESS *dbproc;
int computeid;
int column;
```

dbaltop Devuelve el tipo de operador agregado para una particular columna compute.

Sintaxis

```
int dbaltop(dbproc, computeid, column)
DBPROCESS *dbproc;
int computeid;
int column;
```

dbalttype Devuelve el tipo de datos para una columna compute.

Sintaxis

```
int dbalttype(dbproc, computeid, column)
DBPROCESS *dbproc;
int computeid;
int column;
```

dbaltutype Devuelve el tipo de datos usuario definido para una columna compute.

Sintaxis

```
DBINT dbaltutype(dbproc, computeid, column)
DBPROCESS *dbproc;
int computeid;
int column;
```

dbanullbind Asocia un variable con una columna-renglón de un compute.

Sintaxis

```
RETCODE dbanullbind(dbproc, computeid, column,indicator)
DBPROCESS *dbproc;
int computeid;
int column;
DBINT *indicator;
```

dbbind Asigna una columna de resultados a una variable del programa.

Sintaxis

```
RETCODE dbbind(dbproc, column, vartype, varlen,
varaddr)
DBPROCESS *dbproc;
int column;
int vartype;
```

```
DBINT varlen;
BYTE *varaddr;
```

dbbind_ps Asigna una columna de resultados a una variable del programa, con la precisión y escala soportada por el tipo de datos numeric y decimal.

Sintaxis

```
RETCODE dbbind_ps(dbproc, column, vartype, varlen,
varaddr, typeinfo)
DBPROCESS *dbproc;
DBINT varlen;
BYTE *varaddr;
DBTYPEINFO *typeinfo;
```

dbbufsize Devuelve el tamaño de un renglón del buffer del DBPROCESS.

Sintaxis

```
int dbbufsize(dbproc)
DBPROCESS *dbproc;
```

dbbylist Devuelve el numero de elementos listados en una clausula by para un renglón compute.

Sintaxis

```
BYTE *dbbylist(dbproc, computeid, size)
DBPROCESS *dbproc;
int computeid;
int *size;
```

dbcancel Cancela el lote de comandos actual.

Sintaxis

```
RETCODE dbcancel(dbproc)
DBPROCESS *dbproc;
```

dbcancel_a (VMS sólo) Cancela el lote de comandos actual asincrónicamente.

Sintaxis

```
RETCODE dbcancel_a(dbproc, final_result, ast_proc,
ast_param)
DBPROCESS *dbproc;
RETCODE *final_result;
void (*ast_proc)();
BYTE *ast_param;
```

dbcanquery Cancela cualquier fila pendiente del query recientemente ejecutado.

Sintaxis

```
RETCODE dbcanquery(dbproc)
DBPROCESS *dbproc;
```

dbcanquery_a (VMS sólo) Cancela cualquier fila pendiente del query recientemente ejecutado asincrónicamente.

Sintaxis

```
RETCODE dbcanquery_a(dbproc, final_result, ast_proc, ast_param)
DBPROCESS *dbproc;
RETCODE *final_result;
void (*ast_proc)();
BYTE *ast_param;
```

dbchange Determina si un lote de comandos se ha cambiado de la base de datos actual.

Sintaxis

```
char *dbchange(dbproc)
DBPROCESS *dbproc;
```

dbclose Cierra y libera una estructura de DBPROCESS.

Sintaxis

```
void dbclose(dbproc)
DBPROCESS *dbproc;
```

dbclrbuf Limpia de renglones un buffer.

Sintaxis

```
void dbclrbuf(dbproc, n)
DBPROCESS* dbproc;
DBINT n;
```

dbclropt Limpia las opciones asignadas con dbsetopt.

Sintaxis

```
RETCODE dbclropt(dbproc, option, param)
DBPROCESS *dbproc;int option;
char* param;
```

dbcmd Agrega texto al buffer de comandos de un DBPROCESS.

Sintaxis

```
RETCODE dbcmd(dbproc, cmdstring)
DBPROCESS *dbproc;
char *cmdstring;
```

DBCMDROW Determina si el comando actual puede devolver renglones.

Sintaxis

```
RETCODE DBCMDROW(dbproc)
DBPROCESS *dbproc;
```

dbcolbrowse Determina si la fuente de una columna de resultados es una columna actualizable vía los medios de las DB-library en modo browse.

Sintaxis

```
DBBOOL dbcolbrowse(dbproc, colnum)
DBPROCESS *dbproc;
int colnum;
```

dbcollen Devuelve la longitud máxima de los datos en una columna de resultados.

Sintaxis

```
DBINT dbcollen(dbproc, column)
DBPROCESS *dbproc;
int column;
```

dbcolname Devuelve el nombre de una columna de resultados.

Sintaxis

```
char *dbcolname(dbproc, column)
DBPROCESS *dbproc;
int column;
```

dbcolsource Devuelve un apuntador al nombre de la columna de la base de datos de la columna de resultados que fue

derivada.

Sintaxis

```
char *dbcolsource(dbproc, colnum)
DBPROCESS *dbproc;
int colnum;
```

dbcoltype Devuelve el tipo de datos para una columna de resultados.

Sintaxis

```
int dbcoltype(dbproc, column)
DBPROCESS *dbproc;
int column;
```

dbcoltypeinfo Devuelve información de la precisión y escala para una columna de resultados para tipo numeric o decimal.

Sintaxis

```
DBTYPEINFO * dbcoltypeinfo(dbproc, column)
DBPROCESS *dbproc;
int column;
```

dbcolutype Devuelve el tipo de datos definido por un usuario para una columna de resultados.

Sintaxis

```
DBINT dbcolutype(dbproc, column)
DBPROCESS *dbproc;
int column;
```

dbconvert Convierte los datos de un tipo de datos a otro.

Sintaxis

```
DBINT dbconvert(dbproc, srctype, src, srclen,desttype, dest,
destlen)
DBPROCESS *dbproc;
int srctype;
BYTE *src;
DBINT srclen;
int desttype;
BYTE *dest;
DBINT destlen;
```

dbconvert_ps Convierte los datos de un tipo de datos a otro, con la precisión y escala soportada para el tipo de datos numeric y decimal.

Sintaxis

```
DBINT dbconvert_ps(dbproc, srctype, src, srclen,desttype, dest,
destlen, typeinfo)
DBPROCESS *dbproc;
int srctype;
BYTE *src;DBINT srclen;
int desttype;
BYTE *dest;
DBINT destlen;
DBTYPEINFO *typeinfo;
```

DBCOUNT Devuelve el número de filas afectado por un comando Transact-SQL.

Sintaxis

```
DBINT DBCOUNT(dbproc)
DBPROCESS *dbproc;
```

DBCURCMD Devuelve el número de comandos actual.

Sintaxis

```
int DBCURCMD(dbproc)
DBPROCESS *dbproc;
```

DBCURROW Devuelve el número de la fila que actualmente se esta leyéndo.

Sintaxis

```
DBINT DBCURROW(dbproc)
DBPROCESS *dbproc;
```

dbcursor Inserta, actualiza, borra, bloquea o refresca una fila en particular en el buffer.

Sintaxis

```
RETCODE dbcursor(hc, optype, bufno, table, values)
DBCURSOR *hc;
DBINT optype;
DBINT bufno;
BYTE *table;
BYTE *values
```

dbcursorbind Registra la información ligada sobre las columnas del cursor.

```
RETCODE dbcursorbind(hc, col, vartype, varlen,poutlen, pvaraddr,
typeinfo)
DBCURSOR *hc;
int col;
int vartype;
DBINT varlen;
DBINT *poutlen;
BYTE *pvaraddr;
DBTYPEINFO *typeinfo;
```

dbcursorclose Cierra el cursor asociado a la conexión y libera todos los datos que tomo el cursor.

Sintaxis

```
void dbcursorclose(hc)
DBCURSOR *hc;
```

dbcursorcolinfo Devuelve información de una columna en particular especificando el número de la columna del cursor abierto.

Sintaxis

```
RETCODE dbcursorcolinfo(hcursor, column, colname,
coltype, collen, usertype)
DBCURSOR *hcursor
DBINT column;
DBCHAR *colname;DBINT *coltype;
DBINT *collen;
DBINT *usertype;
```

dbcursorfetch Toma un bloque de renglones y los inserta en variables del programa declaradas por el usuario con el dbcursorbind.

Sintaxis

```
RETCODE dbcursorfetch(hc, fetchtype, rownum)
DBCURSOR *hc;
DBINT fetchtype;
DBINT rownum;
```

dbcursorinfo Devuelve el número de columnas y el número de filas en el keyset y si estos llegaron al final del conjunto de

resultados.

Sintaxis

```
RETCODE dbcursorinfo(hcursor, ncols, nrows);
DBCURSOR *hcursor;
DBINT *ncols
DBINT *nrows;
```

dbcursoropen Abre un cursor y especifica la opción de barrido, la opción de concurrencia y el tamaño del buffer (el número de filas que recupera en un solo fetch).

Sintaxis

```
DBCURSOR *dbcursoropen(dbproc, stmt, scrollopt,concuropt, nrows,
pstatus)
DBPROCESS *dbproc;
BYTE *stmt;
SHORT scrollopt;
SHORT concuropt;
USHORT nrows;
DBINT *pstatus
```

dbdata Devuelve un apuntador a los datos en una columna de resultados.

Sintaxis

```
BYTE *dbdata(dbproc, column)
DBPROCESS *dbproc;
int column;
```

dbdate4cmp Compara dos valores de DBDATETIME4.

Sintaxis

```
int dbdate4cmp(dbproc, d1, d2)
DBPROCESS *dbproc;
DBDATETIME4 *d1;
DBDATETIME4 *d2;
```

dbdate4zero Inicializa una variable DBDATETIME4 a Ene 1, 1900 12:00AM.

Sintaxis

```
RETCODE dbdate4zero(dbproc, dateptr)
DBPROCESS *dbproc;
DBDATETIME4 *dateptr;
```

dbdatechar Convierte un valor entero de un DBDATETIME en formato carácter.

Sintaxis

```
RETCODE dbdatechar(dbproc, charbuf, datepart, value)
DBPROCESS *dbproc;
char *charbuf;int datepart;
int value;
```

dbdatecmp Compara dos valores de DBDATETIME.

Sintaxis

```
int dbdatecmp(dbproc, d1, d2)
DBPROCESS *dbproc;
DBDATETIME *d1;
DBDATETIME *d2;
```

dbdatecrack Convierte un DBDATETIME recuperado de la máquina en formato accesible por el usuario.

Sintaxis

```
RETCODE dbdatecrack(dbproc, dateinfo, datetime)
DBPROCESS *dbproc;
DBDATEREC *dateinfo;
DBDATETIME *datetime;
```

dbdatename Convierte el componente especificado de una estructura de DBDATETIME en su valor correspondiente en carácter.

Sintaxis

```
int dbdatename(dbproc, charbuf, datepart, datetime)
DBPROCESS *dbproc;
char *charbuf;
int datepart;
DBDATETIME *datetime;
```

dbdateorder Devuelve el orden de componente de fecha para un idioma dado.

Sintaxis

```
char *dbdateorder(dbproc, language)
DBPROCESS *dbproc;
char *language;
```

dbdatepart Devuelve la parte especificada de un valor de DBDATETIME como un valor numérico.

Sintaxis

```
DBINT dbdatepart(dbproc, datepart, datetime)
DBPROCESS *dbproc;
int datepart;
DBDATETIME *datetime;
```

dbdatezero Inicializa un valor de DBDATETIME a Ene 1, 1900 12:00:00:000AM.

Sintaxis

```
RETCODE dbdatezero(dbproc, dateptr)
DBPROCESS *dbproc;
DBDATETIME *dateptr;
```

dbdatlen Devuelve la longitud de los datos en una columna de resultados.

Sintaxis

```
DBINT dbdatlen(dbproc, column)
DBPROCESS *dbproc;
int column;
```

dbdayname Determina el nombre de un día de la semana especificado en un idioma especificado.

Sintaxis

```
char *dbdayname(dbproc, language, daynum)
DBPROCESS *dbproc;char *language;
int daynum;
```

DBDEAD Determina si un DBPROCESS está muerto.

Sintaxis

```
DBBOOL DBDEAD(dbproc)
DBPROCESS *dbproc;
```

dberrhandle Instala la función del usuario para ocuparse de errores de las DB-library.

Sintaxis

```
int (*dberrhandle(handler))()
int (*handler)();
```

dbexit Cierra y libera las estructuras de todos los DBPROCESS, y limpia cualquier estructura inicializada por el dbinit.

Sintaxis

```
void dbexit()
```

dbfcmd Agrega el texto al buffer de comandos de un DBPROCESS utilizando parámetros de su programa en C.

Sintaxis

```
RETCODE dbfcmd(dbproc, cmdstring, args...)
DBPROCESS *dbproc;
char *cmdstring;
??? args...;
```

dbfirstrow Devuelve el número de la primera fila buffer de renglones.

Sintaxis

```
DBINT DBFIRSTROW(dbproc)
DBPROCESS *dbproc;
```

dbfreebuf Limpia el buffer de comandos.

Sintaxis

```
void dbfreebuf(dbproc)
DBPROCESS *dbproc;
```

dbfreequal Libera la memoria asignada por el dbqual.

Sintaxis

```
void dbfreequal(qualptr)
char *qualptr;
```

dbfreesort Libera una estructura de orden asignada por el dbloadsort.

Sintaxis

```
RETCODE dbfreesort(dbproc, sortorder)
DBPROCESS *dbproc;
DBSORTORDER *sortorder;
```

dbgetchar Devuelve un apuntador a un carácter en el buffer de comandos.

Sintaxis

```
char *dbgetchar(dbproc, n)
DBPROCESS *dbproc;
int n;
```

dbgetcharset Trae el nombre del juego de caracteres del cliente desde la estructura del DBPROCESS.

Sintaxis

```
char *dbgetcharset(dbproc)
DBPROCESS *dbproc;
```

dbgetloginfo Transfiere la información de un login en su estructura de DBPROCESS TDS (Tabular Data Stream) a una estructura de DBLOGINFO recientemente asignada.

Sintaxis

```
RETCODE dbgetloginfo(dbproc, loginfo)
DBPROCESS *dbproc;
DBLOGINFO **loginfo;
```

dbgetlusername Devuelve el nombre del usuario de una estructura de LOGINREC.

Sintaxis

```
int dbgetlusername(login, name_buffer, buffer_len)
LOGINREC *login;
BYTE *name_buffer;
int buffer_len;
```

dbgetmaxprocs Determina el número máximo actual de DBPROCESSes simultáneamente abierto.

Sintaxis

```
int dbgetmaxprocs()
```

dbgetnatlang Trae el idioma de la estructura de DBPROCESS.

Sintaxis

```
char* dbgetnatlang(dbproc)
DBPROCESS *dbproc;
```

dbgetoff Checa la existencia de una estructura de Transact-SQL en el buffer de comandos.

Sintaxis

```
int dbgetoff(dbproc, offtype, startfrom)
DBPROCESS *dbproc;
DBUSMALLINT offtype;int startfrom;
```

dbgetpacket Devuelve el tamaño del paquete TDS actualmente en uso.

Sintaxis

```
int dbgetpacket(dbproc)
DBPROCESS *dbproc;
```

dbgetrow Lee la fila especificada en el buffer de renglones.

Sintaxis

```
STATUS dbgetrow(dbproc, row)
DBPROCESS *dbproc;DBINT row;
```

DBGETTIME Devuelve el número de segundos que las DB-library esperará por una contestación del servidor a un comando de SQL.

Sintaxis

```
int DBGETTIME()
```

dbgetuserdata Devuelve un apuntador a los datos de usuario-asignados desde una estructura de DBPROCESS.

Sintaxis

```
BYTE *dbgetuserdata(dbproc)
DBPROCESS *dbproc;
```

dbhasretstat Determina si el comando actual o la llamada del procedimiento remoto generaron un número de estado de retorno.

Sintaxis

```
DBBOOL dbhasretstat(dbproc)
DBPROCESS *dbproc;
```

dbinit Inicializa las DB-library.

Sintaxis
```
RETCODE dbinit()
```

DBIORDESC (sólo UNIX y AOS/VS) Proporciona acceso del programa al descriptor de archivo de UNIX o AOS/VS usado por un DBPROCESS para leer datos que vienen del servidor.
Sintaxis
```
int DBIORDESC(dbproc)
DBPROCESS *dbproc;
```

DBIOWDESC (sólo UNIX y AOS/VS) Proporciona acceso del programa al descriptor de archivo de UNIX o AOS/VS usado por un DBPROCESS para escribir los datos al servidor.
Sintaxis
```
int DBIOWDESC(dbproc)
DBPROCESS *dbproc;
```

DBISAVAIL Determina si un DBPROCESS está disponible para el uso general.
Sintaxis
```
DBBOOL DBISAVAIL(dbproc)
DBPROCESS *dbproc;
```

dbisopt Verifica el estatus de un servidor u opciones de las DB-library.
Sintaxis
```
DBBOOL dbisopt(dbproc, option, param)
DBPROCESS *dbproc;
int option;
char *param;
```

DBLASTROW Devuelve el número de la última fila en el buffer de renglones.
Sintaxis
```
DBINT DBLASTROW(dbproc)
DBPROCESS *dbproc;
```

dbloadsort Carga un set order a un servidor.
Sintaxis
```
DBSORTORDER *dbloadsort(dbproc)
DBPROCESS *dbproc;
```

dblogin Asigna un registro del login para el uso en el dbopen.
Sintaxis
```
LOGINREC *dblogin()
```

dbloginfree Libera un registro de login.
Sintaxis
```
void dbloginfree(loginptr)
LOGINREC *loginptr;
```

dbmny4add Agrega dos valores de DBMONEY4.
Sintaxis
```
RETCODE dbmny4add(dbproc, m1, m2, sum)
DBPROCESS *dbproc;
DBMONEY4 *m1;DBMONEY4 *m2;
DBMONEY4 *sum;
```

dbmny4cmp Compara dos valores de DBMONEY4.

Sintaxis

```
int dbmny4cmp(dbproc, m1, m2)
DBPROCESS *dbproc;
DBMONEY4 *m1;
DBMONEY4 *m2;
```

dbmny4copy Copia un valor de DBMONEY4.

Sintaxis

```
RETCODE dbmny4copy(dbproc, src, dest)
DBPROCESS *dbproc;
DBMONEY4 *src;
DBMONEY4 *dest;
```

dbmny4divide Divide un valor DBMONEY4 entre otro.

Sintaxis

```
RETCODE dbmny4divide(dbproc, m1, m2, quotient)
DBPROCESS *dbproc;
DBMONEY4 *m1;
DBMONEY4 *m2;
DBMONEY4 *quotient;
```

dbmny4minus Niega un valor de DBMONEY4.

Sintaxis

```
RETCODE dbmny4minus(dbproc, src, dest)
DBPROCESS *dbproc;
DBMONEY4 *src;
DBMONEY4 *dest;
```

dbmny4mul Multiplica dos valores de DBMONEY4.

Sintaxis

```
RETCODE dbmny4mul(dbproc, m1, m2, product)
DBPROCESS *dbproc;
DBMONEY4 *m1;
DBMONEY4 *m2;
DBMONEY4 *product;
```

dbmny4sub Substrae un valor DBMONEY4 de otro.

Sintaxis

```
RETCODE dbmny4sub(dbproc, m1, m2, difference)
DBPROCESS *dbproc;
DBMONEY4 *m1;
DBMONEY4 *m2;
DBMONEY4 *difference;
```

dbmny4zero Inicializa una variable DBMONEY4 a $0.0000.

Sintaxis

```
RETCODE dbmny4zero(dbproc, mny4ptr)
DBPROCESS *dbproc;
DBMONEY4 *mny4ptr;
```

dbmnyadd Suma dos valores DBMONEY.

Sintaxis

```
RETCODE dbmnyadd(dbproc, m1, m2, sum)
DBPROCESS *dbproc;
DBMONEY *m1;DBMONEY *m2;
DBMONEY *sum;
```

dbmnycmp Compara dos valores DBMONEY.

Sintaxis
```
int dbmnycmp(dbproc, m1, m2)
DBPROCESS *dbproc;
DBMONEY *m1;
DBMONEY *m2;
```

dbmnycopy Copia un valor de DBMONEY.

Sintaxis
```
RETCODE dbmnycopy(dbproc, src, dest)
DBPROCESS *dbproc;
DBMONEY *src;
DBMONEY *dest;
```

dbmnydec Decrementa un valor de DBMONEY en un diez-milésimo de dólar.

Sintaxis
```
RETCODE dbmnydec(dbproc, mnyptr)
DBPROCESS *dbproc;
DBMONEY *mnyptr;
```

dbmnydivide Divide un valor DBMONEY entre otro.

Sintaxis
```
RETCODE dbmnydivide(dbproc, m1, m2, quotient)
DBPROCESS *dbproc;
DBMONEY *m1;
DBMONEY *m2;
DBMONEY *quotient;
```

dbmnydown Divide un valor DBMONEY por un entero positivo.

Sintaxis
```
RETCODE dbmnydown(dbproc, mnyptr, divisor, remainder)
DBPROCESS *dbproc;
DBMONEY *mnyptr;
int divisor;
int *remainder;
```

dbmnyinc Incrementa un valor DBMONEY en un diez-milésimo de dólar.

Sintaxis
```
RETCODE dbmnyinc(dbproc, mnyptr)
DBPROCESS *dbproc;DBMONEY *mnyptr;
```

dbmnyinit Prepara un valor DBMONEY para las llamadas con dbmnyndigit.

Sintaxis
```
RETCODE dbmnyinit(dbproc, mnyptr, trim, negative)
DBPROCESS *dbproc;
DBMONEY *mnyptr;
int trim;DBBOOL *negative;
```

dbmnymaxneg Devuelve el valor DBMONEY negativo máximo soportado.

Sintaxis
```
RETCODE dbmnymaxneg(dbproc,dest)
DBPROCESS *dbproc;
DBMONEY *dest;
```

dbmnymaxpos Devuelve el valor DBMONEY positivo máximo soportado.

Sintaxis
```
RETCODE dbmnymaxpos(dbproc, dest)
```

```
DBPROCESS *dbproc;
DBMONEY *dest;
```

dbmnyminus Niega un valor DBMONEY.

Sintaxis

```
RETCODE dbmnyminus(dbproc, src, dest)
DBPROCESS *dbproc;
DBMONEY *src;
DBMONEY *dest;
```

dbmnymul Multiplica dos valores DBMONEY.

Sintaxis

```
RETCODE dbmnymul(dbproc, m1, m2, product)
DBPROCESS *dbproc;
DBMONEY *m1;
DBMONEY *m2;
DBMONEY *product;
```

dbmnyndigit Devuelve el digito menos significativo de un valor DBMONEY como un DBCHAR.

Sintaxis

```
RETCODE dbmnyndigit(dbproc, mnyptr, value, zero)
DBPROCESS *dbproc;
DBMONEY *mnyptr;
DBCHAR *value;
DBBOOL *zero;
```

dbmnyscale Multiplica un valor DBMONEY por un entero positivo y agrega una cantidad especificada.

Sintaxis

```
RETCODE dbmnyscale(dbproc, mnyptr, multiplier, addend)
DBPROCESS *dbproc;
DBMONEY *mnyptr;
int multiplier;
int addend;
```

dbmnysub Resta un valor DBMONEY de otro.

Sintaxis

```
RETCODE dbmnysub(dbproc, m1, m2, difference)
DBPROCESS *dbproc;
DBMONEY *m1;DBMONEY *m2;
DBMONEY *difference;
```

dbmnyzero Inicializa un valor DBMONEY a $0.0000.

Sintaxis

```
RETCODE dbmnyzero(dbproc, mnyptr)
DBPROCESS *dbproc;
DBMONEY *mnyptr;
```

dbmonthname Determina el nombre de un mes especificado en un idioma especificado.

Sintaxis

```
char *dbmonthname(dbproc, language, monthnum,shortform)
DBPROCESS *dbproc;
char *language;
int monthnum;
DBBOOL shortform;
```

DBMORECMDS Indica si hay más órdenes a ser procesadas.

Sintaxis

```
RETCODE DBMORECMDS(dbproc)
DBPROCESS *dbproc;
```

dbmoretext Envía parte de un texto o de imagen al servidor.

Sintaxis

```
RETCODE dbmoretext(dbproc, size, text)
DBPROCESS *dbproc;
DBINT size;
BYTE *text;
```

dbmsghandle Instala una función para el usuario, para manejar los mensajes del servidor.

Sintaxis

```
int (*dbmsghandle(handler))()
int (*handler)();
```

dbname Devuelve el nombre de la base de datos actual.

Sintaxis

```
char *dbname(dbproc)
DBPROCESS *dbproc;
```

dbnextrow Lee la próxima fila del buffer de resultados.

Sintaxis

```
STATUS dbnextrow(dbproc)
DBPROCESS *dbproc;
```

dbnextrow_a (Sólo VMS) Asincrónicamente lea la próxima fila del buffer de resultados.

Sintaxis

```
STATUS dbnextrow_a(dbproc, final_result, ast_proc,
ast_param)
DBPROCESS *dbproc;
RETCODE *final_result;
void (*ast_proc)();
BYTE *ast_param;
```

dbnpcreate Crea un procedimiento de notificación.

Sintaxis

```
RETCODE dbnpcreate(dbproc)
DBPROCESS *dbproc;
```

dbnpdefine Define un procedimiento de notificación.

Sintaxis

```
RETCODE dbnpdefine(dbproc, procedure_name, namelen)
DBPROCESS *dbproc;
DBCHAR *procedure_name;DBSMALLINT namelen;
```

dbnullbind Asocia una variable con una columna de fila de resultados.

Sintaxis

```
RETCODE dbnullbind(dbproc, column, indicator)
DBPROCESS *dbproc;
int column;
DBINT *indicator;
```

dbnumalts Devuelve el número de columnas compute de una fila.

Sintaxis

```
int dbnumalts(dbproc, computeid)
DBPROCESS *dbproc;
int computeid;
```

dbnumcols Determina el número de columnas regulares para el conjunto actual de resultados.

Sintaxis

```
int dbnumcols(dbproc)
DBPROCESS *dbproc;
```

dbnumcompute Devuelve el número de cláusulas compute en el conjunto actual de resultados.

Sintaxis

```
int dbnumcompute(dbproc)
DBPROCESS *dbproc;
```

DBNUMORDERS Devuelve el número de columnas que estan involucradas en la sentencia order by de una orden de select de transact-SQL.

Sintaxis

