2ej

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Bases, almacenes y mineros de datos

T E S I S

QUE PARA OBTENER EL TÍTULO DE:

Ingeniero en Computación

P R E S E N T A N:

Ivette De Luna Bonilla

Alejo Osvaldo Pacheco Alvarado

Asesor de Tesis Ing. Jorge Gil Mendieta

México, D.F. 1999

Dedicatoria

A mis padres, por haber depositado en mí toda su confianza. Por haberme brindado su tiempo y los mejores consejos. Porque gracias a sus grandes esfuerzos he podido alcanzar esta meta.

A mi tío Manuel, por ser un gran amigo, un buen consejero y estar a mi lado durante mis estudios.

A mis hermanos, por su cariño, comprensión y confianza. Porque su gran entusiasmo me motiva día a día.

Ivette

A mis padres, a quienes debo lo que soy y lo que tengo, y a quienes dedico, además de mi título, toda mi vida.
Gracias...

A Lorena, por haber sido siempre una excelente hermana, gracias.

A Ivette, por permitirme llegar junto a ti al final del camino y compartir conmigo cada momento.

Osvaldo

ÍNDICE

INTRODUCCIÓN

Generalmente las personas prefieren adquirir sus productos o servicios en establecimientos que les hagan sentir que son tomados en cuenta, esto implica, además del trato amable, el hecho de que los vendedores o prestadores de servicios recuerden, en base a las adquisiciones anteriores realizadas en el mismo establecimiento, sus preferencias y que estas sean tomadas en cuenta para promociones y ofrecimientos de otros productos. Este tipo de trato se da en forma natural en establecimientos pequeños, donde existen muy pocos vendedores (en ocasiones uno solo), y estos tienen un trato directo con los clientes, pudiendo recordar las preferencias de cada uno. Sin embargo, las empresas grandes cuyos clientes pueden ser cientos o miles y que en ocasiones ni siquiera tienen un trato directo con ellos, han buscado formas de aplicar esas ventajas de los establecimientos pequeños a sus negocios.

La tecnología de almacenamiento y procesamiento de información ha avanzado a pasos agigantados, esto ha permitido que las grandes corporaciones puedan llevar registros sobre todas las operaciones realizadas por cada uno de sus clientes, lo cual constituye un acervo de información histórica que permanecía en espera de técnicas que permitieran su aprovechamiento.

El surgimiento de la tecnología de Almacenes de Datos o data warehousing ha proporcionado una forma eficiente de almacenar, acceder y manipular todos los datos que se encuentran en la base de datos histórica sin importar el tamaño que ésta pueda llegar a tener (incluso varios terabytes), dotando así a las empresas grandes de "memoria" que les permita recordar cuáles son las preferencias de cada uno de sus clientes, no obstante esto no soluciona el problema, una vez que se ha llegado a este punto se requiere de técnicas que permitan detectar grupos con características similares, predecir comportamientos tanto en los clientes como en los mercados, en fin, dotar a las empresas de "inteligencia" para obtener el mayor provecho posible de los datos almacenados, apoyándolas principalmente en los procesos de toma de decisiones y haciendo posible el trato personalizado de los clientes, aun sin conocerlos. Las técnicas que hacen posible todo ésto han sido agrupadas y reciben el nombre de Minería de Datos. Ambas tecnologías surgieron en el ámbito de los negocios y la mercadotecnia, pero han sido aplicadas con éxito en una gran parte de las áreas del conocimiento humano.

El presente trabajo tiene por objetivo explicar ampliamente la forma de operar de un Almacén de Datos así como las principales técnicas empleadas por la Minería de Datos culminando con un ejemplo de aplicación de éstas técnicas a una base de datos relacional. En el primer capítulo se da un repaso de los conceptos fundamentales de la teoría de bases de datos con el fin de reafirmar los conocimientos sobre el tema, en el segundo capítulo se explican brevemente los principales tipos de bases de datos empleados en la actualidad, esto ayuda a entender de qué manera se pueden aplicar los conceptos de Almacenamiento y Minería de Datos a estos tipos de bases de datos. El tercer capítulo comprende propiamente los fundamentos de las tecnologías de Almacenamiento y Minería de Datos, aquí se tratan puntos como la migración de una base de datos operacional al formato de un Almacén de Datos, su arquitectura y en qué consisten las técnicas de Minería de Datos que se utilizan con mayor frecuencia. En el último capítulo se presentan las diferentes etapas para el desarrollo de una aplicación basada en Minería de Datos, la cual sirve como ejemplo para algunos conceptos desarrollados en el capítulo anterior. Finalmente se presentarán las conclusiones resultantes del desarrollo completo de esta tesis.

CAPÍTULO I

CONCEPTOS SOBRE BASES DE DATOS

1.1- BOSQUEJO HISTÓRICO DE LAS BASES DE DATOS

El gradual incremento en el uso de las computadoras y la evolución del software y hardware de estas ha propiciado avances proporcionales en la tecnología de las bases de datos. Estos avances se han presentado en los siguientes aspectos: forma de almacenamiento de los datos, algoritmos de búsqueda, lenguajes de programación y sistemas manejadores de base de datos entre otros. A continuación se presentan algunos de los eventos más importantes en la historia de las bases de datos.

1945
Se desarrollan las cintas magnéticas, las cuales, en principio, reemplazan a las tarjetas perforadas y a las cintas de papel. Las cintas magnéticas son consideradas como el primer medio que permite realizar búsquedas de información.

1957
Se instala la primer computadora comercial.

1959
En este año se suscitan dos eventos importantes, el primero de ellos es la propuesta de McGee de generalizar el acceso a los datos almacenados electrónicamente. El segundo es la introducción del sistema Ramac por parte de IBM, este sistema lee datos de manera no secuencial, permitiendo de esa manera el acceso a los archivos.

A finales de la década de los años sesenta surgieron sistemas comerciales de base de datos basados en el modelo de red. Los que tuvieron más influencia fueron:

1961
Surge el primer sistema manejador de bases de datos de propósito general (DBMS), el "GE´s Integrated Data Store" diseñado por Bachman, quien popularizó los diagramas de estructuras de datos. La terminología empleada por él, sentó las bases del modelo de datos de red desarrollado por el CODASYL DBTG (*Conference On Data Systems and Languages Database Task Group*).

1965-1970
Algunos de los acontecimientos más importantes en este período fueron:

- El desarrollo de los sistemas manejadores de archivos generalizados.
- IBM desarrolló el Modelo Jerárquico. Estos dos acontecimientos dieron lugar a dos niveles de organización de datos.
- El desarrollo del sistema manejador de información para solucionar problemas de diseño y producción aeroespacial.
- El sistema IMS DB/DC (*database/data communication*) fue el primer sistema DB/DC a gran escala.
- En General Motors, Dodd desarrolla extensiones para el lenguaje PL/I, a las que nombró APL (*Associative Programming Language*).
- IBM y *American Airlines* desarrollan el sistema SABRE, el cual permite acceder a datos en ambiente multiusuario, lo cual involucra una red de comunicaciones.
- TRW desarrolla el sistema de reportes de crédito con el cual experimenta un uso a nivel nacional en los Estados Unidos.

Durante la década de los años setenta la tecnología de las bases de datos experimentó un rápido crecimiento, un gran número de sistemas comerciales siguieron el propósito del CODASYL DBTG pero ninguno lo implantó completamente, algunos otros desarrollaron sus propios modelos de red.

Los proyectos de investigación para esta década fueron: el Sistema R de IBM, INGRES de la Universidad de California, System 2000 de la Universidad de Texas y ADABAS desarrollado en la Universidad Tecnológica de Darmstadt, Alemania.

Los lenguajes de búsqueda desarrollados en los años setenta fueron: SQUARE, SEQUEL(SQL), QBE y QUEL.

1970
Ted Codd, investigador de IBM, desarrolló el modelo relacional seguido del proceso de normalización. Este modelo permite identificar problemas de inserción, borrado y actualización.

1971
Se establece la primera especificación estándar de base de datos, llamada Informe CODASYL DBTG 1971. Desde entonces se han sugerido varias modificaciones a ese informe.

1972
El ACM-SIGMOD (*Association for Computing Machinery - Special Interest Group on Management of Data*) organizó la primera conferencia internacional sobre manejo de datos.

1975
La fundación VLDB (*Very Large Data Base*) organizó la primer conferencia internacional sobre bases de datos muy grandes.

1976
El grupo de trabajo 2.6 de la Federación Internacional sobre procesamiento de información organizó la primera conferencia internacional al respecto.
Chen introduce el modelo entidad-relación (ER).

En la década de los años 80 se desarrolla la familia de los micro DBMSs, los cuales permiten a los usuarios finales definir los datos y manipularlos con facilidad y flexibilidad

1983
ANSI/SPARC reveló que al inicio de los años 80, habían sido implantados más de cien sistemas relacionales.

1985
- Se publicó el estándar SQL preliminar, lo cual permitió que pudieran ser generados programas de aplicación completos empezando desde un lenguaje de interfaz de alto nivel para no programadores.
- El mundo de los negocios es influenciado por los lenguajes de cuarta generación.
- Surge la propuesta de ANSI de crear un lenguaje de definición de red (NDL).

Tendencias
- DBMSs distribuídos

- Sistemas Expertos de bases de datos
- DBMSs orientados a objetos
- Almacenes y mineros de datos.

1.2.- CONOCIMIENTOS GENERALES

1.2.1 Dato

Se entiende por dato, un hecho o imagen que puede ser registrado y almacenado. Los datos pueden ser representados como números, cadenas de texto, imágenes y voz almacenados en archivos en discos o algún otro medio. El hombre es el único ser capaz de darle significado a un dato, ya que por sí solo no lo tiene.

1.2.2 Información

Cuando a un conjunto de datos almacenados en algún medio se le asocia un significado, este conjunto de datos se convierten en información, es decir, información es un conjunto de datos con sentido, como se dijo en la definición anterior, el hombre es el único ser capaz de asociar un significado a un dato o a un conjunto de ellos.

1.2.3 Archivo

Es un conjunto de datos que se encuentra almacenado en algún medio: en memoria principal o en memoria secundaria. uno o varios archivos pueden conformar una base de datos.
Los medios de almacenamiento de las computadoras pueden ser clasificados de dos formas:

Almacenamiento primario.

Incluye los medios de almacenamiento que pueden ser operados directamente por la Unidad Central de Procesamiento (CPU) de la computadora, tales como la memoria principal y la memoria caché. El almacenamiento primario usualmente proporciona acceso rápido a los datos pero es limitado en cuanto a su capacidad de almacenamiento.

Almacenamiento secundario.

Incluye dispositivos como los discos magnéticos y las cintas, usualmente tienen una gran capacidad y son de bajo costo, sin embargo el acceso a los datos es más lento que el del almacenamiento primario. Los datos en el almacenamiento secundario no pueden ser procesados directamente por el CPU, deben ser copiados al almacenamiento primario para su proceso.

En la terminología de las bases de datos, los datos almacenados en disco se organizan como archivos de registro. Cada registro es una colección de valores de datos que pueden ser interpretados como hechos acerca de entidades, sus atributos y sus relaciones. Los registros deben ser almacenados en disco de manera que sea posible su localización en forma eficiente.

Un archivo es una secuencia de registros, todos los registros en un archivo son del mismo tipo. Aunque los bloques son de tamaño fijo determinado por las propiedades físicas del disco y por el sistema operativo, los tamaños de los registros varían, si cada registro en el archivo tiene exactamente el mismo tamaño (en bytes), se dice que el archivo es de registros de longitud fija, si

diferentes registros tienen distintos tamaños, se dice que el archivo es de registros de longitud variable.

1.2.4 Hardware

En el contexto de las bases de datos, hardware se refiere principalmente a los dispositivos de almacenamiento secundario, en los cuales se almacenan físicamente los datos y a los dispositivos de entrada/salida asociados a ellos, así como al procesador o procesadores y su forma de interactuar con la memoria principal ya que de ellos depende el funcionamiento óptimo del software de la base de datos.

1.2.5 Software

Para que un usuario pueda obtener información sobre los datos almacenados y manipularlos a su conveniencia, necesita de un intermediario que sea capaz de acceder a los datos almacenados físicamente de tal forma que el usuario pueda "ver" el contenido de la base de datos. Este intermediario es el software de la base de datos, que como ya se verá, generalmente es un conjunto de programas llamado Sistema Manejador de la Base de Datos (DBMS).

El DBMS constituye el software principal de la base de datos, sin embargo no es el único, ya que dentro de esta categoría se pueden clasificar algunas utilerías, herramientas de desarrollo, programas generadores de reportes, etc.

1.2.6 Usuario

En el entorno de las bases de datos se puede hablar de tres tipos de usuarios principalmente: usuarios finales, programadores de aplicaciones y el administrador de la base de datos.

Usuarios finales

Este tipo de usuario es aquel que interactúa con la base de datos desde una terminal, utilizando programas de aplicación o interfaces incluidas en el software de la base de datos, dentro de éstas, se debe incluir al menos un programa procesador del lenguaje de consulta el cual debe ser capaz de interpretar comandos de alto nivel proporcionados por los usuarios y enviarlos al DBMS. El SQL es el ejemplo más típico de un lenguaje de consulta.

Programadores de aplicaciones

Estos usuarios se encargan de escribir aplicaciones para que el usuario final pueda interactuar con el DBMS de una manera sencilla, o también llamada amistosa. Estas aplicaciones pueden ser escritas, algunas veces, en lenguajes de programación distintos al lenguaje nativo del software de la base de datos.

Administrador de la base de datos

Es el profesional encargado de controlar y supervisar datos y programas, entre sus funciones se definen las siguientes:

- Define el esquema de la base de datos escribiendo un conjunto de definiciones que son traducidas por el compilador del lenguaje de definición de datos a un conjunto de tablas que se almacenan en el diccionario de datos.

- Define las estructuras de almacenamiento y los métodos de acceso escribiendo definiciones que son traducidas por el compilador del lenguaje de almacenamiento y definición de datos.
- Modificaciones del esquema y de la organización física, lo cuál es poco común pero se logra escribiendo un conjunto de definiciones para generar modificaciones a las tablas internas.
- Regula qué partes de la base de datos van a poder ser accedidas y por cuáles usuarios.
- Especifica las restricciones de integridad manteniendo una estructura especial del sistema que es consultada por el manejador de la base de datos cada vez que se actualiza el sistema.

1.2.7 Sistema de base de datos.

Un sistema de base de datos es un sistema computarizado que contiene registros y cuyo propósito general es mantener la información y tenerla disponible cuando se le requiera.

En forma general, un sistema de base de datos se compone de los siguientes elementos: datos, hardware, software y usuarios, los cuales fueron descritos previamente Sin embargo, desde otro punto de vista un sistema de base de datos se puede dividir en módulos que tratan a cada una de las responsabilidades del sistema en general, los componentes funcionales que permiten la interacción entre el sistema de base de datos y el sistema operativo son:

Gestor de archivos

Gestiona la asignación de espacio en la memoria del disco y de las estructuras de datos usadas para representar información almacenada en disco.

Gestor de base de datos

Proporciona la interfaz entre los datos de bajo nivel almacenados en la base de datos y los programas de aplicación y las consultas que se hacen al sistema.

Procesador de consultas

Traduce sentencias en un lenguaje de consultas (pregunta del usuario) a instrucciones de bajo nivel que entiende el gestor de la base de datos, tratando de hacer más eficiente dicha pregunta.

Precompilador de Lenguaje de Manipulación de Datos (DML)

Convierte las sentencias en DML incorporadas en un programa de aplicación en llamadas normales a procedimientos en el lenguaje principal Debe interaccionar con el procesador de consultas para generar el código apropiado.

Compilador de Lenguaje de Definición de Datos (DDL)

Convierte sentencias en DDL en conjunto de tablas que contienen metadatos, es decir, datos sobre datos.

Las estructuras que se requieren para implantar el sistema físico son los *archivos de datos*, que se encargan de almacenar la base y el *diccionario de datos* (que almacena metadatos sobre la estructura de la base de datos).

1.2.8 Bases de Datos

El término base de datos es muy general, ya que en este sentido, una base de datos es simplemente, una colección de datos relacionados. Sin embargo una base de datos debe incluir en forma implícita las siguientes propiedades:

- Debe ser una colección lógicamente coherente de datos con un significado inherente, por lo tanto, un conjunto aleatorio de datos no puede ser una base de datos.
- Deber ser diseñada, construida y alimentada con datos para un propósito específico, es decir, debe ser útil a un conjunto de aplicaciones que a su vez, debe de cubrir las necesidades de un grupo específico de usuarios.
- Debe representar algunos aspectos del mundo real, los cambios que se presenten en ellos deben ser reflejados en la base de datos.

En otras palabras, para que exista una base de datos, debe existir una fuente de donde se derivan los datos, algún grado de interacción con los eventos del mundo real y una audiencia con gran interés en su contenido. Una base de datos puede ser de cualquier tamaño y complejidad.

1.3.- ARQUITECTURA DE UN SISTEMA MANEJADOR DE BASE DE DATOS.

1.3.1 Sistema manejador de base de datos (DBMS)

Un sistema manejador de base de datos es un conjunto de programas que permite a los usuarios crear y mantener una base de datos. El DBMS es, esencialmente, software de propósito general que facilita los procesos de definición, construcción y manipulación de bases de datos para distintas aplicaciones.

La ***definición*** de una base de datos involucra especificar los tipos de datos que serán almacenados, incluyendo una descripción detallada de cada uno de ellos. La ***construcción*** de la base de datos, es el proceso de almacenar los datos propiamente dichos en algún medio de almacenamiento controlado por el DBMS. La ***manipulación*** incluye aquellas funciones como las consultas, la actualización y la generación de reportes.

Cabe señalar, que el DBMS no necesariamente debe ser un software de propósito general, se puede escribir un conjunto de programas para crear y mantener una base de datos en particular, con lo que se tendría un DBMS de propósito específico. En cualquier caso se tendrá software para manipular la base de datos independientemente de la información almacenada.

1.3.2 Abstracción de datos

Un objetivo importante de un sistema de bases de datos es proporcionar a los usuarios una visión abstracta de los datos, es decir, el sistema esconde ciertos detalles de cómo se almacenan y mantienen los datos. Puesto que muchos usuarios de sistemas de bases de datos no tienen experiencia en el uso de computadoras, se les esconde la complejidad a través de diversos niveles de abstracción para simplificar su interacción con el sistema, lo cual da como resultado una *arquitectura de tres niveles.*

Nivel interno o físico

Constituye el nivel más bajo de abstracción y describe cómo se almacenan realmente los datos. Este nivel es una representación de bajo nivel de la base de datos completa. Se define para este nivel un esquema llamado esquema interno, el cual usa un modelo de datos físico y describe en forma detallada las rutas de almacenamiento y acceso a los datos para la base de datos.

Nivel conceptual

Es el siguiente nivel de abstracción y describe qué datos son realmente almacenados en la base de datos y las relaciones que existen entre ellos. En este nivel se define el esquema conceptual, el cual es una descripción global de la base de datos que oculta los detalles de las estructuras del almacenamiento físico y se concentra en describir entidades, tipos de datos y relaciones. El nivel conceptual de abstracción lo usan los administradores de bases de datos, quienes deben decidir qué información se va a guardar en la base de datos.

Nivel externo o de visión.

Es el nivel más alto de abstracción y describe sólo parte de la base de datos completa. Incluye varios esquemas externos, también llamados vistas de usuario. Cada esquema externo describe la vista de la base de datos que tendrá un grupo de usuarios, típicamente, cada vista describe la parte de la base de datos que es de interés para un grupo en particular y le oculta el resto.

1.3.3 Modelos de datos

Un modelo de datos es una colección de herramientas conceptuales para describir datos, relaciones entre ellos, semántica asociada a los datos y restricciones de consistencia. Los modelos de datos existentes se han dividido en tres grupos: modelos lógicos basados en objetos, modelos lógicos basados en registros y modelos físicos de datos.

1.3.3.1 *Modelos lógicos basados en objetos*

Se usan para describir datos en los niveles conceptual y de visión. Se caracterizan por el hecho de que proporcionan capacidad de estructuración bastante flexible y permiten especificar restricciones de datos explícitamente. Algunos ejemplos de este tipo de modelos son el modelo Entidad-Relación y el modelo Orientado a Objetos, los cuales se describen brevemente a continuación y serán retomados a detalle en la siguiente sección de este trabajo.

Modelo Entidad-Relación

Se basa en una percepción de un mundo real que consiste en un conjunto de objetos básicos llamados *entidades* y de las *relaciones* entre estos objetos. Una entidad es un objeto que es distinguible de otros por medio de un conjunto específico de atributos, una relación es una asociación entre varias entidades.

Además de entidades y relaciones, este modelo representa ciertas restricciones a las que deben ajustarse los contenidos de una base de datos. Una restricción importante es la de cardinalidad de asignación, que expresa el número de entidades a las que puede asociarse otra entidad mediante un conjunto de relación.

Modelo Orientado a Objetos

Se basa, al igual que el modelo entidad-relación, en una colección de objetos. Un objeto contiene valores almacenados en variables de instancia dentro del objeto, a diferencia de los modelos orientados a registros, estos valores son objetos por sí mismos. Así, los objetos contienen objetos a un nivel de anidamiento de profundidad arbitraria. Un objeto también contiene partes de código que operan sobre el objeto, estas partes se llaman métodos.

Los objetos que contienen los mismos tipos de valores y los mismos métodos se agrupan en clases. La única forma en la que un objeto puede acceder a los datos de otro es invocando a un método de ese otro objeto. La parte interna del objeto (las variables de instancia y el código de método) no son visibles externamente, como resultado se tienen dos niveles de abstracción de

datos. A diferencia de las entidades, cada objeto tiene su propia identidad única independiente de los valores que contiene, la distinción entre objetos individuales se mantiene en el nivel físico por medio de identificadores de objeto.

1.3.3.2 Modelos Lógicos basados en registros

Se utilizan para describir datos en los niveles conceptual y físico, para especificar la estructura lógica y global de la base de datos y para proporcionar una descripción a nivel más alto de la implantación.

Se llaman así porque la base de datos está estructurada en registros de formato fijo de varios tipos. Cada tipo de registro define un número fijo de campos, o atributos, y cada campo normalmente es de longitud fija lo cual simplifica la implantación del nivel físico de la base de datos.

Estos modelos no incluyen, a diferencia de los modelos orientados a objetos, un mecanismo para la representación directa de código en la base de datos, en su lugar, hay lenguajes separados que se asocian con el modelo para realizar consultas y actualizaciones.

Los tres modelos con mayor aceptación son: el modelo relacional, el modelo de red y el modelo jerárquico A continuación se proporciona una breve explicación de cada uno de ellos.

Modelo relacional

Representa los datos y sus relaciones mediante una colección de tablas cada una de las cuales tiene un número de columnas con nombres únicos

Los modelos relacionales se diferencian de los modelos de red y jerárquico en que no usan punteros o enlaces, en cambio conecta registros mediante los valores que éstos contienen.

Modelo de red

En este modelo los datos se representan mediante colecciones de registros y las relaciones entre los datos se representan por medio de enlaces, los cuáles pueden verse como punteros.

Modelo jerárquico

Es similar al modelo de red en el sentido de que los datos y sus relaciones se representan mediante registros y enlaces respectivamente, en este caso, los registros están organizados como colecciones de árboles.

1.3.3.3 Modelos físicos de datos

Estos modelos se usan para describir datos en el nivel más bajo de abstracción, a diferencia de los modelos lógicos, hay muy pocos modelos físicos de datos en uso.

1.3.4 Independencia de datos

Es la capacidad de modificar una definición de un esquema en un nivel sin afectar la definición de un esquema en el nivel superior siguiente. Existen dos niveles de independencia de datos.

Independencia física

Es la capacidad de modificar el esquema físico sin provocar que se vuelvan a escribir los programas de aplicación. Esto es necesario en algunas ocasiones para mejorar el funcionamiento

Independencia lógica

Es la capacidad de modificar el esquema conceptual sin provocar que se vuelvan a escribir los programas de aplicación. Estas modificaciones son necesarias siempre que se altera la estructura lógica de la base de datos.

La independencia lógica de datos es más difícil de lograr que la física ya que los programas de aplicación dependen fuertemente de la estructura lógica de los datos.

La independencia de datos oculta detalles de implantación a los usuarios, lo cual les permite concentrarse en la estructura general en vez de en los detalles de implantación de bajo nivel.

1.3.5 Lenguaje de Definición de Datos (DDL)

Es un lenguaje especial por medio del cual es posible expresar un conjunto de definiciones que representan un esquema de base de datos. El resultado de la compilación de sentencias de DDL es un conjunto de tablas, las cuáles se almacenan en un archivo especial llamado diccionario de datos (o directorio).

Este diccionario de datos contiene metadatos (que son datos sobre los datos). Este archivo es consultado antes de leer o modificar los datos reales en el sistema.

La estructura de almacenamiento y los métodos de acceso se especifican por medio de un conjunto de definiciones en un tipo especial de DDL llamado lenguaje de almacenamiento y definición de datos

1.3.6 Lenguaje de manipulación de datos (DML)

Por manipulación de datos se entiende:

- La recuperación de información almacenada en la base de datos.
- La inserción de nueva información.
- La supresión de información.
- La modificación de datos almacenados.

Mientras que en el nivel físico se definen algoritmos que permiten el acceso eficiente a los datos, en los niveles de abstracción más altos se debe poner énfasis en la facilidad de uso, con el objeto de proporcionar una interacción eficiente entre las personas y el sistema.

El DML es un lenguaje que permite a los usuarios acceder o manipular datos dependiendo de la manera en que estén organizados por el modelo de datos adecuado.

La sentencia que solicita la recuperación de información se llama consulta, el fragmento de un DML que implica recuperación de información se llama lenguaje de consultas. Aunque técnicamente es incorrecto, generalmente se utilizan los términos lenguaje de consultas y lenguaje de manipulación de datos como sinónimos.

1.4.- ALMACENAMIENTO

El hecho de que los tiempos de acceso a disco son mucho *más lentos* que los tiempos de acceso a la memoria principal, dio lugar al surgimiento de la tecnología de métodos de acceso y a las estructuras de almacenamiento.

El manejador de disco es un componente que reside en el sistema operativo, y es el responsable de todas las operaciones físicas de entrada y salida, por lo tanto debe tener conocimiento de las direcciones físicas, es decir, cuando el manejador de archivos haga referencia a alguna página especifica *p*, el manejador de disco necesita saber exactamente dónde se encuentra la página *p* físicamente. De esta forma, el manejador de archivos "ve" al disco simplemente como una colección lógica de conjuntos de páginas y cada conjunto, a su vez, como una colección de páginas de tamaño fijo. Cada conjunto de páginas tiene un identificador único, y cada página, es identificada por un número de página que es único en el disco. El mapeo entre números de página y direcciones físicas en disco es entendido y mantenido por el manejador de disco.

1.4.1 Paginación

Una forma de optimizar los accesos a disco es a través de la paginación. Bajo este esquema, la base de datos se divide en cierto número de bloques de longitud fija, que se denominan páginas. Supongamos que existen n páginas, numeradas de la 1 a la n, las cuales no necesariamente tienen un determinado orden en el disco, sin embargo, debe haber una forma de encontrar la página *i*-ésima de la base de datos para cualquier *i* dado, esto se logra utilizando una tabla de paginación.

La tabla de paginación tiene *n* entradas, una por cada página de la base de datos. Cada entrada contiene un puntero a una página del disco, no es necesario que el orden lógico de las páginas de la base de datos corresponda al orden físico en que se colocan en el disco.

La idea principal en que se basa la técnica de paginación es mantener dos tablas de paginación durante una transacción, la tabla de paginación actual y la tabla de paginación doble. Cuando comienza la transacción, las dos tablas de paginación son idénticas. La tabla de paginación doble no se modifica en ningún momento durante la transacción, mientras que la tabla de paginación actual se modifica cuando la transacción realiza alguna operación de escritura. Todas las operaciones de entrada y de salida utilizan la tabla de paginación actual para localizar páginas en el disco.

El enfoque de recuperación de la doble paginación consiste en almacenar la tabla de paginación en memoria no volátil de tal forma que el estado de la base de datos antes de ejecutarse la transacción pueda recuperarse en caso de una caída o aborto de la misma. Cuando la transacción termina, la tabla de paginación actual se escribe en memoria no volátil convirtiéndose en la nueva tabla de paginación doble. Es importante que la tabla de paginación doble se almacene en memoria no volátil, ya que es la única forma de localizar las páginas de la base de datos.

1.4.2 Indexación

La estructura de índice se utiliza para permitir el acceso aleatorio rápido a los registros de un archivo. Cada estructura de índice está asociada con una clave de búsqueda determinada. Si el archivo está ordenado secuencialmente y elegimos incluir varios índices en diferentes claves de búsqueda, el índice cuya clave de búsqueda especifica el orden secuencial del archivo es el índice primario. Los demás se llaman índices secundarios. La clave de búsqueda de un índice primario es normalmente la clave primaria.

Los índices secundarios pueden tener una estructura diferente a la de los primarios. Los apuntadores no señalan directamente al archivo sino que apuntan a una pila que contiene apuntadores al archivo, este enfoque permite almacenar juntos todos los apuntadores que corresponden a un valor de llave secundaria de búsqueda determinado, lo cual es útil en algunos tipos de consulta en los que la mayor parte del procesamiento se lleva a cabo utilizando únicamente los apuntadores. Los índices secundarios mejoran el rendimiento cuando se hacen consultas que emplean índices diferentes del primario, por otro lado, implican un gasto extra considerable cuando se modifica la base de datos.

Archivos de índices secuenciales

Los archivos que se encuentran ordenados secuencialmente y que tienen tanto un índice como una clave de búsqueda primaria se llaman archivos de índices secuenciales y se encuentran entre los esquemas de indexación más antiguos usados en los sistemas de bases de datos ya que están diseñados para aplicaciones que requieren tanto un procesamiento secuencial del archivo completo como un acceso aleatorio a registros individuales.

Como su nombre lo indica, los archivos de índice secuencial constan de un archivo secuencial y un índice.

Las técnicas de búsqueda más utilizada en archivos de índices secuenciales son el Árbol B, el Árbol B+ y el Hashing.

1.4.3 Técnicas de compresión

Las técnicas de compresión son formas de reducir la capacidad de espacio de almacenamiento requerido para un conjunto dado de datos. Frecuentemente el resultado de la compresión no sólo reducirá el espacio de almacenamiento, sino también la cantidad de accesos al disco, dado que si los datos ocupan menos espacio, se necesitarán menos operaciones de entrada/salida para acceder a ellos. Sin embargo, se requerirá de procesamiento extra para descomprimir los datos una vez almacenados.

Las técnicas de compresión son diseñadas para aprovechar el hecho de que los datos no son completamente aleatorios sino que son predecibles en cierto grado.

CAPÍTULO II

TIPOS DE BASES DE DATOS

2.1. BASES DE DATOS RELACIONALES

El enfoque relacional busca alcanzar la misma meta de cualquier enfoque para bases de datos, ésta consiste en proporcionar soporte para la administración de los datos como un recurso compartido para un grupo de usuarios.

En 1970 cambió por completo la forma de ver a las bases de datos cuando E. F. Codd introdujo el modelo relacional de datos. En aquel tiempo, los enfoques existentes para bases de datos usaban apuntadores físicos o direcciones de disco para relacionar registros que se encontraban físicamente en archivos distintos, esto traía consigo el problema de tener que hacer una modificación completa de estos apuntadores cada vez que se agregaba una nueva unidad de disco o algún archivo era movido de una localidad física a otra. El modelo relacional , basado en relaciones lógicas entre los datos, puso fin a este tipo de problemas ya que permitía que los usuarios, las aplicaciones y la organización lógica de la base de datos fueran independientes de la forma en que los datos se almacenaban en los medios físicos, en este enfoque, los datos siempre aparecen en el formato de una tabla simple, una base de datos es, entonces, vista como una colección de tablas cuyos datos son presentados externamente en forma tabular independientemente del formato en el que sean organizados dentro de la misma. Desde el punto de vista de los usuarios, no existen conexiones o ligas que mantengan unidos a los datos o que representen caminos de búsqueda a través de la base de datos, y no tiene importancia la posición que guarde alguna tabla o algún renglón en el medio físico de almacenamiento.

A continuación se describen los principales elementos de este enfoque.

2.1.1 Conceptos fundamentales del modelo de datos relacional

Relaciones y Tablas

La palabra relación está basada en la teoría de conjuntos, de donde Codd derivó su modelo y equivale a una tabla simple de dos dimensiones compuesta de columnas y renglones de datos. Con frecuencia, los renglones de una tabla se denominan "tuplas", esta palabra (en inglés *tuple*), se deriva a su vez de la palabra en inglés *couple*, que significa pareja, es decir, representa pares o parejas de valores cuya conexión o liga es la relación en sí misma.

Para aclarar un poco más este concepto pongamos como ejemplo la relación "es padre de", esta relación es de la forma: *a es_padre_de b* , cada elemento de la relación es_padre_de constituye una tupla y cada tupla tiene como componentes al padre y al hijo dentro de esa relación. La siguiente figura es una representación tabular de la relación es_padre_de.

es_padre_de	padre	hijo
tupla →	Juan	Pedro
tupla →	Jose	Luis
tupla →	Jorge	María

Es importante señalar que aún cuando la palabra tupla da la idea de "pares de valores", las relaciones no están limitadas a elementos que se toman de dos en dos, es decir, una relación puede constar de *n* elementos ligados entre sí en forma lógica.

El número de renglones y columnas que puede contener una tabla está dado por la capacidad del software que se esté utilizando como manejador de la base de datos.

Dominios

Los dominios de una relación son los conjuntos de los cuales podemos elegir elementos para formar tuplas o renglones de la tabla y son definidos por los usuarios del DBMS. En la definición de un dominio se debe especificar un conjunto de valores de tipo similar, este tipo puede ser simple, como los tipos de datos utilizados en forma convencional para programar aplicaciones (enteros, fecha, punto flotante, etc.) , o de algún tipo de datos definido por el usuario.

Se llama "integridad de dominio" a la capacidad de un DBMS para permitir que un campo en la tabla sea llenado exclusivamente con un valor "permitido" conforme a la definición del dominio para ese campo, es decir, si se definió el dominio "enteros" para un determinado campo, es labor del DBMS el no permitir la inserción de un valor de tipo "punto flotante" en el mismo.

La definición de un dominio también puede estar dada por rangos de datos de un mismo tipo, por ejemplo, es válido definir que el dominio para el campo "Año" dentro de una tabla esté dado por los valores 1900 a 1999 y para cumplir con la integridad de dominio no se debe permitir la inserción de un valor fuera de ese rango.

Atributos

Los atributos de la relación corresponden a las columnas de una tabla. Cada columna es la implantación de un dominio y el diseñador de la base de datos agrega esa columna o nuevo dominio a la tabla dándole un nombre a ese atributo.

El grado de una relación se refiere al número de dominios para los cuales está definida esa relación, es decir, se refiere al número de columnas de la tabla.

Valores Nulos

Al momento de asignar los datos para los renglones de una relación puede ocurrir que no se conozcan los valores de uno o más de los atributos de ese renglón, tal vez porque ese o esos atributos no se aplican para ese caso en particular, en ese caso y si la definición de dominio para ese atributo lo permite, el campo es llenado con un valor nulo en la base de datos. Un valor nulo no es un espacio en blanco o un cero, es simplemente un dato que denota la ausencia de valor para ese atributo.

Llaves

Las llaves constituyen la forma de que una entidad, dada por un renglón de una tabla, sea única dentro de esa tabla. En general existen dos tipos de llaves, las llaves primarias y las llaves externas.

- *Llave primaria.*- Está constituida por los atributos que identifican a un renglón en forma única dentro de una tabla y previene la aparición de renglones redundantes o duplicados. En ocasiones se llama únicamente llave al conjunto mínimo de estos atributos, es decir, se dice que la llave es una llave primaria mínima.
- *Llave externa.*- También denominada llave foránea, es un conjunto de atributos que, siendo llave primaria de alguna tabla, se encuentran presentes en otra, incluso si la o las columnas no tienen el mismo nombre.

De lo anterior podemos concluir que las características de una relación o tabla deben ser:

- Una tabla es una estructura de dos dimensiones compuesta por las intersecciones de sus renglones y columnas.
- Cada renglón define a una entidad única dentro de el conjunto de entidades, además, cada columna define los atributos de esa entidad.
- Cada intersección entre un renglón y una columna contiene un solo valor. Los datos deben ser clasificados de acuerdo a su formato y a su función y elegidos de un dominio previamente definido para ese campo.
- Cada renglón debe contener una llave primaria.
- Cada columna representa un atributo distinto, por lo tanto debe tener un nombre distinto al de las demás.
- En general, el orden de los renglones y las columnas es irrelevante para el usuario.

2.1.2 Integridad

Las reglas de integridad tienen la función de restringir los valores que pueden presentarse en una base de datos y son un aspecto muy importante que debe tomarse en cuenta para un buen diseño de la base de datos

El modelo relacional de Codd contempla varias reglas que son usadas para verificar la validez de los datos en la base de datos así como para añadir significado a la estructura de los mismos, estas restricciones protegen a la base de datos contra daños accidentales y proporcionan una manera de asegurar que los cambios

que se hacen en la base de datos por usuarios autorizados no den como resultado pérdidas en la consistencia de los mismos.

Ya se mencionó en que consiste la integridad de dominio, ésta constituye la regla de integridad más básica. A continuación se hará mención de otras dos reglas importantes para este enfoque de bases de datos

Integridad de la Entidad

Los renglones de una relación representan instancias de objetos específicos del mundo real en la base de datos, estos objetos específicos son llamados "entidades". Como se dijo anteriormente, la llave de una relación identifica en forma única a cada renglón y, por lo tanto, a cada entidad en esa tabla, es decir, no se puede hablar de que existe una entidad si alguno o algunos de sus atributos llave tienen un valor nulo, la regla de integridad de la entidad dice:

Los valores de los atributos que conforman la llave primaria de una relación deben ser únicos y no deben contener valores nulos

Integridad Referencial

Cuando se construyen relaciones, las llaves externas se usan para ligar a los renglones de una relación con los renglones de otra. Las llaves externas pueden contener valores nulos, sin embargo, cuando tienen valores no nulos estos valores deben tener una correspondencia con algún valor en la relación para la cual esa llave externa es una llave primaria, esto asegura que una llave externa nunca tendrá un valor inválido además de que hace imposible el hecho de borrar un renglón cuya llave primaria esté siendo usado como llave externa en otra relación. La regla de integridad referencial nos dice:

Todas las llaves externas deben contener ya sea un valor nulo o un valor que corresponda con el valor de la llave primaria de otra tabla con la cual esté relacionada

2.1.3 Normalización

El proceso de normalización es una manera semántica de asegurar la mínima redundancia posible de datos entre las relaciones en el diseño lógico de una base de datos. Se basa en el principio de que una entidad debe encontrarse en un solo lugar, pero puede repetirse si es "absolutamente" necesario.

La teoría de la normalización se basa en el concepto de formas normales. Se dice que una relación se encuentra en una forma normal particular si satisface cierto conjunto de especificaciones.

Primera forma normal

Una relación se encuentra en la primera forma normal, si satisface las siguientes condiciones:

1.- No tiene atributos repetidos.
2.- No existen tuplas duplicadas.
3.- El orden de aparición de los atributos es indiferente.

Cada relación en una base de datos relacional se encuentra al menos en la primera forma normal.

Segunda forma normal

Se dice que una relación se encuentra en la segunda forma normal, si ésta se encuentra en la primera forma normal y cumple con las siguientes condiciones:

1 - La llave primaria consiste de un solo atributo
2.- Existe un atributo no-llave.
3.- Cada atributo no-llave depende funcionalmente de la llave primaria, la cual es compuesta.

Tercera forma normal

Se dice que una relación se encuentra en la tercera forma normal si está en la segunda forma normal y todos los atributos no llave son completamente independientes unos de otros

2.1.4 Lenguajes de Consulta

El segundo mayor componente del modelo de datos relacional consiste en los lenguajes que estos sistemas ponen a disposición de los programadores, usuarios finales y diseñadores de la base de datos.

Un lenguaje de consulta es aquel en el que el usuario solicita información a la base de datos, regularmente, estos lenguajes son de más alto nivel que los lenguajes de programación estándar y pueden clasificarse en *procedimentales* y *no procedimentales*. En un lenguaje *procedimental*, el usuario da instrucciones al sistema para que realice una serie de operaciones en la base de datos para encontrar el resultado deseado, por el contrario, en un lenguaje *no procedimental*, el usuario describe la información deseada sin dar un procedimiento específico para obtener esa información. El álgebra relacional es un lenguaje de tipo *procedimental* mientras que el cálculo relacional es del tipo *no procedimental*.

Los sistemas comerciales de bases de datos incluyen los 2 enfoques, el *procedimental* y el *no procedimental*, sin embargo, el lenguaje estándar para un DBMS relacional, el SQL, es un lenguaje con características *no procedimentales*.

Álgebra Relacional

El álgebra relacional es similar al álgebra normal con la diferencia de que las variables son relaciones y los operadores manipulan a estas relaciones para formar nuevas relaciones, de hecho, el álgebra relacional cumple con la propiedad de cerradura, es decir, el resultado de una o más operaciones relacionales es siempre una relación.

A continuación se presenta una breve descripción de las operaciones que se pueden realizar a través del álgebra relacional.

Unión

Combina todos los renglones de las dos tablas, para que pueda darse, las columnas y los dominios de las dos tablas deben ser idénticos, en este caso se dice que las tablas son "compatibles para unión".

Intersección

Da como resultado una tabla que contiene únicamente los renglones que pertenecen a las dos tablas sobre las cuales se efectuó la operación, estas tablas deben ser compatibles para unión.

Diferencia

El resultado de esta operación es una tabla que contiene todos los renglones de la primera tabla que no forman parte de la segunda. En este caso, las tablas también deben ser compatibles para unión.

Producto

Mediante esta operación se obtienen todos los posibles pares de renglones de las dos tablas.

Selección

Esta operación permite encontrar la información que cumpla con una serie de condiciones especificadas a través de un predicado. Para que una condición pueda ser expresada se usan los símbolos de comparación: $=$, $\neq$, $<$, $\leq$, $>$ y $\geq$. Es posible, además, combinar varios predicados mediante la utilización de los conectores "y" ($\wedge$) y "o" ($\vee$).

La operación selección da como resultado un subconjunto "horizontal" de una tabla.

Proyección

Produce el listado de todos los valores para la columna o columnas (atributos) elegidas de una tabla, en otras palabras, da como resultado un subconjunto "vertical" de una tabla.

Join

Permite combinar la información de dos o más tablas mediante la relación de tablas independientes a través de sus atributos comunes. Es una combinación de algunas de las operaciones anteriores dado que para dos

relaciones, digamos A y B, primero se obtiene el producto de A y B, después se hace una selección para eliminar algunas tuplas o renglones (el criterio de selección se especifica como parte de la operación) y finalmente se eliminan algunos atributos o columas de la tabla resultante, es decir, se hace una proyección.

2.1.5 Conceptos sobre implantaciones relacionales

Modelo Entidad-Relación

Según un diccionario, un modelo es una descripción o analogía usada para visualizar algo que no puede ser observado directamente. En este caso, el modelo entidad-relación sirve para tener una visión relacional del modelo de datos en el nivel conceptual de la base de datos. Fue desarrollado en 1976 por Peter Chen en el Instituto Tecnológico de Massachussets y es parte fundamental para un diseño apropiado de una base de datos relacional.

El modelo E-R se basa en diagramas que representan a la base de datos en forma conceptual desde el punto de vista del usuario final. Estos diagramas constan de 3 componentes principales: entidades, relaciones y atributos.

Entidad

En el contexto del modelo Entidad-Relación la palabra entidad se refiere a un "conjunto de objetos" y no a una sola ocurrencia de este objeto, es decir, una entidad en el modelo E-R simboliza una tabla y no un renglón como se había dicho anteriormente En un diagrama entidad-relación, una entidad se representa por un rectángulo.

Existen entidades que no pueden darse en forma aislada, es decir, dependen para su existencia de la existencia de otra entidad con la cual relacionarse, estas entidades son llamadas entidades débiles y se representan por rectángulos dobles en el diagrama E-R.

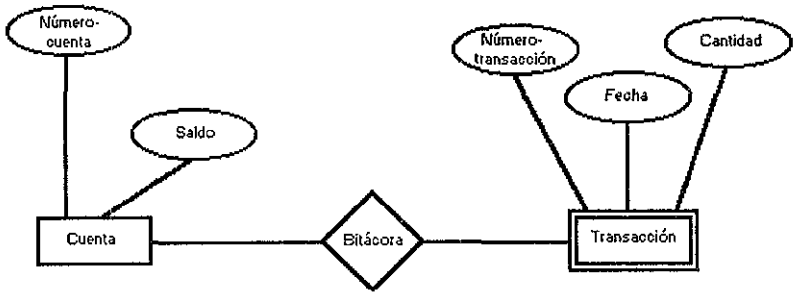
Atributos

Los atributos son características propias de las entidades y se representan en los diagramas por medio de óvalos que se unen a la entidad correspondiente por medio de una línea, por claridad, la palabra que representa al atributo llave de una entidad siempre se subraya. Los atributos que pueden tener más de un valor para una *ocurrencia dentro de la tabla son representados en el diagrama con una línea doble uniéndolos a la entidad* correspondiente.

Relaciones

Las relaciones son asociaciones entre entidades y sus nombres son las descripciones de las relaciones en sí. Son representadas en los diagramas a través de rombos. El grado de una relación está dado por el número de entidades conectadas o participantes en esa relación. La cardinalidad también debe ser expresada, es decir, se debe especificar si la relación entre entidades es uno a uno, uno a muchos o muchos a muchos.

En la figura se muestra un ejemplo de diagrama E/R.



Para convertir un diagrama E/R en tablas de una base de datos se deben de considerar los siguientes criterios:

- Para cada conjunto de entidades fuerte se necesita crear una tabla que lleve el mismo nombre del conjunto y cuyas columnas correspondan en número y nombre a los atributos del mismo.
- Para cada conjunto de entidades débil se necesita crear una tabla que lleve el mismo nombre del conjunto y cuyas columnas correspondan en número y nombre a los atributos del mismo más los atributos que forman la llave primaria del conjunto de entidades fuertes del cual depende.
- Para cada conjunto de relaciones se debe crear una tabla, la cual debe incluir, en sus columnas, por lo menos las llaves primarias de las entidades implicadas en la relación más los atributos de la relación en sí.
- Un conjunto de relaciones de generalización, también llamado "Relación ISA" que proviene de las palabras en inglés *is a*, es aquel que permite establecer diferencias entre entidades que pueden ser agrupadas en su totalidad en una entidad de nivel superior. Para cada conjunto de relaciones de generalización (Relaciones ISA) se tienen dos opciones: La primera consiste en crear una tabla para el conjunto de entidades de nivel superior, esta tabla debe contener todos los atributos de este conjunto, además se debe crear una tabla para cada uno de los conjuntos de entidades de nivel inferior, cada una de estas tablas deberá contener los atributos individuales del conjunto más la llave primaria del conjunto de entidades de nivel superior. En la segunda opción no se crea ninguna tabla para el conjunto de entidades de nivel superior, se debe crear una tabla por cada uno de los conjuntos de entidades de nivel inferior, cada tabla debe contener todos los atributos del conjunto de entidades de nivel superior más todos sus atributos individuales.

2.1.6 Control de concurrencia

En ambientes multiusuario es posible ejecutar varias transacciones de manera concurrente, por lo tanto es necesario que el sistema controle la interacción entre las transacciones concurrentes para evitar que destruyan la consistencia de la base de datos. Este control se logra a través de varios mecanismo llamados esquemas de control de concurrencia, todos trabajan retrasando una operación o abortando la transacción que solicitó la operación. Los mecanismos más comunes son los protocolos de bloqueo y los esquemas de ordenación por hora de entrada.

Un protocolo de bloqueo es un conjunto de reglas que definen cuándo una transacción puede bloquear y desbloquear cada uno de los datos de la base de datos, obviamente, cuando un dato se encuentra bloqueado se encuentra disponible únicamente para la transacción que lo está utilizando. El protocolo de bloqueo permite a una transacción bloquear un nuevo dato sólo si ya ha desbloqueado el anterior.

Un esquema de control de concurrencia por orden de entrada asocia a cada transacción del sistema una hora de entrada única Las horas de entrada de las transacciones determinan el orden en que serán ejecutadas, así,

si la hora de entrada de la transacción Ti es menor que la de Tj, el esquema garantiza que Ti se llevará a cabo antes que Tj, nunca al mismo tiempo.

2.1.7 Recuperación

La tarea principal del componente llamado "administrador de recuperación" del DBMS es regresar a la base de datos a un estado consistente cuando alguna falla ha dejado a la misma en un estado de inconsistencia.

Después de una falla, el administrador de recuperación debe identificar qué transacciones deben deshacerse y cuáles deben volver a efectuarse para garantizar que la base de datos quede en un estado consistente, para esto se vale de la ayuda de un archivo de registro o bitácora, en el cual se registran todas las operaciones llevadas a cabo por cada una de las transacciones de la base de datos. Este archivo de registro se guarda por duplicado en áreas físicamente separadas del área donde se almacena la base de datos con el fin de que siempre se tenga acceso al mismo.

Existen varios tipos de fallas que pueden encontrarse en los sistemas de bases de datos y para los cuales se deben contemplar esquemas de recuperación, los más comunes son.

- *Errores lógicos*: La transacción no puede continuar con su ejecución normal debido a alguna condición interna, como una entrada inválida, información no localizada, desbordamiento o que se exceda el límite de los recursos.
- *Errores de sistema*: El sistema ha entrado en un estado no deseable, como resultado del cual la transacción no puede continuar con su ejecución normal, sin embargo la transacción puede volverse a ejecutar más tarde.
- *Caída del sistema*: El hardware funciona mal, provocando la pérdida del contenido del almacenamiento volátil, mientras que el contenido del almacenamiento no volátil permanece intacto.
- *Fallo de disco*: Un bloque del disco pierde su contenido debido a la rotura de la cabeza o a fallos durante una operación de transacción de información.

El administrador de recuperación puede estar basado en uno de cuatro algoritmos principales, los cuales se describen brevemente a continuación.

Deshacer/Rehacer

Es el más complejo ya que involucra las acciones de "deshacer" y "rehacer" transacciones después de una falla, tiene la ventaja de que el administrador del buffer decide cuando debe vaciar el contenido de éste al disco, reduciendo la sobrecarga durante la operación normal pero incrementándola al reiniciar después de ocurrida alguna falla, debido al número de operaciones que se deben realizar.

Deshacer/No-Rehacer

En este algoritmo el buffer se limpia hasta que una transacción se efectúa con éxito, de esta manera ninguna transacción se tiene que volver a efectuar al reiniciar después de una falla

No-Deshacer/Rehacer

Usando este algoritmo, el administrador de recuperación no escribe las transacciones que aún no se terminan en la base de datos estable, además se forza a que los registros se mantengan en memoria principal hasta que la transacción se efectúe completa y satisfactoriamente, de esta forma, al reiniciar después de una falla, se tienen que "rehacer" las transacciones que estaban en memoria en ese momento, pero no se tiene que "deshacer" ninguna.

No-Deshacer/No-Rehacer

En este algoritmo ninguna actualización hecha por una transacción se debe escribir en la base de datos estable hasta que ésta se termine con éxito pero el contenido del buffer debe ser almacenado antes de que la transacción termine, esta paradoja se resuelve almacenando el contenido del buffer en algún lugar fuera de la base de datos estable, esto permite que, al reiniciar después de una falla, ninguna transacción tenga que ser deshecha o se tenga que volver a efectuar.

2.1.8 Seguridad

El término seguridad de la base de datos normalmente se refiere a la protección contra el acceso no autorizado a la misma. Una base de datos puede protegerse en varios niveles:

- Físico.- El lugar asignado al equipo de cómputo debe ser un área protegida.
- Humano.- Se debe tener control de acceso a dicha área.
- *Sistema operativo.- El sistema operativo debe brindar seguridad* a la base de datos.
- Sistemas de base de datos.- Los usuarios deben de tener autorización para efectuar sólo algunas operaciones sobre la base de datos, sin llegar a la modificación de la misma.

Vistas

Las vistas constituyen una forma de proporcionar al usuario un modelo "personalizado" de la base de datos. Una vista puede ocultar datos que el usuario no debe ver., lo cuál sirve tanto para simplificar la utilización del sistema como para fomentar la seguridad. El uso del sistema es más sencillo debido a que el usuario puede restringir su atención a los datos que le interesan. La seguridad se logra si se cuenta con un mecanismo que limite a los usuarios a su vista o vistas personales.

Permisos

Es importante definir los permisos que tendrá un usuario al trabajar en un sistema de bases de datos, esto con el fin de prevenir daños tanto intencionales como no intencionales provenientes de los mismos usuarios del sistema, en este sentido se deben definir por usuario los permisos de lectura, inserción, actualización y borrado de la información de la base de datos.

2.2 BASES DE DATOS DISTRIBUÍDAS

Durante los años 70 las computadoras fueron ampliamente usadas para construir poderosos sistemas de bases de datos integrados, al mismo tiempo las redes de computadoras han tenido un gran desarrollo permitiendo la conexión de diferentes computadoras y el intercambio de datos y otros recursos entre ellas.

En los 80 se manifestaron una serie de cambios sociales y tecnológicos que afectaron el diseño y desarrollo de las bases de datos:

- La operación de los negocios se volvió descentralizada geográficamente
- La demanda del cliente y las necesidades del mercado favorecieron el estilo de administración descentralizado
- Surgieron las microcomputadoras con la capacidad de grandes computadoras.
- Las corporaciones adoptaron las redes de área local (LAN's).

Una base de datos distribuida puede definirse como una colección de datos distribuidos sobre diferentes computadoras de una red. Cada nodo de la red tiene capacidad de procesamiento autónomo y puede realizar la ejecución de aplicaciones a nivel local, sin embargo, cada nodo debe participar en la ejecución de por lo menos una aplicación global, la cuál requiere el acceso a datos de diferentes nodos a través del uso de un subsistema de comunicaciones. Las bases de datos distribuidas se ajustan en forma natural a las estructuras descentralizadas de muchas organizaciones. Dos características que distinguen a este tipo de bases de datos son:

- *Distribución.* Los datos no residen en el mismo nodo (por lo tanto en el mismo procesador).
- *Correlación lógica* Los datos cuentan con algunas propiedades que los mantienen juntos, aunque físicamente se encuentren separados.

Objetivos de un sistema de base de datos distribuida

Los objetivos principales de una base de datos distribuida son:

- *Autonomía del nodo.* Es un objetivo importane que habilita a cualquier nodo para controlar y procesar sus datos locales. Cada nodo debe almacenar la información necesaria en su diccionario de datos sin tener que enviarla al diccionario de datos global centralizado. La ventaja de esto es que la administración de la base de datos no necesita estar centralizada. Cada base de datos local puede ser controlada por un administrador local. Se requiere que haya coordnación entre los administradores de las bases de datos distribuidas.
- *Transparencia en la ubicación.* Significa que el usuario no necesita saber donde reside un dato específico dentro de un sistema distribuido, es decir, el usuario va a realizar una consulta y va a obtener su información aunque ésta se encuentre en un nodo remoto. La información de la ubicación de los datos se encuentra en el diccionario de datos y es usado por el DBMS para encontrar los datos.
- *Transparencia en la fragmentación.* La forma más simple de almacenar una relación en una base de datos distribuida es en un solo lugar. Sin embargo, por razones de rendimiento, frecuentemente es necesario dividir la relación en fragmentos pequeños almacenados en distintos sitios. La fragmentación es transparente para el usuario. La información de la fragmentación es almacenada en el diccionario de datos y usada por el DBMS distribuido para realizar automáticamente las consultas globales a través de consultas sobre fragmentos.
- *Transparencia en la repetición.* La repetición de los datos se usa por confiabilidad, disponibilidad y rendimiento. Un fragmento es repetido cuando se encuentra almacenado mas de dos veces, cada una en distinto nodo. Lo cual trae como ventaja la alta disponibilidad: si un sitio falla, la copia del fragmento se encuentra disponible en otro lugar. Los principales problemas de la repetición son la complejidad y la sobrecarga requeridas para mantener las copias idénticas. La actualización de una copia debe propagarse a todas las copias. Cuando una copia se recupera de una falla, debe actualizar su información de tal manera que tenga la misma que las otras copias. La repetición de la información es almacenada en el diccionario de datos y utilizada por el DBMS distribuido para administrar la consistencia de las copias.

2.2.1 Bases de datos centralizadas contra distribuidas

Las bases de datos distribuidas permiten el desarrollo de características que difieren de las que se encuentran en los sistemas tradicionales centralizados. A continuación se muestra una comparación entre las bases de datos distribuidas y las centralizadas basada en ciertas características relevantes:

- *Control centralizado.* Uno de los principales motivos para la introducción de las bases de datos fue la posibilidad de tener un control centralizado sobre la información, la función principal del administrador de una base de datos centralizada, es garantizar la seguridad de los datos. En las bases de datos distribuidas se enfatiza mucho menos la idea de un control centralizado, identificando en la mayoría de los casos una estructura de control jerárquico basada en un administrador global que tiene la responsabilidad central de toda la base de datos y en administradores locales que tienen la responsabilidad de sus respectivas bases de datos.
- *Independencia de datos.* La independencia de datos funciona en forma similar tanto en las bases de datos centralizadas como en las distribuidas, es decir, la organización de los datos es transparente para el programador y los programas son escritos teniendo una visión conceptual de los datos, sin embargo, para las bases de datos distribuidas se toma en cuenta otro aspecto que es la transparencia de la distribución, que se refiere a que los programas pueden escribirse como si la base de datos no fuera distribuida.
- *Reducción de redundancia.* En las bases de datos centralizadas, la redundancia se trata de evitar tanto como sea posible por dos razones: primero, por la inconsistencia que acarrea el tener varias copias de los mismos datos lógicos lo cuál se previene teniendo sólo una copia, y segundo, para ahorrar espacio en los dispositivos de almacenamiento. En las bases de datos distribuidas la redundancia de datos es una característica deseable ya que la disponibilidad de los sistemas puede incrementarse debido a que la falla en un nodo no detiene la ejecución de las aplicaciones en los nodos en los cuáles los datos se encuentran repetidos, lo cual aumenta la complejidad de la base de datos.
- *Privacidad y seguridad.* En las bases de datos centralizadas, el administrador puede asegurar que sólo se realizan accesos autorizados a los datos, sin embargo, sin ningún procedimiento especial de control este

enfoque es muy vulnerable a violaciones de seguridad. En las bases de datos distribuidas, cuando se tiene un alto grado de autonomía del nodo, el administrador local debe implantar sus propias medidas de seguridad aunque los problemas de seguridad son propios de los sistemas distribuidos en general ya que las redes de comunicaciones representan un punto vulnerable en cuanto a la protección de los datos.

2.2.2 Sistemas manejadores de bases de datos distribuidas

Un sistema manejador de bases de datos distribuidas (DDBMS) es un sistema que soporta la creación y mantenimiento de bases de datos distribuidas.

Varios de los DDBMSs disponibles comercialmente fueron desarrollados por los creadores de los DBMSs centralizados, estos sistemas contienen componentes adicionales que extienden las capacidades de un DBMS centralizado como el soporte para comunicaciones y la cooperación con otros DBMSs instalados en diferentes nodos de la red.