```
int DBNUMORDERS(dbproc)
DBPROCESS *dbproc;
```

dbnumrets Determina el número de parámetros de retorno generado por un procedimiento almacenado.

Sintaxis

```
int dbnumrets(dbproc)
DBPROCESS *dbproc;
```

dbopen Crea e inicializa una estructura de DBPROCESS.

Sintaxis

```
DBPROCESS *dbopen(login, server)
LOGINREC *login;
char *server;
```

dbopen_a (sólo VMS) Crea e inicializa una estructura de DBPROCESS asincrónicamente.

Sintaxis

```
RETCODE dbopen_a(login, server, dbproc, final_result, ast_proc,
ast_param)
LOGINREC *login;
char *server;
DBPROCESS **dbprocptr;RETCODE *final_result;
void (*ast_proc)();BYTE *ast_param;
```

dbordercol Devuelve el numero de columna que ocupa en la clausula del query y que esta listada en la clausula order by.

Sintaxis

```
int dbordercol(dbproc, order)
DBPROCESS *dbproc;
int order;
```

dbpoll Checa si una contestación del servidor ha llegado por un DBPROCESS.

Sintaxis

```
RETCODE dbpoll(dbproc, milliseconds, ready_dbproc, return_reason)
DBPROCESS *dbproc;
long milliseconds;
DBPROCESS **ready_dbproc;
int *return_reason;
```

dbpoll_a (sólo VMS) Asincrónicamente Checa si una contestación del servidor ha llegado por un DBPROCESS.

Sintaxis

```
RETCODE dbpoll(dbproc, milliseconds, ready_dbproc, return_reason,
final_result, ast_proc, ast_param)
DBPROCESS *dbproc;
long milliseconds;
DBPROCESS **ready_dbproc;
int *return_reason;
RETCODE *final_result;
void (*ast_proc)();
BYTE *ast_param;
```

dbprhead Imprime los títulos de la columna para filas devueltas por el servidor.

Sintaxis

```
void dbprhead(dbproc)
DBPROCESS *dbproc;
```

dbprrow Imprime todas las filas devueltas por el servidor.

Sintaxis

```
RETCODE dbprrow(dbproc)
DBPROCESS *dbproc;
```

dbprtype Convierte una cadena de valor token a una cadena entendible.

Sintaxis

```
char *dbprtype(token)
int token;
```

dbqual Devuelve un apuntador a una clausula where para ser usada en una actualización en el actual renglón de la tabla consultada.

Sintaxis

```
char *dbqual(dbproc, tabnum, tabname)
DBPROCESS *dbproc;
int tabnum;char *tabname;
```

DBRBUF (sólo UNIX y AOS/VS) Determina si las DB-library contienen en el buffer de red algún byte no leido.

Sintaxis

```
DBBOOL DBRBUF(dbproc)
DBPROCESS *dbproc;
```

dbreadpage Lee una página de datos binarios del servidor.

Sintaxis

```
DBINT dbreadpage(dbproc, dbname, pageno, buf)
DBPROCESS *dbproc;
char *dbname;DBINT pageno;
BYTE buf[];
```

dbreadtext Lee parte de un texto o de imagen del servidor.

Sintaxis

```
STATUS dbreadtext(dbproc, buf, bufsize)
DBPROCESS *dbproc;
void *buf;
DBINT bufsize;
```

dbrecftos Graba todos los comandos de SQL enviados de la aplicación al servidor.

Sintaxis

```
void dbrecftos(filename)
char *filename;
```

dbrecvpassthru Recibe un paquete TDS de un servidor.

Sintaxis

```
RETCODE dbrecvpassthru(dbproc, recv_bufp)
DBPROCESS *dbproc;
DBVOIDPTR *recv_bufp;
```

dbregdrop Borra un procedimiento registrado.

Sintaxis

```
RETCODE dbregdrop(dbproc, procedure_name, namelen)
DBPROCESS *dbproc;
DBCHAR *procedure_name;
DBSMALLINT namelen;
```

dbregexec Ejecute un procedimiento registrado.

Sintaxis

```
RETCODE dbregexec(dbproc, options)
DBPROCESS *dbproc;
DBUSMALLINT options;
```

dbreghandle Instala una rutina para una notificación del procedimiento registrado.

Sintaxis

```
RETCODE dbreghandle(dbproc, procedure_name, namelen,
handler)
DBPROCESS *dbproc;
DBCHAR *procedure_name;
DBSMALLINT namelen;
INTFUNCPTR handler;
```

dbreginit Inicializa la ejecución de un procedimiento registrado.

Sintaxis

```
RETCODE dbreginit(dbproc, procedure_name, namelen)
DBPROCESS *dbproc;DBCHAR *procedure_name;
DBSMALLINT namelen;
```

dbreglist Devuelve una lista de procedimientos registrados definida en el Servidor Abierto actualmente.

Sintaxis

```
RETCODE dbreglist(dbproc)
DBPROCESS *dbproc;
```

dbregnowatch Cancela una demanda a ser notificada cuando un procedimiento registrado se ejecuta.

Sintaxis

```
RETCODE dbregnowatch(dbproc, procedure_name,
namelen)
DBPROCESS *dbproc;
DBCHAR *procedure_name;
DBSMALLINT namelen;
```

dbregparam Define o describa un parámetro del procedimiento

registrado.

Sintaxis

```
RETCODE dbregparam(dbproc,param_name, type, datalen,
data)
DBPROCESS *dbproc;
char *param_name;
int type;
DBINT datalen;
BYTE *data;
```

dbregwatch Pide ser notificado cuando un procedimiento registrado se ejecuta.

Sintaxis

```
RETCODE dbregwatch(dbproc, procedure_name,namelen,options)
DBPROCESS *dbproc;
DBCHAR *procedure_name;
DBSMALLINT namelen;
DBUSMALLINT options;
```

dbregwatchlist Devuelve una lista de procedimientos registrados que un DBPROCESS está vigilando.

Sintaxis

```
RETCODE dbregwatchlist(dbproc)
DBPROCESS *dbproc;
```

dbresults Prepara los resultados del siguiente query.

Sintaxis

```
RETCODE dbresults(dbproc)
DBPROCESS *dbproc;
```

dbresults_a (sólo VMS) Asincrónicamente prepara los resultados del siguiente query.

Sintaxis

```
RETCODE dbresults_a(dbproc, final_result, ast_proc,
ast_param)
DBPROCESS *dbproc;RETCODE *final_result;
void (*ast_proc)();BYTE *ast_param;
```

dbretdata Devuelve un apuntador a un valor de parámetro de retorno generado por un procedimiento almacenado.

Sintaxis

```
BYTE *dbretdata(dbproc, retnum)
DBPROCESS *dbproc;
int retnum;
```

dbretlen Determina la longitud de un valor del parámetro de retorno generada por un procedimiento almacenado.

Sintaxis

```
DBINT dbretlen(dbproc, retnum)
DBPROCESS *dbproc;
int retnum;
```

dbretname Determina el nombre del parámetro del procedimiento almacenado asociado con un valor del parámetro de retorno en particular.

Sintaxis

```
char *dbretname(dbproc, retnum)
DBPROCESS *dbproc;
```

```
int retnum;
```

dbretstatus Determina el número de estado del procedimiento almacenado devuelto por el comando actual o la llamada del procedimiento remoto.

Sintaxis

```
DBINT dbretstatus(dbproc)
DBPROCESS *dbproc;
```

dbrettype Determina el tipo de datos de un valor de parámetro de retorno generado por un procedimiento almacenado.

Sintaxis

```
int dbrettype(dbproc, retnum)
DBPROCESS *dbproc;
int retnum;
```

DBROWS Indica si el comando actual devolvió realmente renglones.

Sintaxis

```
RETCODE DBROWS(dbproc)
DBPROCESS *dbproc;
```

DBROWTYPE Devuelve el tipo de la fila actual.

Sintaxis

```
STATUS DBROWTYPE(dbproc)
DBPROCESS *dbproc;
```

dbrpcinit Inicializa una llamada de procedimiento remoto.

Sintaxis

```
RETCODE dbrpcinit(dbproc, rpcname, options)
DBPROCESS *dbproc;
char *rpcname;
DBSMALLINT options;
```

dbrpcparam Agrega un parámetro a una llamada del procedimiento remoto.

Sintaxis

```
RETCODE dbrpcparam(dbproc, paramname, status, type,maxlen,
datalen, value)
DBPROCESS *dbproc;
char *paramname;
BYTE status;
int type;DBINT maxlen;
DBINT datalen;BYTE *value;
```

dbrpcsend Indica el fin de una llamada de procedimiento remoto.

Sintaxis

```
RETCODE dbrpcsend(dbproc)
DBPROCESS *dbproc;
```

dbrpwclr Limpia todas las contraseñas remotas de la estructura de LOGINREC.

Sintaxis

```
void dbrpwclr(loginrec)
LOGINREC *loginrec;
```

dbrpwset Agrega una contraseña remota a la estructura de LOGINREC.

Sintaxis
```
RETCODE dbrpwset(loginrec, srvname, password, pwlen)
LOGINREC *loginrec;
char *srvname;
char *password;
int pwlen;
```

dbsafestr Duplica las cuotas en una cadena de carácteres.
Sintaxis
```
RETCODE dbsafestr(dbproc, src, srclen, dest, destlen, quotetype)
DBPROCESS *dbproc;
char *src;
DBINT srclen;
char *dest;
DBINT destlen;
int quotetype;
```

dbsechandle Instala una función del usuario para ocuparse de login seguro.
Sintaxis
```
RETCODE *dbsechandle(type, handler)
DBINT type;
INTFUNCPTR (*handler)();
```

dbsendpassthru Envía un paquete TDS a un servidor.
Sintaxis
```
RETCODE dbsendpassthru(dbproc, send_bufp)
DBPROCESS *dbproc;
DBVOIDPTR send_bufp;
```

dbservcharset Consigue el nombre del juego de carácter del servidor.
Sintaxis
```
char *dbservcharset(dbproc)
DBPROCESS *dbproc;
```

dbsetavail Marca un DBPROCESS como disponible para el uso general.
Sintaxis
```
void dbsetavail(dbproc)
DBPROCESS *dbproc;
```

dbsetbusy Llama una función del usuario cuando las DB-library están leyendo información del servidor.
Sintaxis
```
void dbsetbusy(dbproc, busyfunc)
DBPROCESS *dbproc;
int (*(*busyfunc)())();
```

dbsetdefcharset Pone un juego de caracteres predefinido para una aplicación.
Sintaxis
```
RETCODE dbsetdefcharset(charset)
char *charset;
```

dbsetdeflang Pone un idioma predefinido para una aplicación.
Sintaxis
```
RETCODE dbsetdeflang(language)
char *language;
```

dbsetidle Llama una función del usuario cuando las DB-library terminaron la lectura desde el servidor.

Sintaxis

```
void dbsetidle(dbproc, idlefunc)
DBPROCESS *dbproc;
void (*idlefunc)();
```

dbsetifile Especifica el nombre y localización del archivo de interfaces de Sybase.

Sintaxis

```
void dbsetifile(filename)
char *filename;
```

dbsetinterrupt Llama a una funcion del usuario para ocuparse de interrupciones mientras espera una lectura desde el servidor.

Sintaxis

```
void dbsetinterrupt(dbproc, chkintr, hndlintr)
DBPROCESS *dbproc;
int (*chkintr)();
int (*hndlintr)();
```

DBSETLAPP Coloca el nombre de la aplicación en la estructura de LOGINREC.

Sintaxis

```
RETCODE DBSETLAPP(loginrec, application)
LOGINREC *loginrec;
char *application;
```

DBSETLCHARSET Coloca el juego de caracteres en la estructura de LOGINREC.

Sintaxis

```
RETCODE DBSETLCHARSET(loginrec, char_set)
LOGINREC *loginrec;
DBCHAR *char_set;
```

DBSETLENCRYPT Especifica si la contraseña será o no encriptada para acceder al Servidor de SQL (10.0+)

Sintaxis

```
RETCODE DBSETLENCRYPT(loginrec, enable)
LOGINREC *loginrec;
DBBOOL enable;
```

DBSETLHOST Coloca el nombre del servidor en la estructura de LOGINREC.

Sintaxis

```
RETCODE DBSETLHOST(loginrec, hostname)
LOGINREC *loginrec;
char *hostname;
```

DBSETLLABELED Pone un bit para informar SQL Server™ Seguro que se enviarán las etiquetas de seguridad del login al momento de login.

Sintaxis

```
RETCODE DBSETLLABELED(loginrec, enable)
LOGINREC *loginrec;
DBBOOL enable;
```

DBSETLNATLANG Pone el idioma en la estructura de LOGINREC.

Sintaxis

```
RETCODE DBSETLNATLANG(loginrec, language)
LOGINREC *loginrec;
char *language;
```

dbsetloginfo Transfiere la información de una estructura de DBLOGINFO a una estructura de LOGINREC.

Sintaxis

```
RETCODE dbsetloginfo(loginrec, loginfo)
LOGINREC *login;
DBLOGINFO *loginfo;
```

dbsetlogintime Pone el número de segundos que las DB-library esperaran por una contestación del servidor a una demanda para una conexión de DBPROCESS.

Sintaxis

```
RETCODE dbsetlogintime(seconds)
int seconds;
```

DBSETLPACKET Pone el tamaño del paquete TDS en la estructura de LOGINREC de una aplicación.

Sintaxis

```
RETCODE DBSETLPACKET(login, packet_size)
LOGINREC *login;short packet_size;
```

DBSETLPWD Pone la contraseña de usuario del servidor en la estructura de LOGINREC.

Sintaxis

```
RETCODE DBSETLPWD(loginrec, password)
LOGINREC *loginrec;
char *password;
```

DBSETLUSER Pone el username en la estructura de LOGINREC.

Sintaxis

```
RETCODE DBSETLUSER(loginrec, username)
LOGINREC *loginrec;
char *username;
```

dbsetmaxprocs Pone el número máximo de estructuras de DBPROCESS simultáneamente abiertas.

Sintaxis

```
RETCODE dbsetmaxprocs(maxprocs)
int maxprocs;
```

dbsetnotifs (sólo VMS) Habilita o desactiva las notificaciones de procedimientos registrados.

Sintaxis

```
RETCODE dbsetnotifs(notif_state)
DBBOOL notif_state;
```

dbsetnull Define el valor de substitución para ser usada al ligar los valores nulos.

Sintaxis

```
RETCODE dbsetnull(dbproc, bindtype, bindlen, bindval)
DBPROCESS *dbproc;int bindtype;
```

```
int bindlen;
BYTE *bindval;
```

dbsetopt Pone opciones para un servidor o las DB-library.

Sintaxis

```
RETCODE dbsetopt(dbproc, option, char_param,int_param)
DBPROCESS *dbproc;
int option;
char *char_param;
int int_param;
```

dbsetrow Pone un renglón del buffered como el actual.

Sintaxis

```
STATUS dbsetrow(dbproc, row)
DBPROCESS *dbproc;
DBINT row;
```

dbsetsecurity Fija los valores de seguridad del login para el uso del servidor al logear en un Servidor de SQL Seguro.

Sintaxis

```
RETCODE dbsetsecurity(loginrec, labelname, labelvalue)
LOGINREC *loginrec;
CHAR *labelname;
CHAR *labelvalue;
```

dbsettime Pone el número de segundos que las DB-library esperará por una contestación del servidor a un comando SQL.

Sintaxis

```
RETCODE dbsettime(seconds)
int seconds;
```

dbsetuserdata Usa una estructura de DBPROCESS para guardar un apuntador a los datos de usuario asignados.

Sintaxis

```
void dbsetuserdata(dbproc, ptr)
DBPROCESS *dbproc;
BYTE *ptr;
```

dbsetversion Especifica un nivel de versión de la DB-library.

Sintaxis

```
RETCODE dbsetversion(version)
DBINT version;
```

dbspid Recupera el id del proceso de servidor para el DBPROCESS especificado.

Sintaxis

```
int dbspid(dbproc)
DBPROCESS *dbproc;
```

dbspr1row Pone un renglón de resultados del servidor en un buffer.

Sintaxis

```
RETCODE dbspr1row(dbproc, buffer, buf_len)
DBPROCESS *dbproc;
char *buffer;DBINT buf_len;
```

dbspr1rowlen Determina el tamaño del buffer para asignar los resultados devueltos por las funciones dbsprhead, dbsprline, y dbspr1row.

Sintaxis
```
DBINT dbspr1rowlen(dbproc)
DBPROCESS *dbproc;
```

dbsprhead Pone los titulos de un query dentro del buffer.
Sintaxis
```
RETCODE dbsprhead(dbproc, buffer, buf_len)
DBPROCESS *dbproc;
char *buffer;
DBINT buf_len;
```

dbsprline Crea una estructura que contiene el subrayado para los nombres de las columnas producido por el dbsprhead.
Sintaxis
```
RETCODE dbsprline(dbproc, buffer, buf_len, linechar)
DBPROCESS *dbproc;
char *buffer;
DBINT buf_len;
DBCHAR linechar;
```

dbsqlexec Envía un lote de comandos al servidor.
Sintaxis
```
RETCODE dbsqlexec(dbproc)
DBPROCESS *dbproc;
```

dbsqlexec_a (sólo VMS) Envía un lote de comados al servidor y verifique su exactitud asincrónicamente.
Sintaxis
```
RETCODE dbsqlexec_a(dbproc,final_result,ast_proc,ast_param)
DBPROCESS *dbproc;
RETCODE *final_result;
void (*ast_proc)();
BYTE *ast_param;
```

dbsqlok Espera por los resultados del servidor y verifica la exactitud de las instrucciones que el servidor está respondiendo.
Sintaxis
```
RETCODE dbsqlok(dbproc)
DBPROCESS *dbproc;
```

dbsqlok_a (sólo VMS) Espera por los resultados del servidor y verifica la exactitud de las instrucciones asincrónicamente.
Sintaxis
```
RETCODE dbsqlok_a(dbproc,final_result,ast_proc,ast_param)
DBPROCESS *dbproc;
RETCODE *final_result;
void (*ast_proc)();BYTE *ast_param;
```

dbsqlsend Envía un lote de comandos al servidor y no espera por una contestación.
Sintaxis
```
RETCODE dbsqlsend(dbproc)
DBPROCESS *dbproc;
```

dbstrbuild Construya una cadena imprimible de texto que contiene el placeholders por variable.
Sintaxis

```
int dbstrbuild(dbproc, charbuf, bufsize,text [, formats [, arg]
... ])
DBPROCESS *dbproc;
char *charbuf;
int bufsize;
char *text;
char *formats;
??? args???;
```

dbstrcmp Compara dos cadenas de caracteres especificando el tipo de ordenamiento.

Sintaxis

```
int dbstrcmp(dbproc, str1, len1, str2, len2,sortorder)
DBPROCESS *dbproc;
char *str1;
int len1;
char *str2;
int len2;
DBSORTORDER *sortorder;
```

dbstrcpy Copia una porción del buffer de comandos a una variable..

Sintaxis

```
RETCODE dbstrcpy(dbproc, start, numbytes, dest)
DBPROCESS *dbproc;
int start;
int numbytes;
char *dest;
```

dbstrlen Devuelve la longitud, en carácteres, del buffer de comandos.

Sintaxis

```
int dbstrlen(dbproc)
DBPROCESS *dbproc;
```

dbstrsort Determina cual de dos cadenas de carateres deben aparecer primero en una lista ordenada.

Sintaxis

```
int dbstrsort(dbproc, str1, len1, str2, len2,
sortorder)
DBPROCESS *dbproc;
char *str1;int len1;
char *str2;int len2;
DBSORTORDER *sortorder;
```

dbtabbrowse Determina si la tabla especificada es una tabla actualizable vía los medios del modo browse de las DB-library.

Sintaxis

```
DBBOOL dbtabbrowse(dbproc, tabnum)
DBPROCESS *dbproc;
int tabnum;
```

dbtabcount Devuelve el número de tablas involucradas en un select.

Sintaxis

```
int dbtabcount(dbproc)
DBPROCESS *dbproc;
```

dbtabname Devuelve el nombre de una tabla basado en su número.

Sintaxis

```
char *dbtabname(dbproc, tabnum)
DBPROCESS *dbproc;
```

```
int tabnum;
```

dbtabsource Devuelve el nombre y número de la tabla desde una columna en particular cuando fue derivada.

Sintaxis

```
char *dbtabsource(dbproc, colnum, tabnum)
DBPROCESS *dbproc;
int colnum;
int *tabnum;
```

DBTDS Determina qué versión de TDS (Tabular Data String protocolo) está usándose.

Sintaxis

```
int DBTDS(dbproc)
DBPROCESS *dbproc;
```

dbtextsize Regresa el número de bytes de texto o datos de la imagen que requiere ser leídos para el renglón actual.

Sintaxis

```
DBINT dbtextsize(dbproc)
DBPROCESS *dbproc;
```

dbtsnewlen Devuelve el tamaño del nuevo valor de la columna del timestamp después de una actualización en modo browse.

Sintaxis

```
int dbtsnewlen(dbproc)
DBPROCESS *dbproc;
```

dbtsnewval Devuelve el nuevo valor de la columna del timestamp después de una actualización en modo browse.

Sintaxis

```
DBBINARY *dbtsnewval(dbproc)
DBPROCESS *dbproc;
```

dbtsput Pone el nuevo valor de la columna del timestamp en la fila actual de la tabla dada en el DBPROCESS.

Sintaxis

```
RETCODE dbtsput(dbproc, newts, newtslen, tabnum,tabname)
DBPROCESS *dbproc;
DBBINARY *newts;
int newtslen;
int tabnum;char *tabname;
```

dbtxptr Devuelve el valor del apuntador del texto para una columna en la fila actual.

Sintaxis

```
DBBINARY *dbtxptr(dbproc, column)
DBPROCESS *dbproc;
int column;
```

dbtxtimestamp Devuelve el valor del texto para una columna timestamp en la fila actual.

Sintaxis

```
DBBINARY *dbtxtimestamp(dbproc, column)
DBPROCESS *dbproc;
int column;
```

dbtxtsnewval Devuelve el nuevo valor de un timestamp de texto después de una llamada a dbwritetext.

Sintaxis

```
DBBINARY *dbtxtsnewval(dbproc)
DBPROCESS *dbproc;
```

dbtxtsput Pone el nuevo valor de un timestamp de texto en la columna especificada de la fila actual en el DBPROCESS.

Sintaxis

```
RETCODE dbtxtsput(dbproc, newtxts, colnum)
DBPROCESS *dbproc;
DBBINARY *newtxts;
int colnum;
```

dbuse Use una base de datos en particular.

Sintaxis

```
RETCODE dbuse(dbproc, dbname)
DBPROCESS *dbproc;
char *dbname;
```

dbvarylen Determina si los datos de la columna de resultados especificada pueden variar en su longitud.

Sintaxis

```
DBBOOL dbvarylen(dbproc, column)
DBPROCESS *dbproc;
int column;
```

dbversion Determine qué versión de DB-library está en uso.

Sintaxis

```
char *dbversion()
```

dbwillconvert Determina si una conversión de tipo de datos específica está disponible dentro de las DB-Library.

Sintaxis

```
DBBOOL dbwillconvert(srctype, desttype)
int srctype;
int desttype;
```

dbwritepage Escribe una página de datos binarios hacia el servidor.

Sintaxis

```
RETCODE dbwritepage(dbproc, dbname, pageno, size, buf)
DBPROCESS *dbproc;
char *dbname;
DBINT pageno;
DBINT size;
BYTE buf[];
```

dbwritetext Envíe un texto o imagen hacia el servidor.

Sintaxis

```
RETCODE    dbwritetext(dbproc,    objname,    textptr,textptrlen,
timestamp, log,size, text)
DBPROCESS *dbproc;
char *objname;
DBBINARY *textptr;
DBTINYINT textptrlen;
DBBINARY *timestamp;
DBBOOL log;DBINT size;
BYTE *text;
```

dbxlate Traduzca una cadena desde un conjunto caracteres hacia otro.

Sintaxis

```
int dbxlate(dbproc, src, srclen, dest, destlen, xlt,
srcbytes_used, srcend, status)
DBPROCESS dbproc;
char *src;
int srclen;
char *dest;
int destlen;
DBXLATE *xlt;
int *srcbytes_used;
DBBOOL srcend;
int *status;
```

4.1.2 Bibliotecas GTK+

g_error()

#define g_error(format, args...)

La función g_error comunica un error grave en una aplicación Muestra un mensaje y aborta el programa. Esta función sólo debe utilizarse para errores que, de todas formas, harían que el programa termine. La función g_set_error_handler puede sustituir el comportamiento de la función g_error, pero no puede prevenir que ésta aborte el programa.

format : el formato del mensaje. Vea la documentación del printf().
args... : los parámetros a insertar en la secuencia del formato.

g_free()

void g_free (gpointer mem);

Libera la memoria a la que apunta mem.

mem : la memoria a liberar

g_strdup()

gchar* g_strdup(const gchar *str);

Duplica los primeros n caracteres de una cadena, retornando un buffer al que le asigna los n +1 caracteres de largo, la cadena siempre será terminada con un nulo. Si el str es menor que los n caracteres del buffer se rellena con nulos. El valor de retorno debe ser liberado cuando este ya no se utilice.

str: la parte de la cadena a duplicar
n : el máximo numero de caracteres a copiar de str

returns : un buffer intermediario nuevo al cual se le asignaran los primeros n caracteres de str terminando en NULL.

gdk_color_alloc()

```
gintgdk_color_alloc (GdkColormap *colormap,
  GdkColor *color);
```

Asigna solo un color de un colormap.

colormap : un GdkColormap.
color : El color a asignar.
returns: TRUE (verdadero) si la asignación tuvo éxito.

gdk_colormap_get_system()

```
GdkColormap* gdk_colormap_get_system(void);
```

Obtiene el colormap estándar del sistema por default

returns : el default colormap.

gdk_pixmap_create_from_xpm ()

```
GdkPixmap*  gdk_pixmap_create_from_xpm  (GdkWindow *window,
GdkBitmap **mask, GdkColor *transparent_color,
  const gchar *filename)
```

Crea un pixmap para un archivo XPM.

window: una GdkWindow, usada para determinar los valores por default del pixmap nuevo.
mask: un puntero a un bitmap que representa la transparencia en mascara del archivo XPM. Puede ser NULL, lo que implicara que se ignora la trasparencia.
transparent_color : el color a ser usado en pixeles que será transparente en el archivo de entrada. Puede ser NULL, en este caso se usara el color por default.
filename : el nombre del archivo que contendrá el XPM.
returns : GdkPixmap

gtk_box_pack_start()