Las principales funciones de un sistema manejador de base de datos distribuida son:

- Administración de un diccionario de datos global que almacena información de los datos distribuidos.
- Definición de los datos distribuidos.
- Control semántico de los datos distribuidos.
- Procesamiento de consultas distribuidas
- Administración de transacciones distribuidas.

Un DDBMS debe incluir por lo menos los siguientes componentes:

- *Nodos*: Son los equipos que forman la red
- *Componentes de red*: Son el software y hardware de red de cada equipo que permiten que haya comunicación entre los nodos así como el intercambio de información..
- *Medio de comunicación*: es aquel que permite la transmisión de datos de un lugar a otro.
- *Procesador de transacciones*: es el software que se encuentra en cada equipo que requiere datos. El procesador de transacciones recibe y procesa los datos de aplicación requeridos.
- *Procesador de datos*: Es el software residente en cada computadora que almacena y recupera datos localizados en ese lugar.

Los DDBMSs pueden ser homogéneos o heterogéneos, se dice que son homogéneos cuando se encuentra el mismo DBMS en cada nodo sin importar si las computadoras y/o el sistema operativo son los mismos, por el contrario, son heterogéneos si existen al menos dos diferentes DBDMSs en el sistema. Los DDBMSs heterogéneos deben realizar la traducción entre los diferentes modelos de datos de los distintos DBMSs locales.

Los sistemas manejadores de bases de datos distribuidas tienen las siguiente ventajas sobre los sistemas tradicionales:

- Los datos están localizados cerca de lugar de mayor demanda
- Acceso rápido a los datos
- Procesamiento rápido de los datos
- Facilidad de crecimiento
- Mejoramiento de la comunicación
- Menos peligro de falla en un punto
- Independencia de procesador

Sin embargo estos sistemas también presentan algunas desventajas:

- Complejidad en la administración y control
- Seguridad
- Los enlaces de comunicación son mas lentos comparados con el tiempo de acceso de los dispositivos de almacenamientos tales como los discos.
- Los sistemas de comunicación presentan retardo de acceso.
- Los mensajes son caros en términos de instrucciones ejecutadas por el CPU.

2.2.3 Arquitectura de referencia de las bases de datos distribuidas

La arquitectura de referencia de las bases de datos distribuidas consta de los siguientes niveles:

- *Esquema global.*- Define a todos los datos contenidos en la base de datos como si ésta no fuera distribuida.
- *Esquema de fragmentación.*- Cada relación global puede dividirse fragmentos. El mapeo entre relaciones globales y fragmentos se define en el esquema de fragmentación, donde muchos fragmentos corresponden a una relación global pero sólo una relación global corresponde a un fragmento.
- *Esquema de asignación.*- El esquema de asignación define en qué nodo o nodos se localiza un fragmento y determina si la base de datos distribuida es redundante o no redundante.
- *Esquema de mapeo local.*- Se refiere al manejo de fragmentos por parte del DBMS local.

2.2.4 Transparencia de las bases de datos distribuidas

Un sistema de base de datos distribuida requiere de características funcionales que pueden ser agrupadas y descritas como un conjunto de características de transparencia, estas tienen la propiedad común de hacer transparente al usuario las complejidades de la base de datos distribuida. Las características de transaprencia de un DDBMS son.

- *Transparencia en la distribución.* Permite que una base de datos distribuida sea tratada como una sola base de datos, es decir, el usuario no debe de saber que los datos están particionados, que los datos pueden estar repetidos o bien la ubicación de los datos.
- *Transparencia en la transacción.* Permite que una transacción actualice los datos en varios nodos de la red. La transparencia de la transacción asegura que la transmisión será completa o abortará para mantener la integridad de la base de datos.
- *Transparencia a las fallas.* Asegura que en la falla de un nodo, el sistema continuará operando. Las funciones que se hayan perdido en la falla de un nodo son resueltas por otro nodo.
- *Transparencia en el rendimiento.* Permite que el sistema funcione como si fuera un DBMS centralizado. El sistema no sufrirá ninguna degradación debido a su uso en la red o debido a las diferentes plataforma de red. Además asegura que el sistema encontrará la mejor ruta para acceder a los datos remotos.
- *Transparencia en la heterogeniedad.* Permite la integración de de varios DBMSs locales bajo un mismo esquema global.

2.2.5 Fragmentación de datos

La descomposición de las relaciones globales en fragmentos se puede realizar de dos formas, mediante la fragmentación horizontal o mediante la fragmentación vertical. En todos los tipos de fragmentación un fragmento puede definirse a través de una expresión en un lenguaje relacional el cual toma a las relaciones globales como operandos y produce fragmentos como resultado. Al momento de definir los fragmentos se deben de seguir las siguientes reglas:

- *Condición de totalidad*: Todos los datos de la relación global deben ser mapeados en fragmentos.
- *Condición de reconstrucción*: Siempre debe ser posible reconstruir cada relación global a partir de sus fragmentos.

Las estrategias de fragmentación de datos se dan a nivel de tablas y consisten en dividir una tabla en fragmentos lógicos.

Fragmentación horizontal

Consiste en particionar los renglones de una relación global en fragmentos o renglones y cada uno de ellos es almacenado en distintos nodos. Sin embargo, todos los renglones únicos tienen los mismos atributos o columnas.

Fragmentación vertical

Es la división de una relación en grupos(fragmentos) de atributos. Cada fragmento es almacenado en nodos diferentes y cada fragmento tienen columnas únicas.

Fragmentación mixta

Los fragmentos que se obtienen de la fragmentación horizontal o vertical son relaciones en si mismas, por lo tanto es posible aplicar operaciones de fragmentación recursivamente. La reconstrucción se puede obtener aplicando las reglas de reconstrucción en orden inverso.

2.2.6 Ubicación de los datos

La ubicación de los datos describe el proceso de decidir donde se localizarán los datos. Las estrategias de ubicación de los datos son:

- *Centralizada*: La base de datos completa es almacenada en un solo lugar.
- *Particionada*: La base de datos es dividida en varios fragmentos y almacenados en varios lugares.
- *Repetida*: Las copias de uno o más fragmentos de la base de datos son almacenados en varios lugares.

Los algoritmos de ubicación de datos toman en cuenta los siguientes factores:

- Rendimiento y disponibilidad de datos
- Tamaño, número de renglones, y relaciones que existen en una entidad.
- Tipos de transacciones que serán aplicadas a la base de datos, los atributos que serán accedidos por esas transacciones.

2.2.7 Recuperación

El proceso de recuperación de una base de datos distribuida debe darse a través del elemento llamado administrador de recuperación, el cual se vale del archivo de registro para regresar a la base de datos a un estado anterior consistente, además cada nodo es encargado de mantener su propio archivo de registro en forma local.

Las fallas que se pueden presentar en una base de datos distribuida se pueden agrupar de la siguiente forma:

- *Fallas locales de las transacciones*. Una falla de tipo local en una transacción puede darse de varias formas, entre ellas errores en la aplicación o conflicto con otras transaccions. Cuando ocurre un error en la transacción se debe de repetir.
- *Fallas del nodo*. Las fallas de un nodo pueden surgir como consecuencia de fallas en el CPU local o por fallas en el suministro de energía, ocasionando que el sistema deje de funcionar.
- *Fallas en el medio*. Se refieren a las fallas que pueden ocurrir en los medios o dispositivos que alojan físicamente a la base de datos y que, generalmente, originan que una porción de ésta se pierda. Existen dos técnicas básicas basadas en la redundancia para prevenir este tipo de fallas y son el almacenamiento de respaldos y la creación de archivos espejo.
- *Fallas en la red*. Una falla puede dividir a una red en subredes, permitiendo la comunicación entre los nodos pertenecientes a una misma subred pero aislándola de las demás subredes.

2.2.8 Integridad

En el caso de las bases de datos distribuidas homogéneas generalmente no existe problema ya que tanto las reglas de integridad implícitas como las explícitas pueden definirse al momento del diseño de la base de datos y pueden ser integradas de forma consistente a los subsistemas de integridad locales de cada DBMS, sin embargo, la división entre los niveles globales y locales de las bases de datos distribuidas heterogéneas hace mucho más difícil el soporte consistente para las reglas de integridad. Este problema puede dividirse en 3 grupos principales:

- Inconsistencias en las reglas de integridad locales. Las reglas de integridad al nivel local deben ser especificadas por el DBMS local y la autonomía de ese nodo juega un papel muy importante dado que se encuentra en un ambiente heterogéneo. Desde el punto de vista del DBMS local no existe ningún problema en manejar diferentes reglas, sin embargo, desde el punto de vista global, esto puede ocasionar una gran confusión.
- Dificultad para especificar las reglas de integridad globales. Las reglas de integridad globales se almacenan en los catálogos globales del sistema y se aplican a las transacciones que se realizan a nivel global. Los catálogos del sistema son aquellos que almacenan la descripción de los datos.
- Inconsistencias entre las reglas locales y globales. Si los dos tipos de integridad son permitidos se pueden asignar prioridades, en un sistema totalmente distribuido con gran autonomía de los nodos, las reglas locales deben tener prioridad sobre las globales.

2.2.9 Seguridad

En una base de datos con autonomía nodal, la seguridad de los datos es responsabilidad del DBMS local, sin embargo, una vez que un usuario remoto ha sido permitido para acceder a los datos, el nodo local no tiene manera para vigilar la seguridad de los mismos, dado que esto implica que los datos sean transportados a través de la red.

Existen cuatro aspectos que deben considerarse por parte del DBMS local para vigilar la seguridad de los datos que están bajo su responsabilidad:

- Identificación y autenticación. Cuando un usuario desea acceder a cualquier sistema de cómputo debe, en primera instancia, identificarse a través de un nombre de usuario o "*login*", y después debe autentificar esta identificación a través de una clave privada o "*password*".
- Distribución de las reglas de autorización. Dado que los datos son distribuidos, también las reglas de autorización para el acceso a los datos tienen que ser distribuidas de tal forma que la regla esté en donde está el objeto al cual hace referencia
- Encripción. Aunque es común encriptar los *passwords*, también es posible encriptar datos y mensajes. En la encripción, los datos o mensajes en texto plano son sometidos a un algoritmo de encripción, el cual transforma ese texto plano en texto cifrado usando una llave de encripción. A menos que se conozca la llave de encripción es virtualmente imposible descifrar el texto encriptado.

2.3 BASES DE DATOS ORIENTADAS A OBJETOS

A finales de los 80 y principios de los 90 los expertos en bases de datos se enfrentaron a la problemática de manejar datos cada vez más complejos que difícilmente podrían haber sido procesados mediante la tecnología relacional estándar. El tamaño de las bases de datos y los cambios en su composición dieron lugar a la reorganización de los sistemas de información existentes. Esta reorganización dio lugar a tecnologías de bases de datos basadas en los conceptos orientados a objetos.

La clave de la programación orientada o objetos es considerar a un programa como si estuviera compuesto de objetos independientes, agrupados en clases, los cuales se comunican con otros objetos por medio de mensajes.

2.3.1 Conceptos Básicos

Objeto

Es una representación abstracta de una entidad del mundo real que tiene una identidad única, propiedades inherentes y la capacidad de interactuar con otros objetos

Identidad de un objeto

La parte más relevante de un objeto es su identidad. La identidad del objeto está representada por un identificador (OID), el cual es único para ese objeto, dos objetos no pueden compartir el mismo OID. El OID es asignado por el sistema en el momento de la creación del objeto y no puede ser cambiado bajo ninguna circunstancia, no depende de los valores de los atributos del objeto. El OID puede ser borrado únicamente si el objeto es borrado, y el mismo OID no puede volver a usarse.

Atributos

Los objetos son descritos por sus atributos, los cuales se convierten en variables de instancia en un ambiente orientado a objetos. Cada atributo tiene un nombre único y un tipo de datos asociado a él. Los atributos del objeto pueden hacer referencia a uno o más objetos.

Método

Cada operación realizada sobre un objeto debe ser implantada por un método. Los métodos son usados para cambiar los valores de los atributos de los objetos, o bien para mostrar el valor de los atributos del objeto seleccionado. Los métodos representan acciones del mundo real, es decir, los métodos son el equivalente a los procedimientos en los lenguajes de programación estructurados. En términos del enfoque orientado a objetos, los métodos representan el comportamiento de los objetos. Cada método es identificado por un nombre y tiene una estructura, la cual está compuesta de instrucciones escritas en algún lenguaje de programación.

Mensaje

Para invocar un método se envía un mensaje al objeto. Un mensaje es enviado especificando un objeto receptor, el nombre del método y algún otro parámetro requerido. La estructura interna del objeto no puede ser accedida directamente por el objeto que envía el mensaje. Denegar el acceso a la estructura interna asegura la integridad del estado del objeto y oculta los detalles internos del mismo.

Clases

Los sistemas orientados a objetos clasifican a los objetos de acuerdo a sus similitudes y diferencias. Los objetos que tienen características comunes se encuentran agrupados en clases, es decir, una clase es una colección de objetos similares que comparten una misma estructura y comportamiento (atributos y métodos). Una clase contiene la descripción de la estructura de datos y los detalles de implantación de los métodos para objetos en esa clase, además todos los objetos de una clase responden a los mismos mensajes.

Herencia

La herencia es la capacidad innata de un objeto dentro de la jerarquía para heredar la estructura de datos y comportamiento de las clases jerárquicamente superiores. Existen dos variantes de la herencia, la herencia simple y la herencia múltiple.

Herencia simple - Existe cuando una clase tiene únicamente una superclase inmediata sobre ella

Herencia múltiple.- Una clase puede derivarse de varias superclases padres localizadas un nivel arriba de esa clase.

Sobrecarga de métodos

Se sobrecarga un método de una superclase cuando se redefine ese método en una o varias de las subclases de esa superclase.

Polimorfismo

Permite que diferentes objetos respondan al mismo mensaje en diferentes formas. El polimorfismo es una característica muy importante de sistemas orientados a objetos porque su existencia permite a los objetos comportarse de acuerdo a sus características específicas En términos del enfoque orientado a objetos, el polimorfismo significa que:

- Se puede usar el mismo nombre para un método definido en diferentes clases.
- El usuario puede enviar el mismo mensaje a diferentes objetos pertenecientes a diferentes clases y siempre obtener la respuesta correcta.

Objetos complejos

Los objetos complejos se forman aplicando constructores a objetos más simples, por ejemplo, enteros, caracteres, cadenas de longitud variable, etc. De esta forma los atributos de un objeto pueden ser otros objetos.

Encapsulamiento

Encapsulación es la capacidad de ocultar los detalles internos de los objetos (atributos y métodos), es decir, la implantación del objeto, dejando "visible" únicamente la interfaz.

2.3.2 Modelo de datos orientado a objetos

El principal problema en la definición de un modelo de datos orientado a objetos es que no hay un modelo de datos orientado a objetos estándar. Algunas de las características que un modelo de datos orientado a objetos debe contener son:

- Debe soportar la representación de objetos complejos.
- Debe ser extensible, es decir, debe ser capaz de definir nuevos tipos de datos así como las operaciones que se realizarán sobre ellos.
- Debe soportar la encapsulación, esto es, la representación de los datos y la implantación de los métodos se deben ocultar a entidades externas.
- Debe permitir la herencia, un objeto debe ser capaz de heredar las propiedades a otros objetos.
- Debe soportar el concepto de identidad de objeto.
- Debe permitir la agrupación de objetos similares en clases y formar con éstas, jerarquías de clase.

2.3.3 Sistema Manejador de Bases de Datos Orientadas a Objetos

Un sistema manejador de bases de datos orientadas a objetos debe ser capaz de combinar las características propias de cualquier DBMS como control de concurrencia, recuperación, persistencia, seguridad, etc , con las características propias de un sistema orientado a objetos como son los conceptos, el modelo de datos y el manejo de interfaces gráficas orientadas a objetos.

Características

Existen 13 características obligatorias que debe cumplir un OODBMS para ser considerado como tal, de las cuales las 8 primeras caracterizan a un sistema orientado a objetos y las 5 restantes son características de un DBMS estándar. A continuación se citan estas características.

- El sistema debe soportar objetos complejos
- Debe soportar el concepto de Identidad de Objeto

- Los objetos deben ser encapsulados
- El sistema debe soportar clases
- El sistema debe soportar herencia
- Debe evitar el traducir los nombres de las operaciones en direcciones en tiempo de compilación.
- Debe ser computacionalmente completo, es decir, se debe poder expresar cualquier función computable usando el lenguaje de manipulación de datos.
- Debe ser extensible en cuanto a tipos de datos.
- Debe ser capaz de recordar la ubicación de los datos (persistencia)
- Debe ser capaz de administrar bases de datos muy grandes.
- Debe aceptar transacciones concurrentes (control de concurrencia).
- Debe ser capaz de recuperarse de las fallas de hardware y software.
- Las consultas deben ser simples

Ventajas de un OODBMS

- Permite que la información almacenada en la base de datos sea una representación más apegada a la realidad.
- El soporte para objetos complejos lo hace indispensable para áreas de aplicación especializadas como CAD/CAM y ambientes multimedia.
- Permite la extensibilidad de tipos de datos incrementando la funcionalidad de la base de datos y sus capacidades de modelado.
- Hacen uso de los avances tecnológicos de las computadoras como la rapidez del CPU y la gran capacidad de memoria lo que se refleja en un mejor rendimiento comparado con los sistemas relacionales.
- La facultad para volver a usar las clases creadas con anterioridad permite el rápido desarrollo y fácil mantenimiento de la base de datos y sus aplicaciones.
- Es una posible solución al problema de integración de las bases de datos existentes y las futuras en un solo ambiente.

Desventajas de un OODBMS

- Está basado en una tecnología que continúa en etapa de crecimiento, además carece de la madurez que define un ambiente de usuario final estable.
- Al ser ésta una tecnología en desarrollo existe una carencia de estándares.
- De acuerdo con los usuarios de los DBMS relacionales, carece de fundamentos teóricos.
- Requiere de programación orientada a objetos.
- No existe variedad en las herramientas de reporte y consulta.
- El control de concurrencia y la administración de transacciones es limitado.

2.3.4 Persistencia

Los lenguajes de programación convencionales se concentran en proporcionar facilidades para la definición y el manejo de estructuras de datos cuya existencia no se extiende más allá de la vida del programa en el cual son creadas. Si se requiere que los datos sobrevivan a la ejecución del programa, entonces tendrán que estar en un archivo, en la mayoría de los casos, externo al lenguaje y sus características. Los datos pueden ser de dos tipos dependiendo de su relación con el programa de aplicación:

- Datos volátiles, los cuales son creados y manipulados por las facilidades que brinda el lenguaje de programación, y cuya existencia termina cuando termina el programa.
- Datos persistentes, los cuales residen en dispositivos de almacenamiento permanentes y deben ser transferidos al espacio de memoria volátil del programa antes de poder ser accedidos o manipulados.

Una base de datos orientada a objetos es una colección de objetos persistentes, cada uno con un identificador único el cual puede ser visto como un apuntador al objeto persistente. Los objetos persistentes

deben ser almacenados en dispositivos de almacenamiento no volátil y deben existir más allá del tiempo de vida del programa que los creó.

Cuando se incorpora la persistencia a un lenguaje de programación orientado a objetos se deben tener en cuenta los siguientes principios:

- La persistencia es una propiedad de las instancias y no de los tipos de objeto.
- Los objetos de cualquier tipo, incluyendo los complejos definidos por el usuario, puede ser alojado en cualquiera de los almacenamientos (volátil o no volátil).
- Los objetos persistentes y los volátiles deben ser accedidos y manipulados en, exactamente, la misma forma.

2.3.5 Concurrencia

Debemos recordar que el enfoque orientado a objetos pretende modelar con mayor fidelidad el mundo real. Dado que en el mundo real muchas actividades son inherentemente concurrentes, los lenguajes que soportan la concurrencia son más expresivos.

Los lenguajes de programación orientados a objetos concurrentes típicamente modelan el mundo real con objetos ejecutables en forma concurrente llamados procesos. Un proceso tiene una interfaz de operaciones ejecutables y uno o más hilos de control, los cuales pueden ser activados o suspendidos en cualquier momento. Un hilo puede ser encolado hasta que un proceso está listo para ejecutarlo y puede ser suspendido y reactivado de acuerdo a las condiciones que prevalezcan en el programa.

En una base de datos orientada a objetos la ejecución concurrente de transacciones que realizan operaciones en los objetos de la base de datos es correcta si equivale a la ejecución serial de las transacciones en las bases de datos relacionales Las transacciones pueden entonces ser vistas como módulos temporales que representan unidades indivisibles de ejecución.

La principal diferencia entre el control de concurrencia basado en objetos y el que se da en bases de datos convencionales es que las unidades de concurrencia pueden ser muy largas. Esto permite considerar varios esquemas de interacción para las transacciones que corren sobre la base de datos. Algunas posibles estrategias son:

- Una estrategia "pre-demanda", en la cual las transacciones deben "adquirir" todos los objetos que necesita antes de que se les permita realizar una acción sobre la base de datos. Los objetos pueden adquirirse ya sea en modo compartido o exclusivo y son liberados sólo cuando la transacción termina. Una transacción termina en forma anormal cuando alguno de los objetos que requiere está bloqueado por otra transacción.
- Una estrategia optimista en la cual cada transacción maneja una copia "sombra" de la base de datos en su propia área de trabajo y entonces envía una petición para hacer sus actualizaciones. El sistema es entonces el encargado de aceptar o rechazar la petición dependiendo de si entra o no en conflicto con otras transacciones.

2.3.6 Recuperación

La recuperación en las bases de datos orientadas a objetos se da en la misma forma en que se da para las bases de datos relacionales centralizadas y distribuidas De esta forma, dado que las posibles causas para que la base de datos quede en un estado de inconsistencia son las mismas, las técnicas de recuperación que se siguen son también las mismas, estas técnicas consisten en el manejo, ya sea de copias de respaldo o de un archivo de registro el cual debe almacenar también la dirección de cada objeto actualizado o creado por la transacción junto con su valor antiguo y su valor nuevo.

Los algoritmos que se mencionaron como parte de la recuperación en bases de datos relacionales: rehacer/deshacer, no-rehacer/deshacer, rehacer/no-deshacer y no-rehacer/no-deshacer tienen el mismo comportamiento en las bases de datos orientadas a objetos y tienen la misma funcionalidad para la recuperación de este tipo de bases de datos.

Se puede hacer mención especial de la forma en que funciona la recuperación para la estrategia optimista que se menciono en el punto anterior. Dado que en el control de concurrencia optimista cada transacción mantiene una copia sombra de la tabla de objetos compartidos, todas las actualizaciones son invisibles a otras transacciones concurrentes. Cuando una transacción se lleva a cabo con éxito, la tabla compartida simplemente es reemplazada por la copia sombra de esa transacción. Entonces, si una transacción termina antes de tener éxito, su copia sombra simplemente es desechada y la base de datos no es afectada.

CAPÍTULO III

ALMACENES Y MINEROS DE DATOS

3.1. ALMACENES DE DATOS

En forma simple, un almacén de datos (*data warehouse*) es la administración de datos situados después y fuera de los sistemas operacionales. Mas adelante se identificarán atributos claves y la definición apropiada de *data warehouse*. A continuación se muestra el uso que históricamente han tenido los datos y los sistemas de análisis y algunos de los factores que influyeron en la evolución de los *data warehouses*.

3.1.1 Historia

A través de la historia, siempre se ha puesto énfasis en los sistemas operacionales y en los datos que estos sistemas procesan, de tal forma que no se permitió que los sistemas de análisis interfirieran con los sistemas operacionales, degradando su rendimiento, además de que se crearon estructuras fuera de las aplicaciones operativas para almacenar los datos que éstas ya habían procesado.

En los años 70 prácticamente todo el desarrollo de sistemas para negocios se hizo sobre mainframes IBM usando herramientas como Cobol, DB2, etc., para la década de los 80 surgieron plataformas de minicomputadoras como la AS/400 y la VAX/VMS. A finales de los 80 y principios de los 90 surgió Unix como la plataforma más popular para servidores debido a la introducción de la arquitectura cliente/servidor. Sin embargo, se calcula que el 70% de los datos de las grandes corporaciones todavía se encuentra en el ambiente de mainframes, la razón más importante para ello, y que es particularmente relevante para el tema del almacenamiento de datos (*data warehousing*), es que durante años estos sistemas han capturado el conocimiento de esas corporaciones, y es difícil llevar este conocimiento a nuevas plataformas o aplicaciones.

Durante la década pasada, la popularidad de la computadora personal trajo consigo nuevas opciones y oportunidades para los sistemas de análisis, la brecha entre los programadores y los usuarios finales se comenzó a cerrar dado que estos últimos contaban ya con herramientas como las hojas de cálculo que les permitían hacer análisis y representaciones gráficas además de trabajar con datos extraídos de los enormes sistemas descritos anteriormente. Sin embargo, esto provocó que el análisis atendiera a necesidades muy específicas dado que cada usuario obtenía únicamente la información que requería.

Otra categoría importante dentro de los sistemas de análisis fue la de los sistemas de apoyo a la toma de decisiones y los sistemas de información ejecutiva, los primeros, enfocados a los administradores de bajo y medio nivel y los segundos, a los ejecutivos de más alto nivel, tienen en común las siguientes características:

- Son los precursores más cercanos de los sistemas de *data warehousing*.
- Presentan resultados en términos descriptivos estándar entendibles por usuarios no técnicos.
- Se basan en vistas que pueden manejar información tan detallada como se desee, sin embargo, raramente son capaces de manejar la información detallada de todas las vistas al mismo tiempo.

Muchos factores han influenciado la rápida evolución de la disciplina del *data warehousing*, el marcado decremento en los precios y el incremento en el potencial del hardware aunado a la facilidad para usar el software de hoy en día han hecho posible el rápido análisis de cientos de gigabytes de información.

El desarrollo más importante de la computación desde la computadora personal es la Internet y las aplicaciones basadas en la Web. Uno de los campos más excitantes en la industria de la computación es el desarrollo de Intranets, que son redes privadas basadas en los estándares de Internet. La tendencia Internet/Intranet ha tenido implicaciones muy importantes para las aplicaciones de *data warehousing*, primero, pueden estar disponibles en todo el mundo a un bajo costo minimizando la necesidad de tener réplicas de la información en diversas regiones geográficas, segundo, este estándar ha permitido que el servidor de Web funcione como intermediario y que el análisis de los datos tenga lugar antes de que los resultados sean presentados en el browser del usuario.

Otra influencia significativa en la evolución del *data warehousing* son los cambios fundamentales en la organización y estructura de los negocios a finales de los 80 y principios de los 90 El surgimiento de una

economía global ha cambiado profundamente la demanda de información por parte de las corporaciones en el mundo entero. Fenómenos como la "reingeniería de procesos" han forzado a las empresas a reevaluar su forma de hacer negocios.

La globalización ha traído consigo no solamente la necesidad de hacer análisis continuos sino también la de administrar los datos en un almacén central cada vez más complicado.

Habiendo revisado el uso que históricamente han tenido los datos y los sistemas de análisis y algunos de los factores que influenciaron la evolución de los sistemas de *data warehouse*, pasemos a identificar sus principales atributos.

3.1.2 Atributos y conceptos

Cabe mencionar que el *data warehousing* es una tecnología en proceso de evolución y como tal, se debe tener mucho cuidado en el manejo de ciertos conceptos, sobre todo, en el que hacen los vendedores para distinguir a sus sistemas de los de la competencia, por ejemplo, el tamaño no determina si un *data warehouse* lo es realmente o no, sino la relevancia que éste tenga para la empresa.

El concepto primario del *data warehousing* es que los datos almacenados para su análisis pueden ser accedidos con mayor eficiencia si se separan de los datos de los sistemas operacionales. Históricamente esto respondía al hecho de que estos datos se almacenaban en cintas y los sistemas de análisis corrían sobre éstas cintas para minimizar el impacto en el rendimiento de los sistemas operacionales, estas razones no han cambiado mucho con la evolución de los sistemas de *data warehousing* excepto que ahora son consideradas formalmente en la construcción de cualquiera de ellos.

Los sistemas de *data warehousing* son más completos cuando los datos provienen de más de un sistema operacional, naturalmente esta integración se lleva a cabo fuera de las aplicaciones de origen. La principal razón para combinar datos de varias aplicaciones fuente es la habilidad para tener referencias cruzadas sobre éstas por ejemplo, el tiempo, en un *data warehouse* típico los datos se ordenan en base al tiempo, al generar reportes para un periodo de tiempo, éste se convierte en el principal atributo de referencias cruzadas sobre las aplicaciones originales. La habilidad para establecer y entender la correlación entre actividades de diferentes grupos de la misma organización es citada como una de las mejores y más avanzadas características de los sistemas de *data warehousing*.

Transformación lógica de los datos operacionales.

Los datos deben ser transformados lógicamente cuando son llevados de los sistemas operacionales a los *data warehouses*, esto requiere considerables esfuerzos de análisis y diseño y se debe a que existen conceptos fundamentales de la teoría de bases de datos relacionales que no pueden ser aplicados a los sistemas de *data warehousing*, aun y cuando generalmente la mayoría de los *data warehouses* son desarrollados sobre plataformas de bases de datos relacionales.

El proceso de modelado de datos necesita estructurar los datos en el *data warehouse* en forma independiente del modelo de datos relacional que existe en la mayoría de los sistemas operacionales, esto es debido a que algunos atributos que resultan esenciales para los sistemas operacionales, dado que contienen información sobre la referencia y relaciones de los datos, se tornan innecesarios para el *data warehouse* y no pueden ni deben ser cargados y almacenados en éste

El modelo del *data warehouse* debe ser extensible y estructurado de tal forma que en cualquier momento puedan ser agregados datos de distintas fuentes, de hecho, en muchos casos, un proyecto de *data warehousing* no incluye los datos de todas las aplicaciones desde el inicio, pueden elegirse algunas y poco a poco ir agregando los datos provenientes de las demás.