```
voidgtk_box_pack_start  (GtkBox *box,
  GtkWidget *child,
  gboolean expand,
  gboolean fill,
  guint padding);
```

Agrega o empaqueta al widget en el box(recipiente).

box : un GtkBox.
child : el GtkWidget a ser adicionado al box.
expand : TRUE (verdadero) si el widget nuevo a adicionar ocupara un espacio extra al adicionarse al box. Se dividirá el espacio extra uniformemente entre todos los widgets hijos de caja si se usa esta opción.
fill : TRUE (verdadero) si el espacio dado al widget se ajusta a él. Este parámetro no tiene efecto si se pone FALSE (falso) este parámetro.
padding : el espacio extra en pixeles que se colocara el widget que sé esta agrandando y los demás widgets contenidos en el box.

gtk_button_new_with_label()

```
GtkWidget*  gtk_button_new_with_label   (const gchar *label);
```

Crea un widget GtkButton con un hijo GtkLabel que contiene el texto dado.

label : El texto que se quiere que contenga GtkLabel.
returns : el nuevo widget GtkButton creado.

gtk_container_add ()

```
voidgtk_container_add   (GtkContainer *container,
 GtkWidget *widget);
```

Agrega un widget a un contenedor. Normalmente se usan contenedores simples tal como GtkWindow, GtkFrame, o GtkButton.
Para contenedores mas complicados tal como GtkBox o GtkTable, esta función obtendrá parámetros estándares que no estarían correctos. Para estos casos considere funciones tal como gtk_box_pack_start() y gtk_table_attach() como una alternativa a gtk_container_add.

container : un GtkContainer
widget : un widget para ser colocado en el contenedor

gtk_container_border_width()

```
#define gtk_container_border_width  gtk_container_set_border_width
```

Crea un gkt_container con borde

gtk_dialog_new()

```
GtkWidget*  gtk_dialog_new  (void);
```

Crea una nueva caja de diálogo. Este wdget no se debe colocar en un GtkWindow directamente, sino en un vbox.
returns : un nuevo GtkDialog

gtk_editable_get_chars()

```
gchar*  gtk_editable_get_chars  (GtkEditable *editable,
 gint start_pos,
 gint end_pos);
```

Recupera una secuencia de caracteres.

editable : un GtkEditable widget.
start_pos : posición inicial.
end_pos : posición final.
returns : los caracteres en la región indicada. El resultado se debe liberar con g_free() cuando se termina de usar.

gtk_entry_new()

```
GtkWidget*  gtk_entry_new   (void);
```

Crea un nuevo GtkEntry widget.

returns : un nuevo GtkEntry.

gtk_entry_set_visibility()

```
voidgtk_entry_set_visibility(GtkEntry *entry,
  gboolean visible);
```

Fija si el contenido de la entrada es visible o no. Cuando la visibilidad se fija a FALSO, los caracteres se exhiben invisibles, y también aparecerán de esa manera cuando el texto en el widget de la entrada se copia a otra parte.

El carácter invisible por defecto es el asterisco " * ", y solamente puede ser cambiado con el gtk_entry_set_invisible_char().

entry : un GtkEntry.
visible : TRUE(verdadero) si el contenido del GtkEntry es visible.

gtk_grab_add()

```
voidgtk_grab_add(GtkWidget *widget);
```

Hace que se bloqueen los otros widgets y cede el control del ratón y el teclado al widget actual.

widget : el widget que recibirá el control

gtk_hbox_new()

```
GtkWidget*  gtk_hbox_new(gboolean homogeneous,
 gint spacing);
```

Crea un nuevo GtkHBox. Caja o contenedor horizontal.

homogeneous : TRUE(verdadero) si todos los widgets que contenga tiene asignado el mismo espacio.
spacing : el número de los pixeles a colocar por defecto entre widgets que contenga.
returns : un nuevo GtkHBox.

gtk_hseparator_new ()

```
GtkWidget*  gtk_hseparator_new  (void);
```

Crea un nuevo GtkHSeparator. Un separador horizontal.

returns : un nuevo GtkHSeparator.

gtk_init ()

```
voidgtk_init(int *argc,
  char ***argv);
```

Se debe llamar esta función antes de usar otras funciones de GTK+. Inicializará todo lo necesario para que funcionen todas las herramientas y analiza un cierto argc estándar de las opciones de la línea de comandos.

argc: Dirección del parámetro del argc de la función principal.
argv: Dirección del parámetro del argv del main().

gtk_label_new()

```
GtkWidget*  gtk_label_new   (const char *str);
```

Crea una nueva etiqueta con el texto dado dentro de él. Si se pasa como parámetro un NULL se puede conseguir un widget con etiqueta vacío.

str : El texto de la etiqueta
returns: el GtkLabel nuevo

gtk_label_set_justify ()

voidgtk_label_set_justify (GtkLabel *label,
GtkJustification jtype);

Fija la alineación de las líneas en el texto de la etiqueta. GTK_JUSTIFY_LEFT es el valor predeterminado cuando el widget se crea con gtk_label_new().

label : una GtkLabel
jtype : una GtkJustification

gtk_main_quit()

voidgtk_main_quit (void);

Invoca para que se recupere el control.

gtk_menu_append()

#define gtk_menu_append(menu,child) gtk_menu_shell_append ((GtkMenuShell *)(menu),(child))

Agrega un GtkMenuItem nuevo al extremo de la lista del menú.

menu : un GtkMenu.
child : El GtkMenuItem a adicionar

gtk_menu_bar_append()

#define gtk_menu_bar_append(menu,child) gtk_menu_shell_append ((GtkMenuShell *)(menu),(child))

Agrega un GtkMenuItem nuevo al final del GtkMenuBar

menu : menu
child : el GtkMenuItem a adicionar

gtk_menu_item_new_with_label ()

GtkWidget* gtk_menu_item_new_with_label(const gchar *label);

Crea un nuevo GtkMenuItem hijo con etiqueta
label : el texto para la etiqueta
returns: el GtkMenuItem nuevo creado

gtk_menu_item_set_submenu ()

```
voidgtk_menu_item_set_submenu   (GtkMenuItem *menu_item,
 GtkWidget *submenu);
```

Fija o cambia el submenu del widget.

menu_item : el menu item widget
submenu: el submenu

gtk_menu_new ()

```
GtkWidget*  gtk_menu_new(void);
```

Crea un nuevo GtkMenu.

returns : un nuevo GtkMenu.

gtk_menu_set_title ()

```
voidgtk_menu_set_title  (GtkMenu *menu,
 const gchar *title);
```

Asigna el título para el menú.

menu : un GtkMenu
title: una secuencia que contiene el título para el menú

gtk_misc_set_alignment()

```
voidgtk_misc_set_alignment  (GtkMisc *misc,
 gfloat xalign,
 gfloat yalign);
```

Controla la alineación del widget.

misc : un GtkMisc.
xalign : la alineación horizontal , con 0 (izquierdo) y con 1 (derecho).
yalign : la alineación vertical, con 0 (arriba) y con 1 (abajo).

gtk_misc_set_padding()

```
voidgtk_misc_set_padding(GtkMisc *misc,
gint xpad,
gint ypad);
```

Controla la cantidad de espacio agregar alrededor del widget.

misc : un GtkMisc.
xpad : la cantidad de espacio a agregar en el lado izquierdo y derecho del widget, en pixeles.
ypad : la cantidad de espacio agregar en la arriba y abajo del widget, en pixeles.

gtk_notebook_append_page ()

```
voidgtk_notebook_append_page(GtkNotebook *notebook,
 GtkWidget *child,
 GtkWidget *tab_label);
```

Añade una página al *notebook*.

notebook : un GtkNotebook
child: el GtkWidget a utilizar como contenido de la página.
tab_label : el GtkWidget que se utilizará como la etiqueta para la página, si se pasa como parámetro NULL se utilizará la etiqueta por defecto, "página N ".

gtk_notebook_new ()

```
GtkWidget*  gtk_notebook_new(void);
```

Crea un nuevo widget de GtkNotebook sin páginas.

returns : el nuevo GtkNotebook creado.

gtk_pixmap_new ()

```
GtkWidget*  gtk_pixmap_new  (GdkPixmap *pixmap,
  GdkBitmap *mask);
```

Crea un GtkPixmap nuevo, usa el pixmap GDK dado y su máscara. Permite desplegar una imagen gráfica o icono en un widget.

gtk_progress_bar_new()

```
GtkWidget*  gtk_progress_bar_new(void);
```

Crea un nuevo GtkProgressBar. Es una barra que indica progreso visualmente.

returns : un GtkProgressBar.

gtk_radio_button_group ()

```
#define gtk_radio_button_group gtk_radio_button_get_group
```

Crea un grupo de radio buttons

gtk_radio_button_new_with_label ()

```
GtkWidget*  gtk_radio_button_new_with_label (GSList *group,
  const gchar *label);
```

Crea un nuevo GtkRadioButton con etiqueta. Crea una opción de check buttons múltiples

group: un grupo radio button existente, o NULL si se creara un nuevo
label : el texto que se observara después del radio button.
returns : un nuevo radio button.

gtk_signal_connect()

```
#define gtk_signal_connect(object,name,func,func_data)
```

Registra la rutina de tratamiento de sucesos ante GTK+. Esta función tiene cuatro parámetros, que indican a GTK+ el widget al que corresponde la retrollamada, qué señal va a tratar ésta, la función que hay que utilizar como retrollamada cuando se envié la señal y que parámetros adicionales hay que pasar a dicha función cuando se invoque la rutina de tratamiento del suceso.

gtk_table_attach ()

```
voidgtk_table_attach(GtkTable *table,
  GtkWidget *child,
  guint left_attach,
  guint right_attach,
  guint top_attach,
  guint bottom_attach,
 GtkAttachOptions xoptions,
 GtkAttachOptions yoptions,
 guint xpadding,
 guint ypadding);
```

Agrega un widget a una tabla. El número de celdas que un widget ocupará es especificado por el left_attach, el right_attach, el top_attach y el bottom_attach. Cada uno representa el extremo izquierdo, de derecha, predominantemente y los números más bajos de la columna y de la fila de la tabla. (las columnas y las filas se ponen en un índice a partir de la cero).

table : la GtkTable a donde se agregara el widget
child: el widget que se adicionara
left_attach : el número de la columna del lado izquierdo donde se colocara el widget.
right_attach : el número de la columna del lado derecho donde se colocara el widget.

top_attach : el número de la fila del lado izquierdo donde se colocara el widget.
bottom_attach : el número de la fila del lado derecho donde se colocara el widget.
xoptions : Especifica las características del widget cuando se vuelve a clasificar según el tamaño de la tabla.
yoptions : Igual que xoptions, solo que este campo determina comportamiento al volver a clasificar según el tamaño vertical.
xpadding : Un valor entero que especifica el espacio izquierdo y derecho del widget que es agregado a la tabla.
ypadding : La cantidad de espacio arriba y debajo del widget.

gtk_table_attach_defaults ()

```
voidgtk_table_attach_defaults   (GtkTable *table,
  GtkWidget *widget,
  guint left_attach,
  guint right_attach,
  guint top_attach,
  guint bottom_attach);
```

Esta función provee al programador los medios para agregar el widget a una tabla con opciones idénticas de espaciado entre celdas.

table : la tabla a la que se adicionara el nuevo widget
widget : el widget a adicionar
left_attach : el número de la columna del lado izquierdo donde se colocara el widget.
right_attach : el número de la columna del lado derecho donde se colocara el widget.
top_attach : el número de la fila del lado izquierdo donde se colocara el widget.
bottom_attach : el número de la fila del lado derecho donde se colocara el widget.

gtk_table_new ()

```
GtkWidget*  gtk_table_new   (guint rows,
  guint columns,
  gboolean homogeneous);
```

Crea un nuevo widget tabla. El tamaño inicial debe ser especificando en número de filas y columnas que tendrá la tabla, aunque esto se puede cambiar más adelante por medio de gtk_table_resize().

rows : El número de filas que la nueva tabla tendrá
columns : El número de columnas que la nueva tabla tendrá

homogeneous: Si el valor es TRUE (verdadero) todas las celdas de la tabla se vuelve a clasificar según el tamaño de la celda que contiene el widget más grande.
returns : Un puntero al nuevo widget tabla creado.

gtk_table_set_col_spacings ()

```
voidgtk_table_set_col_spacings  (GtkTable *table,
  guint spacing);
```

Fija el espacio entre cada columna de la tabla.

table : una GtkTable.
spacing : el numero en pixeles del espacio entre las columnas de la tabla.

gtk_text_freeze ()

```
voidgtk_text_freeze (GtkText *text);
```

Congela el widget GtkText que no permite rediseñar el widget hasta que se descongela. Esto es útil si una gran cantidad de cambios se han hecho al texto dentro del widget, reduciendo la cantidad de parpadeo vista por el usuario.

text : the GtkText widget

gtk_text_insert ()

```
voidgtk_text_insert (GtkText *text,
 GdkFont *font,
 GdkColor *fore,
 GdkColor *back,
 const char *chars,
 gint length);
```

Rellena el widget de GtkText con el texto dado según as características dadas por sus parámetros.

text: el GtkText widget
font: el GdkFont a usar
fore: el color de primer plano con el que se insertara
back : el color del fondo con el que se insertara
chars : el texto que se insertara
length : la longitud del texto que se insertará

gtk_text_new ()

```
GtkWidget*  gtk_text_new(GtkAdjustment *hadj,
GtkAdjustment *vadj);
```

Crea un nuevo widget de tipo texto (GtkText), inicializado con los valores dados a GtkAdjustments. Estos indicadores se pueden utilizar para seguir la posición de la visión del widget de GtkText.

Hadj : ajuste horizontal
vadj : ajuste vertical
returns : el nuevo widget GtkText.

gtk_text_set_editable ()

```
voidgtk_text_set_editable   (GtkText *text,
 gboolean editable);
```

Fija si el widget de GtkText puede ser modificado por el usuario o no. Esto permite que el programador realice cambios con varias de las funciones de GtkText.

text : el GtkText widget
editable : TRUE (verdadero) si el widget GtkText es editable, FALSE (falso) si el GtkText no puede ser modificado por el usuario

gtk_text_set_word_wrap ()

```
voidgtk_text_set_word_wrap  (GtkText *text,
 gboolean word_wrap);
```

Fija si el widget GtkText puede tener mas de una línea o si bien puede ser terminado en la línea actual.

text : el GtkText widget
word_wrap : TRUE(verdadero) si el GtkText puede tener varias líneas, FALSE (falso) de lo contrario.

gtk_timeout_add()

```
gtk_timeout_add (SCROLL_DELAY_LENGTH,
  GtkFunction,
       gpointer);
```

Agrega un contador de tiempo.

gtk_toggle_button_set_state ()

```
#define gtk_toggle_button_set_state gtk_toggle_button_set_active
```

Configura el estado en que se desea que se encuentre un botón.

gtk_tooltips_set_tip ()

```
voidgtk_tooltips_set_tip(GtkTooltips *tooltips,
GtkWidget *widget,
const gchar *tip_text,
const gchar *tip_private);
```

Agrega un tooltip contiene el tip_text del mensaje a un GtkWidget especificó.

gtk_vbox_new ()

```
GtkWidget*  gtk_vbox_new(gboolean homogeneous,
 gint spacing);
```

Crea un nuevo GtkVBox. Caja o contenedor vertical.

homogeneous : TRUE(verdadero) si todos los widgets que contenga tiene asignado el mismo espacio.
spacing : el número de los pixeles a colocar por defecto entre widgets que contenga.
returns : un nuevo GtkVBox.

gtk_vscrollbar_new ()

```
GtkWidget*  gtk_vscrollbar_new  (GtkAdjustment *adjustment);
```

Crea un nuevo scrollbar vertical.

adjustment : el GtkAdjustment en uso, o NULL para crear un nuevo GtkAdjustment.
returns : el nuevo GtkVScrollbar

gtk_widget_add_accelerator()

```
voidgtk_widget_add_accelerator  (GtkWidget *widget,
 const gchar *accel_signal,
 GtkAccelGroup *accel_group,
 guint accel_key,
 GdkModifierType accel_mods,
 GtkAccelFlags accel_flags);
```

Instala un acelerador para el widget del accel_group que hace accel_signal. El accel_group necesita ser agregado al toplevel de los widget vía gtk_window_add_accel_group(), y la señal debe ser del tipo G_RUN_ACTION. Los aceleradores que se agregan con esta función no pueden ser cambiados por el usuario durante el tiempo de ejecución.

widget : widget al que se le instalara un acelerador
accel_signal : señal que el widget debe emitir al activarse el acelerador
accel_group : grupo acelerador para este widget
accel_key: valor GDK del acelerador
accel_mods : combinación dominante del modificante del acelerador
accel_flags : bandera del acelerador

gtk_widget_destroy ()

voidgtk_widget_destroy (GtkWidget *widget);

Destruye un widget. Equivalente al gtk_object_destroy(). Cuando se destruye un widget, romperá cualquier referencia que lleve a cabo a otros objetos. Si el widget está dentro de un contenedor, el widget será quitado de el. Quitar un widget de su contenedor o de una lista de toplevels da lugar a que el widget sea concluido, a menos que se hayan agregado referencias adicionales al widget con el gtk_object_ref.

widget : un GtkWidget

gtk_widget_get_style ()

GtkStyle* gtk_widget_get_style(GtkWidget *widget);

Función que devuelve el estilo de un widget

widget : un GtkWidget
returns : el GtkStyle del widget

gtk_widget_grab_default ()

voidgtk_widget_grab_default (GtkWidget *widget);

Convierte el widget en el widget por defecto. Debe tener la bandera de GTK_CAN_DEFAULT activada; se tiene que fijar manualmente esta bandera llamando a GTK_WIDGET_SET_FLAGS (widget, GTK_CAN_DEFAULT).

widget : a GtkWidget

gtk_widget_hide ()

```
voidgtk_widget_hide (GtkWidget *widget);
```

Invierte los efectos de gtk_widget_show(), haciendo que el widget se oculte (invisible al usuario).

widget : a GtkWidget

gtk_widget_realize ()

```
voidgtk_widget_realize  (GtkWidget *widget);
```

Pone en primer plano la ventana actual. Se utiliza cuando se tienen varias ventanas abiertas y se requiere que solo una este "activa".

widget : a GtkWidget

gtk_widget_set_usize ()

```
voidgtk_widget_set_usize(GtkWidget *widget,
  gint width,
  gint height);
```

Asigna el tamaño mínimo de un widget; es decir, la petición del tamaño de los widget será el ancho por el alto. Se puede utilizar esta función para forzar un widget a ser más grande o más pequeño de lo que es. En muchos casos, el gtk_window_set_default_size() es una opción mejor para las ventanas que esta función; El usar usize forzará a dejar la ventana por lo menos tan grande como usize.

widget : un GtkWidget
width : la anchura mínima, o –1 para el valor por default
height : la altura mínima, o -1 para el valor por default

gtk_widget_show ()

```
voidgtk_widget_show (GtkWidget *widget);
```

Muestra un widget. Todo widget debe ser mostrado en la ventana por medio de esta función de lo contrario no se visualizara. Si se desea mostrar todos los widgets en un contenedor, es más fácil llamar el gtk_widget_show_all() en el contenedor, en vez hacerlo individualmente. Recuerde que se tiene que mostrar el contenedor que contienen un widget, además del widget a sí mismo.

widget : un GtkWidget

gtk_widget_show_all ()

voidgtk_widget_show_all (GtkWidget *widget);

Muestra recurrentemente un widget, y cualquier widget hijo (sí el widget es un contenedor).

widget : un GtkWidget

gtk_window_get_focus ()

GtkWidget* gtk_window_get_focus(GtkWindow *window);

Recupera el widget que se encuentra enfocado actualmente dentro de la ventana.

window : una GtkWindow
returns : el widget actualmente enfocado.

gtk_window_get_resizable ()

gbooleangtk_window_get_resizable(GtkWindow *window);

Obtiene del sistema el valor establecido por medio de la función gtk_window_set_resizable()

window : una GtkWindow
return : TRUE(verdadero) si el usuario puede modificar el tamaño de la ventana

gtk_window_maximize ()

voidgtk_window_maximize (GtkWindow *window);

Maximiza la ventana, de modo que llegue a ocupar toda pantalla. Observe que no se debe asumir que la ventana está maximizada definitivamente, porque podrían otras entidades (por ejemplo usuario de la ventana) minimizarla otra vez. Usted puede seguir la maximización vía la señal "window_state_event" en GtkWidget.

window : una GtkWindow

gtk_window_move ()

```
voidgtk_window_move (GtkWindow *window,
  gint x,
  gint y);
```

Mueve la ventana a la nueva posición dada. Nota: la posición es la posición del punto de referencia gravedad-determinado para la ventana. La gravedad determina dos cosas: primero, la localización del punto de referencia en ventana; y en segundo lugar, que señalan en la ventana se coloca en el punto de referencia.

gtk_window_new ()

GtkWidget* gtk_window_new (GtkWindowType type);

Crea una GtkWindow nueva, que es una ventana de nivel superior que puede contener otros widgets. Casi siempre, el tipo de la ventana debe ser GTK_WINDOW_TOPLEVEL. Si se está poniendo algo en ejecución como un menú del popup, usted debe utilizar GTK_WINDOW_POPUP.

En GTK+, GTK_WINDOW_POPUP significa un menú pop-up.

type : type of window
return : una nueva GtkWindow

gtk_window_set_default_size ()

voidgtk_window_set_default_size (GtkWindow *window,
gint width,
gint height);

Fija el tamaño por defecto de una ventana. Si el tamaño "natural" de la ventana es más grande que el de por defecto, el tamaño por defecto no será tomado en cuenta. El tamaño por defecto de una ventana afecta solamente la primera vez que se muestra una ventana; si se oculta y redemuestra una ventana, recordará el tamaño que tenía antes de ocultar, en vez de usar el tamaño por defecto. Una ventana no puede ser de tamaño 0x0, deben ser por lo menos de 1x1, pero poner 0 de ancho y alto es ACEPTABLE, dando por resultado un tamaño por defecto 1x1.

window : una GtkWindow
width: ancho en pixeles, o -1 para usar el ancho por defecto
height : alto en pixeles, o -1 para usar el alto por defecto

gtk_window_set_destroy_with_parent ()

voidgtk_window_set_destroy_with_parent
(GtkWindow *window,
gboolean setting);

Si se establece como TRUE (verdadero), después de destruir la ventana padre, la ventana hija también se destruirá. Esto es útil para los diálogos que

no deben persistir más allá del curso de la vida de la ventana principal a la que se asocian.

gtk_window_set_focus ()

```
voidgtk_window_set_focus(GtkWindow *window,
 GtkWidget *focus);
```

Si el foco no lo tiene el widget actual, y es un widget con la propiedad focus, lo fija como el widget con el foco de la ventana. Para fijar el foco a un widget en particular es conveniente utilizar la función gtk_widget_grab_focus() en vez de esta función.

window : una GtkWindow
focus: el nuevo widget con el focus

gtk_window_set_modal ()

```
voidgtk_window_set_modal(GtkWindow *window,
  gboolean modal);
```

Establece si una ventana es modal o no-modal. Las ventanas modales previenen la interacción con otras ventanas en el mismo uso.

window : una GtkWindow
modal : establece si una ventana es de tipo modal

gtk_window_set_resizable ()

```
voidgtk_window_set_resizable(GtkWindow *window,
  gboolean resizable);
```

Establece si el usuario puede modificar el tamaño de una ventana. Las ventanas de usuario tienen la propiedad resizable por defecto.

window: una GtkWindow
resizable : TRUE(verdadero) si el usuario puede modificar el tamaño de la ventana

gtk_window_set_title ()

```
voidgtk_window_set_title(GtkWindow *window,
   const gchar *title);
```

Fija el título de la GtkWindow. El título de una ventana será exhibido en su barra del título; en el sistema de la ventana, la barra del título es controlada por el encargado de ventana, El título debe ayudar a un usuario a distinguir esta ventana de otras ventanas que pueden tener abiertas. Un buen título

puede incluir el nombre del uso y el nombre de archivo actual del documento, por ejemplo.

window : una GtkWindow
title : titulo de la ventana

gtk_window_unmaximize ()

voidgtk_window_unmaximize (GtkWindow *window);

Pide minimizar la ventana. Observe que usted no debe asumir la ventana es minimizada definitivamente, porque otras entidades (por ejemplo el usuario de la ventana) podrían maximizarla otra vez. Usted puede seguir la maximización vía la señal "window_state_event" en GtkWidget.

window : una GtkWindow

xpm_label_box()

GtkWidget *xpm_label_box(gchar *xpm_filename,
gchar *label_text)

Se utiliza para colocar imágenes y etiquetas en cualquier widget que pueda ser un contenedor.

xpm_filename: archivo xpm a colocar dentro del widget
label_text : texto de la etiqueta a colocar dentro del widget

4.2 Desarrollo de la interfaz del SABAD

Para la construcción de la interfaz del sistema de administración de bases de datos se requirió de los lenguajes y herramientas que se describieron en el capitulo 3.

La programación de las pantallas se realizó con el lenguaje GTK+, la conexión a la base de datos se desarrollo con lenguaje C y las bibliotecas DB-Library.

4.2.1 Interfaz gráfica

A continuación se mostrará el desarrollo de las pantallas más representativas del sistema.

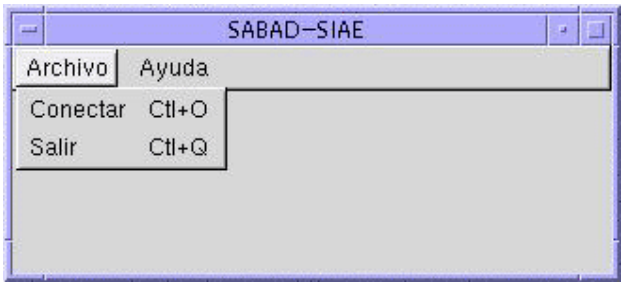


Figura 4.2 Ventana del menú principal

La figura 4.2 muestra la ventana principal con la cual trabajará el sistema de administración de bases de datos (SABAD), tiene un titulo y una barra de menú con las opciones a elegir; en GTK+ los componentes de la interfaz gráfica son conocidos como *widgets*. Ejemplos de *widgets* son las ventanas, casillas de verificación, botones y campos editables. Los *widgets* y las ventanas se definen siempre como punteros a una estructura GtkWidget. Esta estructura es un tipo de datos genérico, utilizado por todos los *widgets.*

El código que permite construir esta ventana es el siguiente:

```
/*Función principal(main)*/
int main( int   argc, char *argv[] ) {
  GtkWidget *window;   /*Declaración de los widgets*/
  GtkWidget *menu;
  GtkWidget *menu1;
  GtkWidget *menuitem;
  GtkWidget *menubar;
  GtkWidget *menu_bar;
  GtkWidget *ayuda;
  GtkWidget *archivo;

  gtk_init (&argc, &argv);
  system("clear"); /*Limpiar pantalla*/
```

```
/*Crea ventana principal*/
  window = gtk_window_new (GTK_WINDOW_TOPLEVEL);

/*Asigna un título*/
 gtk_window_set_title (GTK_WINDOW (window), "SABAD-SIAE");
  …
  menu = gtk_menu_new ();   /*Crea un menú*/

  conectar = gtk_menu_item_new_with_label("Conectar");

/*Asigna una etiqueta al menu*/
gtk_menu_append (GTK_MENU (menu),  conectar);
gtk_widget_show(conectar); /*Muestra la etiqueta*/

  menuitem = gtk_menu_item_new_with_label("Salir");

/*Asigna una etiqueta al menu*/
  gtk_menu_append (GTK_MENU (menu),  menuitem);
  gtk_widget_show(menuitem);

  archivo = gtk_menu_item_new_with_label("Archivo");

/*Asigna una etiqueta al menu*/
  gtk_menu_item_set_submenu (GTK_MENU_ITEM (archivo),
                             menu);
  gtk_widget_show(archivo);

  menu1 = gtk_menu_new ();

  menuitem = gtk_menu_item_new_with_label("Acerca de...");
  gtk_menu_append (GTK_MENU (menu1),  menuitem);
  gtk_widget_show(menuitem);

  ayuda = gtk_menu_item_new_with_label("Ayuda");
  gtk_menu_item_set_submenu (GTK_MENU_ITEM (ayuda),
                             menu1);
  gtk_widget_show(ayuda);

/*crea barra de menu en la ventana principal*/
  menu_bar = gtk_menu_bar_new ();
  gtk_menu_bar_append (GTK_MENU_BAR (menu_bar),  archivo);
  gtk_menu_bar_append (GTK_MENU_BAR (menu_bar),  ayuda);
  gtk_widget_show(menu_bar); /*Muestra la barra de menu*/
  gtk_widget_show(window); /*Muestra la ventana*/
}
```

Cuando se elige la opción conectar del menú, de la figura 4.2 aparecerá la siguiente ventana

Figura 4.3 Conexión al servidor

La figura 4.3 Muestra una ventana la cual recibe como parámetros de entrada el "*login*" y el "*password"* del usuario que se conectará al servidor de base de datos, el usuario debe ser un "*login"* valido dentro del servidor, cuenta con dos botones conectar y cancelar, cuando el usuario llena los campos requeridos y da clic en el botón conectar, se manda a llamar la función que realiza la conexión al servidor de bases de datos, esta función emplea las bibliotecas DB-Library.

La venta de la figura 4.3 se crea con el siguiente código:

```
/*Función que crea la ventana para la conexión al servidor*/
gint conect(void) {
  GtkWidget *label;  /*Declaración de widgets*/
  GtkWidget *entry;
  GtkWidget *button;
  GtkWidget *table;
  GtkWidget *hbox;
  GtkWidget *vbox;

/*Creación de una nueva ventana*/
  window_entry = gtk_dialog_new();

/*tamaño de la ventana*/
  gtk_widget_set_usize(window_entry, 250, 150);

/*Asigna titulo*/
  gtk_window_set_title(GTK_WINDOW(window_entry),
              "Conexión al Servidor");

/*Crea una caja de empaquetado vertical*/
  vbox = gtk_vbox_new(FALSE, 0);

/*Creación widget usuario*/
  hbox=PackNewForm(FALSE,0, FALSE, TRUE, 8, 0, "Usuario:");
```

```
  gtk_box_pack_start(GTK_BOX(vbox), hbox,
                     FALSE, FALSE, 10);

/*Creación widget password*/
  hbox=PackNewForm(FALSE,0,FALSE,TRUE, 8, 1, "Passwd:");
  gtk_box_pack_start(GTK_BOX(vbox), hbox,
                     FALSE, FALSE, 10);

/*Crea una nueva tabla con dos columnas y un renglón*/
  table = gtk_table_new(1, 2, TRUE);
  gtk_table_set_col_spacings(GTK_TABLE(table), 10);
  gtk_box_pack_start(GTK_BOX(GTK_DIALOG(window_entry)
              ->action_area), table, TRUE, TRUE, 10);

/*Crea un Nuevo botón con la etiqueta conectar*/
  button = gtk_button_new_with_label("Conectar");

/*Cuando se da clic en el botón conectar manda a llamar la función
ConectaBase la cual realiza la conexión al servidor*/
  gtk_signal_connect(GTK_OBJECT(button), "clicked",
           GTK_SIGNAL_FUNC(ConectaBase),
                    window_entry);
  gtk_table_attach_defaults(GTK_TABLE(table),
                         button, 0, 1, 0, 1);

/*Crea un Nuevo botón con la etiqueta Cancelar*/
  button = gtk_button_new_with_label("Cancelar");
  gtk_table_attach_defaults(GTK_TABLE(table),
                         button, 1, 2, 0, 1);

/*Muestra todos los widgets*/
  gtk_widget_show_all(window_entry);
}
…
…
…
/*Función ConectaBase (se emplean las bibliotecas DB-Library)*/
gint ConectaBase(void) {
  int no_error;  /*Declaración de variables*/
  char d_tecla[10];

/*Manejo de errores de bibliotecas*/
  if (dbinit() == FAIL) {
  Messages("Error","Las Bibliotecas no sa ha podido
     inicializar", 0);
}

/*Permite tener el control del error*/
  dberrhandle(error);

/*Permite enviar mensajes al usuario*/
```

```
  dbmsghandle(msg);
  dbsetversion(DBVERSION_100); /*Versión de la Biblioteca*/

  login_u = dblogin();/*Login de usuario*/

/*Realiza la conexión al servidor*/
  DBSETLUSER(login_u, host.user);
  DBSETLPWD (login_u, host.clave);
  DBSETLENCRYPT(login_u, (DBBOOL)1);
  DBSETLAPP(login_u, "SABAD-SIAE");

/*Abre un canal de comunicación con el servidor*/
  dbproc1 = dbopen(login_u, "DGAE_F");
  if (dbproc1 == NULL) {
  Messages("Error",  "No se puede establecer la
              conexión con Central", 0);
}
  dbproc2 = dbopen(login_u, "SSRE_F");
  if (dbproc2 == NULL) {
  Messages("Error",  "No se puede establecer la
              conexión con Plantel", 0);
}

  dbcmd(dbproc1, "use master");

/*Si ocurre un error al momento de cambiar de base manda un
mensaje de error*/
  if (dbsqlexec(dbproc1) == FAIL) {
 Messages("Error",  "Error al Momento de Ejecutar", 0);
  }
  else {
  dbresults(dbproc1);
  dbnextrow(dbproc1);
/*Mensaje de bienvenida cuando la conexión tuvo éxito*/
  Messages("Conectado",  "Bienvenido,  Estas conectado
     al Servidor", 1);
}
}
```

Cuando se tiene éxito en la conexión al servidor de bases de datos aparecerá el siguiente menú el cual permite llevar a cabo todas las tareas de administración del servidor. Podemos elegir de entre las diferentes opciones que presenta.

Figura 4.4 Menú principal

Para construir la ventana de la figura 4.4 empleamos el siguiente código:

…
…
…

```
/*Crea un Nuevo botón para las auditorías*/
  btnaud = gtk_button_new ();

/*Crea un mensaje rápido*/
  gtk_tooltips_set_tip (tooltips, btnaud,
                    "Auditorias de las Bases de Datos",
               NULL);


/*Crea un Nuevo botón para los usuarios*/
  btnuser = gtk_button_new ();

/*Crea un mensaje*/
  gtk_tooltips_set_tip (tooltips, btnuser,
                "Altas y Cambios de Usuarios",
          NULL);

/*Cambia el tipo de cursor */
  gdk_window_set_cursor(window->
                window,gdk_cursor_new(GDK_DRAFT_LARGE));

/* Crea un Nuevo botón para las bases*/
  btnbase = gtk_button_new ();

/*Crea un mensaje rápido*/
  gtk_tooltips_set_tip (tooltips, btnbase,
                    "Crear, Modificar y Reorganizar
          Bases de Datos", NULL);
```

```
/* Crea un Nuevo botón para los respaldos*/
  btnresp = gtk_button_new ();

/*Crea un mensaje rápido*/
  gtk_tooltips_set_tip (tooltips, btnresp,
                   "Respaldos Completos y de Logs",
          NULL);

/* Crea un Nuevo botón para la calendarización*/
  btncal = gtk_button_new ();

/*Crea un mensaje*/
  gtk_tooltips_set_tip (tooltips, btncal,
                "Agenda de Actividades", NULL);

/*Coloca las imágenes y las etiquetas*/
  box1 = xpm_label_box(window,"audit.xpm","Auditorias");
  box2 = xpm_label_box(window, "user.xpm", "Usuarios");
  box3 = xpm_label_box(window,"bas.xpm","Bases");
  box4 = xpm_label_box(window,"resp.xpm","Respaldos");
  box5 = xpm_label_box(window,"calendarizacion.xpm",
                       "Calendario");

/*agrega las cajas a cada uno de los botones*/
  gtk_container_add (GTK_CONTAINER (btnaud), box1);
  gtk_container_add (GTK_CONTAINER (btnuser), box2);
  gtk_container_add (GTK_CONTAINER (btnbase), box3);
  gtk_container_add (GTK_CONTAINER (btnresp), box4);
  gtk_container_add (GTK_CONTAINER (button5), box5);

/*Crea una caja de empaquetado horizontal*/
  caja = gtk_hbox_new(FALSE,0);

/*Empaqueta los botones a la caja*/
  gtk_box_pack_start (GTK_BOX (caja),
                      btnaud, FALSE, FALSE, 0);

  gtk_box_pack_start (GTK_BOX (caja),
                      btnuser, FALSE, FALSE, 0);

  gtk_box_pack_start (GTK_BOX (caja),
                      btnbase, FALSE, FALSE, 0);

  gtk_box_pack_start (GTK_BOX (caja),
                      btnresp, FALSE, FALSE, 0);

  gtk_box_pack_start (GTK_BOX (caja),
                      button5, FALSE, FALSE, 0);
```

```
  gtk_widget_show(vbox);   /*Muestra widgets*/
  gtk_widget_show(box);
  gtk_widget_show(caja);
  gtk_widget_show(window);
  gtk_main ();
  return(0);
}
```

Si en la figura 4.4 elegimos la opción de auditorías aparecerá la siguiente ventana

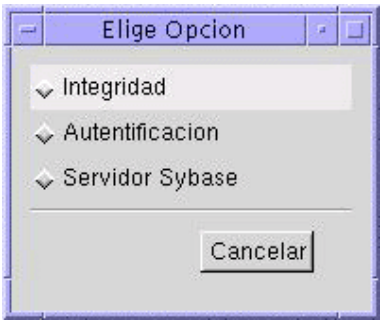


Figura 4.5 Ventana de opciones para la auditoría

Si elegimos la opción de integridad nos permitirá realizar las auditorías relacionadas con las bases de datos, la ventana que aparecerá es la siguiente:

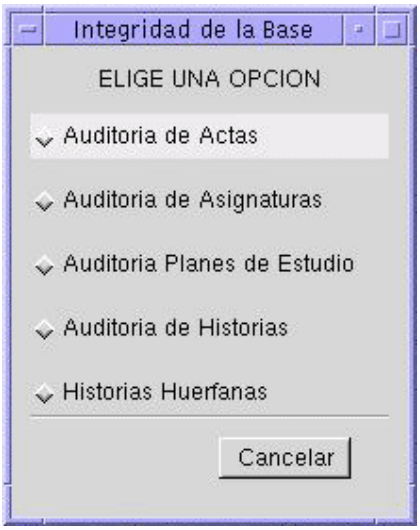


Figura 4.6 opciones para la integridad de las bases de datos

Cada una de las opciones que aparecen en la figura 4.6 tiene asociada una ventana, y son las siguientes:

- *Si se elige la opción de Auditoría de Actas aparecerá la siguiente ventana*

Figura 4.7 Auditoría de las actas

En la figura 4.7 se muestra una ventana la cual pide como datos de entrada la clave del plantel a auditar.

- *Si se elige la opción de Auditoría de Asignaturas aparecerá la siguiente ventana*

Figura 4.8 Auditoría de las asignaturas

Esta ventana contiene opciones a ejecutar, compara los datos de asignaturas de la base central contra las asignaturas de un plantel y viceversa, además revisa si existe ciclo y seriación para estas asignaturas.

- *Si se elige la opción de Auditoría de Planes de Estudio aparecerá la siguiente ventana*

Figura 4.9 Auditoría de los planes de estudio

En la figura 4.9 se muestra la ventana que permite realizar la Auditoría de los Planes de Estudio, se tiene la posibilidad de elegir un plantel a auditar o se puede elegir auditar todos los planteles de la UNAM.

- *Si se elige la opción de Auditoría de Historias aparecerá la siguiente ventana*

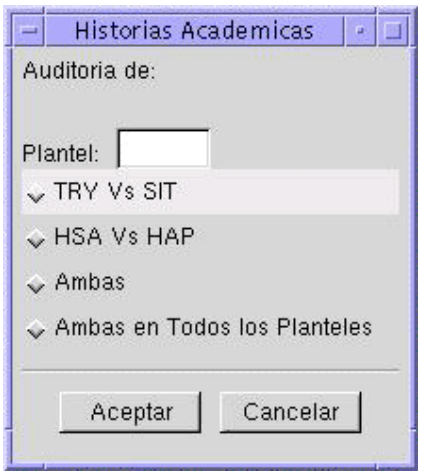


Figura 4.10 Auditoría de las Historias Académicas

En la figura 4.10 se muestra la ventana que realiza la auditoría de las historias académicas, revisa las posibles diferencias que hubiera tanto en los datos de situación e historia de un plantel en comparación con la trayectoria e historia de central, también verifica la ausencia de datos.

- *Si se elige la opción de Auditoría de Historias Huérfanas aparecerá la siguiente ventana*

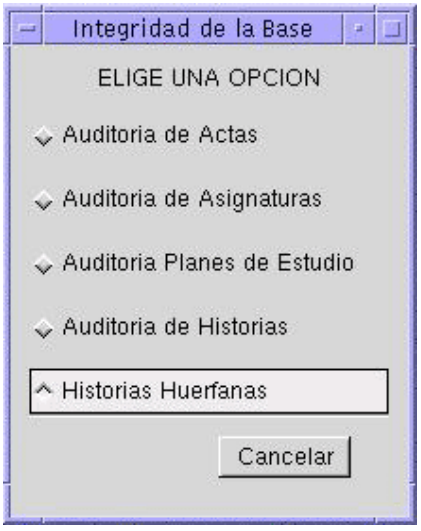


Figura 4.11 Auditoría de las Historias Huérfanas

Esta opción no tiene ventana asociada, la auditoría de historias huérfanas genera un reporte que informa de las historias que no tienen su respectiva trayectoria.

El código que genera estas ventanas se lista a continuación:

- *Para crear la ventana de la figura 4.5 se escribió el siguiente código*

```
/*Llamada a la función de opción para la auditoría*/
void opcion_aud(void) {

/*Se manda a llamar una función que se llama opciones aquí se
construye la siguiente ventana, utiliza banderas las cuales le
indican que ventana creara y que titulo llevara*/
  opciones(1,1,"Integridad","Autentificación","Servidor Sybase");
}

/*Función opciones recibe como parámetros de entrada la señal que
envía, el numero de botones que aparecerán, etiqueta 1 y etiqueta
2*/
gint opciones(int senal, int nbtn,
          char *label1, char *label2,
          char *label3) {
  GtkWidget *label;  /*Declaración de los widgets*/
  GtkWidget *box1;
  GtkWidget *box2;
  GtkWidget *button;
  GtkWidget *table;
  GSList *group = NULL;

/*Crea una nueva ventana con la opción seleccionada*/
  window_op = gtk_dialog_new();

/*Tamaño de la ventana*/
  gtk_widget_set_usize(window_op, 190, 140);

/*Titulo*/
  gtk_window_set_title(GTK_WINDOW(window_op),
                       "Elige opción");

/*Crea una caja de empaquetado vertical*/
  box1 = gtk_vbox_new (TRUE, 0);
  gtk_box_pack_start(GTK_BOX(GTK_DIALOG(window_op)->vbox),
               box1, FALSE, FALSE, 0);

/*Muestra el widget box1*/
  gtk_widget_show(box1);

/*Crea una caja de empaquetado vertical*/
  box2 = gtk_vbox_new (TRUE, 0);
```

```
  gtk_container_border_width (GTK_CONTAINER (box2), 0);
  gtk_box_pack_start (GTK_BOX (box1),
                      box2, FALSE, FALSE, 0);

/*Muestra el widget box2*/
  gtk_widget_show(box2);

/*Crea un Nuevo radio botón con la etiqueta 1*/
  button = gtk_radio_button_new_with_label (NULL, label1);

/*Si la variable senal tiene el valor de 1 entonces realizara lo
siguiente, llama a una función que se llama op_int_base*/
  if(senal==1) gtk_signal_connect(GTK_OBJECT(button),
               "toggled", GTK_SIGNAL_FUNC(op_int_base),
               NULL);

  gtk_box_pack_start (GTK_BOX (box2),
                      button, FALSE, FALSE, 0);

/*Muestra el botón*/
  gtk_widget_show(button);

/*Crea un grupo que pertenece al grupo de los radiobutton*/
  if(nbtn==1) {

/*Crea el grupo*/
group = gtk_radio_button_group (GTK_RADIO_BUTTON
                          (button));

/*Asigna etiqueta3*/
button = gtk_radio_button_new_with_label(group, label3);
gtk_box_pack_start (GTK_BOX (box2),
                    button, FALSE, FALSE, 0);

/*Muestra el botón*/
gtk_widget_show(button);
}

/*Crea una tabla la cual contendrá el botón cancelar*/
  table = gtk_table_new(1, 2, TRUE);  /*Crea una tabla de 2
columnas por 1 renglón*/
  gtk_box_pack_start(GTK_BOX(GTK_DIALOG(window_op)->action_area),
table, TRUE, TRUE, 10);
  gtk_widget_show(table); /*Muestra la tabla*/

/*Crea un Nuevo botón con la etiqueta de cancelar*/
  button = gtk_button_new_with_label("Cancelar");
  gtk_widget_show(button); /*Muestra el botón*/
```

```
/*Coloca el botón en la tabla*/
  gtk_table_attach_defaults(GTK_TABLE(table),
                          button, 1, 2, 0, 1);

/*Muestra la ventana creada con los widgets*/
  gtk_widget_show(window_op);
}
```

- *Para crear la ventana de la figura 4.6 se escribió el siguiente código*

```
/*Cuando se elige la opción de integridad de la base se llama a la
función op_int_base*/
void op_int_base(GtkWidget *widget, gpointer data) {
  if (GTK_TOGGLE_BUTTON(widget)->active) { /*Verifica que el botón
este presionado*/
gtk_widget_destroy(window_op); /*Cierra la ventana anterior*/
integridad_base();  /*Llama a la función integridad_base*/
  }
}
/*Función integridad_base construye la ventana con las opciones
para realizar las auditorías a las bases de datos*/
gint integridad_base(void) {
  GtkWidget *window_int; /*Declaración de widgets*/
  GtkWidget *label;
  GtkWidget *button;
  GtkWidget *table;
  GtkWidget *vbox1;
  GtkWidget *vbox2;
  GSList *group = NULL;

/*Crea una nueva ventana*/
  window_int = gtk_dialog_new();

/*Asigna tamaño a la ventana*/
  gtk_widget_set_usize(window_int, 210, 250);

/*Coloca el titulo a la ventana*/
  gtk_window_set_title(GTK_WINDOW(window_int),
                       "Integridad de la Base");

/*Crea una caja de empaquetado vertical*/
  vbox1 = gtk_vbox_new (FALSE, 0);
  gtk_box_pack_start (GTK_BOX (GTK_DIALOG (window_int)->
                    vbox), vbox1, FALSE, FALSE, 0);

/*Muestra la caja de empaquetado vertical*/
  gtk_widget_show (vbox1);
```

```
/*Crea una caja de empaquetado vertical*/
  vbox2 = gtk_vbox_new (FALSE, 10);
  gtk_box_pack_start (GTK_BOX (vbox1),
                      vbox2, FALSE, FALSE, 0);

/*Muestra la caja de empaquetado vertical*/
  gtk_widget_show (vbox2);

/*Coloca una etiqueta*/
  label = gtk_label_new("ELIGE UNA OPCION");
  gtk_box_pack_start (GTK_BOX (vbox2),
                      label, FALSE, FALSE, 0);

/*Muestra la etiqueta*/
  gtk_widget_show(label);

/*Crea un Nuevo radio botón con la etiqueta de Auditoría de
Actas*/
  button = gtk_radio_button_new_with_label (NULL,
                         "Auditoria de Actas");

/*Cuando se da clic en el botón manda a llamar a la función
op_aud_act*/
  gtk_signal_connect(GTK_OBJECT(button), "toggled",
           GTK_SIGNAL_FUNC(op_aud_act), NULL);
  gtk_box_pack_start (GTK_BOX (vbox2),
                      button, FALSE, FALSE, 0);

/*Muestra el botón*/
  gtk_widget_show(button);

/*Crea un grupo de radio botones*/
  group = gtk_radio_button_group (GTK_RADIO_BUTTON
                                (button));

/*Crea un Nuevo botón con la etiqueta de Auditoría de
Asignaturas*/
  button = gtk_radio_button_new_with_label (group, "Auditoria de
Asignaturas");

/*Cuando se da clic en el botón manda a llamar a la función
op_aud_asg*/
  gtk_signal_connect(GTK_OBJECT(button), "toggled",
           GTK_SIGNAL_FUNC(op_aud_asg), NULL);

  gtk_box_pack_start (GTK_BOX (vbox2),
              button, FALSE, FALSE, 0);
  gtk_widget_show(button); /*Muestra el botón*/

  group = gtk_radio_button_group (GTK_RADIO_BUTTON
                                (button));
```

```
/*Crea un Nuevo botón con la etiqueta de Auditoría Planes de
Estudio*/
  button = gtk_radio_button_new_with_label (group,
                    "Auditoria Planes de Estudio");

/*Cuando se da clic en el botón manda a llamar a la función
op_aud_asg*/
  gtk_signal_connect(GTK_OBJECT(button), "toggled",
           GTK_SIGNAL_FUNC(op_aud_pde), NULL);
  gtk_box_pack_start (GTK_BOX (vbox2),
                      button, FALSE, FALSE, 0);

/*Muestra el botón*/
  gtk_widget_show(button);

  group = gtk_radio_button_group (GTK_RADIO_BUTTON
                                  (button));

/*Crea un Nuevo botón con la etiqueta de Auditoría de Historias */
  button = gtk_radio_button_new_with_label (group, "Auditoria de
Historias");

/*Cuando se da clic en el botón manda a llamar a la función
op_aud_asg*/
  gtk_signal_connect(GTK_OBJECT(button), "toggled",
           GTK_SIGNAL_FUNC(op_aud_hsa), NULL);
  gtk_box_pack_start (GTK_BOX (vbox2),
                      button, FALSE, FALSE, 0);

  gtk_widget_show(button); /*Muestra el botón*/

  group = gtk_radio_button_group (GTK_RADIO_BUTTON
                                  (button));

/*Crea un Nuevo botón con la etiqueta de Historias Huérfanas */
  button = gtk_radio_button_new_with_label (group,
                                  "Historias Huérfanas");
  gtk_box_pack_start (GTK_BOX (vbox2),
                      button, FALSE, FALSE, 0);

  gtk_widget_show(button); /*Muestra el botón*/

  group = gtk_radio_button_group (GTK_RADIO_BUTTON
                                  (button));
  button = gtk_radio_button_new(group);
  gtk_toggle_button_set_state (GTK_TOGGLE_BUTTON (button),
                               TRUE);
```

```
/*Crea una tabla de dos columnas por un renglón*/
  table = gtk_table_new(1, 2, TRUE);
  gtk_box_pack_start(GTK_BOX(GTK_DIALOG(window_int)->
                     action_area),table, TRUE, TRUE, 10);

  gtk_widget_show(table); /*Muestra la tabla*/

/*Crea un Nuevo botón con la etiqueta de Cancelar*/
  button = gtk_button_new_with_label("Cancelar");

/*Coloca el botón en la tabla*/
  gtk_table_attach_defaults(GTK_TABLE(table),
                            button, 1, 2, 0, 1);

  gtk_widget_show(button); /*Muestra el botón*/

  gtk_widget_show(window_int); /*Muestra la ventana*/
}
```

- *Para crear la ventana de la figura 4.7 se escribió el siguiente código*

```
/*Cuando se da clic en el botón auditoría de actas llama a la
función op_aud_act*/
void op_aud_act(GtkWidget *widget, gpointer data) {
/*Verifica que el botón este presionado*/
  if (GTK_TOGGLE_BUTTON(widget)->active) {
/*Hace la llamada a la función audit_act*/
audit_act();
  }
}

gint audit_act(void) {
  GtkWidget *window_acta; /*Declaración de los widgets*/
  GtkWidget *audact;
  GtkWidget *table;
  GtkWidget *button;
  GSList *group = NULL;

/*Crea una nueva ventana*/
  window_acta = gtk_dialog_new();

/*Asigna tamaño a la ventana*/
  gtk_widget_set_usize(window_acta, 200, 100);

/*Asigna Titulo a la ventana*/
  gtk_window_set_title(GTK_WINDOW(window_acta),
                       "Auditoria de Actas");
```

```
/*Crea una tabla de dos columnas por un renglón*/
  table = gtk_table_new(1,2,FALSE);
  gtk_box_pack_start(GTK_BOX(GTK_DIALOG(window_acta)->
                     vbox), table, TRUE, TRUE, 0);
  gtk_widget_show(table); /*Muestra la tabla*/

/*Crea una nueva etiqueta*/
  audact = gtk_label_new("Clave del Plantel: ");

/*Coloca la etiqueta en la tabla*/
  gtk_table_attach_defaults(GTK_TABLE(table),
                            audact, 0, 1, 0, 1);
  gtk_widget_show(audact); /*Muestra la etiqueta*/

/*Crea un campo de texto editable*/
  audact = gtk_entry_new();

/*Asigna tamaño al campo de texto*/
  gtk_widget_set_usize(audact, 30, -1);
  gtk_table_attach_defaults(GTK_TABLE(table),
                            audact, 1, 2, 0, 1);
  gtk_widget_show(audact); /*Muestra el campo de texto*/

/*Crea una nueva tabla con dos columnas y un renglón*/
  table = gtk_table_new(1, 2, TRUE);
  gtk_box_pack_start(GTK_BOX(GTK_DIALOG(window_acta)->
                   action_area), table, TRUE, TRUE, 10);
  gtk_widget_show(table); /*Muestra la tabla*/

/*Crea un Nuevo botón con la etiqueta Aceptar*/
  button =gtk_button_new_with_label("Aceptar");

/*Coloca el botón en la tabla*/
  gtk_table_attach_defaults(GTK_TABLE(table),
                            button, 0, 1, 0, 1);
  gtk_widget_show(button); /*Muestra el botón*/

/*Crea un Nuevo botón con la etiqueta Cancelar*/
  button = gtk_button_new_with_label("Cancelar");

/*Coloca el botón en la tabla
  gtk_table_attach_defaults(GTK_TABLE(table),
                            button, 1, 2, 0, 1);
  gtk_widget_show(button); /*Muestra el botón*/

  gtk_widget_show(window_acta); /*Muestra la ventana*/
}
```

- *Para crear la ventana de la figura 4.8 se escribió el siguiente código*

```
void op_aud_asg(GtkWidget *widget, gpointer data) {
/*Verifica que el botón este presionado*/
  if (GTK_TOGGLE_BUTTON(widget)->active) {

/*Hace una llamada a la función audit_asg*/
audit_asg();
  }
}

gint audit_asg(void) {
  GtkWidget *window_asg; /*Declaración de los widgets*/
  GtkWidget *audasg;
  GtkWidget *table;
  GtkWidget *button;
  GSList *group = NULL;

/*Creación de una nueva ventana*/
  window_asg = gtk_dialog_new();

/*Asigna tamaño a la ventana*/
  gtk_widget_set_usize(window_asg, 230, 160);
/*Asigna titulo a la ventana*/
  gtk_window_set_title(GTK_WINDOW(window_asg),
                       "Auditoria de Asignaturas");

/*Crea una nueva tabla de una columna por cuatro renglones*/
  table = gtk_table_new(4,1,FALSE);
  gtk_box_pack_start(GTK_BOX(GTK_DIALOG(window_asg)->vbox),
                          table, TRUE, TRUE, 5);
  gtk_widget_show(table);/* Muestra la tabla*/

/*Crea una nueva etiqueta*/
  audasg = gtk_label_new("Comparacion de: ");

/*Coloca la etiqueta en la tabla*/
  gtk_table_attach_defaults(GTK_TABLE(table),
                            audasg, 0, 1, 0, 1);
  gtk_widget_show(audasg); /*Muestra la etiqueta*/

/*Crea un Nuevo radio botón*/
  button = gtk_radio_button_new_with_label (NULL,
                   "Asignaturas Central Vs Plantel");

/*Coloca el botón en la tabla*/
  gtk_table_attach_defaults(GTK_TABLE(table),
                            button, 0, 1, 1, 2);
  gtk_widget_show(button); /*Muestra el botón*/
```

```
/*Crea un Nuevo grupo de radio botones*/
  group = gtk_radio_button_group (GTK_RADIO_BUTTON
                                  (button));

/*Crea un Nuevo radio botón*/
  button = gtk_radio_button_new_with_label (group,
                    "Datos entre APP vs CCL y SER");

/*Coloca el botón en la tabla*/
  gtk_table_attach_defaults(GTK_TABLE(table),
                              button, 0, 1, 2, 3);
  gtk_widget_show(button); /*Muestra el botón*/

  group = gtk_radio_button_group (GTK_RADIO_BUTTON
                                  (button));

/*Crea un Nuevo radio botón*/
  button = gtk_radio_button_new_with_label (group,
                    "Asignaturas con Planes no 900");

/*Coloca el botón en la tabla*/
  gtk_table_attach_defaults(GTK_TABLE(table),
                              button, 0, 1, 3, 4);
  gtk_widget_show(button); /*Muestra el botón*/

  group = gtk_radio_button_group (GTK_RADIO_BUTTON
                                  (button));

/*Crea un Nuevo radio botón*/
  button = gtk_radio_button_new(group);
  gtk_toggle_button_set_state (GTK_TOGGLE_BUTTON (button),
                               TRUE);

/*Crea una nueva tabla de dos columnas por un renglón*/
  table = gtk_table_new(1, 2, TRUE);

/*Coloca la tabla en la ventana*/
  gtk_box_pack_start(GTK_BOX(GTK_DIALOG(window_asg)->
                     action_area), table, TRUE, TRUE, 10);
  gtk_widget_show(table); /*Muestra la tabla*/

/*Crea un Nuevo botón con la etiqueta Aceptar*/
  button =gtk_button_new_with_label("Aceptar");

/*Coloca el botón en la tabla*/
  gtk_table_attach_defaults(GTK_TABLE(table),
                              button, 0, 1, 0, 1);
  gtk_widget_show(button); /*Muestra el botón*/

/*Crea un Nuevo botón con la etiqueta Cancelar*/
  button = gtk_button_new_with_label("Cancelar");
```

```
/*Coloca el botón en la tabla*/
  gtk_table_attach_defaults(GTK_TABLE(table),
                            button, 1, 2, 0, 1);
  gtk_widget_show(button); /*Muestra el botón*/

  gtk_widget_show(window_asg); /*Muestra la ventana*/
}
```

- *Para crear la ventana de la figura 4.9 se escribió el siguiente código*

```
/*Cuando se da clic en la opción auditoría de los planes de
estudio se manda a llamar a la función op_aud_pde*/
void op_aud_pde(GtkWidget *widget, gpointer data) {
/*Verifica que se haya presionado el botón*/
  if (GTK_TOGGLE_BUTTON(widget)->active) {

/*Manda a llamar al a función audit_pde*/
audit_pde();
  }
}

gint audit_pde(void) {
  GtkWidget *window_pde; /*Declaración de widgets*/
  GtkWidget *audpde;
  GtkWidget *table;
  GtkWidget *button;
  GSList *group = NULL;

/*Crea una nueva ventana*/
  window_pde = gtk_dialog_new();

/*Asigna tamaño a la ventana*/
  gtk_widget_set_usize(window_pde, 200, 150);

/*Asigna titulo a la ventana*/
  gtk_window_set_title(GTK_WINDOW(window_pde),
                       "Planes de Estudio");

/*Crea una nueva tabla de dos columnas por tres renglones*/
  table = gtk_table_new(3,2,FALSE);

/*Coloca la tabla en la ventana*/
  gtk_box_pack_start(GTK_BOX(GTK_DIALOG(window_pde)->vbox),
                     table, TRUE, TRUE, 5);
  gtk_widget_show(table); /*Muestra la tabla*/

/*Crea una nueva etiqueta*/
  audpde = gtk_label_new("Auditoria de: ");
```

```
/*Coloca la etiqueta en la tabla*/
  gtk_table_attach_defaults(GTK_TABLE(table),
                            audpde, 0, 1, 0, 1);
  gtk_widget_show(audpde); /*Muestra la etiqueta*/

/*Crea un Nuevo campo de texto editable*/
  plt = gtk_entry_new();

/*Asigna tamaño al campo de texto editable*/
  gtk_widget_set_usize(plt, 30, -1);

/*Coloca el campo de texto en la tabla*/
  gtk_table_attach_defaults(GTK_TABLE(table),
                            plt, 1, 2, 1, 2);

/*Crea un nuevo radio botón*/
  button = gtk_radio_button_new_with_label (NULL,
                          "Un plantel: ");

/*Coloca el botón en la tabla*/
  gtk_table_attach_defaults(GTK_TABLE(table),
                            button, 0, 1, 1, 2);
  gtk_widget_show(button); /*Muestra el botón*/

/*Crea un Nuevo grupo*/
  group = gtk_radio_button_group (GTK_RADIO_BUTTON
                               (button));

/*Crea un Nuevo radio botón*/
  button = gtk_radio_button_new_with_label (group,
                             "Todos los Planteles");

/*Coloca el botón en la tabla*/
  gtk_table_attach_defaults(GTK_TABLE(table),
                            button, 0, 1, 2, 3);
  gtk_widget_show(button); /*Muestra el botón*/

  group = gtk_radio_button_group (GTK_RADIO_BUTTON
                               (button));

/*Crea un Nuevo radio botón*/
  button = gtk_radio_button_new(group);
  gtk_toggle_button_set_state (GTK_TOGGLE_BUTTON (button),
                               TRUE);

/*Crea una nueva tabla de dos columnas por un renglón*/
  table = gtk_table_new(1, 2, TRUE);
  gtk_box_pack_start(GTK_BOX(GTK_DIALOG(window_pde)-
                   >action_area), table, TRUE, TRUE, 10);
  gtk_widget_show(table); /*Muestra la tabla*/
```

```
/*Crea un nuevo botón con la etiqueta Aceptar*/
  button =gtk_button_new_with_label("Aceptar");

/*Coloca el botón en la tabla*/
  gtk_table_attach_defaults(GTK_TABLE(table),
                            button, 0, 1, 0, 1);
  gtk_widget_show(button); /*Muestra el botón*/

/*Crea un Nuevo botón con la etiqueta Cancelar*/
  button = gtk_button_new_with_label("Cancelar");

/*Coloca el botón en la tabla*/
  gtk_table_attach_defaults(GTK_TABLE(table),
                            button, 1, 2, 0, 1);
  gtk_widget_show(button); /*Muestra el botón*/

  gtk_widget_show(window_pde); /*Muestra la ventana*/
}
```

- *Para crear la ventana de la figura 4.10 se escribió el siguiente código*

```
/*Cuando se da clic en el botón de Auditar Historias Académicas se
manda a llamar la función op_aud_hsa*/
void op_aud_hsa(GtkWidget *widget, gpointer data) {

/*Verifica que el botón este presionado*/
  if (GTK_TOGGLE_BUTTON(widget)->active) {

/*Llama la función audit_hsa*/
audit_hsa();
  }
}

/*Función audit_hsa*/
gint audit_hsa(void) {
  GtkWidget *window_hsa; /*Declaración de widgets*/
  GtkWidget *audhsa;
  GtkWidget *table;
  GtkWidget *button;
  GtkWidget *vbox;
  GSList *group = NULL;

/*Crea una nueva ventana*/
  window_hsa = gtk_dialog_new();

/*Asigna tamaño a la ventana*/
  gtk_widget_set_usize(window_hsa, 210, 220);

/*Asigna titulo a la nueva ventana*/
  gtk_window_set_title(GTK_WINDOW(window_hsa),
                       "Historias Académicas");
```

```
/*Crea una nueva caja de empaquetado vertical*/
  vbox = gtk_vbox_new(FALSE,0);

/*Coloca el empaquetado en la ventana*/
  gtk_box_pack_start(GTK_BOX(GTK_DIALOG(window_hsa)->vbox),
                     vbox, TRUE, TRUE, 0);
  gtk_widget_show(vbox); /*Muestra la ventana*/

/*Crea una nueva etiqueta*/
  audhsa = gtk_label_new("Auditoria de: ");

/*Coloca la etiqueta en el empaquetado*/
  gtk_box_pack_start (GTK_BOX (vbox),
                      audhsa, FALSE, FALSE, 0);
  gtk_widget_show(audhsa); /*Muestra la etiqueta*/

/*Crea una nueva tabla de 4 columnas por dos renglones*/
  table = gtk_table_new(2,4,TRUE);

/*Coloca la tabla en el empaquetado*/
  gtk_box_pack_start (GTK_BOX (vbox),
                      table, FALSE, FALSE, 0);
  gtk_widget_show(table); /*Muestra la tabla*/

/*Crea una nueva etiqueta*/
  audhsa = gtk_label_new("Auditoria de: ");

/*Coloca la etiqueta en la tabla*/
gtk_table_attach_defaults(GTK_TABLE(table),
                                audhsa, 0, 1, 0, 1);
gtk_widget_show(audhsa); /*Muestra la etiqueta*/

/*Crea una nueva etiqueta*/
  audhsa = gtk_label_new("Plantel: ");

/*Coloca la etiqueta en la tabla*/
  gtk_table_attach_defaults(GTK_TABLE(table),
                            audhsa, 0, 1, 1, 2);
  gtk_widget_show(audhsa); /*Muestra la etiqueta*/

/*Crea un Nuevo campo de texto editable*/
  audhsa = gtk_entry_new();

/*Coloca el campo de texto en la tabla*/
  gtk_table_attach_defaults(GTK_TABLE(table),
                            audhsa, 1, 2, 1, 2);
  gtk_widget_show(audhsa); /*Muestra el campo de texto*/

/*Crea una nueva tabla de una columna por cuatro renglones*/
  table = gtk_table_new(4,1,FALSE);
```

```
/*Coloca la nueva tabla en una celda de la tabla anterior*/
  gtk_box_pack_start (GTK_BOX (vbox),
                      table, FALSE, FALSE, 0);
  gtk_widget_show(table); /*Muestra la tabla*/

/*Crea un Nuevo radio botón con la etiqueta TRY Vs SIT*/
  button = gtk_radio_button_new_with_label (NULL,
                                        "TRY Vs SIT ");

/*Coloca el botón en la tabla*/
  gtk_table_attach_defaults(GTK_TABLE(table),
                            button, 0, 1, 0, 1);
  gtk_widget_show(button); /*Muestra el botón*/

/*Crea un Nuevo grupo de radio botones*/
  group = gtk_radio_button_group (GTK_RADIO_BUTTON
                                (button));

/*Crea un Nuevo radio botón con la etiqueta HAS Vs HAP*/
  button = gtk_radio_button_new_with_label (group,
                                        "HSA Vs HAP");

/*Coloca el botón en la tabla*/
  gtk_table_attach_defaults(GTK_TABLE(table),
                            button, 0, 1, 1, 2);
  gtk_widget_show(button); /*Muestra el botón*/

  group = gtk_radio_button_group (GTK_RADIO_BUTTON
                                (button));

/*Crea un Nuevo radio botón con la etiqueta Ambas*/
  button = gtk_radio_button_new_with_label (group,
                                     "Ambas");

/*Coloca el botón en la tabla*/
  gtk_table_attach_defaults(GTK_TABLE(table),
                            button, 0, 1, 2, 3);
  gtk_widget_show(button);

  group = gtk_radio_button_group (GTK_RADIO_BUTTON
                                (button));

/*Crea un Nuevo radio botón con la etiqueta Ambas en Todos los
Planteles*/
  button = gtk_radio_button_new_with_label (group,
                    "Ambas en Todos los Planteles");

/*Coloca el botón en la tabla*/
  gtk_table_attach_defaults(GTK_TABLE(table),
                            button, 0, 1, 3, 4);
  gtk_widget_show(button); /*Muestra el botón*/
```

```
  group = gtk_radio_button_group (GTK_RADIO_BUTTON
                                  (button));


  button = gtk_radio_button_new(group);
  gtk_toggle_button_set_state (GTK_TOGGLE_BUTTON (button),
                               TRUE);

/*Crea una nueva tabla de dos columnas por un renglón*/
  table = gtk_table_new(1, 2, TRUE);

/*Coloca la tabla en la ventana*/
  gtk_box_pack_start(GTK_BOX(GTK_DIALOG(window_hsa)->
                     action_area), table, TRUE, TRUE, 10);
  gtk_widget_show(table); /*Muestra la tabla*/

/*Crea un Nuevo botón con la etiqueta Aceptar*/
  button =gtk_button_new_with_label("Aceptar");

/*Coloca el botón en la tabla*/
  gtk_table_attach_defaults(GTK_TABLE(table),
                            button, 0, 1, 0, 1);
  gtk_widget_show(button); /*Muestra el botón*/

/*Crea un Nuevo botón con la etiqueta Cancelar*/
  button = gtk_button_new_with_label("Cancelar");

/*Coloca el botón en la tabla*/
  gtk_table_attach_defaults(GTK_TABLE(table),
                            button, 1, 2, 0, 1);
  gtk_widget_show(button); /*Muestra el botón*/

  gtk_widget_show(window_hsa); /*Muestra la ventana*/
}
```

- *Para crear la ventana de la figura 4.11 se escribió el siguiente código*

```
/*Función integridad_base*/
gint integridad_base(void) {
  GtkWidget *window_int; /*Declaración de widgets*/
  GtkWidget *label;
  GtkWidget *button;
  GtkWidget *table;
  GtkWidget *vbox1;
  GtkWidget *vbox2;
  GSList *group = NULL;

/*Crea una nueva ventana*/
  window_int = gtk_dialog_new();
```

```
/*Asigna tamaño a la ventana*/
  gtk_widget_set_usize(window_int, 210, 250);


/*Crea una caja de empaquetado vertical*/
  vbox2 = gtk_vbox_new (FALSE, 10);

/*Coloca la caja en la ventana*/
  gtk_box_pack_start (GTK_BOX (vbox1),
                      vbox2, FALSE, FALSE, 0);

/*Muestra la caja de empaquetado*/
  gtk_widget_show (vbox2);

/*Crea un Nuevo radio botón con la etiqueta Historias Huérfanas*/
button = gtk_radio_button_new_with_label (group,
                              "Historias Huérfanas");

/*Coloca el botón en la caja de empaquetado*/
  gtk_box_pack_start (GTK_BOX (vbox2),
                      button, FALSE, FALSE, 0);

/*Muestra el botón*/
  gtk_widget_show(button);

/*Crea una nueva tabla de dos columnas por un renglón*/
  table = gtk_table_new(1, 2, TRUE);

/*Coloca la tabla en la ventana*/
  gtk_box_pack_start(GTK_BOX(GTK_DIALOG(window_int)->
                    action_area), table, TRUE, TRUE, 10);

/*Muestra la tabla*/
  gtk_widget_show(table);

/*Crea un nuevo radio botón con la etiqueta Cancelar*/
  button = gtk_button_new_with_label("Cancelar");

/*Coloca el botón en la tabla*/
  gtk_table_attach_defaults(GTK_TABLE(table),
                            button, 1, 2, 0, 1);

/*Muestra el botón*/
  gtk_widget_show(button);

/*Muestra la ventana*/
  gtk_widget_show(window_int);
```

Los códigos anteriores permiten construir las ventanas para realizar las auditorías a las bases de datos, la siguiente opción que encontramos en la figura 4.4 es la de usuarios, la cual permite crear un nuevo usuario o modificar un usuario existente.

Cuando se elige la opción de usuarios la ventana que aparecerá será la siguiente:

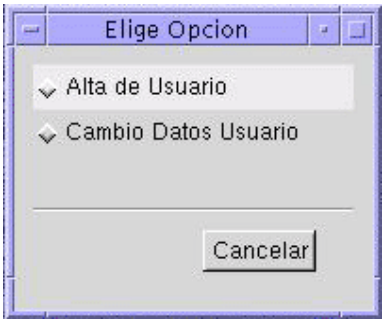


Figura 4.12 Ventana de opciones para usuarios

Si en la figura 4.12 elegimos la opción de Alta de usuario aparecerá la siguiente ventana

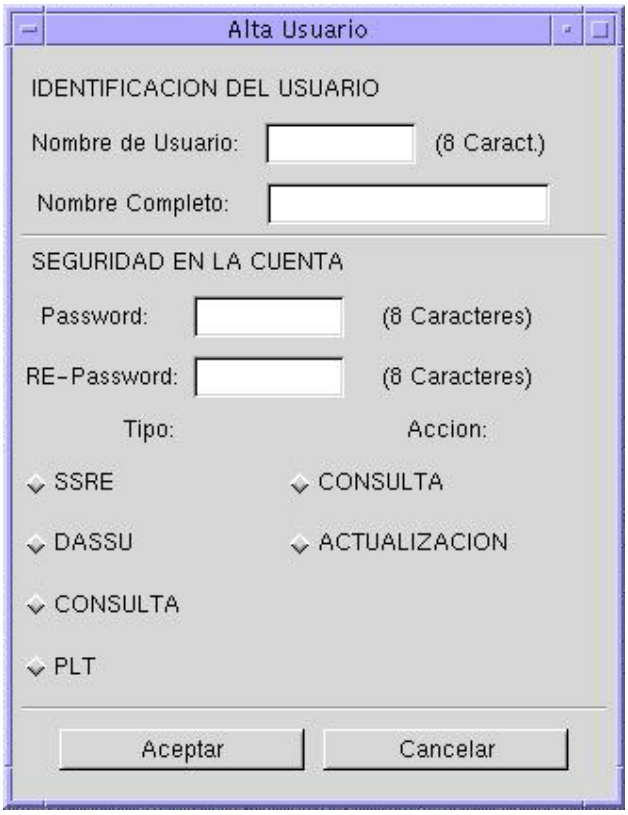


Figura 4.13 Ventana para alta de usuarios

Aquí se tienen que capturar los datos del nuevo usuario que se agregará a los servidores de bases de datos, como datos de entrada pide el nombre de usuario que tendrá en el servidor, su nombre completo y su password, se debe seleccionar un tipo de usuario, que es el grupo al que el usuario esté asignado y la acción es lo que puede realizar dentro del servidor, si son consultas o actualizaciones.

Cuando se hayan capturado los datos requeridos se tiene que dar clic en el botón aceptar y automáticamente el usuario se agregará a los servidores de bases de datos, con sus respectivos grupos y en el caso de los usuarios de plantel, al plantel asignado.

Tenemos la posibilidad de realizar cambios en los datos del usuario, si seleccionamos la opción cambio datos usuario de la figura 4.12 la ventana que aparecerá será la siguiente:

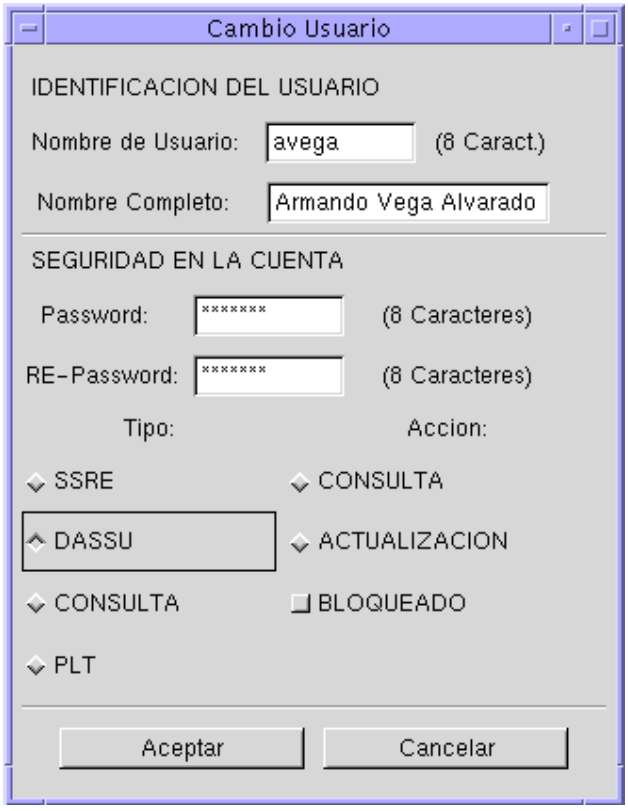


Figura 4.14 Ventana para alta de usuarios

La ventana de la figura 4.14 es similar a la de alta de usuario, pero ésta contiene los datos del usuario que se agregó al servidor de bases de datos, el campo que no se puede modificar es el de nombre de usuario, también aparece una opción que permite bloquear o desbloquear a un usuario, un usuario no se borra de la base de datos sólo se bloquea.

Para crear las ventanas para los usuarios se empleó el siguiente código:

- *Para crear la ventana de la figura 4.12 se escribió el siguiente código*

```
…
…
…
/*Crea un nuevo botón*/
btnuser = gtk_button_new ();

/*Cuando se da clic en el botón se manda a llamar a la función
opcion_usr*/
  gtk_signal_connect (GTK_OBJECT (btnuser), "clicked",
                      GTK_SIGNAL_FUNC (opcion_usr), NULL);
…
…
…

/*Función para la opción de usuario*/
void opcion_usr(void) {

/*Manda a llamar a la función opciones*/
  opciones(0,0,"Alta de Usuario",
          "Cambio Datos Usuario","");
}

/*Función que crea los botón de opción esta función es un molde
que permite crear varios botones de opción*/
gint opciones(int senal, int nbtn,
          char *label1, char *label2,
          char *label3) {
  GtkWidget *label;  /*Declaración de widgets*/
  GtkWidget *box1;
  GtkWidget *box2;
  GtkWidget *button;
  GtkWidget *table;
  GSList *group = NULL;


/*Crea una nueva ventana*/
  window_op = gtk_dialog_new();

/*Asigna tamaño a la ventana*/
  gtk_widget_set_usize(window_op, 190, 140);

/*Asigna un titulo a la ventana*/
  gtk_window_set_title(GTK_WINDOW(window_op),
                       "Elige opción");

/*Crea una nueva caja de empaquetado vertical*/
  box1 = gtk_vbox_new (TRUE, 0);
```

```
/*Coloca la caja en la ventana*/
  gtk_box_pack_start(GTK_BOX(GTK_DIALOG(window_op)->vbox),
                     box1, FALSE, FALSE, 0);

/*Muestra la caja*/
  gtk_widget_show(box1);

/*Crea una caja de empaquetado vertical*/
  box2 = gtk_vbox_new (TRUE, 0);

/*Coloca la caja2 en la caja1*/
  gtk_box_pack_start (GTK_BOX (box1),
                      box2, FALSE, FALSE, 0);

/*Muestra la caja*/
  gtk_widget_show(box2);

/*Crea un nuevo radio botón*/
  button = gtk_radio_button_new_with_label (NULL, label1);

/*Se coloca el botón en la caja de empaquetado*/
  gtk_box_pack_start (GTK_BOX (box2),
                      button, FALSE, FALSE, 0);

/*Muestra el botón
  gtk_widget_show(button);

/*Crea una nueva tabla con dos columnas y un renglón*/
  table = gtk_table_new(1, 2, TRUE);

/*Coloca la tabla en la ventana*/
  gtk_box_pack_start(GTK_BOX(GTK_DIALOG(window_op)->
                     action_area), table, TRUE, TRUE, 10);

/*Muestra la tabla*/
  gtk_widget_show(table);

/*Crea un nuevo botón con la etiqueta cancelar*/
  button = gtk_button_new_with_label("Cancelar");

  gtk_widget_show(button); /*Muestra el botón*/

/*Coloca el botón en la tabla*/
  gtk_table_attach_defaults(GTK_TABLE(table),
                            button, 1, 2, 0, 1);

  gtk_widget_show(window_op); /*Muestra la ventana*/
}
```

- *Para crear la ventana de la figura 4.13 se escribió el siguiente código*