El modelo lógico de un *data warehouse* se parece o trata de alinearse mas a la estructura de la empresa que al modelo particular de cualquier aplicación. Las entidades definidas y mantenidas en el *data warehouse* son paralelas a entidades de la organización como clientes, productos, distribuidores, etc.

Una transformación importante que deben sufrir a nivel lógico los datos antes de ser almacenados en un *data warehouse* es la desnormalización. El proceso de normalización generalmente descompone una

tabla en muchas tablas independientes lo cual da por resultado flexibilidad para hacer consultas en bases de datos relacionales, sin embargo, un modelo de datos totalmente normalizado puede resultar también muy ineficiente al manejar la cantidad de datos que almacena un *data warehouse*, debido a la cantidad de uniones y relaciones que existen entre las tablas.

Otra razón importante para la desnormalización es que los sistemas operacionales mantienen relaciones dinámicas entre tablas mientras que el *data warehouse* contempla exclusivamente relaciones estáticas.

Transformación física de los datos operacionales

Este proceso homogeneiza los datos para el *data warehouse* y generalmente se conoce como "*data scrubbing*" o "*data staging*". Contempla los siguientes aspectos:

Los nombres y términos usados en los sistemas operacionales deben ser transformados a términos estándar uniformes para la organización.

Cuando un atributo es definido físicamente por el *data warehouse* se deben usar el tipo de datos y la longitud estándar de dato que corresponden a ese atributo en cualquier lugar en que este sea utilizado, esto con el objeto de no permitir mas de una definición física para un mismo atributo, además se deben usar en forma consistente los valores predefinidos en el *data warehouse* para un atributo, independientemente del valor predefinido que este tenia en su aplicación de origen

Una recomendación que se hace durante el proceso de la transformación física de los datos es definir valores por *default* que tendrán los datos cuya transformación no pueda ser llevada a cabo apropiadamente. Estos valores por *default* deben tener un significado de tal forma que el *data warehouse* pueda interpretarlos correctamente.

Una característica muy importante de los *data warehouses* es el manejo de vistas resumidas que son tablas que se crean independientemente del momento en que serán consultadas por un usuario y permiten hacer análisis iniciales, obtener ganancias considerables en cuanto al desempeño del sistema y manejar varias vistas con el mismo detalle. Estas vistas resumidas, en general, son creadas alrededor de entidades como clientes, productos, etc. y ocultan la complejidad de manejar datos muy detallados.

Definición

Un *data warehouse* es un ambiente extensible estructurado diseñado para el análisis de datos no volátiles (estáticos), lógica y físicamente transformados de los formatos que guardaban en sus aplicaciones de origen y alineados con la estructura de la organización, actualizados y mantenidos por largos periodos de tiempo, expresados en términos simples y resumidos para un análisis más rápido.

3.1.3 Diferencia entre *data warehouse* y bases de datos operacionales

Tal vez uno de los conceptos más importantes que ha traído el *data warehousing* es el reconocimiento de dos tipos diferentes de sistemas de información: los sistemas operacionales y los sistemas informativos.

Los sistemas operacionales son aquellos que, como su nombre lo indica, ayudan a que la empresa opere día a día. En la empresa existen sistemas principales por ejemplo: sistema para inventario, sistema para manufactura y sistemas para contabilidad Debido a la importancia de los sistemas operacionales en la organización, casi siempre son los primeros en ser computarizados. A través de los años, estos sistemas operacionales se han extendido, manejado y mantenido hasta el punto en que han sido integrados completamente en la organización. La mayor parte de las organizaciones ya no pueden operar sin sus sistemas operacionales y de los datos que estos sistemas mantienen.

Por otro lado, hay otras funciones dentro de la empresa que tienen que ver con la planeación, pronóstico y administración de la organización; estas funciones son criticas para la sobrevivencia de la organización. Funciones como "planeación de mercado", "planeación de ingeniería" y "análisis financiero" también requieren de sistemas de información para soportarlo. Pero estas funciones son diferentes de las

operacionales, y los tipos de sistemas y la información requerida también son diferentes. Las funciones basadas en el conocimiento son sistemas informativos.
Los sistemas informativos tienen que ver con el análisis de datos y la toma de decisiones, frecuentemente las decisiones acerca de como operara la empresa, ahora y en el futuro. Los datos operacionales normalmente están enfocados a una sola área, los datos informativos necesitan cubrir diferentes áreas y necesitan gran cantidad de datos relacionados operacionales.

Un *data warehouse* es diferente de las bases de datos operacionales que soportan las aplicaciones de "Procesamiento de Transacción en Línea" (OLTP, On-Line-Transaction-Processing). El *data warehouse*:

- *Está orientado a la facilidad de consulta.*- Mientras que la aplicación operativa está orientada al "performance" operacional: 2 segundos por transacción, el *data warehouse* está orientado a la facilidad de consulta: 1 minuto, 3 horas o más por consulta, pero que sea fácil definirla.

- *Contiene datos temporizados.*- Los datos de la aplicación operativa son detallados, comprenden pocos períodos y generalmente no tienen una temporización, los datos del *data warehouse* también son detallados, comprenden muchos períodos y siempre son temporizados.

- *Administra grandes cantidades de información.*- La mayoría de los *data warehouses* contiene información histórica que se retira con frecuencia de las aplicaciones operativas porque ya no es necesaria para éstas, también debe ofrecer opciones para la adición y condensación que clasifican esta gran cantidad de datos. Por la necesidad de administrar toda la información histórica y además los datos actuales, un *data warehouse* es mucho mayor que las bases de datos operacionales.

- *Guarda información en diversos medios de almacenamiento.*- Por los volúmenes de información que deben manejarse, un *data warehouse* frecuentemente guarda información en diferentes medios de almacenamiento.

- *Comprende múltiples versiones de un esquema de base de datos.*- Debido a que el *data warehouse* tiene que guardar información histórica y administrarla, y cómo la información histórica ha sido manejada en distintos momentos por diferentes versiones de esquemas de bases de datos, en ocasiones el *data warehouse* tiene que controlar información originada en organizaciones de bases de datos diferentes.

- *Condensa y agrega información.*- Con frecuencia, es muy alto el nivel de detalle de la información guardada por bases de datos operacionales para cualquier toma de decisiones sensata Un *data warehouse* condensa y agrega información para presentarla en forma comprensible a las personas.

- *Integra y asocia información de muchas fuentes.*- Debido a que las organizaciones han administrado sus operaciones utilizando múltiples bases de datos y numerosas aplicaciones de software, se requiere de *data warehousing* para recopilar y organizar en un solo lugar la información que estas aplicaciones han acumulado al paso de los años.

A continuación se presenta una tabla comparativa entre el modelo de datos del *data warehouse* y una base de datos operativa.

Modelo de datos de las bases de datos operacionales	Modelo de datos del *Data warehouse*
• El modelo de datos se concentra en eliminar la redundancia, coordinar las actualizaciones y soportar las transacciones que realizan el mismo tipo de operaciones muchas veces al día sobre los mismos datos	• El modelo de datos se orienta a manejar un amplio rango de consultas y recuperación de información en forma periódica. El usuario final realiza muy pocas actualizaciones.
• Contiene muchas normas para manejar actualizaciones consistentes y mantener la integridad referencial.	• Tiene muy pocas normas, para proporcionar acceso instantáneo sin necesidad de tener que realizar una gran cantidad de uniones.
• El modelo de datos es solo el producto de la base de datos y de los desarrolladores de aplicaciones. A menudo es complejo y grande	• Los usuarios finales deben entender el modelo de datos, ya que la finalidad principal del *data warehouse* es que la visibilidad y el acceso a los datos sea fácil y conveniente.
• Los datos son necesarios para las operaciones actuales. Los datos innecesarios se archivan para evitar una degradación en el desempeño de los sistemas operacionales.	• Los datos son tanto operacionales como históricos. Los volúmenes de datos que contiene el *data warehouse* se ubican entre los cientos de megabytes a los cientos de gigabytes.
• Almacenan muy pocos datos derivados. La mayoría de las aplicaciones que operan con este tipo de bases de datos derivan los datos, siempre que se requiera	• Almacena grandes cantidades de datos derivados para ahorrar el esfuerzo del cálculo repetitivo de las derivaciones. Esto se debe a que las derivaciones se hacen no sólo con los datos operacionales, sino también con los históricos.
• Estas bases de datos contienen todos los datos que requiere una empresa para sustentar sus operaciones	• Un *data warehouse* sólo contiene datos que tienen valor a través del tiempo.
• Los datos operacionales están ligeramente resumidos, generalmente en forma de reporte.	• Un *data warehouse* precalcula y almacena datos muy resumidos.

3.1.4 Arquitectura de referencia

Una arquitectura de referencia es un diagrama que separa los distintos componentes de una solución basada en un *data warehouse*. Este esquema permite integrar en una clasificación común los distintos tipos de información necesarios para construir un *data warehouse*, esta clasificación permite comparar las opciones de distribuidores competidores, evaluar ventajas y desventajas y descubrir deficiencias que deben corregirse mediante el desarrollo de software interno. La palabra referencia significa que la arquitectura es independiente de un distribuidor y que caracteriza la naturaleza genérica de todos los *data warehouses*

La arquitectura de un *data warehouse* es una forma de representar una estructura completa de datos, comunicación, procesamiento y presentación que existe para el usuario final. La arquitectura se conforma de las siguientes partes interrelacionadas:

- *Base de datos operacional/Capa de base de datos externa.* La meta del *data warehouse* es liberar la información que esta almacenada en las bases de datos operacionales y mezclarla con otras fuentes de datos, frecuentemente externas. Las organizaciones están adquiriendo información adicional de bases de datos externas. Esta información incluye aspectos demográficos, económicos y competitivos entre otros.

- *Capa de acceso a la información.* Esta capa trata directamente con el usuario final. En particular representa las herramientas que el usuario final usa diariamente: Excel, Lotus1-2-3, Access, SAS, etc. Esta capa también incluye el hardware y software involucrado en el despliegue e impresión de reportes, hojas de calculo, gráficas y diagramas para análisis y presentación.

- *Capa de acceso de datos.* Esta capa permite que exista una comunicación entre la capa de Acceso a la Información y la capa de Operación. Actualmente el lenguaje común de datos es SQL. Uno de los avances en los últimos años ha sido el desarrollo de una serie de "filtros" de accesos a los datos tales como EDA/SQL que hace posible que SQL acceda a todos los DBMSs y a los sistemas de archivos de datos, relacionales y no relacionales. Estos filtros hacen posible que las herramientas de Acceso a la Información accedan a los datos almacenados en los sistemas de administración de base de datos que tienen 20 años de haber sido creados La capa de Acceso de Datos no solo cubre los diferentes *DBMSs y sistemas de archivos en el mismo hardware, sino también fabricantes y protocolos de red.*

- *Capa de Directorio de Datos (Metadatos)* Para proporcionar el acceso a los datos es necesario contar con un directorio de datos o almacén de metadatos Para tener un almacén funcional y completo es necesario tener una variedad de metadatdos disponible, datos acerca de las vistas del usuario final y datos acerca de las bases d4e datos operacionales. Idealmente, los usuarios finales deben ser capaces de acceder a los datos del data warehouse (o desde las bases de datos operacionales) sin tener que saber donde residen los datos o la forma en la cual han sido almacenados.

- *Capa de administración de procesos.* Esta capa esta involucrada con la programación de varias tareas que deben ser llevadas a cabo para construir y mantener la información del data warehouse y del directorio de datos. Esta capa puede ser vista como el programador o el control de trabajo de alto nivel para muchos procesos (procedimientos) que deben ocurrir para que el *data warehouse* se encuentre funcionando.

- *Capa de Mensajes de Aplicación.* Esta capa tiene que ver con el transporte de la información a través de la red de la empresa. Los mensajes de aplicación también se conocen como "*middleware*", sin embargo no solo involucran protocolos de red Estos mensajes de aplicación pueden ser utilizados por ejemplo para aislar aplicaciones, operacionales o informativas, desde el formato exacto de los datos en cualquier área. También pueden ser usados para coleccionar transacciones o mensajes y distribuirlos a cierto lugar en cierto tiempo.

- *Capa de data warehouse (fisica).* El núcleo del *Data warehouse* es donde se encuentran los datos actuales que se utilizan principalmente para fines informativos. En esta capa las copias de los datos operacionales y/o externos son almacenados en una forma accesible y altamente flexible.

- *Capa de montaje*. Esta capa incluye todos los procesos necesarios para seleccionar, editar, resumir, combinar y cargar el *data warehouse* y los datos de acceso a la información desde las bases de datos operacionales y/o externas. Con frecuencia involucra la programación compleja, incrementando las herramientas de *data warehousing* que están siendo creadas para ayudar en este proceso. Esta capa también puede involucrar los programas de análisis de calidad de datos y filtros que identifican patrones y estructuras de datos dentro de los datos operacionales existentes

3.1.5 Construcción de un *data warehouse*

Al igual que todos los sistemas de información, el desarrollo de un *data warehouse* sigue un ciclo cuyas fases se explican a continuación:

Planeación

La fase de planeación consta de los siguientes pasos, algunos de los cuales pueden llevarse a cabo en paralelo para reducir la duración de esta fase:

- Selección de la estrategia de implantación: Esta decisión tiene mucho que ver con la organización y se basa en como se llevan a cabo las tareas dentro de ésta. Estas son algunas estrategias que han demostrado su popularidad:

  - *Enfoque de arriba hacia abajo*.- En este enfoque se identifican primero los requerimientos empresariales que debe cubrir el *data warehouse* y éstos, son los principales conductores de la implantación. Se recomienda usar este enfoque cuando la empresa ya ha tenido experiencia anterior desarrollando aplicaciones basadas en este enfoque, cuando se puede prever un conjunto claro de objetivos para el *data warehouse*, cuando se tiene una idea clara de donde encaja el *data warehouse* en la estructura organizacional y/o cuando se tiene una idea clara de cómo la utilización del *data warehouse* es un subproceso de un proceso empresarial ya establecido.

  - *Enfoque de abajo hacia arriba*.- Generalmente comienza con experimentos y prototipos basados en la tecnología, se selecciona un subconjunto bien definido de la problemática empresarial y se formula una solución para este subconjunto. En general es más rápido y permite que una organización avance con un gasto considerablemente menor y que evalúe los beneficios de la tecnología antes de establecer compromisos significativos.

  - *Enfoque combinado*.- Puede explotar la naturaleza planeada y estratégica del enfoque de arriba hacia abajo al tiempo que conserva la rápida implantación y aplicación oportuna del enfoque de abajo hacia arriba. Este enfoque reúne las ventajas de los otros dos pero es más complejo de manejar como proyecto. Este enfoque se adapta mejor a la tecnología del *data warehouse*.

- Selección de la metodología de desarrollo: En teoría, un *data warehouse* puede desarrollarse por medio de cualquier metodología de desarrollo de software, sin embargo, en la realidad, se descarta el uso de cualquier tecnología que requiera fases de acopio de requerimientos, análisis, desarrollo y despliegue muy largas. A continuación se muestran dos metodologías que pueden ser usadas para este fin:

  - *Método de análisis y diseño estructurado (en cascada)*.- Una vez que se tienen los requerimientos se analizan y subdividen en forma progresiva, construyéndose después un diseño basado en los resultados del análisis El diseño se divide en niveles más concretos hasta que aparece el código del sistema. Este método se dedica a la construcción de sistemas que satisfagan requerimientos concisos y específicos.

- *Método de desarrollo espiral.*- Aquí los requerimientos no se pueden identificar con claridad al inicio. El método espiral es partidario de la rápida generación de sistemas cada vez más funcionales con intervalos cortos entre versiones sucesivas. El intervalo entre versiones se emplea para identificar e implantar funcionalidad adicional Dado que el desarrollo de un *data warehouse* presenta estas características, ésta se constituye en una buena elección de metodología de desarrollo.

- *Selección de un ámbito inicial de implantación:* Después de definir un rumbo general y un conjunto general de objetivos para el *data warehouse*, es necesario derivar con rapidez un ámbito limitado para la primera implantación, para lo cuál pueden formularse preguntas como las siguientes: ¿Cuáles son los departamentos que necesitan utilizar inicialmente el *data warehouse*?, ¿Cuál es el tamaño de los datos dentro del *data warehouse*?, ¿Cuáles y cuántas son las funciones de entrada de datos?, ¿Qué tan utilizables son los datos de las fuentes?, ¿Qué tan bien documentadas están las fuentes de datos?, etc.

- *Desarrollo de un programa y un presupuesto para planes del proyecto*: Comprende los siguientes puntos:
  - Articular tanto un plan de programa como un conjunto de planes de proyecto. Un plan de programa es una visión general de la actividad del *data warehouse* y su función en la vida diaria de la organización.
  - Reservar un presupuesto adecuado para el programa.
  - Proporcionar medidas para la estimación de la retribución del *data warehouse*.

- *Desarrollo de escenarios de uso empresarial:* Dado que el *data warehouse* lo utilizan personas distintas a los desarrolladores, es necesario que los usuarios finales se involucren en el establecimiento de expectativas de lo que puede ofrecer el *data warehouse*, estos escenarios son una importante herramienta del prototipo de requerimientos, los cuales generalmente se satisfacen mediante el uso de fuentes de datos apropiadas.

- *Recopilación de los metadatos.* Aquí se deben reunir los metadatos provenientes de todas las fuentes de datos utilizadas, tanto internas como externas.

Requerimientos

Consiste en una especificación precisa de las funciones que se obtendrán del *data warehouse*, además de describir con claridad el ambiente operativo en el que se entregará.

- *Requerimientos del propietario*· Algunas preguntas que proponen los propietarios del *data warehouse* son: ¿Por qué construir un *data warehouse*?, ¿Qué problema empresarial abordará?, ¿Cuáles son los objetivos empresariales?, ¿Cuánto costará?, ¿Cuándo estará listo?, ¿Cuáles son los riesgos?, ¿Se tiene capacidad para hacerlo?.
  Parte de los requerimientos empresariales son también especificaciones de ámbito para el *data warehouse*, en términos de:
  - *Areas tema.*- Son los temas de interés de diversas funciones empresariales. Una selección cuidadosa de las áreas tema contiene el ámbito de implantación del *data warehouse* y maximiza su utilidad
  - *Granularidad.*- Se refiere al nivel de detalle de la información requerida.
  - *Categorías.*- Un *data warehouse* organiza un gran conjunto de datos operacionales e históricos mediante múltiples categorías. Las siguientes categorías son de uso común en las consultas empresariales: tiempo, grupos de clientes, familia de productos, geografía y ubicación, estructura de la organización, etc

- *Requerimientos del arquitecto*: El arquitecto es la persona o personas responsables de diseñar los componentes del *data warehouse* para sustentar necesidades actuales y futuras. Los arquitectos deben compilar una serie de requerimientos que coincidan con la visión del propietario, así como un conjunto de requerimientos que refleje la implantación de la tecnología.

- *Requerimientos del desarrollador*: El desarrollador visualiza al *data warehouse* en forma concreta, es decir, requiere que las arquitecturas de datos de aplicación y de tecnología formuladas por el arquitecto se subdividan aún más en aplicaciones, interfaces, computadoras, bases de datos, comunicaciones y pantallas de interfaz de usuario específicas. También los requerimientos del desarrollador se relacionan con descripciones detalladas de elementos tales como el lenguaje de programación, el acceso al RDBMS y los protocolos de comunicación.

- *Requerimientos del usuario final*. Para el usuario final, el acceso al *data warehouse* se realiza a través de herramientas de consulta y de reportes dado que para él, el *data warehouse* es una gran caja negra. Sus requerimientos pueden ubicarse en una o más de las siguientes categorías:
  - Requerimientos de consulta
  - Requerimientos de reportes
  - Visualización de datos

Análisis

Esta fase consiste en convertir los requerimientos obtenidos en la fase anterior en especificaciones que puedan apoyar el diseño.

El proceso de análisis consiste en derivar modelos físicos y lógicos de datos para el *data warehouse* y los mercados de datos y definir los procesos necesarios para conectar las fuentes de datos, el *data warehouse*, los mercados de datos y las herramientas de acceso del usuario final

Diseño

En esta fase, los modelos lógicos desarrollados en la fase de análisis se convierten en modelos físicos, también se identifican y detallan los procesos que requiere cada bloque de la arquitectura de referencia del *data warehouse*.

Dentro de esta fase se realiza el diseño de las arquitecturas de datos y aplicación. En el diseño de la arquitectura de datos se lleva a cabo el desarrollo de modelos físicos de datos para las bases de datos de almacenamiento del *data warehouse*. También se verifica la correspondencia de los modelos físicos de datos de las fuentes de datos con el modelo físico del *data warehouse*, lo cual ayuda a los procesos de extracción y refinamiento a efectuar sus funciones dentro del mismo. El diseño de la arquitectura de aplicación comprende procesos que son internos a las fuentes de datos y se relacionan con depuraciones o extracciones parciales de información y procesos que conectan la fuente de datos con el *data warehouse*.

Construcción

Esta fase es responsable de implantar físicamente los diseños desarrollados en la etapa anterior, su construcción es similar a la construcción de un sistema de base de datos relacional grande, es decir, se necesitan construir aplicaciones que:

- Creen y modifiquen las bases de datos para el *data warehouse* y los mercados de datos
- Extraigan datos de fuentes relacionales y no relacionales
- Realicen transformaciones de datos como integración, resumen y adición
- Realicen actualizaciones
- Efectúen búsquedas en bases de datos muy grandes

Despliegue

Tiene que ver con los retos de instalación, puesta en servicio y uso de la solución de *data warehouse*. Como en cualquier otro sistema, algunas de las actividades requeridas en esta etapa son: instalación inicial

- Planeación y entrega de implantación por etapas
- Capacitación y orientación a todo tipo de usuarios
- Administración de usuarios y sistemas
- Capacidad de generar archivos permanentes y respaldos
- Capacidad de recuperación
- Controles de acceso y seguridad

Además, el sistema de *data warehouse* debe contemplar las siguientes actividades:

- El nivel de documentación de los metadatos en el sistema es a nivel técnico, sin embargo, los usuarios finales requieren ver definiciones de esta información en un lenguaje que comprendan.
- Se debe promover entre los usuarios la información contenida en el *data warehouse* para que éste se convierta en un sistema de misión crítica.

Expansión

Al comenzarse a utilizar regularmente el *data warehouse*, se podrían prever algunas de las siguientes áreas de mejoramiento:

- Consultas que no pudieran formularse o satisfacerse debido a limitaciones impuestas por la implantación inicial o porque comprenden fuentes de datos externos que no formaron parte de ésta.
- Desempeño no satisfactorio de componentes clave del *data warehouse*.
- Integración de nuevos departamentos al proyecto del *data warehouse*.

3.1.6 Administración del *data warehouse*

La administración del *data warehouse* comienza desde el momento del despliegue inicial, y debe comprender los siguientes puntos:

- Actualización/duplicación de datos.- La tecnología de duplicación es la capacidad persistente de hacer copias de datos dentro de una base de datos en otra base de datos conectada a ella en la misma máquina u otra diferente. La duplicación es un método conveniente para:
  - Mantener al *data warehouse* sincronizado con sus fuentes
  - Mantener a los mercados sincronizados con los almacenes de datos
  - Construir sistemas redundantes, donde el sistema redundante es una base de datos idéntica al sistema que respalda y que entra en funcionamiento al ocurrir una falla en el sistema original.

  La duplicación se tiene que planear tomando en cuenta factores como la periodicidad con que se llevará a cabo, la cantidad de tráfico que aportará a la red y si el proceso de duplicación interrumpirá o no el funcionamiento de *data warehouse*.

- Sincronización de fuentes.- La sincronización de datos es la coordinación de actualizaciones simultáneas a los mismos datos por varios usuarios. En los casos de *data warehouse* con datos duplicados en múltiples lugares se aplican controles de concurrencia similares a los utilizados para las bases de datos distribuidas.

- Recuperación - Consiste en la definición anticipada de un plan para la recuperación del sistema después de un desastre, ya que, una vez instalado y funcionando el *data warehouse*, es difícil detener todo, recuperar el servidor y volver a instalar el sistema de administración de base de datos.

- Controles de acceso y seguridad.- Se debe tener en cuenta la seguridad y la administración de acceso a los datos tanto para personas ajenas a la empresa, dado el valor que tienen los datos contenidos en el mismo, como para los propios usuarios ya que los recursos deben administrarse con el fin de evitar que unos cuantos usuarios acaparen grandes cantidades de recursos.

- Administración del crecimiento de datos.- Consiste en determinar qué datos deben permanecer y qué datos deben ser desechados del *data warehouse* y cuál es el mejor nivel de resumen de los mismos, tratando de evitar costos excesivos en dispositivos y sistemas de almacenamiento así como lentitud en la búsqueda de información.

- Administración del desempeño de la base de datos.- Dado que el *data warehouse* es una aplicación exigente, el desempeño de las consultas lo determina en gran medida la organización física de la base de datos, aunque se deben tomar en cuenta algunas otras técnicas para mejorar el desempeño:
  - Ejecución de consultas en paralelo
  - Segmentación inteligente de tablas
  - Uso de métodos avanzados de indexación
  - Uso de almacenamiento contiguo de tablas relacionadas entre sí
  - Eliminar verificaciones de integridad referencial dado que, en teoría, sólo se cargan o actualizan datos ya verificados provenientes de bases de datos operacionales
  - Protección contra consultas escape especificando límites de recursos para una consulta

- Mejoras y ampliación del *data warehouse*.- Consiste en planear la incorporación de nuevos datos al modelo de datos del *data warehouse* y su posible repercución, tanto positiva como negativa.

3.1.7 Metadatos

Los metadatos se definen como "datos acerca de los datos". En una base de datos, los metadatos son la representación de los diversos objetos que definen una base de datos. En una base de datos relacional, esta representación consiste en las definiciones de tablas, columnas, bases de datos, visualización y otros objetos. El término metadatos se usará para hacer referencia a todo lo que defina un objeto del *data warehouse*, por ejemplo una tabla, una columna, un reporte, una consulta, una regla empresarial o una transformación dentro de un *data warehouse*.

En forma muy parecida a la que una ficha de catálogo de biblioteca apunta tanto al contenido como a la ubicación de los libros de una biblioteca, los metadatos apuntan a la ubicación y al significado de información diversa dentro del *data warehouse*.

El *data warehouse* debe contener un componente que satisfaga las funciones del catálogo para la información que maneja. Los metadatos son el mapa de carreteras hacia el *data warehouse*.

Los metadatos del *data warehouse* y también los metadatos capturados durante el proceso de extracción de las fuentes de datos deben manejarse centralmente en un directorio de información. El directorio de información tiene tres componentes: un directorio técnico para apoyar el desarrollo y operación del *data warehouse*, un directorio empresarial para comunicar el contenido del *data warehouse* a los usuarios empresariales, y un navegador de información para permitir a los usuarios finales acceder a la información del *data warehouse* en distintas formas.

3.1.8 Procesamiento analítico en línea (OLAP)

El procesamiento informático y el procesamiento analítico son dos formas básicas de aprovechar el *data warehouse* para operar la empresa de manera eficiente y planear el futuro.

El procesamiento informático consta de tres componentes: consultas para acceder y recuperar los datos del *data warehouse*, el análisis de los datos y la presentación del análisis en forma de reportes. Por lo regular el ámbito del procesamiento informático está limitado a un procesamiento de dos o tres dimensiones.

En el análisis multidimensional, los datos se representan mediante dimensiones, por ejemplo, producto, territorio y cliente. Por lo regular las dimensiones se relacionan en jerarquías, por ejemplo, ciudad, estado, región, país y continente. El tiempo es también una dimensión estándar con su propia jerarquía, como día, semana, mes, trimestre y año.

El ámbito del procesamiento informático es por lo regular un análisis más sencillo de datos históricos (estáticos) para comprender el pasado. El procesamiento analítico se emplea para análisis históricos complejos con amplia manipulación (análisis de datos dinámicos), así como para planeación a futuro y pronósticos.

Al procesamiento analítico o análisis multidimensional se le conoce también como procesamiento analítico en línea (OLAP), el cual se apoya en una visión multidimensional de los datos en el *data warehouse*.

El procesamiento analítico en línea es una tecnología de análisis de datos que, entre otras cosas:

- Presenta una visión multidimensional lógica de los datos en el *data warehouse*. Esta visión es independiente de cómo se almacenan los datos.
- Comprende el análisis de los datos incluyendo la profundización en niveles cada vez más detallados o el ascenso a niveles superiores de resumen.
- Maneja modelos funcionales de pronóstico, análisis de tendencias y análisis estadísticos.
- Recupera y exhibe datos tabulares en dos o tres dimensiones, cuadros y gráficas.
- Responde con rapidez a las consultas, de modo que el proceso de análisis no se interrumpe y la información no se desactualiza.

3.2. MINEROS DE DATOS

Cuando el proveedor de servicios y el cliente, pueden convivir, es muy fácil que el proveedor de servicios conozca los gustos de cada cliente, su rango de precios, sugerencias futuras, etc. Este conocimiento lo va adquiriendo a la vez que el cliente utiliza sus servicios, y ese conocimiento que tiene sobre el cliente hace que este regrese ya que siente que es tomado en cuenta. Las grandes compañías se han percatado de este suceso, y por esta razón han puesto a la venta libros que tratan de hacer más estrecha la relación cliente-proveedor de servicios.

Las pequeñas empresas construyen relaciones con sus clientes considerando sus necesidades, recordando sus preferencias, y aprendiendo de sucesos pasados como servirles mejor en el futuro. Pero, como puede una empresa grande lograr obtener todos estos datos?, pareciera ser que no hay forma de lograrlo. Sin embargo, las grandes empresas tienen almacenada una gran cantidad de información (utilizada para algún fin específico: control de inventario o facturas, etc.) que después de ser utilizada, se guarda en algún dispositivo para no volver a ser usada. Esta información es muy valiosa, pero para poder ser utilizada como una forma de conocer a los clientes, es necesario tenerla almacenada en un solo lugar, y organizarla en una forma consistente y utilizable, a este se le conoce como "*data warehousing*". El *data warehouse* permite que la empresa recuerde que información se tiene acerca de sus clientes. Posteriormente, los datos deben de ser analizados, entendidos y devueltos en una forma servible, es aquí donde interviene el "minería de datos". El *data warehouse* provee a la empresa de memoria, pero es de muy poca utilidad si no se tiene inteligencia. La inteligencia permite a través de la memoria, encontrar patrones, deducir reglas, obtener ideas para hacer predicciones acerca del futuro.