```
/*Función op_alta*/
void op_alta(GtkWidget *widget, gpointer data) {
/*Verifica que el botón se haya presionado*/
  if (GTK_TOGGLE_BUTTON(widget)->active) {

/*Cierra la ventana anterior*/
gtk_widget_destroy(window_op);

/*Llama a la función alta_usr*/
alta_usr();
  }
}

/*Función alta_usr*/
gint alta_usr(void) {
  GtkWidget *window_alta; /*Declaración de widgets*/
  GtkWidget *label;
  GtkWidget *separator;
  GtkWidget *hbox;
  GtkWidget *hbox1;
  GtkWidget *vbox;
  GtkWidget *vbox1;
  GtkWidget *vbox2;
  GtkWidget *box;
  GtkWidget *table;
  GtkWidget *button;
  GtkWidget *button1;
  GSList *group = NULL;
  GSList *group1 = NULL;

/*Crea una nueva ventana*/
  window_alta = gtk_dialog_new();

/*Asigna tamaño a la ventana*/
  gtk_widget_set_usize(window_alta, 320, 400);

/*Asigna titulo a la ventana*/
  gtk_window_set_title(GTK_WINDOW(window_alta),
                       "Alta Usuario");

/*Crea una caja de empaquetado vertical*/
  vbox = gtk_vbox_new(FALSE, 0);

/*Muestra la caja*/
  gtk_widget_show(vbox);
```

```
/*Llama a una función que se llama form_alta_cambio_usr y le
asigna los valores correspondientes*/
  hbox = form_alta_cambio_usr(FALSE,0,FALSE,
     TRUE,5,0,"IDENTIFICACION DEL USUARIO","");

/*El valor que recibe se almacena en hbox y lo coloca en la caja
de empaquetado*/
  gtk_box_pack_start(GTK_BOX(vbox), hbox, FALSE, FALSE, 5);

/*Muestra los valores de hbox*/
  gtk_widget_show(hbox);

/*Llama a una función que se llama form_alta_cambio_usr y le
asigna los valores correspondientes*/
  hbox = form_alta_cambio_usr(FALSE,0,FALSE,
     TRUE,5,1,"Nombre de Usuario: ","(8 Caract.)");

/*El valor que recibe se almacena en hbox y lo coloca en la caja
de empaquetado*/
  gtk_box_pack_start(GTK_BOX(vbox), hbox, FALSE, FALSE, 5);

/*Muestra los valores de hbox*/
  gtk_widget_show(hbox);

/*Llama a una función que se llama form_alta_cambio_usr y le
asigna los valores correspondientes*/
  hbox = form_alta_cambio_usr(FALSE,0,FALSE,
     TRUE,8,2,"Nombre Completo: ","");

/*El valor que recibe se almacena en hbox y lo coloca en la caja
de empaquetado*/
  gtk_box_pack_start(GTK_BOX(vbox), hbox, FALSE, FALSE, 5);

/*Muestra los valores de hbox*/
  gtk_widget_show(hbox);

  separator = gtk_hseparator_new ();
  gtk_box_pack_start (GTK_BOX (vbox), separator, FALSE, TRUE, 0);
  gtk_widget_show(separator);

/*Llama a una función que se llama form_alta_cambio_usr y le
asigna los valores correspondientes*/
  hbox = form_alta_cambio_usr(FALSE,0,FALSE,
     TRUE,5,0,"SEGURIDAD EN LA CUENTA","");

/*El valor que recibe se almacena en hbox y lo coloca en la caja
de empaquetado*/
  gtk_box_pack_start(GTK_BOX(vbox), hbox, FALSE, FALSE, 5);

  gtk_widget_show(hbox); /*Muestra los valores de hbox*/
```

```
/*Llama a una función que se llama form_alta_cambio_usr y le
asigna los valores correspondientes*/
  hbox = form_alta_cambio_usr(FALSE,0,FALSE,
     TRUE,10,3,"Password: ","(8 Caracteres)");
/*El valor que recibe se almacena en hbox y lo coloca en la caja
de empaquetado*/
  gtk_box_pack_start(GTK_BOX(vbox), hbox, FALSE, FALSE, 5);

/*Muestra los valores de hbox*/
  gtk_widget_show(hbox);

/*Llama a una función que se llama form_alta_cambio_usr y le
asigna los valores correspondientes*/
  hbox = form_alta_cambio_usr(FALSE,0,FALSE,
     TRUE,2,4,"RE-Password: ","(8 Caracteres)");

/*El valor que recibe se almacena en hbox y lo coloca en la caja
de empaquetado*/
  gtk_box_pack_start(GTK_BOX(vbox), hbox, FALSE, FALSE, 5);

/*Muestra los valores de hbox*/
  gtk_widget_show(hbox);

/*Llama a una función que se llama form_alta_cambio_usr y le
asigna los valores correspondientes*/
  /*  hbox = form_alta_cambio_usr(FALSE,0,FALSE,
     TRUE,5,0,"Tipo:","");

/*El valor que recibe se almacena en hbox y lo coloca en la caja
de empaquetado*/
  gtk_box_pack_start(GTK_BOX(vbox), hbox, FALSE, FALSE, 5);

/*Muestra los valores de hbox*/
  gtk_widget_show(hbox);

/*Crea una nueva tabla de dos columnas por cinco renglones*/
  table = gtk_table_new(5, 2, FALSE);

/*Muestra la tabla*/
  gtk_widget_show(table);

/*Coloca la tabla en la caja de empaquetado*/
  gtk_box_pack_start(GTK_BOX(vbox), table, TRUE, TRUE, 0);

/*Crea una etiqueta*/
  label = gtk_label_new("Tipo:");

/*Coloca la etiqueta en la tabla*/
  gtk_table_attach_defaults(GTK_TABLE(table),
                            label, 0, 1, 0, 1);
```

```
/*Muestra la etiqueta*/
  gtk_widget_show(label);

/*Crea un nuevo radio botón con la etiqueta SSRE*/
  button = gtk_radio_button_new_with_label (NULL, "SSRE");

/*Coloca el botón en la tabla*/
  gtk_table_attach_defaults(GTK_TABLE(table),
                            button, 0, 1, 1, 2);

/*Muestra el botón*/
  gtk_widget_show(button);

/*Crea un nuevo grupo para los radio botones*/
  group = gtk_radio_button_group (GTK_RADIO_BUTTON
                                  (button));

/*Crea un nuevo radio botón con la etiqueta DASSU*/
  button = gtk_radio_button_new_with_label (group,
                                            "DASSU");

/*Coloca el botón en la tabla*/
  gtk_table_attach_defaults(GTK_TABLE(table),
                            button, 0, 1, 2, 3);

/*Muestra el botón*/
  gtk_widget_show(button);

/*Crea un nuevo grupo para los radio botones*/
  group = gtk_radio_button_group (GTK_RADIO_BUTTON
                                  (button));

/*Crea un nuevo radio botón con la etiqueta CONSULTA*/
  button = gtk_radio_button_new_with_label(group,
                                           "CONSULTA");

/*Coloca el botón en la tabla*/
  gtk_table_attach_defaults(GTK_TABLE(table),
                            button, 0, 1, 3, 4);

/*Muestra el botón*/
  gtk_widget_show(button);

/*Crea un nuevo grupo para los radio botones*/
  group = gtk_radio_button_group (GTK_RADIO_BUTTON
                                  (button));

/*Crea un nuevo radio botón con la etiqueta PLT*/
  button = gtk_radio_button_new_with_label(group, "PLT");
```

```
/*Coloca el botón en la tabla*/
  gtk_table_attach_defaults(GTK_TABLE(table),
                            button, 0, 1, 4, 5);

/*Muestra el botón*/
  gtk_widget_show(button);

/*Crea una nueva etiqueta*/
  label = gtk_label_new("Accion:");

/*Coloca la etiqueta en la tabla*/
  gtk_table_attach_defaults(GTK_TABLE(table),
                            label, 1, 2, 0, 1);

/*Muestra la etiqueta*/
  gtk_widget_show(label);

/*Crea un nuevo radio botón con la etiqueta CONSULTA*/
  button1 = gtk_radio_button_new_with_label(NULL,
                                        "CONSULTA");

/*Coloca el botón en la tabla*/
  gtk_table_attach_defaults(GTK_TABLE(table),
                            button1, 1, 2, 1, 2);

/*Muestra el botón*/
  gtk_widget_show(button1);

/*Crea un nuevo grupo para los radio botones*/
  group1 = gtk_radio_button_group (GTK_RADIO_BUTTON
                                   (button1));

/*Crea un nuevo radio botón con la etiqueta CONSULTA*/
  button1 = gtk_radio_button_new_with_label(group1,
                                   "ACTUALIZACION");

/*Coloca el botón en la tabla*/
  gtk_table_attach_defaults(GTK_TABLE(table),
                            button1, 1, 2, 2, 3);

/*Muestra el botón*/
  gtk_widget_show(button1);

/*Crea un nuevo grupo para los radio botones*/
  group = gtk_radio_button_group (GTK_RADIO_BUTTON
                                  (button));

/*Crea un campo de texto editable*/
  plt = gtk_entry_new();
```

```
/*Coloca el campo de texto en la tabla*/
  gtk_table_attach_defaults(GTK_TABLE(table),
                            plt, 1, 2, 4, 5);

/*Crea una nueva tabla de dos columnas por un renglón*/
  table = gtk_table_new(1, 2, TRUE);

/*Coloca la tabla en la ventana*/
  gtk_box_pack_start(GTK_BOX(GTK_DIALOG(window_alta)->
                     action_area), table, TRUE, TRUE, 10);

/*Muestra la tabla*/
  gtk_widget_show(table);

/*Crea un nuevo botón con la etiqueta Aceptar*/
  button =gtk_button_new_with_label("Aceptar");

/*Coloca el botón en la tabla*/
  gtk_table_attach_defaults(GTK_TABLE(table),
                            button, 0, 1, 0, 1);

/*Muestra el botón*/
  gtk_widget_show(button);

/*Crea un nuevo botón con la etiqueta cancelar*/
  button = gtk_button_new_with_label("Cancelar");

/*Coloca el botón en la tabla*/
  gtk_table_attach_defaults(GTK_TABLE(table),
                            button, 1, 2, 0, 1);

/*Muestra el botón*/
  gtk_widget_show(button);

/*Coloca la caja de empaquetado en la ventana*/
  gtk_box_pack_start(GTK_BOX(GTK_DIALOG(window_alta)->
                          vbox), vbox, TRUE, TRUE, 5);

/*Muestra la ventana*/
  gtk_widget_show(window_alta);
}
...
...
...
```

```
/*Esta es la función form_alta_cambio_usr que se manda a llamar*/

static GtkWidget *form_alta_cambio_usr(gint homogeneous,
                        gint spacing,
                        gint expand,
                        gint fill,
                        gint padding,
                        gint wid,
                        char *szlabel,
                        char *szlabel1) {

  GtkWidget *userwid; /*Declaración de widgets*/
  GtkWidget *vbox;
  GtkWidget *hbox;
  GtkWidget *box;
  GtkWidget *combo;
  GtkWidget *button;
  GList *items = NULL;
  GSList *group = NULL;

/*Crea una nueva caja de empaquetado vertical*/
  vbox = gtk_vbox_new(FALSE,0 );

/*Crea una nueva caja de empaquetado horizontal*/
  box = gtk_hbox_new(homogeneous, spacing);

/*Si la variable wid es igual a cero*/
 if(wid == 0) {
/*Crea una nueva etiqueta*/
userwid = gtk_label_new(szlabel);

/*Coloca la etiqueta en el empaquetado*/
gtk_box_pack_start(GTK_BOX(box),
                        userwid, expand, fill, padding);

/*Muestra la etiqueta*/
gtk_widget_show(userwid);

/*Devuelve el empaquetado*/
return(box);
  }

/*Si la variable wid es igual a uno*/
  if(wid == 1) {

/*Crea una nueva etiqueta*/
userwid = gtk_label_new(szlabel);
```

```
/*Coloca la etiqueta en el empaquetado*/
gtk_box_pack_start(GTK_BOX(box),
                        userwid, expand, fill, padding);

/*Muestra la etiqueta*/
gtk_widget_show(userwid);

/*Crea un nuevo campo de texto editable*/
userwid = gtk_entry_new();

/*Coloca el campo de texto en el empaquetado*/
gtk_box_pack_start(GTK_BOX(box),
                        userwid, expand, fill, padding);

/*Muestra el campo de texto*/
gtk_widget_show(userwid);

/*Crea una nueva etiqueta*/
userwid = gtk_label_new(szlabel1);

/*Coloca la etiqueta en el empaquetado*/
gtk_box_pack_start(GTK_BOX(box),
                        userwid, expand, fill, padding);

/*Muestra la etiqueta*/
gtk_widget_show(userwid);

/*Devuelve el empaquetado*/
return(box);
  }

/*Si la variable wid es igual a dos*/
  if(wid == 2) {

/*Crea una nueva etiqueta*/
userwid = gtk_label_new(szlabel);

/*Coloca la etiqueta en el empaquetado*/
gtk_box_pack_start(GTK_BOX(box),
                        userwid, expand, fill, padding);

/*Muestra la etiqueta*/
gtk_widget_show(userwid);

/*Crea un nuevo campo de texto editable*/
userwid = gtk_entry_new();

/*Coloca el campo de texto en el empaquetado*/
gtk_box_pack_start(GTK_BOX(box),
                        userwid, expand, fill, padding);
```

```
/*Muestra el campo de texto*/
gtk_widget_show(userwid);

/*Devuelve el empaquetado*/
return(box);
  }

/*Si la variable wid es igual a tres*/
  if(wid == 3) {

/*Crea una nueva etiqueta*/
userwid = gtk_label_new(szlabel);

/*Coloca la etiqueta en el empaquetado*/
gtk_box_pack_start(GTK_BOX(box),
                       userwid, expand, fill, padding);

/*Muestra la etiqueta*/
gtk_widget_show(userwid);

/*Crea un nuevo campo de texto editable*/
userwid = gtk_entry_new();

/*Permite que los valores que se escriben no se vean, esta función
sirve para los passwords*/
gtk_entry_set_visibility(GTK_ENTRY(userwid), FALSE);

/*Coloca el campo de texto en el empaquetado*/
gtk_box_pack_start(GTK_BOX(box),
                       userwid, expand, fill, padding);

/*Muestra el campo de texto editable*/
gtk_widget_show(userwid);

/*Crea una nueva etiqueta*/
userwid = gtk_label_new(szlabel1);

/*Coloca la etiqueta en el empaquetado*/
gtk_box_pack_start(GTK_BOX(box),
                       userwid, expand, fill, padding);

/*Muestra la etiqueta*/
gtk_widget_show(userwid);

/*Devuelve la etiqueta*/
return(box);
  }
```

```
/*Si la variable wid es igual a cuatro*/
  if(wid == 4) {

/*Crea una nueva etiqueta*/
 userwid = gtk_label_new(szlabel);

/*Coloca la etiqueta en el empaquetado*/
gtk_box_pack_start(GTK_BOX(box),
                       userwid, expand, fill, padding);

/*Muestra la etiqueta*/
gtk_widget_show(userwid);

/*Crea un nuevo campo de texto editable*/
userwid = gtk_entry_new();

/*Oculta la cadena que se escribe*/
gtk_entry_set_visibility(GTK_ENTRY(userwid), FALSE);

/*Coloca el campo de texto editable en el empaquetado*/
gtk_box_pack_start(GTK_BOX(box),
                       userwid, expand, fill, padding);

/*Muestra el campo de texto*/
gtk_widget_show(userwid);

/*Crea una nueva etiqueta*/
userwid = gtk_label_new(szlabel1);

/*Coloca la etiqueta en el empaquetado*/
gtk_box_pack_start(GTK_BOX(box),
                       userwid, expand, fill, padding);

/*Muestra la etiqueta*/
gtk_widget_show(userwid);

/*Devuelve el empaquetado*/
return(box);
  }
}
```

- *Para crear la ventana de la figura 4.14 se escribió el siguiente código*

```
/*Función op_cambio*/
void op_cambio(GtkWidget *widget, gpointer data) {
/*Verifica que el botón este presionado*/
  if (GTK_TOGGLE_BUTTON(widget)->active) {

/*Cierra la ventana anterior*/
gtk_widget_destroy(window_op);
/*Llama a la función cambio_usr*/
```

```
cambio_usr();
  }
}

/*Función cambio_usr*/
gint cambio_usr(void) {
  GtkWidget *window_cambio; /*Declaración de widgets*/
  GtkWidget *label;
  GtkWidget *separator;
  GtkWidget *hbox;
  GtkWidget *hbox1;
  GtkWidget *vbox;
  GtkWidget *vbox1;
  GtkWidget *vbox2;
  GtkWidget *box;
  GtkWidget *table;
  GtkWidget *button;
  GtkWidget *button1;
  GtkWidget *togbtn;
  GSList *group = NULL;
  GSList *group1 = NULL;

/*Crea una nueva ventana*/
  window_cambio = gtk_dialog_new();

/*Asigna tamaño a la ventana*/
  gtk_widget_set_usize(window_cambio, 320, 400);

/*Asigna titulo a la ventana*/
  gtk_window_set_title(GTK_WINDOW(window_cambio),
                       "Cambio Usuario");

/*Crea un caja de empaquetado vertical*/
  vbox = gtk_vbox_new(FALSE, 0);

/*Muestra la caja de empaquetado*/
  gtk_widget_show(vbox);

/*Llama a una función que se llama form_alta_cambio_usr y le
asigna los valores correspondientes*/
  hbox = form_alta_cambio_usr(FALSE,0,FALSE,
     TRUE,5,0,"IDENTIFICACION DEL USUARIO","");

/*El valor que recibe se almacena en hbox y lo coloca en la caja
de empaquetado*/
  gtk_box_pack_start(GTK_BOX(vbox), hbox, FALSE, FALSE, 5);

/*Muestra los valores de hbox*/
  gtk_widget_show(hbox);
```

```
/*Llama a una función que se llama form_alta_cambio_usr y le
asigna los valores correspondientes*/
  hbox = form_alta_cambio_usr(FALSE,0,FALSE,
     TRUE,5,1,"Nombre de Usuario: ","(8 Caract.)");

/*El valor que recibe se almacena en hbox y lo coloca en la caja
de empaquetado*/
  gtk_box_pack_start(GTK_BOX(vbox), hbox, FALSE, FALSE, 5);

/*Muestra los valores de hbox*/
  gtk_widget_show(hbox);

/*Llama a una función que se llama form_alta_cambio_usr y le
asigna los valores correspondientes*/
  hbox = form_alta_cambio_usr(FALSE,0,FALSE,
     TRUE,8,2,"Nombre Completo: ","");

/*El valor que recibe se almacena en hbox y lo coloca en la caja
de empaquetado*/
  gtk_box_pack_start(GTK_BOX(vbox), hbox, FALSE, FALSE, 5);

/*Muestra los valores de hbox*/
  gtk_widget_show(hbox);

  separator = gtk_hseparator_new ();
  gtk_box_pack_start (GTK_BOX (vbox), separator, FALSE, TRUE, 0);
  gtk_widget_show(separator);

/*Llama a una función que se llama form_alta_cambio_usr y le
asigna los valores correspondientes*/
  hbox = form_alta_cambio_usr(FALSE,0,FALSE,
     TRUE,5,0,"SEGURIDAD EN LA CUENTA","");

/*El valor que recibe se almacena en hbox y lo coloca en la caja
de empaquetado*/
  gtk_box_pack_start(GTK_BOX(vbox), hbox, FALSE, FALSE, 5);

  gtk_widget_show(hbox); /*Muestra los valores de hbox*/
/*Llama a una función que se llama form_alta_cambio_usr y le
asigna los valores correspondientes*/
  hbox = form_alta_cambio_usr(FALSE,0,FALSE,
     TRUE,10,3,"Password: ","(8 Caracteres)");
/*El valor que recibe se almacena en hbox y lo coloca en la caja
de empaquetado*/
  gtk_box_pack_start(GTK_BOX(vbox), hbox, FALSE, FALSE, 5);

/*Muestra los valores de hbox*/
  gtk_widget_show(hbox);
```

```
/*Llama a una función que se llama form_alta_cambio_usr y le
asigna los valores correspondientes*/
  hbox = form_alta_cambio_usr(FALSE,0,FALSE,
     TRUE,2,4,"RE-Password: ","(8 Caracteres)");
--
/*El valor que recibe se almacena en hbox y lo coloca en la caja
de empaquetado*/
  gtk_box_pack_start(GTK_BOX(vbox), hbox, FALSE, FALSE, 5);

/*Muestra los valores de hbox*/
  gtk_widget_show(hbox);

/*Llama a una función que se llama form_alta_cambio_usr y le
asigna los valores correspondientes*/
  /*  hbox = form_alta_cambio_usr(FALSE,0,FALSE,
     TRUE,5,0,"Tipo:","");

/*El valor que recibe se almacena en hbox y lo coloca en la caja
de empaquetado*/
  gtk_box_pack_start(GTK_BOX(vbox), hbox, FALSE, FALSE, 5);

/*Muestra los valores de hbox*/
  gtk_widget_show(hbox);

/*Crea una nueva tabla de dos columnas por cinco renglones*/
  table = gtk_table_new(5, 2, FALSE);

/*Muestra la tabla*/
  gtk_widget_show(table);

/*Coloca la tabla en la caja de empaquetado*/
  gtk_box_pack_start(GTK_BOX(vbox), table, TRUE, TRUE, 0);

/*Crea una etiqueta*/
  label = gtk_label_new("Tipo:");

/*Coloca la etiqueta en la tabla*/
  gtk_table_attach_defaults(GTK_TABLE(table),
                            label, 0, 1, 0, 1);
/*Muestra la etiqueta*/
  gtk_widget_show(label);

/*Crea un nuevo radio botón con la etiqueta SSRE*/
  button = gtk_radio_button_new_with_label (NULL, "SSRE");

/*Coloca el botón en la tabla*/
  gtk_table_attach_defaults(GTK_TABLE(table),
                            button, 0, 1, 1, 2);

/*Muestra el botón*/
  gtk_widget_show(button);
```

```
/*Crea un nuevo grupo para los radio botones*/
  group = gtk_radio_button_group (GTK_RADIO_BUTTON
                                  (button));

/*Crea un nuevo radio botón con la etiqueta DASSU*/
  button = gtk_radio_button_new_with_label (group,
                                            "DASSU");

/*Coloca el botón en la tabla*/
  gtk_table_attach_defaults(GTK_TABLE(table),
                            button, 0, 1, 2, 3);

/*Muestra el botón*/
  gtk_widget_show(button);

/*Crea un nuevo grupo para los radio botones*/
  group = gtk_radio_button_group (GTK_RADIO_BUTTON
                                  (button));

/*Crea un nuevo radio botón con la etiqueta CONSULTA*/
  button = gtk_radio_button_new_with_label(group,
                                           "CONSULTA");

/*Coloca el botón en la tabla*/
  gtk_table_attach_defaults(GTK_TABLE(table),
                            button, 0, 1, 3, 4);

/*Muestra el botón*/
  gtk_widget_show(button);

/*Crea un nuevo grupo para los radio botones*/
  group = gtk_radio_button_group (GTK_RADIO_BUTTON
                                  (button));

/*Crea un nuevo radio botón con la etiqueta PLT*/
  button = gtk_radio_button_new_with_label(group, "PLT");


/*Coloca el botón en la tabla*/
  gtk_table_attach_defaults(GTK_TABLE(table),
                            button, 0, 1, 4, 5);

/*Muestra el botón*/
  gtk_widget_show(button);

/*Crea una nueva etiqueta*/
  label = gtk_label_new("Accion:");

/*Coloca la etiqueta en la tabla*/
  gtk_table_attach_defaults(GTK_TABLE(table),
                            label, 1, 2, 0, 1);
```

```
/*Muestra la etiqueta*/
  gtk_widget_show(label);

/*Crea un nuevo radio botón con la etiqueta CONSULTA*/
  button1 = gtk_radio_button_new_with_label(NULL,
                                        "CONSULTA");

/*Coloca el botón en la tabla*/
  gtk_table_attach_defaults(GTK_TABLE(table),
                            button1, 1, 2, 1, 2);

/*Muestra el botón*/
  gtk_widget_show(button1);

/*Crea un nuevo grupo para los radio botones*/
  group1 = gtk_radio_button_group (GTK_RADIO_BUTTON
                                   (button1));

/*Crea un nuevo radio botón con la etiqueta CONSULTA*/
  button1 = gtk_radio_button_new_with_label(group1,
                                   "ACTUALIZACION");

/*Coloca el botón en la tabla*/
  gtk_table_attach_defaults(GTK_TABLE(table),
                            button1, 1, 2, 2, 3);

/*Muestra el botón*/
  gtk_widget_show(button1);

/*Crea un botón de selección el cual permite bloquear o no a un
usuario, dentro del servidor de bases de datos los usuarios no se
borran solo se bloquean ya que tienen registros asociados.*/
  togbtn = gtk_check_button_new_with_label("BLOQUEADO");

/*Coloca el botón en la tabla*/
  gtk_table_attach_defaults(GTK_TABLE(table),
                            togbtn, 1, 2, 3, 4);

/*Muestra el botón*/
  gtk_widget_show(togbtn);

---
/*Crea un nuevo grupo para los radio botones*/
  group = gtk_radio_button_group (GTK_RADIO_BUTTON
                                  (button));

/*Crea un campo de texto editable*/
  plt = gtk_entry_new();
```

```
/*Coloca el campo de texto en la tabla*/
  gtk_table_attach_defaults(GTK_TABLE(table),
                            plt, 1, 2, 4, 5);

/*Crea una nueva tabla de dos columnas por un renglón*/
  table = gtk_table_new(1, 2, TRUE);

/*Coloca la tabla en la ventana*/
  gtk_box_pack_start(GTK_BOX(GTK_DIALOG(window_alta)->
                     action_area), table, TRUE, TRUE, 10);

/*Muestra la tabla*/
  gtk_widget_show(table);

/*Crea un nuevo botón con la etiqueta Aceptar*/
  button =gtk_button_new_with_label("Aceptar");

/*Coloca el botón en la tabla*/
  gtk_table_attach_defaults(GTK_TABLE(table),
                            button, 0, 1, 0, 1);

/*Muestra el botón*/
  gtk_widget_show(button);