La minería de datos es la exploración y análisis, por medios automáticos o semiautomáticos, de grandes cantidades de datos con la finalidad de descubrir patrones y reglas significativos. Las técnicas y herramientas de minería de datos son aplicables de la misma forma en radioastronomía, medicina, control de procesos industriales, etc. Los mineros comerciales emplean las técnicas apropiadas de estadística, ciencia

de la computación e inteligencia artificial. La selección apropiada para una situación dada depende de las tareas que realizará la minería de datos y de los datos disponibles.

Otra de las tareas que se han desarrollado de manera muy cercana es lo que se conoce como descubrimiento del conocimiento (*Knowledege Discovery*), el cual no asume nada, los datos hablan por si mismos. Existen dos tipos de descubrimiento del conocimiento: directo e indirecto El primero intenta explicar algunos campos particulares de datos, el segundo intenta encontrar patrones o similitudes entre grupos o registros sin el uso de un campo particular o colección predefinida de clases. Todas estas actividades caen dentro de la definición de minería de datos.

Aunque las técnicas de minería de datos han existido desde hace algunos años, la minería de datos comercial se ha dado recientemente

La mayoría de los algoritmos de minería de datos requieren grandes cantidades de datos para construir los modelos que serán usados para efectuar la clasificación, predicción, estimación u otras tareas de minería de datos.

La introducción de manejadores de bases de datos relacionales por parte de proveedores como Oracle, Informix, Red Brick, Sybase, Tandem e IBM, han proporcionado un ambiente excelente para la minería de datos a gran escala.

La minería de datos puede realizar un conjunto limitado de tareas y únicamente bajo circunstancias limitadas Muchos problemas de interés intelectual, económico y de negocios pueden ser expresados en términos de las siguientes 6 tareas:

- Clasificación.
- Estimación.
- Predicción.
- Agrupamiento por afinidad.
- Agrupamiento.
- Descripción.

Clasificación

La clasificación es la más común de las tareas de minería de datos. Para entender el mundo que nos rodea, recurrimos a la clasificación, grados y categorías, por ejemplo, dividimos la materia en elementos, los perros en razas, etc.

La clasificación consiste en examinar las características de un nuevo objeto y asignarlo a una clase ya definida En este caso, los objetos que serán clasificados están representados generalmente por registros en una base de datos y la clasificación consiste en actualizar cada registro llenando un campo con el código de la clase de algún tipo.

La clasificación se caracteriza por la definición de clases, y su tarea principal es construir un modelo de algún tipo que puede ser aplicado a datos no clasificados para poder clasificarlos. Los árboles de decisión y el razonamiento basado en el conocimiento son técnicas muy apropiadas para la clasificación. El análisis de enlaces también es muy usual para ciertas circunstancias.

Estimación

La estimación trata con resultados, dados algunos datos de entrada, la estimación se usa para proponer un valor para una variable continua desconocida tal como ingresos, altura o el balance de una tarjeta de crédito. En la práctica, la estimación es frecuentemente usada para realizar tareas de clasificación. Las redes neuronales son las más apropiadas para tareas de estimación

Predicción

La predicción es lo mismo que la clasificación o la estimación excepto que los registros son clasificados de acuerdo a la predicción de comportamientos futuros o valores futuros estimados. En una tarea de predicción, la única manera de verificar la precisión de la clasificación es esperar y ver.

Cualquiera de las técnicas usadas para la clasificación y la estimación pueden ser adaptadas para usarse en la predicción utilizando ejemplos donde el valor de la variable que se va a predecir, ya se conoce, así como los datos históricos para tales ejemplos.

Los datos históricos son usados para construir un modelo de datos que explica el comportamiento actual. Cuando este modelo se aplica a entradas actuales, el resultado es una predicción del comportamiento futuro.

El análisis del carrito del supermercado, el razonamiento basado en la memoria, los árboles de decisión y las redes neuronales son apropiadas para el uso de la predicción. La selección de la técnica depende de la naturaleza de los datos de entrada, del tipo de valor que se va a predecir y a la explicación de la predicción.

Agrupamiento por afinidad ó análisis del carrito del supermercado.

La tarea de agrupamiento por afinidad se usa para determinar cuales cosas van juntas. La venta al pormenor puede usar el agrupamiento por afinidad para planear el acomodo de los artículos en los anaqueles o en un catálogo tal que los artículos que frecuentemente se compran juntos se acomodan juntos.

El agrupamiento por afinidad también se puede utilizar para identificar las oportunidades de venta, y diseñar paquetes atractivos o agrupaciones de productos y servicios.

La agrupación por afinidad es una forma simple de generar reglas a partir de los datos.

Agrupamiento

El agrupamiento se encarga de segmentar una población heterogénea en subgrupos homogéneos. Lo que distingue la agrupación de la clasificación es que el primero no depende de clases predefinidas. En la clasificación, la población está subdividida a través de la asignación de cada elemento o registro a una clase predefinida en base a un modelo desarrollado a través del entrenamiento de ejemplos preclasificados. En el agrupamiento, no hay clases predefinidas y tampoco ejemplos, los registros se agrupan juntos en base a sus similitudes. El agrupamiento frecuentemente se da como un preludio a alguna otra forma de minería o modelado de datos.

Descripción

Algunas veces el propósito de la minería de datos es simplemente describir lo que hay en una base de datos complicada de tal manera que incremente el entendimiento de la gente, productos o procesos que produjeron los datos en primera instancia.

Algunas de las herramientas de la minería de datos, por ejemplo, el análisis de la carro del supermercado, son puramente descriptivos. Otras, como las redes neuronales, no proporcionan casi nada de información.

3.2.1 Panorama de las técnicas de minería de datos

Evolución del análisis de datos.

Sin la ayuda de las computadoras, la gente ha estado analizando los datos y buscando patrones desde antes de los primeros registros de la historia

Lo que empezó a cambiar hace unos siglos, ha sido la codificación de las matemáticas y la creación de máquinas que facilitaran la toma de medidas: su almacenamiento y su análisis.

En la historia de las técnicas de minería de datos destaca la influencia de otras disciplinas. Los algoritmos genéticos y las redes neuronales surgen de los intentos de modelar procesos biológicos sobre computadoras. El razonamiento basado en memoria es una técnica que surge del campo de la Inteligencia Artificial, y el análisis de enlaces surge de la teoría de gráficas y su aplicación a la estructura de datos en la computación.

La estadística se originó como una forma de entender las observaciones acerca del mundo real, generalmente en las ciencias naturales o en la política. La estadística descriptiva proporciona información

general acerca de las observaciones (valores promedio y media). El análisis de regresión se refiere a las técnicas usadas para interpolar y extrapolar estas observaciones.

El muestreo es una técnica muy importante usada ampliamente por la estadística (y un poco menos en la minería de datos) para reducir el tamaño de los datos. Una muestra aleatoria toma "n" registros, por ejemplo 100 registros para reducir la cantidad con la que se va a trabajar.

Hoy en día la estadística cuenta con algunas ventajas que no son compartidas por la minería de datos. La estadística es muy útil pero no resuelve todos los problemas de la minería de datos. El muestreo reduce el tamaño del conjunto de datos, de tal forma que pueden perderse datos importantes. Debido a su énfasis en funciones continuas y en formas ya conocidas , la regresión no se considera de propósito general como las técnicas de minería de datos. La complejidad computacional de los avances estadísticos no se desarrollan con las misma rapidez que las bases de datos grandes.

Modelos

Un modelo produce una o más salidas para un conjunto de datos de entrada. El análisis de datos es a menudo el proceso de construir un modelo apropiado para los datos. Una regresión lineal, por ejemplo, crea un modelo que es una línea, que tiene la forma: $aX + bY + c = 0$, donde a, b, y c son parámetros del modelo y X y Y son variables. Para un valor dado de X es posible usar la línea para estimar un valor de Y. Este tipo de modelo es uno de los modelos disponibles más simple.

El que exista un modelo, no significa que éste proporcione los resultados certeros. Para un conjunto de puntos dados, hay una línea que se ajusta mejor a dichos puntos. Un modelo puede ser usado para agrupación y clasificación entre otras funciones. Los modelos proporcionan un lenguaje común en términos de minería de datos.

Un modelo de clasificación toma un nuevo registro y lo asigna a una clasificación. También puede asignar una nueva clasificación, una probabilidad de corrección y cualquier otra información dependiendo de la naturaleza del modelo. Un modelo predictivo es similar al modelo de clasificación, excepto que la salida no está limitada a un grupo de clases. Un modelo de agrupamiento toma muchos registros y devuelve un pequeño número de grupos, los cuáles pueden ser aplicados a nuevos registros para crear un modelo de clasificación.

Cuando los modelos son creados, generalmente la entrada se especifica claramente. Las entradas al modelo pueden afectar la selección de la técnica. Para problemas físicos con muchas variables de entrada continuas, las técnicas de regresión estadística regularmente trabajan muy bien. Cuando las entradas tienen muchas variables categóricas, los árboles de decisión son la mejor opción. Con frecuencia un modelo proporciona un grado de confiabilidad en la salida, lo cual es muy útil para determinar cuando aplicar los resultados del modelo.

Cuando se crean modelos para minería de datos, hay algunos aspectos que se deben de tener en mente:

1. Uno de los peligros de los modelos es el "*underfitting*" y el "*overfitting*" de los datos.
2. La minería de datos dirigida y la no dirigida usan modelos ligeramente diferentes.
3. Algunos modelos proponen que trabajan mejor que otros.
4. Algunos modelos son más fáciles de aplicar que otros.

1) *Underfitting y Overfitting*

Algunas veces los modelos no trabajan muy bien, dos causas comunes son el "*underfitting*" y el "*overfitting*" de los datos. Un ejemplo muy ilustrativo es el croquis de una ciudad. Supongamos que se tienen dos croquis, uno muy detallado de tal forma que indica los nombres de todas las calles de una ciudad, los edificios y hasta los nombres de los parques. Y el otro solo indica las avenidas principales. En el caso de que la ciudad forme parte de la ruta hacia algún destino, el plano detallado no es necesario, ya que en lugar de ayudar solo causaría confusiones debido a la cantidad de información que contiene, lo cual se conoce como "*overfitting*". Sin embargo, si lo que se requiere es saber el nombre de alguna calle, el croquis general no serviría, por lo tanto se dice que hay "*underfitting*", es decir, el croquis general o no contiene la información requerida o no la describe bien.

En términos de minería de datos, el "*overfitting*" de los datos se da cuando el modelo memoriza los datos y predice resultados basados en los datos usados para pruebas. De esta manera, el modelo produce buenos resultados pero solo para el conjunto de datos de prueba, pero no para otros datos..

El "*underfitting*" ocurre cuando un modelo resultante falla en patrones de coincidencia de interés en los datos. El "*underfitting*" es común cuando se aplican técnicas estadísticas a los datos y una causa muy frecuente es la eliminación de variables que tienen poder predictivo pero que no son incluidas en el modelo.

2) Dirigida contra no-dirigida

La diferencia entre la minería de datos dirigida y no dirigida ocurre cuando se crea el modelo de minería de datos. En la minería de datos dirigida, la salida del modelo es especificado antes de crear el modelo. En un modelo no dirigido, el modelo determina la salida y el analista determina lo que es importante de los resultados.

3) Explicabilidad

Para algunos propósitos el saber porque un modelo genera ciertos resultados, no es importante, sin embargo, para otros propósitos, esto puede ser absolutamente insignificante. Algunos tipos de modelos son más fáciles de entender que otros. Por ejemplo, los árboles de decisión y el análisis del carro del supermercado producen reglas claras, mientras que las redes neuronales y las técnicas de agrupación sólo proporcionan una vaga idea.

4) Fácil aplicación del modelo.

Si los datos están almacenados en una base de datos relacional, entonces se puede implantar un modelo que utilice sentencias de SQL, y éste sería preferible a uno que requiera exportar los datos con otras herramientas.

Técnicas

Análisis del carrito del supermercado

Este análisis es una forma de agrupación usada para encontrar grupos de artículos que tienden a ocurrir juntos en una transacción. Los modelos que construye devuelven la probabilidad de que diferentes productos sean comprados juntos y puede expresar los resultados en forma de reglas.

Como una técnica de agrupamiento, el análisis del carrito del supermercado es muy útil en problemas donde se desea saber cuales artículos ocurren juntos o en una secuencia particular. Los resultados son muy significativos ya que presenta los artículos específicos.

Razonamiento basado en memoria (MBR - Memory Based Reasoning)

Es una técnica de minería de datos dirigida que utiliza instancias conocidas como un modelo para hacer predicciones acerca de instancias desconocidas. Busca los "vecinos" más cercanos dentro de las instancias conocidas y combina sus valores para asignar valores de predicción o clasificación.

Una de las principales ventajas del razonamiento basado en memoria, es su capacidad para correr sobre cualquier fuente de datos, aun sin la modificación de datos. Los dos elementos clave en esta técnica son: la función de distancia usada para encontrar los vecinos más cercanos y la función que combina valores de los vecinos más cercanos para hacer la predicción. El RBM puede estar basado en tipos de datos más complejos tales como texto e imágenes.

Otras de las ventajas del RBM es su capacidad para aprender acerca de nuevas clasificaciones introducidas por nuevas instancias dentro de la base datos.

Análisis de Enlaces

El análisis de enlaces sigue las relaciones entre registros para desarrollar modelos basados en patrones en las relaciones. Esta es una aplicación de la teoría de gráficas construida para la minería de datos. Con frecuencia, el uso de análisis de enlaces requiere que el código sea dependiente de la aplicación.

Árboles de decisión y reglas de inducción.

Los árboles de decisión son usados para la minería de datos dirigida, principalmente para la clasificación. Dividen los registros en subconjuntos no relacionados, cada uno de los cuales está descrito por una regla simple en uno o más campos.

Una de las ventajas de los árboles de decisión es que el modelo es muy explicativo dado que presenta reglas de manera explícita, esto permite a la gente evaluar los resultados, identificando los atributos clave en un proceso. Las reglas pueden ser expresadas fácilmente como sentencias lógicas en un lenguaje como SQL.

Redes neuronales artificiales.

Las redes neuronales son las técnicas más comunes en la tecnología de minería de datos. Son modelos simples de las interconexiones neuronales en el cerebro, adaptados para usarse en las computadoras. De un conjunto de pruebas generalizan patrones que haya dentro de ellas, para obtener la clasificación y la predicción. Las redes neuronales son interesantes debido a que detectan patrones en los datos de manera análoga al pensamiento humano. Sin embargo, es difícil entender los modelos que producen.

Algoritmos genéticos.

Los algoritmos genéticos aplican los mecanismos de la selección genética y natural para encontrar los conjuntos óptimos de parámetros que describen una función predictiva También se usa para mejorar el razonamiento basado en memoria y las redes neuronales.

Procesamiento Analítico en Línea (OLAP)

No todo lo que se es usado para el análisis de datos es necesariamente minería de datos. OLAP es una forma de representar datos relacionales a los usuarios para facilitar el entendimiento de los mismos y los patrones importantes dentro de ellos. No es específicamente una herramienta para minería de datos, pero es una herramienta importante para extraer y presentar información.

3.2.2 Análisis del carrito del supermercado

Con frecuencia este tipo de análisis es usado como punto de partida cuando está disponible la transacción de los datos y no se sabe qué patrones buscar.

El uso del análisis del carrito del supermercado se da debido a la claridad y utilidad de sus resultados, los cuales se presentan en forma de reglas de asociación. Una regla de asociación expresa que tan relacionados pueden estar los productos y los servicios y como tienden a agruparse. Una regla como la siguiente resulta ser muy clara: "Si un cliente solicita el servicio de tres llamadas a la vez, entonces ese cliente también solicita el servicio de llamada en espera".

Existen tres tipos comunes de reglas producidas por el análisis del carrito del supermercado: las útiles, las triviales y las inexplicables.

- *Las reglas útiles contienen información relevante y no es difícil de justificar.* Además permite encontrar las causas más probables y definir estrategias que permitan obtener mejores resultados. Por ejemplo, la siguiente regla: "Los jueves, los clientes de la tienda de abarrotes frecuentemente compran pañales y cervezas". La regla es muy clara, se puede deducir, que las parejas se preparan para el fin de semana. Debido a que la regla es fácil de entender, se pueden tomar las siguientes medidas basadas en

la regla: poner otros productos de bebé a la vista de los consumidores de cervezas y poner algun tipo de botanas cerca de los artículos para el bebé.

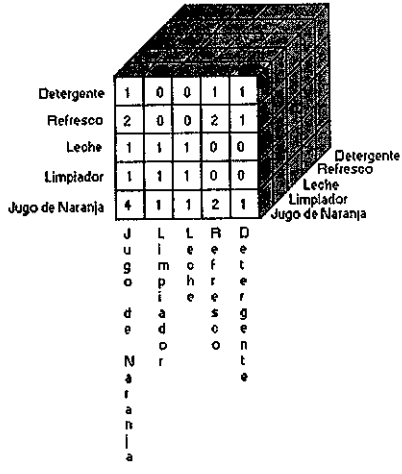
- *Los resultados triviales generalmente son conocidos por cualquier elemento de la compañía.* Es decir son reglas muy triviales: "La gente que compra pintura, también compra brochas para pintar". Esto se debe a que los datos que utiliza el análisis del carrito del supermercado son los mismos que se utilizan por las campañas de mercadeo anteriores.

- Los resultados inexplicables no sugieren una acción acertada.

Cuando se aplica el análisis del carrito del supermercado, muchos de los resultados frecuentemente son triviales o inexplicables. Con frecuencia los resultados triviales miden acciones previas , como las campañas de mercadeo, pero no proporcionan ninguna guía para acciones futuras. La determinación de las reglas que son valiosas puede requerir conocimientos de campañas de mercadeo previas y otros factores históricos.

El análisis del carrito del supermercado inicia con las transacciones que contienen uno o más productos o servicios ofrecidos e información acerca de la transacción. Cada una de las transacciones proporciona información acerca de qué objetos van acompañados de otros objetos. Estos datos se utilizan para crear tablas de concurrencia, la cual indica el numero de veces que un par de objetos se encuentran juntos.

Las tablas de concurrencia no solo sirven para pares de objetos, sino para cualquier número de objetos. En el caso de tres objetos, el número de subcubos es proporcional a la potencia de 3 del número diferente de objetos tal como se muestra en la siguiente figura. En general, el número de combinaciones con n objetos es proporcional al número de objetos que resulten de elevar éste número a la n-ésima potencia.



Los objetos de interés pueden cambiar con el tiempo. escoger el nivel correcto de detalle es crítico para el análisis. En el mundo real los objetos tienen categorías jerárquicas. El número de combinaciones crece tan rápido como el número de objetos, por lo que se debe utilizar objetos con niveles jerárquicos más

altos, por ejemplo: "postres fríos" en lugar de "helados". Sin embargo, los objetos más específicos tienen más probabilidad de generar resultados favorables.

La complejidad de una regla se refiere al número de objetos que contiene. La complejidad deseada en las reglas también determina que tan específicos o generales deben ser los objetos. Los datos usados en el análisis del carrito del supermercado no son de muy buena calidad ya que generalmente son usados principalmente para propósitos operacionales como el control de inventario. Los datos provenientes de los sistemas operacionales frecuentemente se encuentran "sucios" y necesitan una limpieza para poder ser usados como fuente para el soporte de decisiones. Los datos tienen múltiples formatos, correcciones, códigos incompatibles, etc.

El cálculo de número de veces que una combinación de objetos dada aparece en la transacción de los datos es bueno, sin embargo un número de combinaciones no es una regla. Una regla tiene dos partes, una condición y un resultado, y regularmente se representa de la siguiente manera.

*Si condición **entonces** resultado.*

La generación de reglas de asociación es un proceso de varios pasos, el cual consiste en generar la matriz de concurrencia para objetos solos, generar la matriz de concurrencia para dos objetos y utilizarla para encontrar reglas con dos objetos, generar la matriz de concurrencia para tres objetos y utilizarla para encontrar reglas con tres objetos, y así subsecuentemente.

Si se incrementa el número de objetos en la combinación, también se requerirá mayor procesamiento. En este caso, la solución es reducir el número de objetos y combinaciones de objetos que son considerados en cada paso. Cada paso del cálculo de la tabla de concurrencia puede eliminar combinaciones de artículos que no se encuentran dentro del límite, reduciendo su tamaño y el número de combinaciones a considerar durante las próximas pasadas.

El número de combinaciones puede crecer exponencialmente, por ejemplo, un supermercado tiene al menos 10,000 objetos diferentes y a veces hasta 20,000 o 30,000. Una regla de disociación es similar a una regla de asociación excepto que ésta puede tener el conector "y no" en la condición y el conector "y". Una regla de disociación típica como:

*Si A y no B **entonces** C*

Las reglas de disociación pueden ser generadas por una simple adaptación del algoritmo de análisis del carrito del supermercado. La adaptación introduce un nuevo grupo de objetos que son inversos de los objetos originales. Entonces modificar cada transacción incluye un objeto inverso si, y sólo si no contiene el objeto original. Por ejemplo, la siguiente tabla muestra la transformación de algunas transacciones. El símbolo ¬ antes del objeto denota que es el inverso.

Cliente	Artículos		Cliente	
1	{A,B,C}	→	1	{A,B,C}
2	{A}	→	2	{A,¬B,¬C}
3	{A,C}	→	3	{A,¬B,C}
4	{A}	→	4	{A,¬B,¬C}
5	{}	→	5	{¬A,¬B,¬C}

Para usar series de tiempo, la transacción debe tener dos características adicionales:

- Información secuencial para determinar cuando ocurrieron transacciones relativas a otras.
- Identificar información, tal como número de cuenta o el identificador del cliente, que pueden identificar las transacciones que pertenecen al mismo cliente.

Las transacciones que corresponden al mismo cliente son reunidas en series de tiempo. Estas series muestran cuales objetos sucedieron antes, después o al mismo tiempo que otros.

Una forma simple pero poderosa para usar las series de tiempo es analizar la causa y el efecto. Este es un método directo para encontrar las causas de un evento que ocurre en un tiempo particular. Se trata de convertir el problema de la serie de tiempo en un problema de análisis del carrito del supermercado. Cada serie de tiempo es convertida en una transacción incluyendo los objetos antes del evento de interés (para causas) o después de los objetos de interés (para efectos) eliminando objetos duplicados en la transacción.

Ventajas

- Produce resultados claros y entendibles
- Soporta minería de datos no dirigida
- Trabaja con datos de longitud variable
- Las operaciones que usa son fáciles de entender.

Desventajas

- Requiere de mayor procesamiento dado el crecimiento exponencial
- Es difícil determinar el número correcto de objetos
- Elimina objetos raros.

3.2.3 Razonamiento basado en la memoria

Cuando una persona ve una cara en una multitud, compara esa cara con otras que conoce. Cuando los médicos diagnostican alguna enfermedad, están aplicando su experiencia con pacientes con síntomas similares en el caso actual. La identificación de un rostro en una multitud y el diagnóstico de una enfermedad siguen un proceso similar: el primer paso es identificar casos similares en la experiencia, y entonces aplicar la información de estos casos para el problema en cuestión. Esto es la esencia del Razonamiento Basado en la Memoria (RBM), que es una técnica de minería de datos dirigida que, de manera similar se basa en la experiencia. Para mantener una base de datos con registros conocidos, el RBM encuentra vecinos similares a los nuevos registros y usa los vecinos para la clasificación y predicción.

A diferencia de otras técnicas de minería de datos, ésta técnica no se preocupa por el formato de los registros, únicamente se preocupa por la existencia de dos operaciones: la función de distancia que asigna una distancia entre dos registros, y la función de combinación que combina el resultado de los vecinos para entregar un a respuesta.

Las áreas de aplicación que comprende el RBM son:

- Detección de fraudes: Los casos nuevos de fraudes, probablemente son similares a casos que se presentaron con anterioridad.
- Predicción de la respuesta del cliente: Los próximos clientes pueden responder casi de la misma manera que clientes anteriores.
- Tratamientos médicos: El tratamiento que recibe un paciente probablemente es el mismo que recibió con anterioridad un paciente con los mismos síntomas.

RBM tiene dos fases: la fase de aprendizaje que genera la base de datos histórica y la fase de predicción que aplica el RBM en nuevos casos. Los tres puntos principales en la resolución de un problema con RBM son:

1. Elegir el conjunto adecuado de datos históricos.
2. Determinar la forma más eficiente de representar los datos históricos
3. Determinar la función distancia, la función de combinación y el número de vecinos.

El rendimiento del RBM en las predicciones depende de como los conjuntos de entrenamiento o datos históricos son representados en la computadora. La forma más simple de encontrar el vecino más cercano requiere encontrar la distancia entre el desconocido con cada uno de los registros de los datos históricos y escogiendo los que tengan menor distancia. A medida que el número de registros crece, el tiempo para encontrar a los vecinos también.

La función de distancia, la función de combinación y el número de vecinos son ingredientes clave para determinar que tan buenos resultados puede producir el RBM.

La distancia es la forma en que el RBM mide las similitudes. La distancia de un punto A a un punto B, denotado como d(A,B), tiene las siguientes propiedades:

Bien definida. La distancia entre dos puntos esta siempre definida y es un número real no negativo d(A,B) >= 0,

Identidad: La distancia de un punto a él mismo siempre es cero, d(A,A) = 0.

Conmutatividad: La dirección no hace una distancia, es decir, la distancia de A a B es la misma que de B a A d(A,B) = d(B,A).

La distancia es usada para encontrar el vecino más cercano, el conjunto de vecinos más cercanos puede tener las mismas propiedades peculiares. Hay tres funciones de distancia más comunes para campos numéricos:

El valor absoluto de las diferencias: |A - B|

El cuadrado de la diferencia: $(A - B)^2$

El valor absoluto normalizado: |A - B|/ diferencia máxima.

La ventaja de la última función es que el valor resultante siempre se encuentra entre 0 y 1.

El RBM es una técnica muy poderosa de la minería de datos que puede ser usada para resolver una amplia variedad de problemas de minería de datos dirigida.

Ventajas

- Produce resultados entendibles.
- Se aplica a tipos de datos arbitrarios.
- Trabaja eficientemente en casi cualquier tipo de datos.
- El mantenimiento del conjunto de entrenamiento requiere muy poco esfuerzo.

Desventajas

- Es muy caro realizar las tareas de clasificación y predicción.
- Requiere gran cantidad de almacenamiento para los datos.
- Los resultados dependen de la función de distancia y el número de vecinos.

3.2.4 Análisis de Enlaces

El mundo de los negocios es un mundo de relaciones que enlaza gente, lugares y cosas, las líneas aéreas enlazan ciudades, la gente se comunica a través de compañías de telecomunicaciones que ofrecen servicios de larga distancia, los propietarios de tarjetas de crédito prefieren algunos tipos de restaurantes, los usuarios de Internet tienen sitios de interés, etc. En fin, las relaciones están en todas partes y contienen una gran cantidad de información de la cual no pueden tomar ventaja la mayoría de las técnicas para minería de datos, el análisis de enlaces es la técnica capaz de explotar estas relaciones.

El análisis de enlaces esta basado en una rama de las matemáticas llamada teoría de gráficas y no es aplicable a todos los tipos de datos o para resolver cualquier tipo de problema. Existen pocas herramientas que soportan explícitamente el análisis de enlaces, las que lo hacen, se enfocan a la visualización de los enlaces y son más útiles para el área del descubrimiento del conocimiento que para encontrar patrones

Las consultas en SQL pueden ser la forma más básica de realizar análisis de enlaces, sin embargo, estas consultas demandan mucho tiempo de ejecución dado que los enlaces son equivalentes a uniones (*joins*) entre tablas en el modelo relacional.

Otro problema común para el análisis de enlaces es poder reconocer cuando existe una relación o conexión entre dos entidades, existen casos obvios, por ejemplo, en el caso de una llamada telefónica una persona llama a otra y la llamada en sí misma es la relación o enlace, sin embargo, existen casos en los que poder reconocer estos elementos es muy difícil.

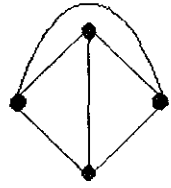
A continuación se citan algunos conceptos sobre teoría de gráficas que son útiles para entender como funciona el análisis de enlaces.

Conceptos sobre teoría de gráficas

Las gráficas son una abstracción desarrollada específicamente para representar relaciones y han probado ser muy útiles tanto en las matemáticas como en la ciencia de la computación para el desarrollo de algoritmos que exploten estas relaciones Una gráfica consta de dos partes:

- Nodos.- Algunas veces llamados vértices, representan a los elementos que tienen relaciones, tienen nombres y con frecuencia, algunas otras propiedades.
- Enlaces.- Son pares de nodos conectados o unidos por una relación, se representan por los pares de nodos a los que conecta, por ejemplo AB representa al enlace que conecta a los nodos A y B.

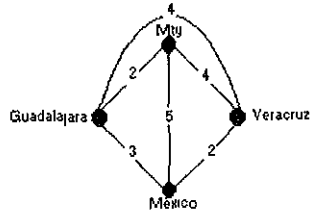
La siguiente gráfica tiene cuatro nodos unidos a través de seis enlaces y tiene la propiedad de que existe un enlace entre cada par de nodos, en la teoría de grafos se dice que esta gráfica está *totalmente conectada*.