/*Crea un nuevo botón con la etiqueta cancelar*/
  button = gtk_button_new_with_label("Cancelar");

/*Coloca el botón en la tabla*/
  gtk_table_attach_defaults(GTK_TABLE(table),
                            button, 1, 2, 0, 1);

/*Muestra el botón*/
  gtk_widget_show(button);

/*Coloca la caja de empaquetado en la ventana*/
  gtk_box_pack_start(GTK_BOX(GTK_DIALOG(window_alta)->
                          vbox), vbox, TRUE, TRUE, 5);

/*Muestra la ventana*/
  gtk_widget_show(window_alta);
}
```

El código anterior también hace uso de la función form_alta_cambio_usr.

Si en la figura 4.2 elegimos ayuda de la opción del menú mostrará información acerca de los desarrolladores así como una descripción general del Sistema de Administración de Bases de Datos (SABAD), las pantallas son las siguientes:

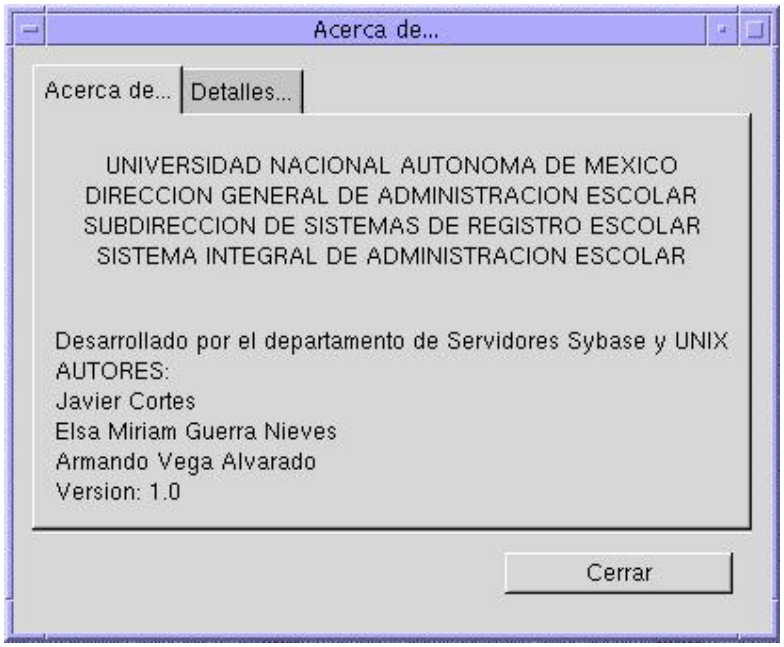


Figura 4.15 Ventana Acerca de…

En esta pantalla se muestran los detalles del SABAD

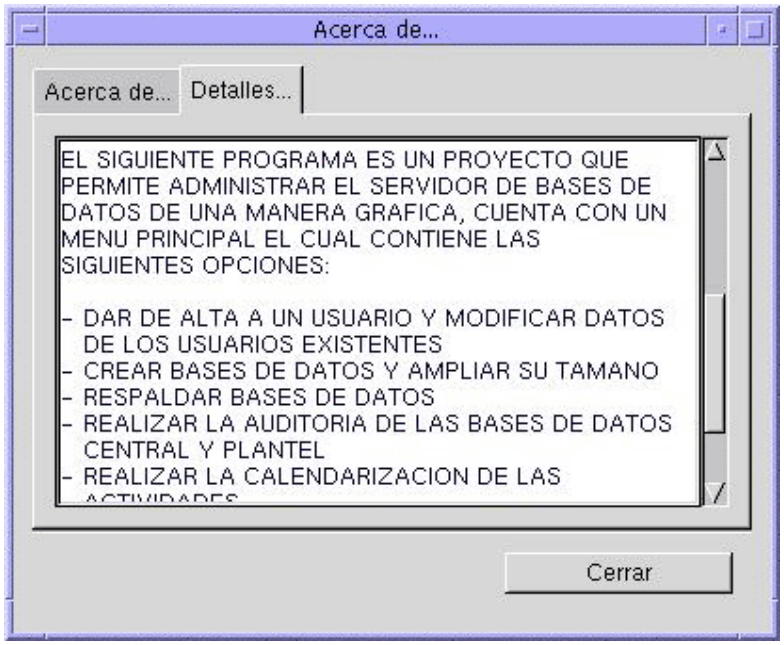


Figura 4.16 Ventana para los detalles

Cuando se desea cerrar la sesión en la que se está trabajando el sistema presentará la siguiente ventana de confirmación

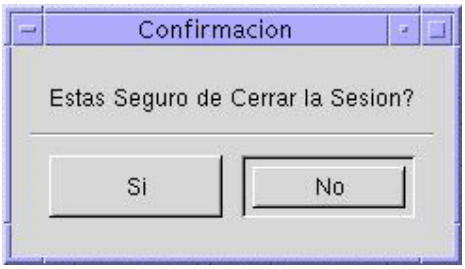


Figura 4.17 Ventana de confirmación

El código que permite generar estas ventanas es el siguiente:

- *Para crear la ventana de la figura 4.15 y 4.16 se escribió el siguiente código*

```
/*Función que construye la ventana acerca de…, cuando se elige la
opción del menú ayuda*/
gint ShowMessage(void) {
  GtkWidget *label; /*Declaración de widgets*/
  GtkWidget *tmp;
  GtkWidget *view;
  GtkWidget *vscroll;
  GtkWidget *hscroll;
  GtkWidget *scrolled_window;
  GtkWidget *box;
  GtkWidget *button;
  GtkWidget *table;
  GtkWidget *dialog_window;
  GtkWidget *notebook;
  GdkColormap *cmap;
  GdkColor color;

  FILE *infile; /*Apuntador de tipo archivo*/

/*Crea una nueva ventana*/
  dialog_window = gtk_dialog_new();

/*Asigna un titulo*/
  gtk_window_set_title(GTK_WINDOW(dialog_window),
                       "Acerca de...");

/*Coloca en primer plano a la ventana*/
  gtk_widget_realize (dialog_window);
```

```
/*Crea un widget notebook*/
  notebook = gtk_notebook_new ();

/*Coloca el widget en la ventana*/
  gtk_box_pack_start (GTK_BOX (GTK_DIALOG (dialog_window)->
                      vbox), notebook, TRUE, TRUE, 0);

/*Muestra el widget*/
  gtk_widget_show (notebook);

/*Crea una caja de empaquetado vertical*/
  box = gtk_vbox_new (TRUE, 5);

/*Asigna un borde para la caja*/
  gtk_container_border_width (GTK_CONTAINER (box), 10);

/*Muestra la caja de empaquetado*/
  gtk_widget_show (box);

/*Crea un nueva etiqueta*/
  label = gtk_label_new("UNIVERSIDAD NACIONAL AUTONOMA DE
MEXICO\nDIRECCION GENERAL DE ADMINISTRACION ESCOLAR\nSUBDIRECCION
DE SISTEMAS DE REGISTRO ESCOLAR\nSISTEMA INTEGRAL DE
ADMINISTRACION ESCOLAR\n");

/*Coloca la etiqueta en la caja de empaquetado*/
  gtk_box_pack_start (GTK_BOX (box),
                      label, FALSE, FALSE, 0);

/*Muestra la etiqueta*/
  gtk_widget_show(label);

/*Crea una nueva etiqueta*/
  label = gtk_label_new("Desarrollado por el departamento de
Servidores Sybase y UNIX\nAUTORES:\nJavier Cortes\nElsa Miriam
Guerra Nieves\nArmando Vega Alvarado\nVersion: 1.0");

/*Alinea el texto a la izquierda*/
  gtk_label_set_justify (GTK_LABEL (label), GTK_JUSTIFY_LEFT);

/*Coloca la etiqueta en la caja de empaquetado*/
  gtk_box_pack_start (GTK_BOX (box),
                      label, FALSE, FALSE, 0);

/*Muestra la etiqueta*/
  gtk_widget_show(label);
```

```
/*Crea una nueva etiqueta*/
  label = gtk_label_new("Acerca de...");

/*Coloca la etiqueta en la pestaña del widgets notebook*/
  gtk_notebook_append_page (GTK_NOTEBOOK (notebook),
                            box, label);

/*Muestra la etiqueta*/
  gtk_widget_show(label);

/*Crea una caja de empaquetado vertical*/
  box = gtk_vbox_new (FALSE, 5);

/*Asigna borde en la caja*/
  gtk_container_border_width (GTK_CONTAINER (box), 10);

/*Muestra la caja*/
  gtk_widget_show (box);

/*Crea una nueva tabla de dos columnas por un renglón*/
  table =gtk_table_new (1, 2, FALSE);

/*Coloca la tabla en la caja de empaquetado*/
  gtk_box_pack_start (GTK_BOX (box),
                      table, TRUE, TRUE, 0);

/*Muestra la tabla*/
  gtk_widget_show (table);

/*Crea un nuevo widget de tipo texto*/
  view = gtk_text_new (NULL, NULL);

/*El texto contenido en el widget no se puede editar*/
  gtk_text_set_editable (GTK_TEXT (view), FALSE);
  gtk_text_set_word_wrap (GTK_TEXT (view), TRUE);

/*Coloca el widget de texto en la tabla*/
  gtk_table_attach (GTK_TABLE (table),
          view, 0, 1, 0, 1,
          GTK_FILL | GTK_EXPAND,
          GTK_FILL | GTK_EXPAND | GTK_SHRINK, 0, 0);

/*Muestra el widget de texto*/
  gtk_widget_show (view);

/*Crea una nueva barra de desplazamiento vertical*/
  vscroll = gtk_vscrollbar_new (GTK_TEXT (view)->vadj);
```

```
/*Coloca la barra en la tabla*/
  gtk_table_attach (GTK_TABLE (table),
          vscroll, 1, 2, 0, 1,
          GTK_FILL, GTK_EXPAND | GTK_FILL |
          GTK_SHRINK, 0, 0);

/*Muestra la barra de desplazamiento*/
  gtk_widget_show (vscroll);

/*Crea una nueva etiqueta*/
  label = gtk_label_new ("Detalles...");

/*Muestra la etiqueta*/
  gtk_widget_show (label);

/*Asigna la etiqueta a una de las pestañas del widget de
notebook*/
  gtk_notebook_append_page (GTK_NOTEBOOK (notebook),
                            box, label);

/*Obtiene los colores del S.O.*/
  cmap = gdk_colormap_get_system();/*Llamada a sistema para saber
su mapa de colores*/
  color.blue = 0x3c82; /*Definición de Colores*/
  color.green = 0;
  color.red = 0;

/*Si ocurre un error al momento de obtener los colores*/
  if (!gdk_color_alloc(cmap, &color)) {
g_error("couldn't allocate color");
  }

/*Coloca en primer plano al widget de texto*/
  gtk_widget_realize(view);

/*Congela el widget de texto, para poder escribir en el*/
  gtk_text_freeze (GTK_TEXT (view));

/*Abre archivo para lectura*/
  infile = fopen("acerca.txt", "r");

  if (infile) {
char buffer[1024];
int nchars;

while (1) {
     nchars = fread(buffer, 1, 1024, infile);

/*Inserta el contenido del archivo en el widget de texto*/
     gtk_text_insert (GTK_TEXT (view), NULL, &color, NULL,
                buffer, nchars);
```

```
     if (nchars < 1024)
       break;
  }

/*Cierra el archivo leído*/
fclose (infile);

/*Libera el buffer usado para almacenar el archivo leído*/
g_free(buffer);
  }


/*Crea una nueva tabla de tres columnas por un renglón*/
  table = gtk_table_new(1, 3, TRUE);

/*Asigna espacio entre celdas*/
  gtk_table_set_col_spacings(GTK_TABLE(table), 0);

/*Coloca la tabla en la ventana*/
  gtk_box_pack_start(GTK_BOX(GTK_DIALOG(dialog_window)->
                   action_area), table, TRUE, TRUE, 0);

/*Crea un nuevo botón con la etiqueta Cerrar*/
  button = gtk_button_new_with_label("Cerrar");

/*Coloca el botón en la tabla*/
  gtk_table_attach_defaults(GTK_TABLE(table),
                            button, 2, 3, 0, 1);

/*Muestra el botón*/
  gtk_widget_show(button);

/*Muestra la tabla*/
  gtk_widget_show(table);

/*Muestra la ventana
  gtk_widget_show(dialog_window);

/*Congela la ventana*/
  gtk_grab_add(dialog_window);
}
```

Para crear la ventana de la figura 4.17 se escribió el siguiente código

```
/*Función op_salir que llama a la función mensajes y le pasa como
parámetros los mensajes a imprimir*/
void op_salir(GtkWidget *widget, gpointer data) {

/*Llamada a la función messages*/
  Messages("Confirmacion",
           "Estas Seguro de Cerrar la Sesion?",2);
}

/* Función messages*/
gint Messages(char *sztitle,
       char *szmessage,
       int flag) {

  GtkWidget *label; /*Declaración de widgets*/
  GtkWidget *button;
  GtkWidget *table;
  GtkWidget *pbar;
  GtkWidget *dialog_window;

/*Crea una nueva ventana*/
  dialog_window = gtk_dialog_new();

/*Asigna un titulo a la ventana*/
  gtk_window_set_title(GTK_WINDOW(dialog_window), sztitle);

/*Asigna borde para la ventana*/
  gtk_container_border_width(GTK_CONTAINER(dialog_window),
                         10);

/*Coloca en primer plano a la ventana*/
  gtk_widget_realize (dialog_window);

/*Crea una nueva etiqueta*/
  label = gtk_label_new(szmessage);

/*Espaciado entre los caracteres de la etiqueta*/
  gtk_misc_set_padding(GTK_MISC(label), 10, 10);

/*Coloca la etiqueta en la ventana*/
  gtk_box_pack_start(GTK_BOX(GTK_DIALOG(dialog_window)->
                     vbox), label, TRUE, TRUE, 0);

/*Si el valor de flan es cero*/
  if(flag == 0) {
```

```
/*Crea una nueva tabla de tres columnas por un renglón*/
table = gtk_table_new(1,3,TRUE);

/*Coloca la tabla en la ventana*/
gtk_box_pack_start(GTK_BOX(GTK_DIALOG(dialog_window)->
                      action_area), table, TRUE, TRUE, 0);

/*Crea un nuevo botón con la etiqueta Aceptar*/
button = gtk_button_new_with_label("Aceptar");

/*Coloca el botón en la tabla*/
gtk_table_attach_defaults(GTK_TABLE(table),
                             button, 1, 2, 0, 1);

/*Si el valor de la variable es igual a 1*/
  else if (flag == 1) {

/*Crea una nueva tabla de tres columnas por un renglón*/
table = gtk_table_new(1,3,TRUE);

/*Coloca la tabla en la ventana*/
gtk_box_pack_start(GTK_BOX(GTK_DIALOG(dialog_window)->
                      action_area), table, TRUE, TRUE, 0);

/*Crea un nuevo botón con la etiqueta Aceptar*/
button = gtk_button_new_with_label("Aceptar");

/*Coloca el botón en la tabla*/
gtk_table_attach_defaults(GTK_TABLE(table),
                             button, 2, 3, 0, 1);

  }

/*Si el valor de la variable es dos*/
  else if (flag == 2) {

/*Crea una nueva tabla de dos columnas por un renglón*/
table = gtk_table_new(1, 2, TRUE);

/*Asigna espacio entre las celdas*/
gtk_table_set_col_spacings(GTK_TABLE(table), 10);

/*Coloca la tabla en la ventana*/
gtk_box_pack_start(GTK_BOX(GTK_DIALOG(dialog_window)->
                      action_area), table, TRUE, TRUE, 0);

/*Crea un nuevo botón con la etiqueta Si*/
button = gtk_button_new_with_label("Si");
```

```
/*Coloca el botón en la tabla*/
gtk_table_attach_defaults(GTK_TABLE(table),
                          button, 0, 1, 0, 1);

/*Crea un nuevo botón con la etiqueta No*/
 button = gtk_button_new_with_label("No");

/*Coloca el botón en la tabla*/
 gtk_table_attach_defaults(GTK_TABLE(table),
                           button, 1, 2, 0, 1);
  }

/*Asigna el enfoque al botón*/
  GTK_WIDGET_SET_FLAGS(button, GTK_CAN_DEFAULT);
  gtk_widget_grab_default(button);

/*Muestra todos los widgets*/
  gtk_widget_show_all(dialog_window);

/*Enfoque para la ventana del widget de dialogo*/
  gtk_grab_add(dialog_window);
}
```

4.3 Generación de reportes

Los reportes que generará el SABAD son principalmente los de auditorías, pues se tiene que verificar la información para que los datos sean los correctos.

Algunos de los reportes que se generan son los siguientes:

En estos reportes obtenemos la comparación de los datos de asignaturas de central contra las asignaturas de un plantel y viceversa, revisando si existe ciclo y seriación para esta asignatura y finalmente verifica si existe ciclo y seriación para una asignatura determinada de central.

```
================================================================================
                    UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO
                 DIRECCION GENERAL DE ADMINISTRACION ESCOLAR
              SUBDIRECCION DE SISTEMAS DE REGISTRO ESCOLAR
                 SISTEMA INTEGRAL DE ADMINISTRACION ESCOLAR
--------------------------------------------------------------------------------
          Reporte de asignaturas contenidas en ASG contra APP, CCL y SER
--------------------------------------------------------------------------------
          NUMERO DE REGISTROS COMPARADOS en Plantel  = [800]      09/08/2002 10:31:42
--------------------------------------------------------------------------------
               CLAVE DEL PLANTEL [999]FACULTAD DE AAAAAAAAAA AAAAAAA
--------------------------------------------------------------------------------
================================================================================

 ---------------------------------------------------------------
 | CLAVE  |NOMBRE DE LA ASIGNATURA  |  EXISTE (---)  |  NO EXISTE (X)  |
 ---------------------------------------------------------------
 |        |                         |APP|CCL|SER|
 ---------------------------------------------------------------
 |  9990  |AAAAAAAAAA AA AAAAAAAA  |---|---| X |
 ---------------------------------------------------------------
 |  9991  |AAAAAAAAAA AA AAAAAAAA  |---|---| X |
 ---------------------------------------------------------------
 |  9992  |  AAAAAAAAA             |---| X | X |
 ---------------------------------------------------------------
 |  9993  |  AAAAAAAAA             |---| X | X |
 ---------------------------------------------------------------
```

```
================================================================================
                    UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO
                 DIRECCION GENERAL DE ADMINISTRACION ESCOLAR
              SUBDIRECCION DE SISTEMAS DE REGISTRO ESCOLAR
                   SISTEMA INTEGRAL DE ADMINISTRACION ESCOLAR
================================================================================
                  Reporte de asignaturas contenidas en APP contra ASG
================================================================================
        NUMERO DE REGISTROS COMPARADOS en Central  = [666]      09/08/2002 10:27:24
================================================================================
                CLAVE DEL PLANTEL [999]FACULTAD DE AAAAAAAAAA AAAAAAA
================================================================================

  ------------------------------------------------------------------------------
  |     Claves de asignaturas de APP que no estan en ASG del plantel 999       |
  ------------------------------------------------------------------------------
  | CLAVE |                              NOMBRE                                |
  ------------------------------------------------------------------------------
  | 9990  |          AAAAAAA AA AAAAAAAAAAAAAAAAAA AAAAAAAAA                   |
  ------------------------------------------------------------------------------
  | 9991  |          AAAAAAA AA AAAAAAAAAAAAAAAAAA AAAAAAAAA                   |
  ------------------------------------------------------------------------------
  | 9992  |          AAAAAAA AA AAAAAAAAAAAAAAAAAA AAAAAAAAA                   |
  ------------------------------------------------------------------------------

  ------------------------------------------------------------------------------
  |     Claves de asignaturas de ASG que no estan en APP del plantel 999       |
  ------------------------------------------------------------------------------
  | CLAVE |                              NOMBRE |                              |
  ------------------------------------------------------------------------------
  | 8990  |          AAAAAAA AA AAAAAAAAAAAAAAAAAA AAAAAAAAA                   |
  ------------------------------------------------------------------------------
  | 8991  |          AAAAAAA AA AAAAAAAAAAAAAAAAAA AAAAAAAAA                   |
  ------------------------------------------------------------------------------
  | 8992  |          AAAAAAA AA AAAAAAAAAAAAAAAAAA AAAAAAAAA                   |
  ------------------------------------------------------------------------------
```

```
================================================================================
                    UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO
                 DIRECCION GENERAL DE ADMINISTRACION ESCOLAR
                 SUBDIRECCION DE SISTEMAS DE REGISTRO ESCOLAR
                    SISTEMA INTEGRAL DE ADMINISTRACION ESCOLAR
          ------------------------------------------------------------------=
            Reporte de asignaturas contenidas en APP y que no tienen Ciclo ni Seriacion
          ------------------------------------------------------------------=
               NUMERO DE REGISTROS COMPARADOS en Central  = [666]       09/08/2002 10:36:10
          ------------------------------------------------------------------=
                     CLAVE DEL PLANTEL [999]FACULTAD DE AAAAAAAAAA AAAAAAA
          ------------------------------------------------------------------=

                  ---------------------------------------------------
                  |   ASIGNATURAS QUE NO SE ENCUENTRAN EN PLANES DE ESTUDIO VALIDOS    |
                  ---------------------------------------------------
                  | CLAVE |       NOMBRE DE LA ASIGNATURA      | CRED| HORAS  | HORAS  |
                  |       |                                    |     | PRACTC | TEORIC |
                  ---------------------------------------------------
                  | 9995 |     FUNDAMENTOS DE AAAAAAAAAAA    |  12 |  00:00 |  00:00 |
                  ---------------------------------------------------
                  | 9996 |       INTRODUCCION A AAAAAAAAAA    |   8 |  00:00 |  00:00 |
                  ---------------------------------------------------
```

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO
DIRECCION GENERAL DE ADMINISTRACION ESCOLAR
SUBDIRECCION DE SISTEMAS DE REGISTRO ESCOLAR
SISTEMA INTEGRAL DE ADMINISTRACION ESCOLAR

Reporte de asignaturas contenidas en APP y que no tienen Ciclo ni Seriacion

NUMERO DE REGISTROS COMPARADOS en Central = [666]   09/08/2002 10:33:03

CLAVE DEL PLANTEL [999]FACULTAD DE AAAAAAAAAA AAAAAAA

ASIGNATURAS CON PLANES DE ESTUDIO NO 900						
CLAVE	NOMBRE DE LA ASIGNATURA	EXISTE (---)		NO EXISTE (X)		
		CCL	SER	HAI	HAE	HAP
9905	FUNDAMENTOS DE AAAAAAAAAAAA	X	X	0	X	0
9907	INTRODUCCION A AAAAAAAAAA	X	X	0	X	0

ASIGNATURAS CON PLANES DE ESTUDIO 900						
CLAVE	NOMBRE DE LA ASIGNATURA	EXISTE (---)		NO EXISTE (X)		
		CCL	SER	HAI	HAE	HAP
8905	FUNDAMENTOS DE AAAAAAAAAAAA	X	X	0	X	0
8907	INTRODUCCION A AAAAAAAAAA	X	X	0	X	0

En este reporte obtenemos la comparación de las actas que se encuentran en central contra las de un plantel, las diferencias encontradas así como el total de registros encontrados, registros erróneos y aquellos registros que no encontró ya sea en central o plantel.

```
==========================================================================
                UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO
             DIRECCION GENERAL DE ADMINISTRACION ESCOLAR
          SUBDIRECCION DE SISTEMAS DE REGISTRO ESCOLAR
              SISTEMA INTEGRAL DE ADMINISTRACION ESCOLAR
      ------------------------------------------------------------------
            Reporte de consistencia en las actas de central y las actas de un plantel
      ------------------------------------------------------------------
                  FECHA DE HOY [31/01/2002] HORA [17:38:17]
        ----------------------------------------------------------------
                        [999]   FACULTAD AAAAAAAAAA
      ------------------------------------------------------------------
       ACTAS
       ---------------------------------------------------------------
       |  NUME   | NUME_SER | CLAVE DE GRUPO |  CAMPO ERRONEO  |CENTRAL  |PLANTEL  |
       ===============================================================
       | 9999990 |    3     |      0001      |       cust      |    2    |    1    |
       ---------------------------------------------------------------
       | 9999991 |    3     |      EA01      |     tipo_exm    |    E    |    O    |
       ---------------------------------------------------------------
       | 9999992 |    3     |      EA02      |       estd      |    6    |    7    |
       ---------------------------------------------------------------
       | 9999993 |    3     |      EC02      |       estd      |    6    |    7    |
       ---------------------------------------------------------------

          **********************************************************
          * TOTAL DE REGISTROS  * *
          **********************************************************
          *   ACTAS   *
          **********************************************************
          * PLANTEL  | CENTRAL  * REGISTROS ERRONEOS | ERRORES*
          **********************************************************
          *22250 |22250 *659 |659 *
          **********************************************************
```

En estos reportes se obtienen las diferencias que hay en los datos de situación e historia de un plantel en comparación contra trayectoria e historia de central y también verifica la ausencia de datos en estas tablas.