Una gráfica sin etiquetas puede representar muchas cosas, la gráfica de arriba puede representar los vuelos de una aerolínea entre cuatro ciudades, cuatro personas que se conocen, etc. Otra característica importante de la gráfica anterior es que no existe ninguna intersección entre las líneas que unen a los nodos, a las gráficas que cumplen con esta condición se les llama gráficas planas dado que pueden ser dibujadas en una hoja de papel, es decir, en un plano. No todas las gráficas pueden cumplir con esta condición, a las gráficas que tienen una o más intersecciones entre sus enlaces se les llama gráficas no planas.

Un camino, como su nombre lo indica, es una secuencia ordenada de nodos conectados por enlaces.

La gráfica que se muestra a continuación se denomina gráfica pesada ya que tiene etiquetas y pesos asociados a las líneas que unen a los nodos, el peso representa el costo, en algún tipo de unidades, que tiene el ir de un nodo a otro. Este tipo de gráficas son la representación de casos concretos y ayudan a la solución de problemas como encontrar el camino más corto, el de menor costo, etc.



Existe otro tipo de gráficas llamadas *gráficas dirigidas*, en una gráfica dirigida se puede identificar plenamente cual es el nodo origen y cual es el nodo destino de un enlace, es decir un enlace que va del nodo A al nodo B se dice que *sale* de A y *entra* a B, por lo tanto es distinto de un enlace que va del nodo B al nodo A Existen dos tipos de nodos que tienen un interés particular dentro de las gráficas dirigidas para el análisis de enlaces, estos son los llamados *nodos origen* que son aquellos de donde salen una gran cantidad

de enlaces pero a los cuales no llega ninguno, y los *nodos destino*, a los cuales, en contraparte, llega una gran cantidad de enlaces pero no sale ninguno

Otra propiedad de las gráficas dirigidas es la existencia de ciclos, un ciclo es un camino que comienza y termina en el mismo nodo y puede repetirse en forma infinita, una gráfica que contiene al menos un ciclo es llamada *gráfica cíclica*, si la gráfica no contiene ciclos es llamada *gráfica acíclica*. En una gráfica acíclica cualquier par de nodos tiene una precedencia bien definida, es decir, si el nodo A precede al nodo B en un camino que contenga a ambos, entonces el nodo A siempre precederá al nodo B en cualquier camino que contenga a ambos nodos, de otra forma se establecería un ciclo. En este caso, el nodo A es el predecesor del nodo B y el nodo B es el sucesor del nodo A. Este tipo de consideraciones son muy útiles para los propósitos de minería de datos.

Existe otro tipo de gráficas donde los nodos no representan cosas del mismo tipo sino entidades diferentes interactuando unas con otras, este tipo de gráficas son llamadas redes. Los enlaces en una red también representan relaciones, sin embargo, el tipo de relación depende de lo que representen los nodos que une ese enlace.

Ventajas

Existen 3 ventajas que se distinguen principalmente para el análisis de enlaces.

- Es el más apropiado para datos ligados.- Las bases de datos y en particular algunos problemas de minería de datos involucran el concepto de relaciones o datos ligados, para estos casos la técnica de análisis de enlaces es la más apropiada ya que es la manera natural de representar ese tipo de problemas, ejemplos existen muchos como el de las compañías de larga distancia dado que la comunicación se realiza siempre entre dos personas y tiene un origen y un destino, el de las compañías de transportación donde los mapas pueden ser representados como gráficas, etc.
- Es muy útil para visualización.- Las gráficas son una manera muy natural de visualizar algunos tipos de datos, lo cual contribuye ampliamente al descubrimiento de conocimientos. Aún en circunstancias donde se puedan encontrar patrones en forma automática, la visualización ayuda a entender qué es lo que está pasando y propone formas alternativas a las que proponen los formatos de las bases de datos relacionales y las herramientas OLAP Los enlaces pueden, además sugerir patrones importantes en los datos, sin embargo, se requiere de una persona para su interpretación.
- Ayuda a crear atributos derivados.- La información contenida en las gráficas frecuentemente es usada para crear nuevos atributos para los datos que no son visibles cuando éstos son presentados en otros formatos.

Desventajas

Algunas desventajas de esta técnica son:

- No es aplicable a todos los tipos de datos.- Aunque esta técnica es muy poderosa no puede ser aplicada a muchos tipos de problemas dado que no es una herramienta de predicción o clasificación como las redes neuronales. Existen muchos tipos de datos que no pueden ser representados en forma de gráficas y por tanto son incompatibles con el análisis de enlaces.
- Existen pocas herramientas que soportan esta técnica.- Para hacer análisis de enlaces generalmente es necesario escribir código que satisfaga situaciones particulares dado que existen pocas herramientas de propósito general y las que existen proporcionan la visualización únicamente para algunos cientos de elementos.
- Las implantaciones en bases de datos relacionales resultan ineficientes.- Dado que las gráficas representan ligas entre tablas, la implantación de esta técnica mediante SQL en bases de datos muy grandes resulta ineficiente.

Finalmente cabe señalar que la técnica de análisis de enlaces es una técnica para descubrir, no es una herramienta de predicción o clasificación, sin embargo, una vez que se ha detectado algún patrón a través de ella, éste se puede convertir en un atributo que puede llegar a tener mucho valor para otras técnicas de minería de datos como las redes neuronales o los árboles de decisión.

3.2.5 Árboles de decisión

Los árboles de decisión son herramientas muy poderosas y populares para clasificación y análisis. El atractivo de los métodos basados en árboles radica en que, en comparación con las redes neuronales, los árboles representan reglas y las reglas pueden expresarse fácilmente en un lenguaje entendible por los humanos o en algún lenguaje entendible por una computadora.

Existen varios algoritmos para construir árboles de decisión, estos algoritmos se explicarán más adelante en esta sección.

Un árbol de decisión representa una serie de preguntas, la respuesta a la primer pregunta determina cuál será la pregunta que se tendrá que responder a continuación. Generalmente son dibujados con la raíz en la parte superior y las hojas en la inferior. Un registro entra al árbol en el nodo raíz, aquí se debe hacer una pregunta para determinar cuál es el *nodo hijo* que el registro encontrará en seguida, este proceso debe ser repetido hasta que el registro llegue a un *nodo hoja*. Todos los registros que terminen en una misma hoja son clasificados de la misma manera. Existe un solo camino desde la raíz a cada hoja y este camino es una expresión de la regla usada para clasificar esos registros.

Los árboles pueden crecer en diferentes formas, el número de hijos que puede tener un nodo puede variar desde dos (árboles binarios) hasta *n*, así como la profundidad del mismo, es decir, la distancia desde la raíz hasta las hojas.

La efectividad de un árbol se comprueba tomando un grupo de registros, haciéndolos pasar por el árbol y determinando qué porcentaje de ellos fueron clasificados correctamente, además, es posible obtener estadísticas de cada nodo acerca del número de registros que entran y el porcentaje de registros clasificados correctamente en ese nodo.

Otra manera de representar la forma en que los registros son clasificados en un árbol de decisión es mediante "cajas", es decir, se puede ver a las hojas de los árboles como cajas donde son guardados un cierto número de registros que han cumplido con la regla para llegar hasta ahí, a partir de estas cajas pueden ser elaborados histogramas que indiquen cuántos registros se han guardado en cada caja.

Arboles de clasificación y regresión (CART)

Este algoritmo para la construcción de árboles de decisión es el más popular y fue publicado por L. Briemen y asociados en 1984.

Al comenzar el proceso se tiene un conjunto de registros preclasificados, es decir, que tienen una clase conocida, la meta es construir un árbol que permita asignar una clase al campo destino de un nuevo registro en base a los valores de los otros campos o variables independientes. El algoritmo CART construye un árbol binario separando los registros en cada nodo de acuerdo a la función de un solo campo de entrada La primera tarea consiste en decidir a través de cual de los campos independientes se puede hacer una mejor separación. Se considera que el mejor separador es aquel que puede tomar una menor cantidad de valores, es decir, el que tiene el mayor decremento en la diversidad de valores que puede adquirir. Existen funciones matemáticas que pueden ser maximizadas con el objeto de encontrar cual es el campo que puede ser usado como separador, sin embargo, en la práctica, el método que se sigue es considerar todos los campos independientes y ordenarlos de acuerdo al número de valores que pueden tomar, el que pueda tomar el menor número de valores es considerado como el mejor y es usado como separador para el nodo raíz. Este proceso debe ser repetido tantas veces como sea necesario, cuando un campo puede tomar únicamente un valor entonces es excluido del proceso y es considerado un nodo hoja. Al terminar el proceso, cada registro del conjunto inicial debe haber sido clasificado en una hoja y se debe determinar el grado de error del árbol a través de métodos estadísticos

Dado que el árbol resultante surgió de un conjunto de datos preclasificados es muy posible que tenga algunos errores, es necesario entonces "podar" el árbol, la forma de hacerlo es dividiéndolo en subárboles más pequeños a partir de los nodos que tuvieron más errores y aplicar a estos subárboles algunos otros conjuntos de registros, de tal forma que el árbol ganador sea el que tiene menos probabilidad de error.

C4.5

El algoritmo C4.5 es relativamente reciente y fue desarrollado por el profesor australiano J. Ross Quinlan. C4.5 es un algoritmo usado en un paquete comercial de minería de datos de la compañía Integral Solutions, Ltd. llamado *"Clementine"*.

La primera diferencia que se encuentra entre el algoritmo CART y el C4.5 es que el primero siempre construye árboles binarios mientras que el segundo genera árboles con números variables de hijos por nodo, esto se debe al hecho de que, aunque las variables continuas son tratadas en forma muy similar, las variables categóricas son tratadas en forma diferente. Cuando C4.5 determina que un campo categórico será usado como separador asume que se debe crear una rama por cada valor que pueda tomar esa variable, por ejemplo, si se elige el campo "color" como el mejor campo para hacer la separación en la raíz y el primer conjunto de datos tiene registros que en el campo "color" tienen los valores rojo, amarillo, verde, azul y blanco, entonces habrán 5 nodos en el siguiente nivel del árbol.

El algoritmo C4.5 hace uso del concepto "ganancia de información" para decidir qué campo puede ser usado como mejor separador. La ganancia de información se puede entender de la siguiente forma: el número de bits para representar un campo depende de el número de valores que puede tomar ese campo, en otras palabras, si un campo puede tomar ocho distintos valores igualmente probables se puede representar con $\log_2(8)$ o 3 bits, si un campo puede tomar 4 distintos valores puede ser representado por $\log_2(4)$ o 2 bits, si después de una división en la cual el árbol tuvo que ser dividido en 8 nodos, éstos pueden tener un máximo de 4 hijos cada uno, se dice que en ese nivel el árbol tuvo una ganancia de información de 1 bit. La ganancia de información es un concepto muy útil que permite que un árbol tenga muy poca profundidad, sin embargo debe tenerse cuidado ya que si siempre se busca hacer divisiones con mucha ganancia de información, se puede dar que el árbol llegue a su final sin que se hubieran considerado una gran cantidad de atributos que también aportarían información al mismo.

La forma de convertir un árbol en reglas también requiere especial cuidado dado que un árbol generado mediante C4.5 tiene más hijos por nodo que un árbol generado mediante CART, por razones obvias, el conjunto de reglas que produce un árbol C4.5 es mucho mayor, ésto se contrapone a los esfuerzos de la minería de datos ya que esto amplía el problema en lugar de ofrecer una solución, de esta forma, el programa encargado de generar las reglas a partir de los árboles C4.5 debe agrupar las reglas que llegan al mismo resultado y tratar de generalizarlas eliminando de éstas las características que no aportan información de tal manera que el número de reglas sea el mínimo.

Detección Automática de Interacciones Chi-cuadrada (CHAID)

El algoritmo CHAID fue publicado por J. A. Hartigan en 1975, es el más antiguo pero también el más usado ya que forma parte de muchos paquetes estadísticos como SPSS y SAS. La frase detección automática de interacciones nos deja ver que la motivación para la creación de este algoritmo fue la idea de detectar relaciones estadísticas entre variables.

A diferencia de los otros dos algoritmos, el algoritmo CHAID detiene el crecimiento del árbol hasta que se realiza una verificación a los datos, otra diferencia importante es que CHAID está restringido a variables categóricas, las variables continuas deben ser separadas en rangos o reemplazadas por clases como alto, medio o bajo.

La forma de elegir un campo separador consiste en agrupar registros cuyos valores en los campos elegidos como variables de interés no tengan diferencias "significativas" a fin de que en el siguiente nivel todos estos registros puedan tener una misma clasificación. La forma de saber si la diferencia es significativa o no, es aplicando el análisis estadístico de la χ^2 (Chi-cuadrada) a los valores del campo elegido. Así, el árbol crecerá hasta que ya no se puedan hacer más divisiones a través de este método.

Simulación

Otra característica de los árboles de decisión es la de poder extrapolar patrones secuenciales, esto permite hacer simulaciones, es decir, proyectar los valores de todas las variables hacia el futuro, el resultado es una base de datos extendida que contiene exactamente los mismos campos que la original pero con valores proporcionados por la simulación y no por la observación.

Dentro de la terminología utilizada para la simulación cada "fotografía" obtenida de los datos históricos para un momento determinado es llamada *caso*. Un caso está compuesto de *atributos* que constituyen los campos de un registro caso, los atributos pueden estar compuestos de cualquier tipo de dato y pueden ser continuos o categóricos. Los atributos son usados para formar *características*, que son variables booleanas que se combinan en varias formas para formar los nodos del árbol de decisión Los atributos son utilizados también para formar *interpretaciones*, las cuales son nuevos atributos derivados de los anteriores y generalmente reflejan conocimiento sobre el dominio e importancia de algunas relaciones.

La idea central de la visualización proyectiva o simulación es usar los casos históricos para generar un conjunto de reglas capaces de generar el caso n+1 a partir del caso n, así cuando estas reglas son aplicadas al último caso observado se genera el primer caso proyectado o predictivo pudiéndose generar tantos casos como se quiera aunque, entre más casos se generen las reglas tienen más probabilidad de fallar.

Actualmente, con el fin de mejorar cada vez más la técnica de los árboles de decisión varias compañías y laboratorios se encuentran haciendo pruebas para implantar en sistemas comerciales mejoras a los algoritmos normales como el poder elegir más de una variable a la vez para hacer la separación en un nodo o implantar en cada nodo una pequeña red neuronal que trate de "aprender" a donde debe dirigir un registro que haya entrado a ese nodo, a esta mejora al algoritmo se le ha llamado "árboles neuronales".

Ventajas

- Son capaces de generar reglas entendibles.
- Llevan a cabo clasificaciones sin necesidad de muchos requerimientos de cómputo.
- Son capaces de manejar tanto variables continuas como categóricas.
- Ofrecen una clara idea de qué campos son los más importantes para predicción y clasificación.

Desventajas

- Requieren de mucho trabajo de cómputo para el crecimiento del árbol.
- Generalmente sólo examinan un campo a la vez esto ocasiona que las "cajas" de clasificación no correspondan con la distribución de los registros en el espacio de decisiones.

3.2.6 Redes Neuronales

Las redes neuronales son una clase de herramientas de propósito general muy utilizadas para predicción, clasificación y agrupamiento. El nombre de redes neuronales se debe a que este tipo de herramientas buscan modelar en una computadora digital las conexiones entre las neuronas de un cerebro humano. Cuando son usadas en dominios bien definidos, su habilidad para aprender de los datos se parece a nuestra habilidad para aprender de la experiencia, esta característica es muy usada dentro de la minería de datos y convierte a las redes neuronales en un área muy interesante para su estudio dado que promete nuevos y mejores resultados en el futuro.

Sin embargo existe un problema, los resultados de entrenar una red neuronal son pesos distribuidos internamente a través de la red, estos pesos proporcionan la mayor parte de las veces una respuesta correcta, pero no es posible explicar la razón por la cual ésa es la mejor respuesta.

Historia

Los primeros estudios sobre la forma de trabajar de las neuronas tuvieron lugar en 1943, es decir, antes de que aparecieran las computadoras digitales, en ese año, Warren McCulloch y Walter Pits postularon un modelo simple para explicar la forma en la que trabajan las neuronas biológicas, su intención era entender la anatomía del cerebro y no pensaron que este modelo aportara un nuevo enfoque para resolver cierta clase de problemas fuera de la rama de la neurobiología.

En la década de los 50, una vez que las primeras computadoras digitales ya estaban disponibles, algunos científicos de la computación implantaron modelos llamados perceptrones basados en el trabajo de McCulloch y Pits, sin embargo, el éxito obtenido con los perceptrones en los laboratorios fue escaso y no se consideró que fueran una herramienta para solucionar problemas generales. Las razones por las cuales los perceptrones no tuvieron mucho éxito fueron las limitaciones de los equipos de cómputo de esa época y algunas deficiencias teóricas de los modelos de redes neuronales, esto provocó que durante las décadas de los 60s y 70s la investigación en esta área decayera dramáticamente.

En 1982, John Hopfield inventó la propagación hacia atrás (backpropagation), que es una forma de entrenar redes neuronales que deja a un lado los errores de las primeras redes, este descubrimiento hizo renacer las investigaciones en el área de las redes neuronales. Fue también a partir de ese año que estas herramientas comenzaron a salir de los laboratorios para integrarse al mundo comercial donde comenzaron a ser aplicadas para resolver una gran cantidad de problemas. Al mismo tiempo, una técnica llamada regresión logística aportó varios conocimientos para entender funciones complejas de muchas variables. Esto es importante porque tanto la regresión logística como la regresión lineal pueden ser representadas como casos particulares de una red neuronal. Las redes neuronales son usadas para resolver problemas con las siguientes características:

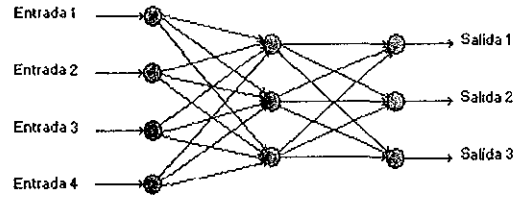
- *Los datos de entrada están bien definidos.-* Se tiene una idea clara de qué características son importantes en los datos aunque no se sepa exactamente como combinarlas.
- *Las salidas están bien definidas.-* Se conoce exactamente qué se está tratando de obtener o de predecir.
- *Existe experiencia disponible.-* Se cuenta con una gran cantidad de ejemplos donde tanto los datos de entrada como las salidas son conocidas Esta experiencia se utilizará para entrenar a la red.

Los pasos que se siguen para el proceso de construcción de un modelo para clasificación o predicción mediante el uso de redes neuronales son:

1. Identificar las características de entrada y de salida.
2. Convertir los valores que pueden tomar tanto las entradas como las salidas para que este valor esté entre 0 y 1.
3. Elegir una red con la topología adecuada.
4. Entrenar a la red mediante un conjunto representativo de ejemplos.
5. Probar a la red mediante un conjunto de datos de prueba estrictamente independiente del conjunto de datos de entrenamiento, si los resultados no son los deseados, repetir el entrenamiento.
6. Aplicar el modelo generado por la red para comenzar a predecir salidas a partir de datos de entrada diferentes.

Es importante señalar que una red neuronal se debe mantener actualizada ya que los modelos propuestos por la red no pueden reaccionar automáticamente a la aparición de nuevas variables, la desaparición de algunas de ellas o a algún cambio en la forma en que estas variables interactúan en el mundo real. Cuando esto sucede, la red debe ser sometida a un nuevo proceso de entrenamiento.

Las redes neuronales se forman a partir de unidades básicas llamadas neuronas, cada unidad tiene una o varias entradas que combina para dar como resultado un sólo valor de salida. Las unidades se interconectan unas con otras formando una red, de esta forma, las salidas de algunas neuronas son usadas como entradas para otras. Este concepto se representa en la siguiente figura:



La figura anterior representa una red neuronal *"feed-forward"*, esto significa que el flujo va en una sola dirección desde las entradas hacia las salidas, es decir, no hay ciclos en la red, las redes de este tipo son la más simples y las más usadas.

Al proceso en el cual una neurona combina las entradas y da como resultado un solo valor de salida se le llama *función de activación* de la unidad. La forma general de como trabaja la función de activación se basa en la forma en como trabajan las neuronas humanas, si el valor de las entradas se encuentra bajo, de tal forma que la combinación de las entradas no alcance cierto umbral, la salida también se mantiene baja, si por el contrario, la combinación de las entradas sobrepasa el umbral predefinido, el valor de la salida se eleva, activando a la neurona. Una característica importante de las neuronas computacionales que también existe en las neuronas humanas es el hecho de que presenten comportamientos no lineales, esto es, pequeños cambios en los valores de las entradas, cuando éstas están cerca del umbral, originan grandes cambios en la salida, por el contrario, grandes cambios en los valores de las entradas, cuando éstas se encuentran lejos del umbral, originan cambios muy pequeños en la salida de la neurona.

La función de activación tiene dos partes: la *función de combinación*, que es la encargada de agrupar los valores de todas las entradas en un solo valor y la *función de transferencia*, que es la encargada de llevar el valor entregado por la función de combinación a la salida de la unidad.

Aunque el caso ideal es que todas las unidades o neuronas de una red tuvieran una función de transferencia lineal, en la realidad la mayoría de estas neuronas tienen una función de transferencia sigmoidal, la fórmula de la función sigmoide es: Sigmoide $(x) = 1 / (1+ e^{-x})$, es decir, las neuronas tienen un comportamiento casi lineal para valores de entrada pequeños (entre 1 y -1) mientras que para valores grandes la función comienza a tener un comportamiento no lineal acercándose a sus valores límite: 0 y 1.

Dentro de la topología de las redes sin ciclos (*Feed-Forward*), que son las más usadas para predicción y clasificación, las unidades son organizadas generalmente en tres capas:

- *La capa de entrada.*- está conectada directamente con las entradas las cuales han sido modificadas de tal forma que sus valores siempre estén entre 0 y 1. Cada unidad es conectada a una fuente
- *La capa oculta.*- se llama así porque no está conectada ni a las entradas ni a las salidas de la red. Cada unidad en esta capa está totalmente conectada con las unidades de la capa de entrada. La forma en que estas unidades calculan su salida es multiplicando el valor de la entrada por su peso correspondiente y aplicando la función sigmoidal.
- *La capa de salida.*- por lo general consta de una sola neurona, sin embargo para los casos en que se desea obtener más de una respuesta se pueden tener tantas unidades como respuestas se deseen. Las unidades en esta capa se encuentran totalmente conectadas con las unidades de la capa oculta.

Un ejemplo de esta arquitectura se encuentra en la figura anterior. Existen varias formas de entrenar una red, la forma más utilizada es la de propagación hacia atrás de Hopfield, los pasos que se siguen para este tipo de entrenamientos son:

1. Cuando llega a la red un ejemplo de entrenamiento se obtiene su salida a partir de los pesos que se encuentran actualmente en la red.
2. Se calcula el error tomando la diferencia entre el valor calculado y el valor esperado.

3. El error es propagado hacia atrás en la red y los pesos son ajustados.

Una red neuronal también puede ser entrenada por medio de algoritmos genéticos, ésta forma será tratada más adelante, en la siguiente sección del presente capítulo.

Existen algunos puntos que tienen que ser tomados muy en cuenta para que una red neuronal pueda trabajar adecuadamente, uno de estos puntos, al igual que para la técnica de los árboles de decisión, debe ser el elegir con cuidado los datos que servirán como ejemplo y que, en el caso de las redes, servirá para entrenar a la red, para este caso deben considerarse el número de ejemplos que serán tomados en cuenta, el número de entradas en cada uno, el número de salidas esperadas, etc. Otro punto importante es la correcta preparación de los datos para que puedan servir como datos de entrada a la red neuronal, como ya se mencionó, los valores de cualquier tipo de dato de entrada deben ser convertidos de tal forma que siempre se encuentren entre 0 y 1, esto es relativamente fácil para valores continuos, sin embargo, para valores discretos ordenados y para datos categóricos (valores discretos no ordenados como el código postal o el sexo de una persona) se tiene que recurrir incluso a tablas de asignación de valores, de tal forma que arbitrariamente se asigne el valor a la variable, por ejemplo, se puede decidir asignar un valor de entrada de 0.5 al sexo masculino y 0.75 al femenino.

En muchos problemas de minería de datos, los datos caen en forma natural en series dependientes del tiempo o series de tiempo. En estos casos, las redes neuronales pueden ser fácilmente entrenadas para que, en base a resultados obtenidos en forma histórica las neuronas puedan predecir el siguiente resultado de la serie, un ejemplo de este tipo de aplicación es la que se usa para predecir la paridad de una moneda contra otra, donde pueden ser considerados valores históricos en un período de, por ejemplo, dos semanas y en base a estos datos se hace la predicción de qué valor tendrá la paridad al día siguiente.

Dado que el número de unidades en la capa oculta determina el número de patrones que la red puede reconocer, podría pensarse que entre más grande sea esta capa es mejor, sin embargo esto no es cierto, dado que una capa oculta muy grande puede redituar en un sobreentrenamiento de la red. Una red está sobreentrenada cuando memoriza el conjunto de entrenamiento, de tal forma que da excelentes resultados cuando se aplica un ejemplo de este conjunto pero falla cuando se aplica un dato desconocido. Una regla que se utiliza es no colocar en la capa oculta más de dos veces el número de unidades de la capa de entrada, generalmente lo que se hace es colocar el mismo número de unidades en ambas capas, si la red se sobreentrena, se quitan unidades a la capa oculta, si por el contrario, le falta entrenamiento, se agregan unidades a esta capa.

Dado que la forma en la que trabajan las redes neuronales internamente no nos permite conocer como fue obtenida alguna respuesta (en realidad se desconocen qué pesos se asocian a cada entrada), la forma de darnos una idea es haciendo un análisis de sensibilidad, esto es, medir cómo influyen las variaciones en cada una de las entradas en el resultado final, de tal forma que podamos conocer cuáles entradas tienen un mayor peso y de esta forma saber cuáles son los factores más importantes para la red.

Mapas auto-organizables

Estos mapas fueron inventados por el Dr. Tuevo Kohonen y originalmente fueron usados para imágenes y sonidos. sin embargo han sido de gran importancia para la Minería de Datos no dirigida ya que son capaces de detectar grupos en los datos.

La estructura de un mapa auto-organizable es similar a la de una red neuronal sin ciclos en el hecho de que las neuronas en su capa de entrada están conectadas directamente a las entradas y estas entradas tienen pesos asociados, sin embargo, no existe una capa oculta, las neuronas de la capa de salida se encuentran totalmente conectadas a las unidades de entrada, pero además se encuentran conectadas con sus vecinos adyacentes de tal manera que se forma una especie de "tablero de juego de mesa" en esta capa.

La forma de entrenar a este tipo de redes para el reconocimiento de grupos es que, cuando un elemento de entrenamiento entra a la red se elige un peso y la salida que tendrá ese dato, de tal manera que elementos con características similares siempre tendrán el mismo punto de salida. Cabe señalar que en la práctica, el número de grupos que se pueden detectar es menor que el número de salidas del mapa

Ventajas

- Pueden manejar una amplia gama de problemas
- Producen muy buenos resultados aún en dominios muy complejos
- Pueden manejar variables tanto continuas como discretas
- Se encuentran disponibles en muchos paquetes comerciales

Desventajas

- Requieren que todos sus valores de entrada se encuentren entre 0 y 1.
- No pueden "explicar" el porque de sus resultados
- Pueden converger prematuramente en una solución no apta dado que no hay forma de asegurar que el modelo generado por la red sea el más adecuado para esa situación

3.2.7 Algoritmos genéticos

Al igual que el razonamiento basado en la memoria y las redes neuronales, los algoritmos genéticos están basados en una analogía con los procesos biológicos. La evolución y la selección natural han dado como resultado, a través de millones de años, especies adaptables e individuos altamente integrados con su medio ambiente. Estos procesos tratan de optimizar las características de los individuos a través de generaciones propagando el material genético de los individuos más fuertes de una generación a la generación siguiente, los algoritmos genéticos aplican la misma idea a los problemas donde las soluciones pueden ser vistas como "individuos" y los problemas consisten en optimizar las mejores características de esos individuos, obviamente, los individuos más débiles y sus características genéticas no sobreviven.

Los algoritmos genéticos han sido usados recientemente en varias áreas, sin embargo, la más importante de todas ha sido el entrenamiento de redes neuronales. Los trabajos sobre algoritmos genéticos comenzaron a finales de los 50 cuando un grupo de matemáticos y biólogos trataron de modelar mecanismos genéticos en las primeras computadoras. Más tarde, a principios de los 60 John Holland y algunos científicos de la Universidad de Michigan aplicaron conceptos como cromosoma y gene para la optimización de funciones muy grandes. Fue hasta la década de los 70s cuando John Holland recibió la aprobación de su teoría ya que pudo sustentarla con importantes fundamentos teóricos.

En la minería de datos, los algoritmos genéticos no han tenido tanto desarrollo como algunas otras técnicas dado que la minería se enfoca a tareas como la predicción y la clasificación, no tanto a la optimización de funciones, sin embargo, muchos problemas de la minería de datos pueden ser modelados en forma de optimización.

Para entender la forma en que trabajan los algoritmos genéticos para la optimización de funciones se puede considerar el siguiente ejemplo: obtener el valor de p donde p puede tomar valores entre 0 y 31 para que la función $31p - p^2$ tenga el máximo valor.

Un algoritmo genético sigue los siguientes pasos:

1. Identifica el genoma y la función que evalúa la fortaleza de estos genomas, en este momento se crea una primera generación de genomas.
2. Modifica la población inicial aplicando selección, combinación y mutación.
3. Repite el paso 2 hasta que la fortaleza de la población ya no pueda ser mejorada.