```
=================================================================================
                    UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO
                DIRECCION GENERAL DE ARMINISTRACION ESCOLAR
              SUBDIRECCION DE SISTEMAS DE REGISTRO ESCOLAR
                SISTEMA INTEGRAL DE ADMINISTRACION ESCOLAR
=--------------------------------------------------------------------------------=
                          REPORTE DE TRAYECTORIA E
                              HISTORIAS ACADEMICAS
=--------------------------------------------------------------------------------=
                    ESC[999]   XXXXXX XXXXXXXXXXXXXXXXXXXX  [31/01/2002]  [19:21:50]
================================================================================
  HISTORIA ACADEMICA DE CENTRAL SIN CORRESPONDIENTE EN EL PLANTEL
================================================================================
| NUME_CTA   | CRR | PDE   | CI | CGPO | ASG   | PERD | NUM_ACT | S | CA | CLV_AUT | E | U | OBSERVACION|
=--------------------------------------------------------------------------------=
| 099999991 | 502 | 0835 | 52 | ET39 | 1304 | 1992 | 1111111 | 2 | 08 | xxxxxxx | E | 1 | NO TIENE HISTORIA EN EL PLANTEL
| 099999992 | 502 | 0835 | 52 | ET39 | 1304 | 1992 | 1111121 | 2 | 08 | xxxxxxx | E | 1 | NO TIENE HISTORIA EN EL PLANTEL
================================================================================
HISTORIA ACADEMICA DEL PLANTEL SIN CORRESPONDIENTE EN CENTRAL
================================================================================
| NUME_CTA   | CRR | PDE   | CI | CGPO | ASIG | PERD | NUM_ACT | S | CA | CLV_AUT | E | U | OBSERVACION|
=--------------------------------------------------------------------------------=
| 099999991 | 502 | 0330 | 52 | 1116 | 3001 | 1851 | 0000000 | 2 | 04 | zzzzzz | O | 0 | NO TIENE HISTORIA EN CENTRAL
| 099999991 | 502 | 0330 | 52 | 1116 | 3002 | 1851 | 0000001 | 2 | 04 | zzzzzz | O | 0 | NO TIENE HISTORIA EN CENTRAL
| 099999977 | 502 | 0330 | 52 | 1116 | 3003 | 1851 | 0000002 | 2 | 04 | zzzzzz | O | 0 | NO TIENE HISTORIA EN CENTRAL
| 099999977 | 502 | 0330 | 52 | 1116 | 3004 | 1851 | 0000003 | 2 | 03 | zzzzzz | O | 1 | NO TIENE HISTORIA EN CENTRAL
| 099999977 | 502 | 0330 | 52 | 1116 | 3005 | 1851 | 0000004 | 2 | 03 | zzzzzz | O | 1 | NO TIENE HISTORIA EN CENTRAL
| 099999977 | 502 | 0330 | 52 | 1168 | 3015 | 1861 | 0000005 | 2 | 04 | zzzzzz | O | 0 | NO TIENE HISTORIA EN CENTRAL
| 099999977 | 502 | 0330 | 52 | 4315 | 3009 | 1962 | 0000006 | 2 | 13 | zzzzzz | O | 1 | NO TIENE HISTORIA EN CENTRAL
=--------------------------------------------------------------------------------=
REGISTROS PROCESADOS: 1890295   REGISTROS FALTANTES: 0997
```

```
=============================================================================
                    UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO
                 DIRECCION GENERAL DE ADMINISTRACION ESCOLAR
              SUBDIRECCION DE SISTEMAS DE REGISTRO ESCOLAR
                 SISTEMA INTEGRAL DE ADMINISTRACION ESCOLAR
-----------------------------------------------------------------------------
                          REPORTE DE TRAYECTORIA E
                            HISTORIAS ACADEMICAS
-----------------------------------------------------------------------------
                   PLT [999]   FACULTAD DE AAAAAAAAAAA AAAAA
=============================================================================
| NUME_CTA  | CRR |  PDE | CI | CGPO | ASIG | PERD | **********  |        CAMPO ERRONEO  |VALOR EN CENTRAL |VALOR EN PLANTEL |
=============================================================================
| 099999995 | 304 | 0347 | 54 | EF41 | 1316 | 1942 | **********  | CLAVE DE AUTORIZACION  | XXXXXXX | zzzzzzz |
| 099999995 | 304 | 0347 | 54 | EF41 | 1316 | 1942 | **********  |    ULTIMO REGISTRO     |       1 |   0     |
-----------------------------------------------------------------------------
| 099999996 | 411 | 0836 | 56 | 1121 | 3116 | 1991 | **********  |    ULTIMO REGISTRO     |       0 |   1     |
-----------------------------------------------------------------------------
| 099999997 | 414 | 0560 | 72 | 0002 | 3604 | 1992 | **********  | CLAVE DE AUTORIZACION  | ZZZZZZZ | xxxxxxx |
-----------------------------------------------------------------------------
```

```
=============================================================================
                    UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO
                DIRECCION GENERAL DE ADMINISTRACION ESCOLAR
             SUBDIRECCION DE SISTEMAS DE REGISTRO ESCOLAR
                SISTEMA INTEGRAL DE ADMINISTRACION ESCOLAR
=-------------------------------------------------------------------------=
                          REPORTE DE TRAYECTORIA E
                            HISTORIAS ACADEMICAS
=-------------------------------------------------------------------------=
                           PLT [999]   FACULTAD AAAAAAAAAAAAAAA
=============================================================================
| NUME_CTA   | CRR |  PDE | CI |  ************ |     CAMPO ERRONEO     | VALOR EN CENTRAL | VALOR EN PLANTEL |
=============================================================================
| 099999991 | 102 | 0842 | 54 |  ************ | ESCUELA DE PROCEDENCIA |      502027     |      2701      |
-----------------------------------------------------------------------------
| 499999999 | 102 | 0842 | 56 |  ************ |    ARTICULO 19        |      2071       |      2072      |
-----------------------------------------------------------------------------
| 099999994 | 102 | 0842 | 56 |  ************ |    ARTICULO 19        |      2062       |      2072      |
| 099999994 | 102 | 0842 | 56 |  ************ |    ARTICULO 24        |      2091       |      2101      |
-----------------------------------------------------------------------------
| 099999990 | 102 | 0842 | 59 |  ************ | ESCUELA DE PROCEDENCIA |     2030009     |      203009    |
-----------------------------------------------------------------------------
| 099999993 | 102 | 0842 | 69 |  ************ | ESCUELA DE PROCEDENCIA |     1020201     |      102201    |
-----------------------------------------------------------------------------
| 099999999 | 102 | 0842 | 59 |  ************ | ESCUELA DE PROCEDENCIA |     3120015     |      312015    |
-----------------------------------------------------------------------------
| 099999991 | 102 | 0842 | 59 |  ************ | ESCUELA DE PROCEDENCIA |     2100519     |      210519    |
-----------------------------------------------------------------------------
Hora y fecha de terminacion de el reporte: 14/01/2002    17:19:03
```

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO
DIRECCION GENERAL DE ARMINISTRACION ESCOLAR
SUBDIRECCION DE SISTEMAS DE REGISTRO ESCOLAR
SISTEMA INTEGRAL DE ADMINISTRACION ESCOLAR

REPORTE DE TRAYECTORIA E HISTORIAS ACADEMICAS

ESC[999] FACULTAD DE AAAAAAAAAAAAAAAAA [31/01/2002] [07:41:11]

TRAYECTORIAS QUE NO TIENEN SITUACION EN EL PLANTEL

NUME_CTA	CRR	PDE	CI	CF	AR19	AR24	ESC_PRO	FECH_FIN	PERF	GEN	T	BSERVACION
099999993	304	0348	54		2002	1000	0304006		1941	193	0	NO TIENE SITUACION EN EL PLANTEL
099999992	304	0348	56		1962	1000	0304006		1962	189	0	NO TIENE SITUACION EN EL PLANTEL
099999997	304	0548	54		2032	1000	0304006		1961	195	0	NO TIENE SITUACION EN EL PLANTEL
099999992	304	0352	54		1922	1000	0304006		1902	185	0	NO TIENE SITUACION EN EL PLANTEL
099999993	304	0548	56		2052	1000	0304006		1972	197	0	NO TIENE SITUACION EN EL PLANTEL
099999994	304	0544	54		2022	1000	0304006		1972	194	0	NO TIENE SITUACION EN EL PLANTEL

SITUACIONES QUE NO TIENEN TRAYECTORIA EN CENTRAL

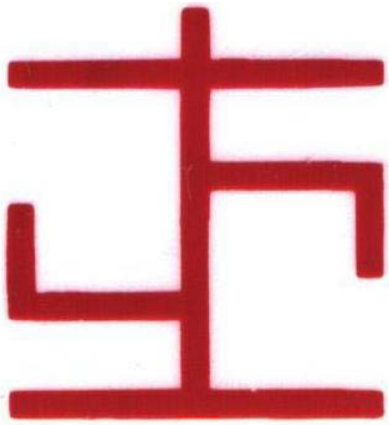
NUME_CTA	CRR	PDE	CI	CF	AR19	AR24	ESC_PRO	FECH_FIN	PERF	GEN	T	OBSERVACION
094559644	301	0179	56	00	2061	2081	4000009		0000	199	0	NO TIENE TRAYECTORIA EN CENTRAL
090372872	301	0348	59	00	2041	1000	0002601		0000	196	0	NO TIENE TRAYECTORIA EN CENTRAL

REGISTROS PROCESADOS: 84088

En este reporte obtenemos los registros de historias academicas que no tienen trayectoria.

```
=======================================================================================================
                                   UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO
                                  DIRECCION GENERAL DE ADMINISTRACION ESCOLAR
                                 SUBDIRECCION DE SISTEMAS DE REGISTRO ESCOLAR
                                  SISTEMA INTEGRAL DE ADMINISTRACION ESCOLAR
              =---------------------------------------------------------------------------------------------------=
                                      Reporte de historias sin trayectorias
             =--------------------------------------------------------------------------------------------------  -=
FECHA [31/01/2002]       HORA [15:29:00]
=------------------------------------------------------------------------------------------------------=

=------------------------------------------------------------------------------------------------------=
=------------------------------------------------------------------------------------------------------=
|         NOMBRE              |  NUM_CTA  | CRR | PDE  | PLT | CAI | GRPO | ASIG | PERD | NUM_ACT | SER | CAL | CLV_AUT | EXM | ULT
=------------------------------------------------------------------------------------------------------=
 -----------
|  =>HAP<=  |
 -----------
=------------------------------------------------------------------------------------------------------=
| XXXXXXXXXX XXXXXX XXXXX XXXXX| 999999999 | 201 | 0215 | 003 | 056 | 1107 | 0241 | 1801 | 8888888 | 003 | 002 | zzzzzzz |   O |   1
=------------------------------------------------------------------------------------------------------=
                               | 999999999 | 201 | 0215 | 003 | 056 | 1107 | 0484 | 1801 | 8888888 | 003 | 001 | zzzzzzz |   O |   1
                          -------------------------------------------------------------------------=
                               | 999999999 | 201 | 0215 | 003 | 056 | 1107 | 0643 | 1801 | 8888888 | 003 | 003 | zzzzzzz |   O |   1
                          -------------------------------------------------------------------------=
                               | 999999999 | 201 | 0215 | 003 | 056 | 2015 | 0485 | 1802 | 8888888 | 003 | 001 | zzzzzzz ¦  O |   1
|=-----------------------------------------------------------------------------------------------------=
| XXXXXXXXXX XXXXXX XXXXX XXXXX| 999999999 | 412 | 0325 | 010 | 072 | EB64 | 1523 | 2001 | 8888888 | 003 | 008 | xxxxxxx |   E |   1
=------------------------------------------------------------------------------------------------------=
| XXXXXXXXXX XXXXXX XXXXX XXXXX| 999999999 | 303 | 0220 | 004 | 056 | 9103 | 1110 | 1991 | 8888888 | 003 | 013 | zzzzzzz |   O |   1
=------------------------------------------------------------------------------------------------------=
| XXXXXXXXXX XXXXXX XXXXX XXXXX| 999999999 | 302 | 0222 | 004 | 056 | 9103 | 1102 | 1991 | 8888888 | 003 | 013 | zzzzzzz |   O |   1
=------------------------------------------------------------------------------------------------------=
| XXXXXXXXXX XXXXXX XXXXX XXXXX| 999999999 | 302 | 1054 | 004 | 056 | 0001 | 1111 | 1991 | 8888888 | 003 | 013 | zzzzzzz |   O |   1
=------------------------------------------------------------------------------------------------------=
| XXXXXXXXXX XXXXXX XXXXX XXXXX| 999999999 | 303 | 1064 | 004 | 056 | 0003 | 1118 | 1991 | 8888888 | 003 | 013 | zzzzzzz |   O |   1
```



Conclusiones

En este trabajo de tesis se diseño el Sistema de Administración de Bases de Datos (SABAD) para facilitar las tareas de administración de los servidores de bases de datos del SIAE, por la complejidad y forma de trabajo del mismo, esta tarea es realmente monumental, por lo mismo, la complejidad de desarrollo del SABAD también de grandes dimensiones, debido a la gran cantidad de información que se procesa y a lo delicado que es su manejo.

Para el diseño de este sistema se utilizaron tecnologías que nos permiten la creación de herramientas de administración. Estas tecnologías engloban lenguajes de programación, DBMS, Sistemas Operativos y técnicas de análisis y diseño de sistemas, etc.

Con toda la información obtenida del análisis se definió una estructura del sistema integrada en cuatro módulos con base a su función y tipo. Para posteriormente diseñar dicha estructura y crear un sistema de fácil manejo para el responsable de la administración.

Es importante resaltar que con el SABAD se logra cierto grado de independencia de las personas con la administración de los servidores de base de datos, es decir que con esta herramienta se cuenta con elementos de trabajo para cualquier otro equipo de administradores sin requerirles forzosamente un grado alto de conocimientos, con SABAD las tareas rutinarias se mantendrán controladas y podrán realizarlas con muy poco conocimiento sobre las características particulares del SIAE.

El SABAD es un sistema que brinda muchas ventajas al personal que trabaja en la administración técnica, ya que dará oportunidad de dedicar menos tiempo a tareas rutinarias y mas tiempo a tareas de investigación y capacitación del mismo personal, de igual modo se puede realizar de forma automática y con cierta autonomía las tareas fundamentales como son los respaldos de las bases de datos y el resguardo de los mismos bajo los esquemas de trabajo ya explicados.

Los sistemas día con día tienden a modificarse es por ello que en este trabajo de tesis se proporciona la documentación necesaria para que el SABAD pueda adaptarse a futuras adiciones y/o modificaciones, esto es una gran ventaja ya que las tareas de administración aumentan, debido al incremento en el volumen de la información.

Basándonos en todo lo expuesto en el presente documento concluimos que el uso del SABAD-SIAE permite una Administración de Bases de datos confiable, segura, y de fácil manejo.

Glosario de términos

alu Tabla que contiene a todos los estudiantes que concluyeron los tramites de inscripción en la UNAM, con lo cual adquieren los derechos y obligaciones que establece la Legislación Universitaria.

app Tabla que contiene el programa o la actividad académica, teorica o práctica, establecida por un plan de estudios para un plantel.

apuntador Un puntero o apuntador es una variable que contiene la dirección de una variable

archivo Es una estructura que organiza registros. Es sólo un conjunto de información con un nombre (llamado *nombre del archivo).* Grupo de datos relacionados entre sí que se procesan juntos

asg Tabla que contiene el programa o actividad académica, teórica o práctica establecida por un Plan de Estudio.

auditoría Es una técnica especializada, orientada al examen y evaluación de los sistemas de información automatizadas, verificando la corrección y confiabilidad de la información procesada por medios electrónicos, de acuerdo con normas técnicas y reglamentarias.

Backup Copia de seguridad o respaldo de una base de datos, servidor, computadora o archivos.

BCPL Lenguaje antiguo, desarrollado por Martín Richards. BCPL sirvió de base a un lenguaje llamado B, que inventó Ken Thompson, y que condujo al desarrollo de C en los años setenta.

bibliotecas colección o conjunto de programas desarrollados por un mismo fabricante que suelen ser compatibles e interoperables entre sí.

browser Programa que despliega la información almacenada en páginas HTML que se encuentran disponibles en servidores del World Wide Web. Como ejemplo de visualizadores tenemos Cello, Internet Explorer, Mosaic, Netscape, Plugins, etc.

buffer Espacio de memoria que se utiliza como regulador y sistema de almacenamiento intermedio entre dispositivos de un sistema informático. Así, por ejemplo, las impresoras suelen contar con un buffer donde se almacena temporalmente la información a imprimir, liberando a la memoria del ordenador de

dichos datos, y permitiendo que el usuario pueda seguir trabajando mientras se imprimen los datos. También existen buffers entre diferentes dispositivos internos del ordenador.

cache de memoria Llamada también a veces almacenamiento caché ó RAM caché, es una parte de memoria RAM estática de alta velocidad (SRAM) más que la lenta y barata RAM dinámica (DRAM) usada como memoria principal. La memoria caché es efectiva dado que los programas acceden una y otra vez a los mismos datos o instrucciones.

ccl Tabla que contiene el semestre o año escolar que establece el plan de estudios para cursar una asignatura.

ciclo Semestre o año escolar que establece el plan de estudios para cursar una asignatura.

cinta Las unidades de cintas magnéticas son utilizadas para copias de archivos o backup debido a su alta capacidad en un espacio reducido y bajo costo por Mbyte en comparación a los disquetes y los discos ópticos. Puede almacenar fácilmente varios miles de MB. La cinta contiene información en forma secuencial, por lo cual, la lectura debe hacerse de la misma manera, haciendo muy lento el proceso de búsqueda de archivos y la transferencia de información entre ésta y la unidad de proceso.

comando Instrucción que un usuario da al sistema operativo de la computadora para realizar determinada tarea.

compactacion Proceso de reducir el tamaño de un archivo para ahorrar espacio o para transmitirlo a mayor velocidad.

compilador Programa traductor que genera lenguaje máquina a partir de un lenguaje de programación de alto nivel basado en el lenguaje. Programa capaz de traducir un código fuente, escrito en el lenguaje de alto nivel que sea, a un código objeto escrito en lenguaje de maquina.

compress Reduce el tamaño de un archivo utilizando codificación Lempel-Ziv. Cuando sea posible cada archivo es reemplazdo por uno con extensión .Z. La cantidad de compresión depende del tamaño de la entrada, típicamente es reducido en 50-60%.

compute Cláusula que genera valores totales que aparecen

como filas adicionales en los resultados de las consultas.

configurar Adaptar una aplicación software o un elemento hardware al resto de los elementos del entorno y a las necesidades específicas del usuario. Es una tarea esencial antes de trabajar con cualquier nuevo elemento. La tendencia actual es a reducir las necesidades de configuración mediante sistemas que permiten al nuevo elemento detectar en qué entorno se instala, configurándose automáticamente sin requerir la participación del usuario. Cuando ésta es necesaria, se intenta facilitar al máximo el proceso de configuración.

consulta Interrogación realizada a una base de datos, en la que se requiere una información o informaciones concretas en función de unos criterios de búsqueda definidos.

crr Tabla que contiene el conjunto de estudios que capacitan para ejercer una profesión a nivel Técnico o Licenciatura.

DASSU Departamento de Administración Servidores Sybase y UNIX

DML (Data Management Language)Lenguaje de Manipulación de Datos

DBMS (DataBase Management System) Sistema de Administración de Base de Datos. Conjunto de programas necesarios para facilitar el almacenamiento, el acceso a bases de datos, la seguridad, y el mantenimiento de su integridad.

DOS Disk Operating System -- DOS (Sistema Operativo en Disco) DOS fue el primer sistema operativo para ordenadores personales. Se basa en mandatos que se escriben línea por línea y fue desarrollado por Bill Gates para IBM, si bien antes de la aparición de los ordenadores personales IBM desarrolló otro DOS para anteriores ordenadores. No confundir con DoS (*Denial of Service*), con o minúscula.

encriptar Técnica por la que la información se hace ilegible para terceras personas. Para poder acceder a ella es necesaria una clave que sólo conocen el emisor y el receptor. Se usa para evitar el robo de información sensible, como números de tarjetas de crédito

FORTRAN (FORmula TRANslator) Traductor de fórmulas. Primer lenguaje de programación de alto nivel y compilador,

desarrollado en 1954 por IBM. Originalmente fue diseñado para expresar fórmulas matemáticas, y aunque en ocasiones se emplea para aplicaciones comerciales, es aún el lenguaje que más se usa para problemas científicos, de ingeniería y matemáticos. FORTRAN IV es un estándar ANSI, pero FORTRAN V tiene varias versiones patentadas.

hardware Dícese de cualquier componente físico relacionado con el sector informático. Antónimo 'software' (Soft = Blando) por oposición a 'hardware' (Hard = Duro). Componentes materiales del ordenador pantalla, chips, etc. Conjunto de dispositivos de los que consiste un sistema. Comprende componentes tales como el teclado, el Mouse, las unidades de disco y el monitor.

hsa Tabla que contiene el resultado académico que un alumno activo obtuvo en la evaluación de una asignatura de un plan de estudios.

indices Se usan en las aplicaciones de bases de datos para indicar la operación de ordenar los registros contenidos en ella de manera especial, en función de unos parámetros definidos previamente.

interfaz. Conexión mecánica o eléctrica que permite el intercambio de información entre dos dispositivos o sistemas. Habitualmente se refiere al 'software' y 'hardware' necesarios para unir dos elementos de proceso en un sistema o bien para describir los estándares recomendados para realizar dichas interconexiones. Es más conocido por su denominación inglesa 'interface'.

Linux Versión bajo la licencia GPL/GNU (que permite la copia y distribución junto al código fuente y sólo se paga el "medio físico") del conocido sistema operativo UNIX. Es un sistema multitarea multiusuario para PC´s.

log Archivo que registra movimientos y actividades de un determinado programa (log file).

login Entrada de identificación.

pantalla Denominada a veces display. Periférico generalmente integrado en el monitor.

password O contraseña. Se denomina así al método de seguridad que se utiliza para identificar a un usuario. Es frecuente su uso en redes, sistemas operativos y DBMS. Se utiliza para dar acceso a personas con

	determinados permisos.
pde	Tabla que contiene el conjunto de asignaturas, exámenes y requisitos necesarios, aprobados por consejeros técnicos de facultades y escuelas, así como por el Consejo Universitario, que al acreditarse, aseguran obtener la preparación teórica y práctica necesarias para el ejercicio eficaz y responsable de una profesión.
plantel	Unidad Académica que imparte planes de estudio a los alumnos.
plataforma	Se refiere a la arquitectura física, de hardware de la computadora.
procedimientos almacenados	es un conjunto de sentencias SQL, que han sido almacenados en una base de datos que pueden ser ejecutados por su nombre.
query	Consulta. Búsqueda en una base de datos.
regla	Una regla puede especificar una máscara para una columna o un tipo definido por un usuario. Una regla puede ser una lista de valores, un rango o un patrón; una regla puede ser aplicada para varias columnas de una o distintas tablas.
roles	son conjuntos de privilegios que pueden concederse 'de golpe' a un usuario. Los Privilegios pueden ser de Objetos (para INSERT, SELECT, UPDATE, DELETE, EXECUTE...) o del Sistema (para crear tablas, vistas...).
rsa	Tabla que contiene el avance escolar de un alumno con respecto al tiempo, actividades académicas y créditos, establecidos en un plan de estudios.
SABAD	Sistema de Administración de base de datos.
SARCE	Sistema Automatizado de Registro y Control Escolar.
ser	Tabla que contiene el orden que se establece en el plan de estudios para cursar las asignaturas correspondientes.
seriación	Orden que se establece en el plan de estudios para cursar las asignaturas correspondientes.
software	Dícese de cualquier componente lógico (programas, aplicaciones) relacionado con el sector informático. Componentes inmateriales del ordenador programas, sistemas operativos, etc. Conjunto de instrucciones mediante las cuales la computadora puede realizar tareas. Los programas, los sistemas operativos y las aplicaciones son ejemplos de software.
SQL	(Structured Query Language). Es un estándar en el

	lenguaje de acceso a bases de datos. Originalmente, era un lenguaje de acceso al sistema de gestión de bases de datos denominado DB2 en plataformas 390 de IBM
SSRE	Subdirección de Sistemas de Registro Escolar
tabla	Es una unidad donde se almacena el conjunto de datos para la base de datos. Estos datos estarán ordenados en columnas verticales, a las cuales se les denomina campos.
Transact-SQL	Lenguaje de programación, que permite realizar consultas a bases de datos, ejecutar comandos de mantenimiento y administración de servidores de bases de datos.
trigger	O disparador. Se defina asi a una subrutina que es ejecutada de manera automática cuando se produce algún tipo de transacción (inserción, borrado o actualización) en la tabla de una base de datos**.**
try	Tabla que contiene el antecedente académico de los diferentes niveles y planes de estudio en que está registrado un alumno.
UNIX	Potente y complejo sistema operativo multiproceso/multitarea y multiusuario orientado a comunicaciones y gran devorador de 'RAM'. Fue creado en 1969 por Ken Thompson y Dennis Ritchie (de la empresa norteamericana 'AT&T Laboratories') coincidiendo con el nacimiento de 'Internet'.
utilerias	Herramientas de hardware o de software con un propósito o utilidad especifica.
variable	Es un identificador que se utiliza para representar un dato individual; es decir, una cantidad numérica o carácter de forma parecida a la constante pero en este caso, su valor es variable, asignado en alguna parte del programa. El valor que el programa asigna a la variable puede ser recuperado referenciando al nombre de la variable. Sin embargo, el tipo de dato asociado a la variable no puede cambiar, esto se refiere a que no se puede asignar por ejemplo un carácter a una variable de tipo int.
widgets	Objeto en GTK+ este puede ser una ventana, un cuadro combinado, un cuadro de texto, un botón, etc.)

Bibliografía

David A. Ruble; tr. Sergio Luis Maria Ruiz Faudon, Análisis y diseño practico para sistemas cliente/servidor con GUI, Prentice-Hall, 1998

Antonio de Amescua Seco, Ingeniería del software de gestión: análisis y diseño de aplicaciones, Paraninfo, 1995

Kenneth E. Kendall y Julie E. Kendall, Análisis y Diseño de Sistemas, Prentice-Hall, 1997

Edward Yourdon, Analisis Estructurado Moderno, Prentice-Hall, 1993

Irene Luque Ruiz, Miguel Angel Gomez-Nieto, Enrique Lopez Espinosa, Gonzalo Cerruela Garcia, Bases de Datos, Alfaomega, 2002

Eric Harlow, Desarrollo de aplicaciones Linux con GTK+ y GDK, Prentice-Hall, 1999

Pegler Swift, Stacia Sambar (Traducción al español Armando Vega Alvarado),. Open Client DB-Library/C, Sybase, 1994

Carlos Muñoz Razo, Como Elaborar y Asesorar una Investigación de Tesis, Prentice-Hall, 1998

Roger S. Presuman, Ingeniería del software enfoque práctico, McGraw-Hill, 1998

Alice Y. H. TSAI, Sistemas de Bases de Datos Administración y uso, Prentice-Hall, 1990

Steve McConnell, Desarrollo y gestión de proyectos informáticos, McGraw-Hill, 1997

Direcciones electrónicas

Información acerca de arquitectura abierta
http://el-nacional.terra.com.ve/tecnologia/noticias115.htm

Información acerca de cliente/servidor
http://www.inei.gob.pe/cpi-mapa/bancopub/libfree/lib616/

Información acerca de productos y documentación Sybase
http://www.sybase.com

Información de administración en general
http://www.monografias.com/trabajos/administracion/administracion.shtml

Pagina oficial de SQShell, documentación, fuentes y binarios
http://www.sqsh.org/

Pagina oficial del compilador *GCC*, documentación, fuentes y binarios
http://www.gnu.org/software/gcc/gcc.html

Pagina oficial de las bibliotecas GTK+, documentación, fuentes y binarios
http://www.gtk.org/