En el ejemplo, los genomas se constituyen por cadenas de 5 bits dado que son 5 bits los que se necesitan para representar los números entre 0 y 31 y la función para evaluarlos es precisamente la función que se quiere maximizar. El algoritmo genera un cierto número de genomas iniciales, que para el ejemplo son cadenas de 5 bits, al azar y evalúa el desempeño de éstos dentro de la función. Una vez evaluados se lleva a cabo el primer proceso de selección donde los genomas más débiles, es decir los que lograron un menor valor de la función, son deshechados. A continuación se realiza una combinación, es decir, aleatoriamente se elige un número de bits de alguno de los genomas fuertes que reemplazará al mismo

número de bits en otro de los genomas mientras que los bits que fueron reemplazados en este último se combinan con los del primero, una vez hecho esto, se mide el desempeño tanto de los genomas que se habían elegido como los más fuertes como el de los genomas de la siguiente generación que fueron generados a partir de los primeros y el proceso se repite. En combinación con estos procesos se puede llevar a cabo alguna mutación, para los algoritmos genéticos la mutación consiste en cambiar arbitrariamente la información de alguno o algunos de los bits, ya sea de 0 a 1 o de 1 a 0 y evaluar el comportamiento de estos genomas en la función.

Para el caso del ejemplo, al cabo de dos o tres iteraciones y sin emplear mutaciones ya se tiene al genoma 10001 que representa un valor de 17 para p, este valor está muy cercano a los valores que en realidad puede tomar p de tal forma que el valor de $31p - p^2$ sea el máximo, que son 16 y 17.

Como se pudo ver en el ejemplo para que los problemas puedan ser analizados por medio de algoritmos genéticos tienen que poder ser presentados a través de cadenas de bits lo que lleva en muchas ocasiones a la necesidad de usar, al igual que para las redes neuronales, tablas de asignación donde se asignen arbitrariamente los valores. En este caso se tiene que tener mucho cuidado de dar un significado a todas las combinaciones de bits de tal forma que alguna combinación o alguna mutación no generen un genoma sin significado, es decir, si para representar el sexo en algún registro se opta por hacer la siguiente asignación: 00 - masculino, 01 - femenino y 10 - desconocido, es necesario definir el caso 11 aunque sea repitiendo el valor "desconocido" ya que si alguna combinación o mutación llevan al genoma a ese estado, éste no tendría significado. Una ventaja que ofrece todo esto es que los algoritmos genéticos tratan a los datos como cadenas de bits sin importar lo que éstas representen.

Uno de los aspectos más fuertes de los algoritmos genéticos es su habilidad para trabajar sobre *cajas negras*, esto es, problemas donde la función de evaluación se conoce pero los detalles de los cálculos no. Esto los hace ideales para trabajar en conjunto con las redes neuronales. La forma en que las redes neuronales y los algoritmos genéticos pueden trabajar conjuntamente se puede explicar fácilmente.

Como ya se dijo, las redes neuronales necesitan que ciertos pesos sean asignados a las variables de entrada y el entrenamiento de una red consiste en ir modificando estos pesos de tal manera que dado un conjunto de datos de entrada, estas entradas sean combinadas de tal forma que la salida sea la correcta. La forma en que interactúan estas dos técnicas es que estos pesos pueden ser representados en forma de cadenas de bits, es decir pueden ser representados por genomas dentro de los algoritmos genéticos, como esta interacción se da durante el proceso de entrenamiento de la red, la función que evalúa la fortaleza de los genomas es la diferencia entre el valor de la salida entregada por la red y el valor real de esa salida. De esta forma los pesos adecuados pueden ser encontrados a través de las técnicas de selección, combinación y mutación, y la red neuronal, en general, es entrenada a través de algoritmos genéticos.

Ventajas

- Producen resultados que pueden ser explicados
- Los resultados son fácilmente aplicables
- Son capaces de manejar una gran variedad de tipos de datos
- Se aplican para la optimización de funciones
- Se integran muy bien con las redes neuronales

Desventajas

- La forma en que se tienen que presentar los datos (cadenas de bits) llega a hacer difícil la representación de algunos problemas
- Garantizan un muy buen resultado, pero no en todos los casos, el resultado óptimo
- Requieren muchos recursos de cómputo
- Se encuentran disponibles en muy pocos paquetes comerciales, la mayoría de ellos, de redes neuronales.

3.2.8 Minería y almacenamiento de datos

Desde la introducción de las computadoras en los centros de procesamiento de datos la década de los años 60, prácticamente todos los sistemas operacionales de los negocios han sido automatizados. Estos sistemas tienen la característica de generar grandes cantidades de datos que son almacenados y están disponibles, sin embargo, los datos están disponibles pero no la información. Esta es la meta del "almacenamiento de datos" o *data warehousing*, como ya se dijo anteriormente, agrupar los datos de una organización que normalmente se encuentran dispersos con el propósito de que la información obtenida de ellos pueda ser usada como apoyo a los procesos de toma de decisiones. La toma de decisiones puede hacerse de muchas maneras, desde la manera tradicional, a partir de reportes hasta el complejo modelado de un proceso a través de redes neuronales para determinar, por ejemplo, a qué clientes se debe ofrecer un producto.

La minería de datos, por su parte, busca encontrar patrones que tengan un significado dentro de los datos, para lograr su propósito, los datos deben encontrarse en forma "limpia" y "consistente" y en el formato requerido por las herramientas de la minería de datos. El esfuerzo de identificar, adquirir y adecuar los datos, generalmente se realiza en forma adicional a la aplicación de minería de datos propiamente dicha, sin embargo, cuando se dispone de un almacén de datos bien diseñado, éste puede proporcionar los datos necesarios para el análisis mediante las técnicas de minería. Mejor aún, si el almacén de datos fue diseñado con el propósito de interactuar con aplicaciones de minería de datos, entonces los esfuerzos del primero serán encaminados a obtener los datos que requieren esas aplicaciones.

Los sistemas manejadores de bases de datos relacionales, que conforman el corazón de la mayoría de los *data warehouses*, proporcionan cada vez mayor número de herramientas avanzadas de acceso a los datos además del SQL, el ejemplo más claro es la inclusión de herramientas de procesamiento analítico en línea (OLAP).

Tan útiles como son, los almacenes de datos no son un prerequisito para la minería o el análisis de datos dado que los estadistas y analisitas han usado paquetes estadísticos desde hace mucho tiempo con buenos resultados sin contar con almacenes de datos como tales, sin embargo, dependiendo de la cantidad de datos que se maneje, los almacenes de datos resultan una necesidad imperiosa para cualquier tipo de toma de decisiones o análisis de información.

La justificación y características de los almacenes de datos fue tratada en una sección previa dentro de este mismo capítulo.

La minería de datos juega un papel muy importante en el entorno de los almacenes de datos. El resultado inicial de un almacén viene del hecho de automatizar algunos procesos existentes y es el hecho de colocar reportes en línea y servir a otras aplicaciones como una fuente de datos "limpios", sin embargo, el resultado más importante es que un mejor acceso a los datos puede inspirar innovación y creatividad. Este es el papel de la minería de datos ya que proporciona herramientas para un mejor entendimiento e inspira creatividad basada en la observación de los datos. La forma en que estas dos tecnologías se complementan se describe en los siguientes puntos:

- *Grandes cantidades de datos.*- Para que una herramienta de minería de datos pueda resolver una pregunta como ¿Cuál será el próximo producto que este cliente querrá comprar?, debe tener una gran cantidad de información acerca de productos que ese cliente ha comprado anteriormente. Este nivel de detalle para cantidades inimaginables de datos únicamente puede ser proporcionado por un almacén de datos bien diseñado.
- *Datos limpios y consistentes.*- Los algoritmos de minería de datos generalmente trabajan sobre varios gigabytes de información obtenidos de distintas fuentes. Generalmente el 80% del tiempo necesario para obtener un resultado se gasta en reunir y adecuar los datos cuando no se tiene a la mano un almacén de datos, además el proceso se puede volver muy lento dado que las herramientas que encuentran patrones en base a procesos iterativos requieren estar tomando cada vez distintos elementos de datos. Un almacén de datos prácticamente resuelve estos problemas.
- *Generación de hipótesis y análisis de resultados.*- La cantidad de datos manejada por un almacén, además de la forma en que estos datos son guardados, permite tener la suficiente cantidad de

información para generar y probar hipótesis surgidas en las herramientas de minería de datos, estas hipótesis son del tipo ¿Realmente los productos en colores tropicales se venden más en la costa? o ¿En verdad la gente prefiere hacer llamadas de larga distancia después de cenar?. El análisis de resultados por su parte permite a las empresas aprender de sus éxitos y fracasos.

- *Sistemas escalables* - El complemento entre estas dos técnicas se da también a nivel de los sistemas, dado que el mismo hardware que hace posible el almacenamiento y el hacer consultas a bases de datos muy grandes generalmente también hace posible el correr algoritmos avanzados de análisis sobre grandes cantidades de datos. Por otra parte, las limitantes propias del lenguaje SQL se han venido superando mediante la incorporación de nuevas herramientas a los manejadores de bases de datos relacionales, por ejemplo, las herramientas de procesamiento analítico en línea.

CAPÍTULO IV

EJEMPLO DE APLICACIÓN

4.1 PROBLEMÁTICA

La idea de desarrollar un sistema de minería de datos surge en el Laboratorio de Redes, el cual pertenece al Departamento de Modelación Matemática en Sistemas Sociales del Instituto de Investigaciones Aplicadas y en Sistemas (IIMAS) de la UNAM.

Actualmente uno de los proyectos más importantes del Departamento es el "Modelado de la Red Política de México", la cual consta de la biografía de 5452 personajes políticos mexicanos, de los cuales se obtienen grupos cuya característica o características similares, en un momento determinado, son de interés para el estudio de la red.

La finalidad del proyecto es explicar o representar mediante ecuaciones obtenidas como resultado de aplicar los conceptos sobre teoría de gráficas a estos grupos, en un momento determinado o a través del tiempo, siendo estos datos analizados por un especialista en la materia con el fin de determinar la influencia de estos modelos en la vida política y social de México.

Una vez que se han obtenido los grupos de interés y las relaciones entre los individuos de estos grupos, las cuales son representadas en forma de matrices, comienza la fase de aplicar la teoría de gráficas de tal suerte que se obtengan los modelos de esos grupos, sin embargo, la forma en que se obtienen estos grupos y sus relaciones no es fácil debido principalmente a dos factores:

- *El tamaño y la forma en que está diseñada la base de datos.*- Actualmente la base de datos se encuentra disponible en tres formatos intercambiables entre sí: Access (donde actualmente se maneja el sistema), Sybase (para plataforma Unix) y en modo texto. Consta de 5454 registros de personajes políticos agrupados en 17 tablas, cada una de las cuales representa una actividad o característica de un funcionario público.
  Originalmente los datos que contiene la base de datos se encuentran en la serie de libros "Diccionario biográfico del Gobierno Mexicano", las versiones más recientes cuentan con una versión en CD-ROM, una de estas versiones fue modificada de tal manera que tuviera el formato de una base de datos en modo texto, sin embargo esto ocasionó que se presentara una falta de normalización en la base de datos ya que originó campos descriptivos muy grandes en cada actividad, como se muestra en el siguiente ejemplo:

Id_fp	Nombre	Generales	Padres	Cónyuge
2000	REYES COLIN, PEDRO FERNANDO	NACIO EN MEXICO, DF, EL 17 DE ABR DE , 1950	HIJO DE JORGE REYES TAYABAS, LIC. EN DERECHO, Y DE BERTHA EUGENIA COLIN CASSAIGNE.	MARIA STELLA SEEGROVE CUSPINERA.

Ejemplo de un registro correspondiente a un funcionario en la tabla "*Generales*"

Id_fp	Actividad	Año_ini	Año_fin
2000	SECRETARIO GENERAL DE ACUERDOS, TRIBUNAL SUPERIOR DE JUSTICIA, DF	1975	1977
2000	SEGUNDO SECRETARIO, JUZGADO PRIMERO DE DISTRITO EN MATERIA ADMINISTRATIVA, DF,	1977	1978
2000	SECRETARIO DE ESTUDIO Y CUENTA, SCJN,	1978	1981
2000	JUEZ PRIMERO DE DISTRITO EN HERMOSILLO, SON,	1981	1986
2000	MAGISTRADO EN EL PRIMER TRIBUNAL COLEGIADO DEL DECIMOQUINTO CIRCUITO, MEXICALI, BC, DE	1986	9999

Ejemplo de los registros correspondientes a un funcionario en la tabla "*Cargos en el gobierno*".

- *La falta de flexibilidad para elegir las características de interés.*- La herramienta que permite obtener grupos con las mismas características fue desarrollada en Access, estas características son dictadas por el usuario de acuerdo a la experiencia que ha tenido al formar grupos en ocasiones anteriores. Es decir, tiene que reconocer cuáles características pueden ser comunes a un grupo de individuos, pudiendo quedar fuera algunas características igualmente importantes, pero de las cuáles se desconocía su utilidad en el grupo.

Estos dos factores, aunados al surgimiento de técnicas como el "Almacenamiento y la Minería de Datos", influyeron en la decisión del Departamento de Modelación Matemática en Sistemas Sociales de hacer pruebas en el sistema basadas en las técnicas de minería de datos. Estas pruebas debían estar orientadas a mejorar la forma en que actualmente se lleva a cabo la elección de los grupos de interés.

El desarrollo de los sistemas de prueba consistió de las siguientes fases.

4.2 ANÁLISIS

Antes de comenzar con el desarrollo del proyecto, fue necesario realizar una labor de investigación acerca de las técnicas y requerimientos que conlleva la implantación de un sistema de almacenamiento y minería de datos con el fin de determinar la factibilidad de implantar alguna de estas tecnologías en el proyecto del laboratorio. Cabe señalar que la información disponible al inicio de la investigación era escasa pero fue aumentando con la misma proporción que el avance en las tecnologías, además su estudio no forma parte de ninguna de las materias de nivel Licenciatura para la carrera de Ingeniero en Computación.

Como se observó en el capítulo anterior, una de las características de los sistemas de almacenamiento de datos es la capacidad de integrar datos históricos de múltiples fuentes y sistemas operacionales, ésta característica no justifica el desarrollo de un sistema de almacenamiento de datos para el proyecto de la Red Política de México, dado que la información se encuentra almacenada en una sola base de datos y no es necesario exportar datos de ninguna otra fuente. Esto, aunado a los requerimientos de hardware y software, además del costo de cualquier sistema de almacenamiento de datos comercial nos llevó a la decisión de descartar el desarrollo de un almacén de datos.

Por otro lado, el estudio de las técnicas de minería de datos realizada en el capítulo anterior sustentó que era factible implantar un sistema de minería de datos que trabajara sobre el sistema actual. Inicialmente se contemplaron dos opciones, la primera consistía en adaptar un sistema o alguna herramienta de minería de datos de propósito general, la segunda contemplaba el desarrollo del sistema en el Laboratorio con las características exigidas por el proyecto.

Con el fin de evaluar la primera opción se obtuvieron de Internet algunas demostraciones para analizar su funcionamiento y adaptabilidad al sistema. Las aplicaciones que se encontraron disponibles en el momento de iniciar el proyecto fueron "Alice" de Isoft y "SuperQuery Office" de Azmy Thinkware, Inc. Ambas resultaron ser herramientas de "escritorio", es decir, fueron pensadas para ser usadas por usuarios finales en su escritorio sin necesidad de que éstos tuvieran conocimientos sobre minería de datos, además de que pueden obtener los datos de tablas con formato de paquetes comerciales para plataforma PC como Excel, Access, SPSS, Foxpro y Paradox para el caso de Alice y Excel y Access para SuperQuery Office. Al no requerir de conocimientos previos sobre minería de datos, los fabricantes no incluyen en estas aplicaciones algún tipo de lenguaje para programación de tal forma que se permita modificar o agregar funcionalidad a las mismas. Otra desventaja que se apreció en el funcionamiento de estas aplicaciones fue que operan

sólo sobre una tabla a la vez y no son capaces de seguir las relaciones dictadas por una base de datos relacional. Otro punto importante es que se encuentran enfocadas al área de Mercadotecnia de las empresas, lo cual origina que sólo reconozcan tipos de datos numéricos, de fecha y booleanos, obviamente al no contemplar los campos de tipo texto no resulta eficiente usarlas sobre el sistema actual ya que la mayoría de los campos son de tipo texto

Las características expuestas anteriormente hicieron que la primera opción fuera desechada, con lo que se prosiguió al análisis de la segunda opción.

Las aplicaciones que puede tener la minería de datos en el sistema actual pueden ser muchas, por este motivo se decidió limitar el problema de tal forma que el sistema de prueba se ajustara a los siguientes requerimientos

- ✓ El costo del proyecto debe ser mínimo.
- ✓ El manejo de la base de datos debe hacerse en Sybase para UNIX
- ✓ El sistema debe tener la mayor disponibilidad posible.
- ✓ Debe facilitar la obtención de grupos, identificando automáticamente características similares.
- ✓ Debe explotar las relaciones familiares entre individuos, las cuales no se han explotado hasta el momento.
- ✓ Los resultados deben presentarse de una manera comprensible para el usuario.

Propuesta

Una vez establecidos los requerimientos para el desarrollo del sistema, se hizo la siguiente propuesta. Se realizará una aplicación que se conforme de dos programas que apliquen técnicas de minería de datos, uno se orientará a la obtención automática de características similares y el otro se enfocará a la explotación de relaciones familiares. Para facilitar la disponibilidad el sistema será basado en la arquitectura cliente/servidor y podrá ser accedido mediante reglas de seguridad a través de Internet. Como interfaz gráfica se utilizará la WWW y la base de datos será accedida desde este medio a través de programas CGI.

4.3 DISEÑO

4.3.1 Diagramas de flujo

De acuerdo a la propuesta, el sistema completo constará de dos aplicaciones las cuales se basarán en forma general en los siguientes diagramas de flujo.

A) Diagrama de flujo del programa encargado de obtener relaciones familiares. (Fig. 4.1)
B) Diagrama de flujo del programa que detecta automáticamente características similares. (Fig. 4.2)

Diagrama de Flujo Programa Relaciones Familiares

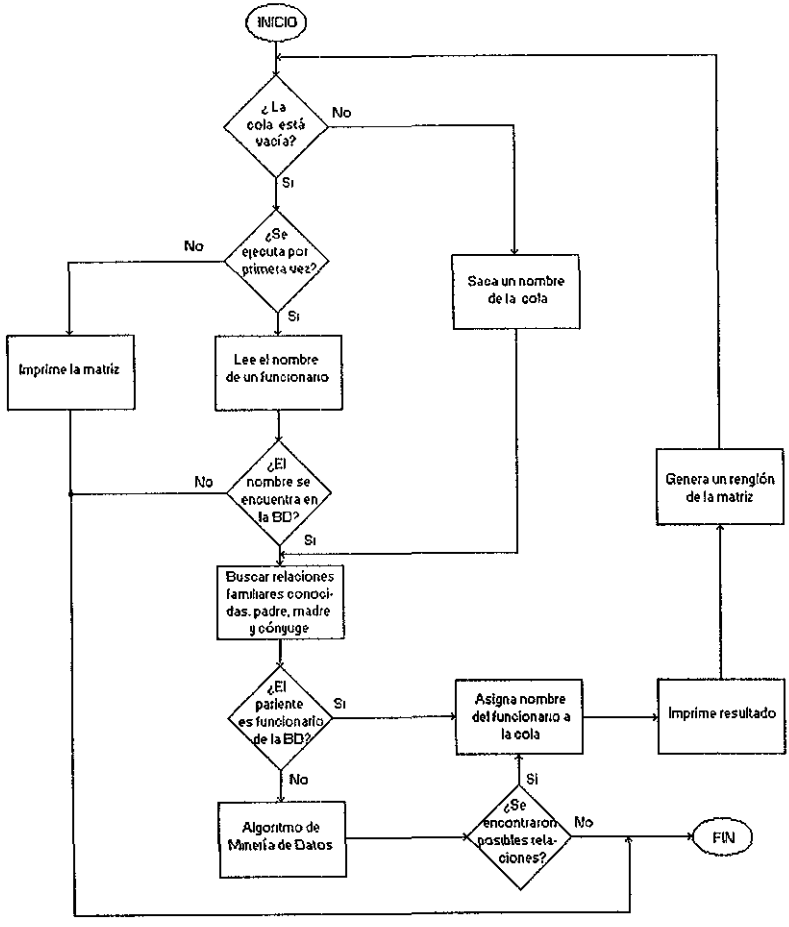


Fig. 4.1 *Diagrama de flujo del programa encargado de obtener relaciones familiares*

Diagrama de Flujo Programa Características Similares

Fig. 4.2 Diagrama de flujo del programa encargado de obtener características similares para un grupo de personas.

4.3.2 Selección de técnicas de minería de datos

En cada uno de los diagramas presentados anteriormente se observa la presencia de un proceso de minería de datos, estos procesos representan la parte del programa donde se debe implantar alguna o algunas técnicas de minería de datos encargadas de resolver el problema en cuestión. La naturaleza de los problemas que se busca solucionar es distinta, por lo tanto fue necesario elegir la técnica de minería de datos que se adecuara más a cada uno de ellos.

En el caso del programa encargado de obtener relaciones familiares. la técnica de minería de datos que se adapta en forma natural a este tipo de problemas es la clasificación por medio de Árboles de decisión. El algoritmo que se utilizó para la creación del árbol de decisión fue el Algoritmo CART explicado en el capítulo anterior, es decir, el árbol resultante es un árbol binario cuyas variables de evaluación son:

Variable	Abreviatura
Apellido paterno del funcionario elegido	AP1
Apellido materno del funcionario elegido	AM1
Apellido paterno del padre del funcionario elegido	APP1
Apellido materno del padre del funcionario elegido	AMP1
Nombre del padre del funcionario elegido	NP1
Apellido paterno de la madre del funcionario elegido	APM1
Apellido materno de la madre del funcionario elegido	AMM1
Nombre de la madre del funcionario elegido	NM1
Apellido paterno del cónyuge del funcionario elegido	APC1
Apellido materno del cónyuge del funcionario elegido	AMC1
Apellido paterno de cualquier funcionario en la BD	AP2
Apellido materno de cualquier funcionario en la BD	AM2
Apellido paterno del padre de cualquier funcionario en la BD	APP2
Apellido materno del padre de cualquier funcionario en la BD	AMP2
Nombre del padre de cualquier funcionario en la BD	NP2
Apellido paterno de la madre de cualquier funcionario en la BD	APM2
Apellido materno de la madre de cualquier funcionario en la BD	AMM2
Nombre de la madre de cualquier funcionario en la BD	NM2
Apellido paterno del cónyuge de cualquier funcionario en la BD	APC2
Apellido materno del cónyuge de cualquier funcionario en la BD	AMC2

De acuerdo al algoritmo, en la raíz del árbol debe evaluarse la variable que tenga la menor variabilidad, es decir, la que asegure que el crecimiento del árbol será óptimo. En este caso se encontraron varias condiciones que hacían que el árbol se comportara de la misma manera, de éstas se eligió una, con la cual se obtuvo el siguiente árbol. (Fig. 4.3)

Para probar éste árbol se generó un conjunto de registros de prueba que consistían en los nombres de los integrantes de una familia, de la cual se conocían sus relaciones familiares. El resultado obtenido de la prueba fue que el 100% de los registros fueron clasificados correctamente. Cabe señalar que para aplicar esta técnica durante la fase de desarrollo se requerirá de la normalización sobre la tabla que contiene la información empleada, de tal forma que las variables arriba mencionadas sean identificadas fácilmente en la base de datos.

Árbol de decisión

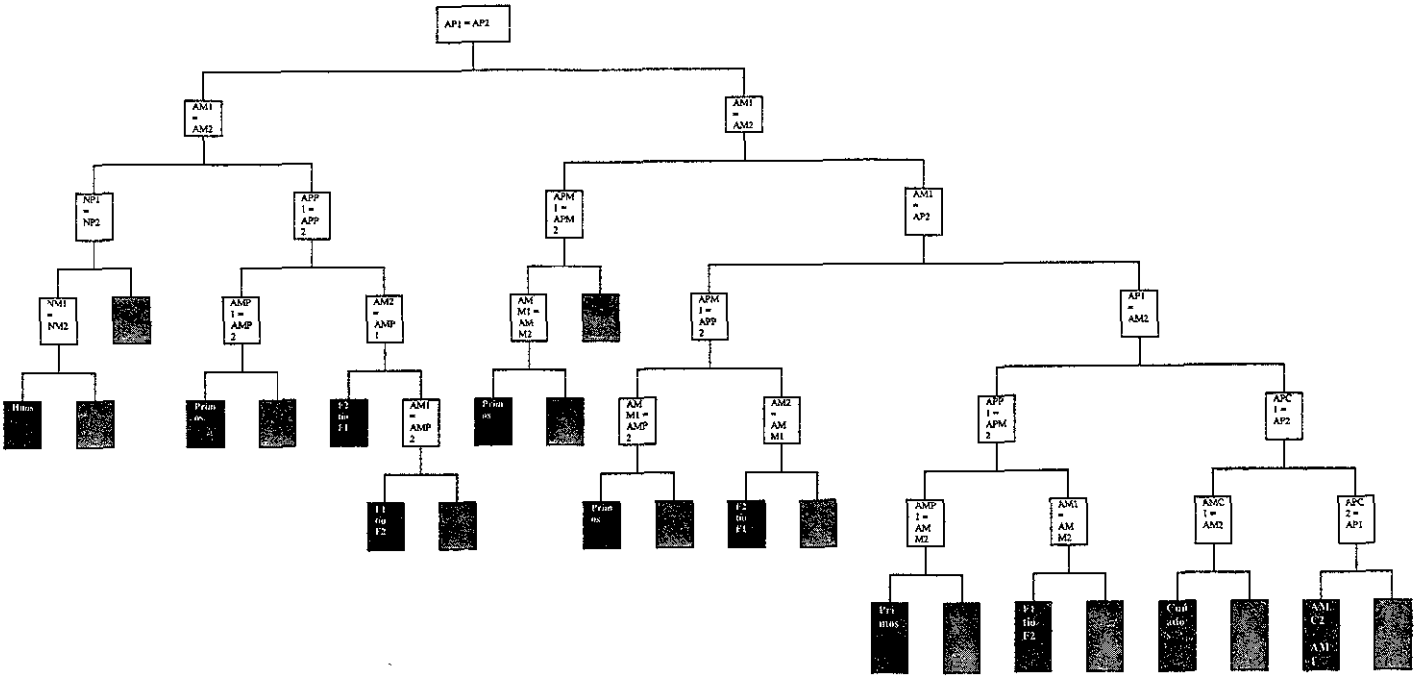


Fig. 4.3. Árbol de decisión.

Para el programa encargado de encontrar automáticamente características similares, no se encontró una técnica que se aplicara en forma tan transparente como en el caso anterior, por lo tanto fue necesario elegir aquella que pudiera ser modificada para que se acercara a los requerimientos de este caso. La técnica elegida fue el Análisis del carrito del supermercado, la cual en su forma original permite conocer el comportamiento de los clientes al comprar en un supermercado, sin considerar la identidad del cliente. Si consideramos a las características de un funcionario como artículos que éste va depositando en un carrito de supermercado y ponemos atención en la identidad de éste funcionario, al final del análisis podemos saber qué funcionarios reúnen las mismas características. Para efectos del proyecto se logra un resultado cuando podemos decir cuántos y cuáles son los funcionarios que cumplen con una cierta característica, sin embargo, tal como se hace en la técnica del Análisis del carrito del supermercado, se podrían generar reglas de comportamiento, en este caso éstas reglas no nos indicarían que productos se deben de colocar cerca de otros sino, por ejemplo, qué características debe poseer un funcionario para ocupar un cierto puesto.

4.3.3 Selección de hardware y software

Debido a que se requería trabajar con un costo mínimo para la implantación del sistema y dado que en el laboratorio se contaba con una versión de Sybase para HP-UX, se decidió utilizar la estación de trabajo HP 9000 del laboratorio, sin embargo este equipo contaba con una versión de software para la cual no se pudo encontrar en Internet un compilador de C, por lo tanto se decidió iniciar en una estación de trabajo Sun Sparc4.

Como se ha mencionado, el manejador de base de datos elegido, dado que ya se contaba con él, fue Sybase para plataforma Unix de esta manera se comenzaron a buscar opciones de software que fueran compatibles con esta plataforma. Por cuestiones de desempeño se decidió colocar un servidor de Web en la misma estación de trabajo que contenía al servidor de base de datos, las opciones que se presentaron en este caso fueron, el servidor Web de NCSA, el "*FastTrack Server*" de Netscape y el servidor "*Apache*" de "*Apache HTTP Server Project*". Los aspectos que se consideraron para la elección del servidor de Web son los siguientes: desempeño, cantidad de recusos utilizados (espacio en disco y memoria) y facilidad de administración. El que cumplió con todas las características fue "Apache".

Para la creación de los programas CGIs encargados tanto de la ejecución de los algoritmos de minería de datos como de la interacción entre el servidor de Web y la base de datos, se eligió el lenguaje de programación C por los siguientes motivos:

- Debido a que Unix fue programado en C, la ejecución de uno de estos programas resulta más eficiente que la ejecución de un programa en cualquier otro lenguaje.
- Los compiladores de C para Unix se encuentran en forma gratuita en Internet.
- Fue posible conseguir, también en forma gratuita, las bibliotecas de C conocidas como "*DBLibraries*" que permiten a un programa hecho en C interactuar con una base de datos en *Sybase*.

La validación de las formas encargadas de ejecutar a los programas CGI, fue pensada para que se ejecutara en el cliente, por lo tanto se recurrirá a la programación en *JavaScript*, que es un lenguaje interpretado, orientado a objetos, gratuito y que se inserta en el código de una página HTML permitiendo, entre otras cosas, la validación de los elementos de una forma.

Al insertar código *JavaScript* en las páginas HTML se limitó el acceso a estas páginas a versiones 3.0 o superiores de los clientes de WWW "*Netscape Navigator*" e "*Internet Explorer*".

4.4 DESARROLLO E IMPLANTACIÓN

Después de haber seleccionado cuidadosamente el software para el desarrollo de las aplicaciones, se prosiguió con la elaboración de los programas.

A manera de presentación del sistema se creó la página "aplicaciones.html" que se muestra en la figura 4.4 y que tiene la finalidad de que el usuario pueda elegir de manera sencilla con cuál de las aplicaciones desea trabajar.

El funcionamiento del programa encargado de la obtención de relaciones familiares entre funcionarios es el siguiente:

- El nombre del funcionario sobre el cuál se desea encontrar todas relaciones familiares existentes se introduce en el campo de texto de la forma de envío que se encuentra en la página "matriz.html", la cual se muestra en la figura 4.5.
- Al ser enviada, la forma ejecuta el programa CGI llamado "matriz.cgi", el cual sigue los siguientes pasos:
  1. Toma el nombre introducido por el usuario y lo busca en la tabla de la base de datos que contiene los nombres de los funcionarios, no importando el orden en que haya sido escrito el nombre del funcionario (*Nombre Apellido_Paterno Apellido_Materno, Apellido_Paterno Nombre, Apellido_Materno Nombre*, etc.).
  2. Si el nombre introducido se ha encontrado una sola vez en la base de datos, el programa continuará con su ejecución normal, en el caso de existir dos o más coincidencias el programa generará una página HTML en donde el usuario elegirá de entre los nombres coincidentes aquél que corresponda al que pretendía buscar en la consulta inicial.
  3. Si el nombre no corresponde al de ningún funcionario en la base de datos se generará una página de error.
  4. Una vez que se tiene el nombre del funcionario sobre el que se va a trabajar, se obtienen de la base de datos sus relaciones familiares conocidas, padre, madre y cónyuge, ya que éstos son datos que se tienen en dicha base de datos, éstos datos son almacenados en memoria.
  5. Realiza la búsqueda del nombres del padre, de la madre y del cónyuge en la tabla de funcionarios de la base de datos. En caso de encontrarse en la tabla, el nombre, el identificador del funcionario, la relación y la distancia que guarda con el funcionario inicial son insertados en un arreglo del cual se obtendrá mas adelante la matriz.
  6. Ejecuta la implantación del algoritmo de minería de datos para encontrar las relaciones familiares no conocidas tales como, tíos, sobrinos, abuelos, primos (maternos y paternos), hijos, hermanos, cuñados y suegros. Las relaciones encontradas son válidas cuando el nombre del pariente pertenezca a la tabla de funcionarios, sólo entonces es agregado al arreglo de relaciones familiares.
  7. Para cada uno de los funcionarios que ingrese al arreglo mencionado, se repite el procedimiento a partir del paso 5.
  8. Cuando ya no quede ningún funcionario por buscar, se genera la matriz de relaciones y se almacena en un archivo de texto.
  9. Finalmente, se despliega en una página HTML una tabla que contiene las relaciones familiares encontradas para el funcionario en cuestión, así como la matriz de relaciones.

El modo de operación del programa encargado de encontrar características similares en un grupo de personas es el siguiente.

- Para iniciar el proceso de detección de características similares, se debe elegir a un grupo de personas que cumplan con una característica indicada por el usuario, la cual se hace directamente sobre la base de datos y debe ser definida en el área de texto correspondiente en

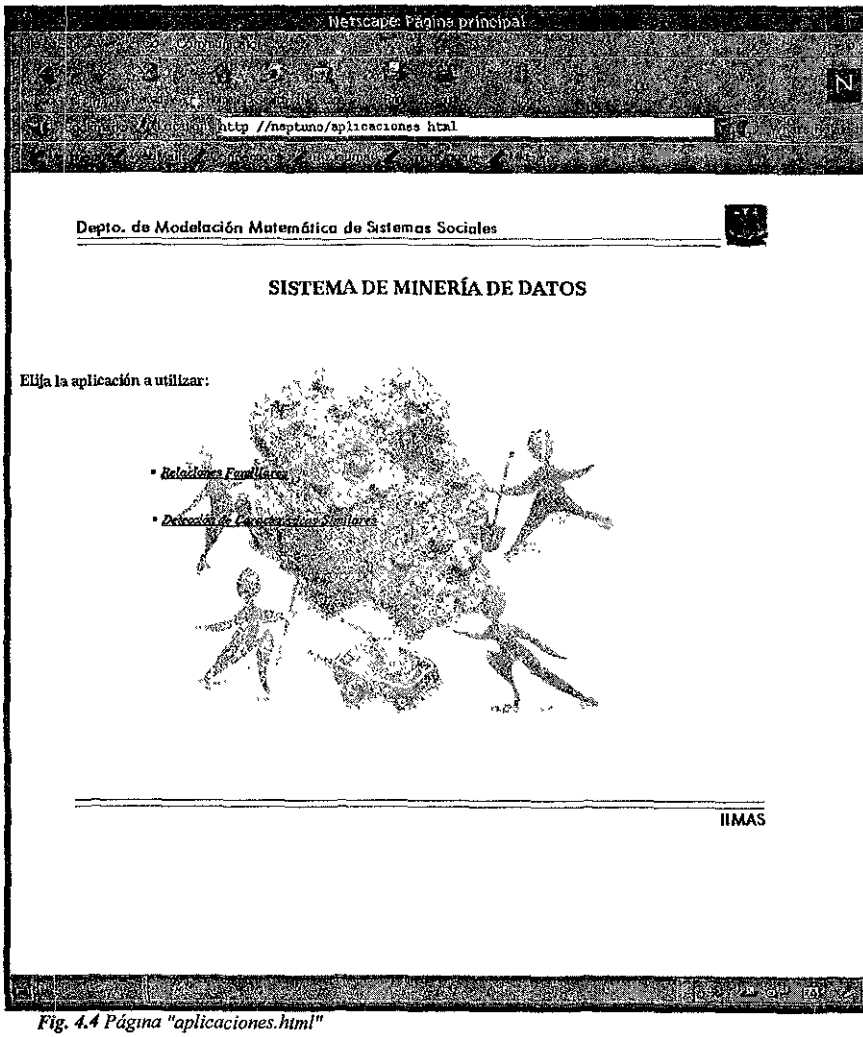


Fig. 4.4 Página "aplicaciones.html"

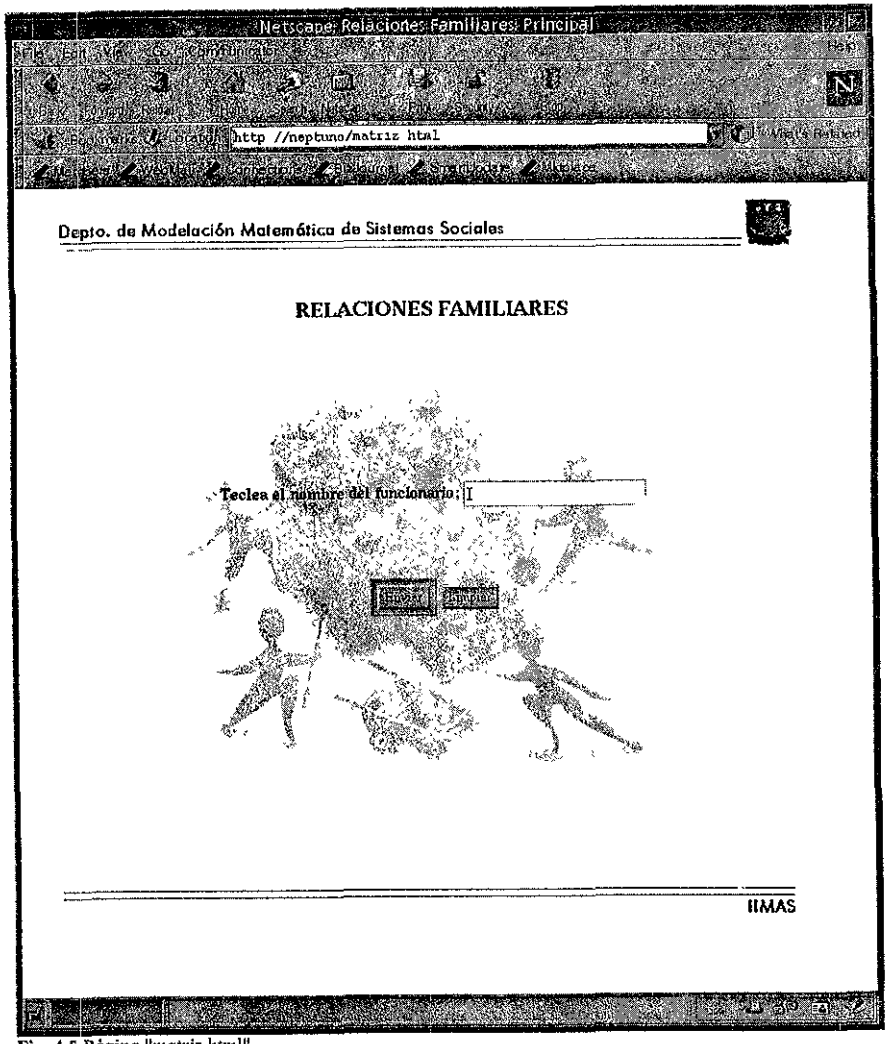


Fig. 4.5 Página "matriz.html"

la página "analisis.html", la cual se muestra en la figura 4.6. La consulta debe seguir la estructura de una expresión lógica, para lo cual se ponen a disposición del usuario una serie de botones que le ayuden a estructurar la expresión de tal forma que sea entendible por el programa. La sintaxis de la expresión se valida mediante un programa escrito en *JavaScript*, que se encuentra inmerso dentro del código HTML de dicha página.

- Una vez que se ha definido la consulta, es necesario indicar sobre qué tablas se realizará. En este punto, mediante programación en *JavaScript*, el usuario tiene dos opciones, la primera se encuentra seleccionada desde el inicio de la consulta y selecciona automáticamente todas las tablas de la lista, pero no permite al usuario desactivar alguna de ellas, la segunda es elegir las tablas sobre las cuales desea que se realice la consulta La figura 4.6 muestra la página HTML en la que se inserta la consulta y se eligen las tablas.
- La integridad de esta forma también se verifica mediante *JavaScript* al momento de ser enviada, en caso de no presentarse ningún error se ejecuta el programa "analisis.cgi" que sigue los siguientes pasos:

1. La consulta que ha sido definida en la página HTML en forma de expresión lógica, es la variable de entrada al programa cgi, el cual la adapta a una consulta entendible por SQL.
2. Se ejecuta la consulta anterior (incluyendo el período) y se almacenan en memoria los identificadores de los funcionarios que cumplieron con ella.
3. Con los datos obtenidos se efectúa una nueva consulta que obtiene toda la información de dichos funcionarios en las tablas elegidas, considerando para ello el período indicado.
4. La información devuelta por esta consulta es clasificada y separada en tres archivos que corresponden al período en que fue realizada la actividad devuelta en ese instante con respecto al período original de consulta, pudiendo haber sido realizada en un período previo, posterior o durante el mismo. Los archivos reciben los nombres de res_pre (actividades realizadas durante el período original), res_pas (actividades realizadas antes del período original) y res_fut (actividades realizadas en un período posterior al original), y contienen una línea por cada registro devuelto por la consulta. Cabe señalar que en caso de no haberse especificado un período sobre el cual realizar la primera consulta, el programa calculará este tiempo en forma independiente para cada funcionario tomando en cuenta los años de inicio y de fin de la actividad consultada.
5. Utilizando la misma información se crean otros tres archivos con las mismas características de los anteriores, con la diferencia de que todos los registros devueltos para un mismo funcionario son concatenados y se escriben en cada archivo como una sola línea. Estos archivos reciben el nombre de linea_pre, linea_pas y linea_fut.
6. Para cada una de las líneas contenidas en los archivos res_pas, res_pre y res_fut, se hace una comparación con cada una de las líneas en el archivo *linea* correspondiente (linea_pre, linea_pas y linea_fut). Esta comparación sigue el siguiente algoritmo para cada par de archivos:
    - Se toma la primer palabra de la línea del archivo "res" y mediante la utilería "grep" de Unix, se obtiene cuantas y cuales líneas del archivo "linea" tienen alguna coincidencia con esa palabra. Esto se almacena en un archivo temporal.
    - Se toma la siguiente palabra de la línea del archivo "res" y se sigue el procedimiento anterior.
    - Se concatenan la primera y la segunda palabra y ahora se hace la búsqueda de ésta cadena en el archivo "linea".
    - Se toma la tercera palabra del archivo "res" y se busca mediante "grep" en el archivo "linea".
    - Se concatenan la primera, la segunda y la tercera palabra de la línea del archivo "res" y

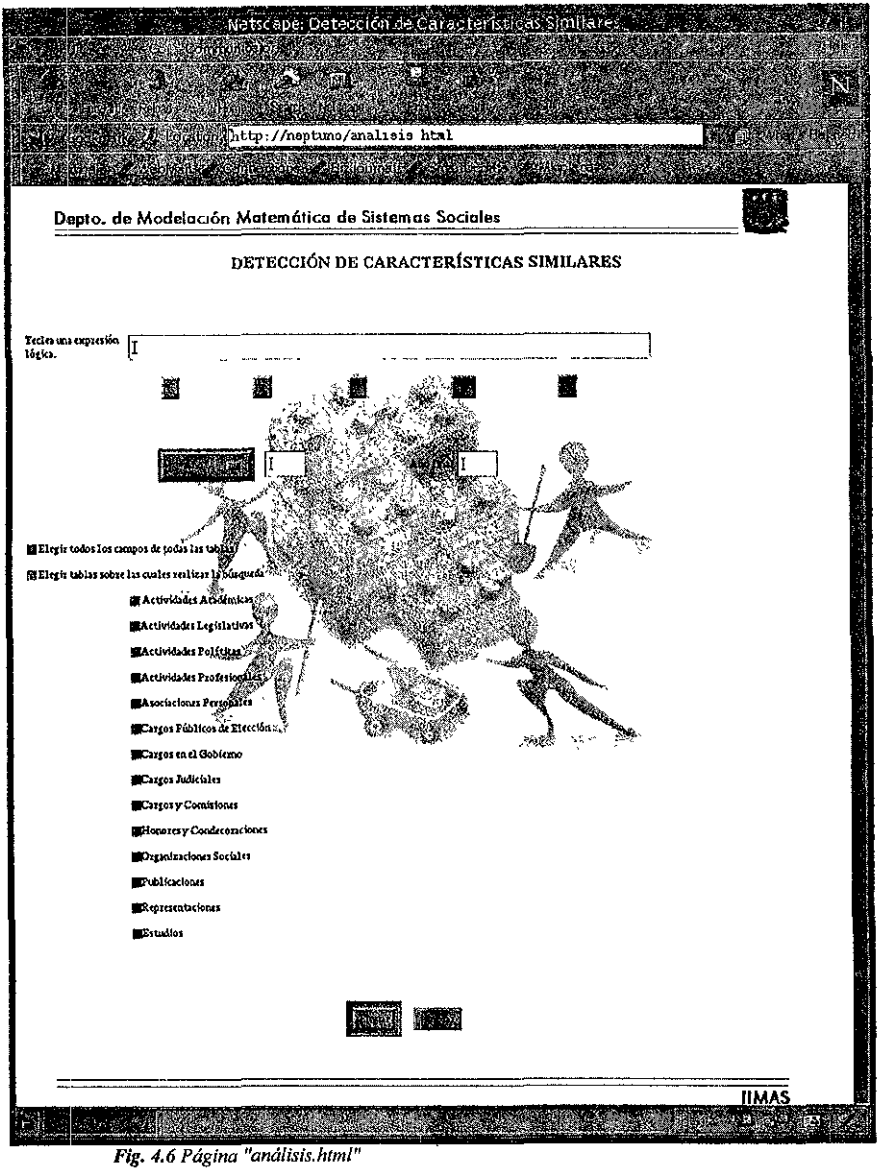


Fig. 4.6 Página "análisis.html"

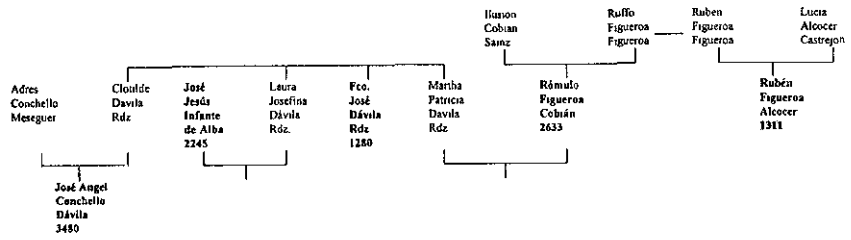
se hace la búsqueda de ésta cadena en el archivo "linea".
- Este procedimiento se repite para cada palabra en la línea y para todas las líneas del archivo "res".

7. Al final del proceso anterior se tiene la información de cuáles funcionarios cumplen con un determinado patrón, divididos en tres archivos que representan los períodos de tiempo antes citados. De estos archivos se eliminan las características que resultaron ser únicas para un funcionario, es decir, aquellas que tuvieron una sola coincidencia, por no considerarse representativas.
8. Se eliminan palabras que se encuentran solas y que por sí mismas no se considera que proporcionen información útil.
9. Después de esta depuración los archivos resultantes se almacenan en memoria para facilitar su manejo al momento de imprimir los resultados.
10. El despliegue de los resultados se realiza a través de una página HTML dividida en dos partes, la primera muestra cuáles fueron los funcionarios que coincidieron con la consulta inicial, en la segunda se muestran en forma tabular y gráfica las características similares encontradas para esos funcionarios en cada uno de los períodos. Para los resultados en forma tabular se muestran únicamente aquellas características que tuvieron 25% o más de coincidencias en los funcionarios encontrados, para el caso gráfico, únicamente se muestran un máximo de cinco características (las más significativas), considerando nuevamente al 25% de coincidencia como el límite inferior.

4.5 RESULTADOS

Una vez implantados los programas se pasó a una fase de pruebas y obtención de los primeros resultados mediante la ejecución de los mismos. A continuación se presentan una serie de imágenes obtenidas como resultado de una corrida para cada uno de los programas, en ellas se trata de mostrar los diferentes casos que se pueden tener durante su ejecución normal.

Para el caso del programa encargado de obtener las relaciones familiares de un funcionario se eligió al funcionario ***Rubén Figueroa***, la forma en que se introdujo su nombre como entrada para el programa se muestra en la figura 4.7. El nombre de esta persona forma parte de la base de datos por lo tanto no se genera ningún error. Los resultados obtenidos se muestran en las figuras 4.8 (resultados en forma tabular) y 4.10 (matriz de relaciones para ese funcionario). La primera columna de la matriz de relaciones corresponde a los identificadores de los funcionarios que se encontraron, siendo el primero el que corresponde al funcionario sobre el cual se hizo la consulta original, las demás columnas corresponden a la distancia a la que se encuentran los otros funcionarios con respecto al primero, de tal forma, un 1 en la segunda columna de la matriz indica que ese funcionario se encuentra a distancia 1, es decir, es familiar directo del funcionario original, un 1 en la cuarta columna representa que ese funcionario se encuentra a distancia 3 del original y así sucesivamente. En el ejemplo, se encontraron las relaciones del funcionario Rubén Figueroa Alcocer (1311) con otros cuatro funcionarios, la comprobación de las relaciones mostradas en la figura 4.8 y de las distancias que conforman la matriz de relaciones de la figura 4.10 se muestra en el siguiente árbol obtenido de seguir las éstas relaciones directamente en la base de datos.



Con el fin de facilitar la construcción del árbol anterior, en la tercera columna de la página que presenta los resultados en forma tabular, se muestra un mnemónico que indica el nivel, en un árbol de referencia, en que se debe situar ese funcionario con respecto a aquél del cuál se obtuvo directamente su nombre, ya que, como se mencionó anteriormente, el algoritmo se ejecuta en forma recursiva con el fin de obtener todas las relaciones posibles. En ésta página existe una referencia a la imagen de este árbol con el fin de hacer más fácil su consulta. La imagen del árbol de referencia se muestra en la figura 4.9.

Considerando el caso en que un usuario no recordara el nombre completo de un funcionario, se pensó en la posibilidad de que se alimentara al programa únicamente con la fracción del nombre que recordara del mismo, lo cual trae como consecuencia que en algunas ocasiones el nombre completo de más de un funcionario coincida con los datos proporcionados por el usuario. Como se dijo anteriormente, los nombres completos de todos estos funcionarios se presentan al usuario para que éste pueda elegir sobre cuál de ellos ejecutar el programa. Para ilustrar esta situación se ejecutó el programa introduciendo el nombre **Raúl Salinas** (fig. 4.11), para el cuál el programa encontró dos coincidencias en la base de datos, éstas y la pantalla que se presenta al usuario para que éste haga su elección se muestran en la figura 4.12. Una vez hecha la selección, en este caso se eligió al funcionario **Raúl Salinas de Gortari**, el programa continúa con su ejecución normal, los resultados se muestran en la figura 4.13.

Si el nombre usado por el usuario como entrada al programa no corresponde con el de ningún funcionario en la base de datos, el programa genera una página de error donde se advierte al usuario de esta inconsistencia. Para ejemplificar este caso se alimentó al programa con el nombre **Porfirio Díaz**, éste nombre no se encuentra en la base de datos, la ejecución de esta consulta se muestra en las figuras 4.14 y 4.15

Para mostrar la ejecución normal del programa que detecta características similares en un grupo de individuos se alimentó al programa con la expresión **CONASUPO y DIRECTOR** (los botones marcados en la página como "y", "o" y "no" corresponden a los operadores "*and*". "*or*" y "*not*" de cualquier expresión lógica), sobre las tablas "Actividades Políticas", "Actividades Profesionales" y "Cargos en el Gobierno", para éste caso no se consideró un período de tiempo en particular (figura 4.16). En las figuras 4.17 a 4.20 se muestran los resultados obtenidos para dicha consulta, cabe señalar que éstas figuras muestran distintos fragmentos de la misma página. Como se mencionó anteriormente, la página de resultado está dividida en dos partes principales, la primera de ellas contiene los nombres y los identificadores en la base de datos de los funcionarios que cumplieron con la consulta original (CONASUPO y DIRECTOR en las tablas citadas para cualquier período), la segunda muestra los patrones o características encontradas para estos funcionarios en forma tanto tabular como gráfica, indicando en cada caso la característica, el porcentaje y los identificadores de los funcionarios que cumplen con ella, divididos en tres etapas

que corresponden a períodos anterior, posterior y durante el calculado por el programa en base a la consulta original.

La consistencia de la forma que se envía como entrada al programa anterior es validada a través de un programa en *JavaScript*, en la figura 4.21 se muestra un error detectado por el programa al intentar enviar la forma sin haber escrito una expresión lógica.

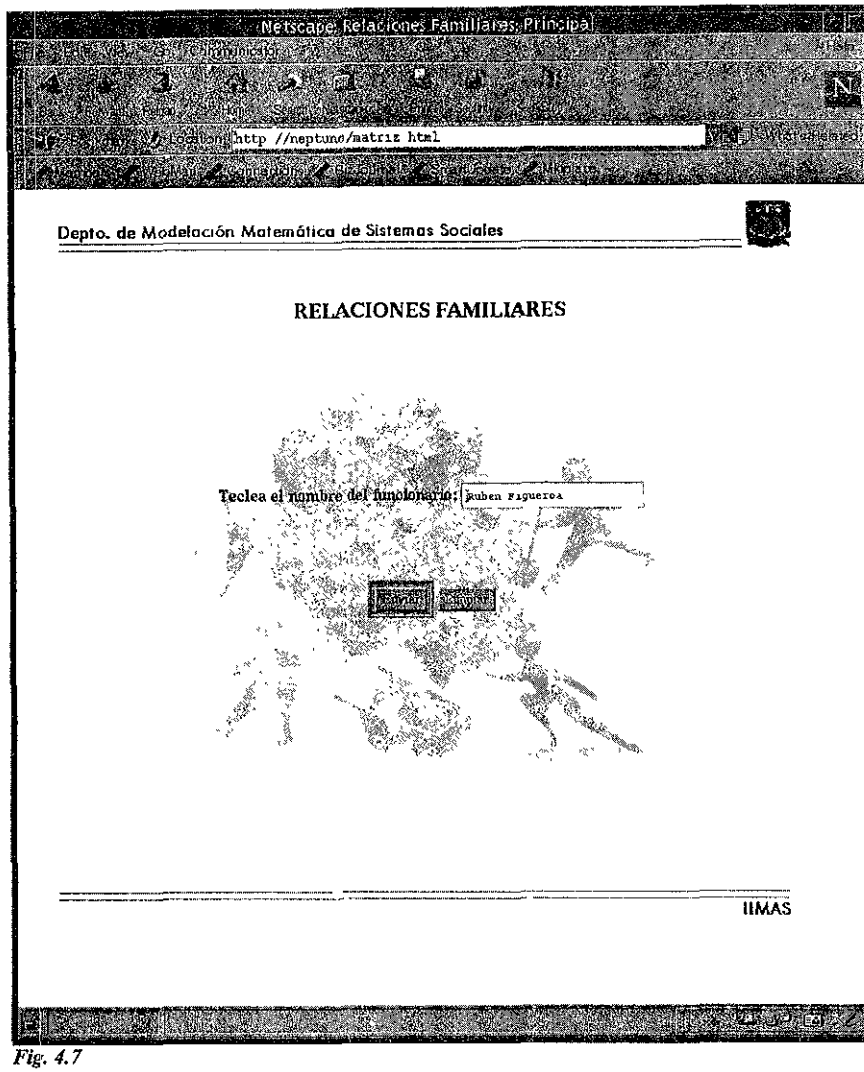

Netscape: Relaciones Familiares: Principal
http //neptuno/matriz html
Depto. de Modelación Matemática de Sistemas Sociales
RELACIONES FAMILIARES
Teclea el nombre del funcionario:
Ruben Figueroa
IIMAS

Fig. 4.7

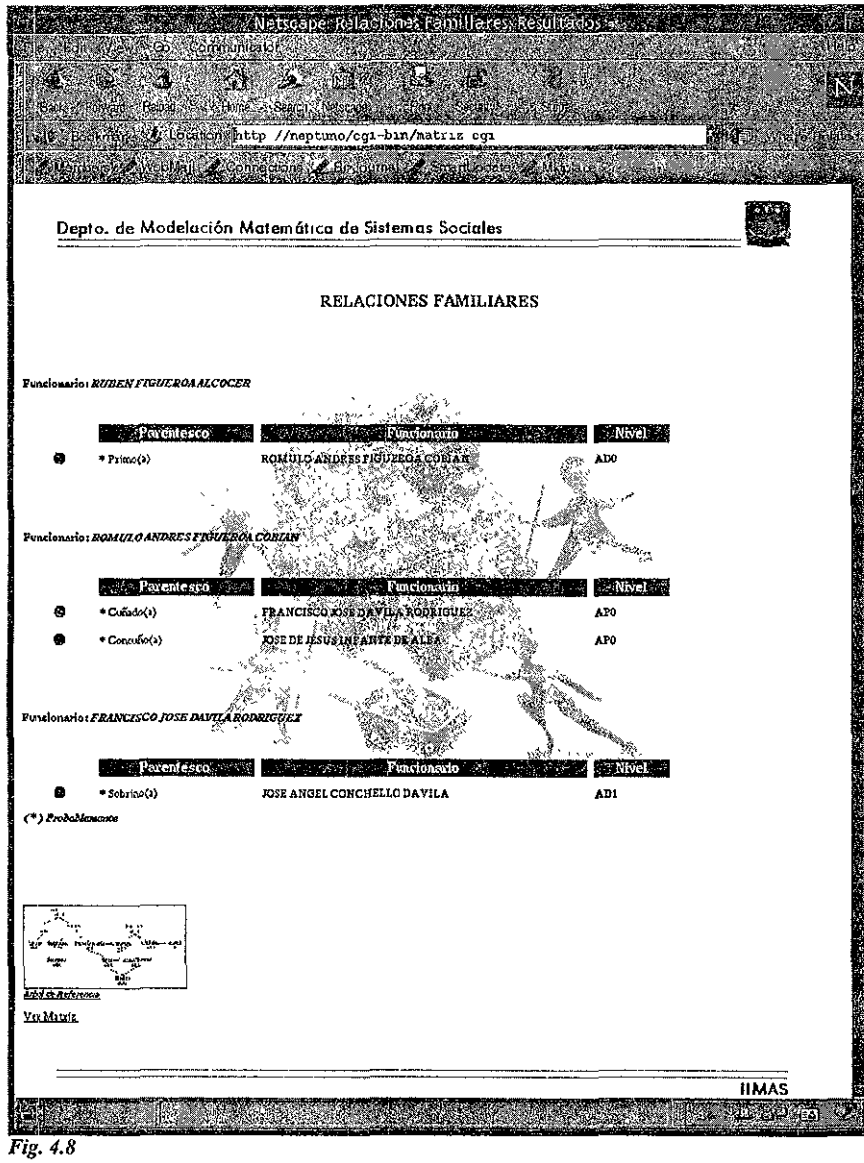

Netscape: Relaciones Familiares, Resultados
Location: http //neptuno/cgi-bin/matriz cgi
Depto. de Modelación Matemática de Sistemas Sociales
RELACIONES FAMILIARES
Funcionario: RUBEN FIGUEROA ALCOCER
Parentesco
Funcionario
Nivel
* Primo(s)
ROMULO ANDRES FIGUEROA CORIAN
AD0
Funcionario: ROMULO ANDRES FIGUEROA CORIAN
Parentesco
Funcionario
Nivel
* Cuñado(s)
FRANCISCO JOSE DAVILA RODRIGUEZ
AP0
* Consuño(s)
JOSE DE JESUS INFANTE DE ALBA
AP0
Funcionario: FRANCISCO JOSE DAVILA RODRIGUEZ
Parentesco
Funcionario
Nivel
* Sobrino(s)
JOSE ANGEL CONCHELLO DAVILA
AD1
(*) Probablemente
Ver Matriz
IIMAS

Fig. 4.8

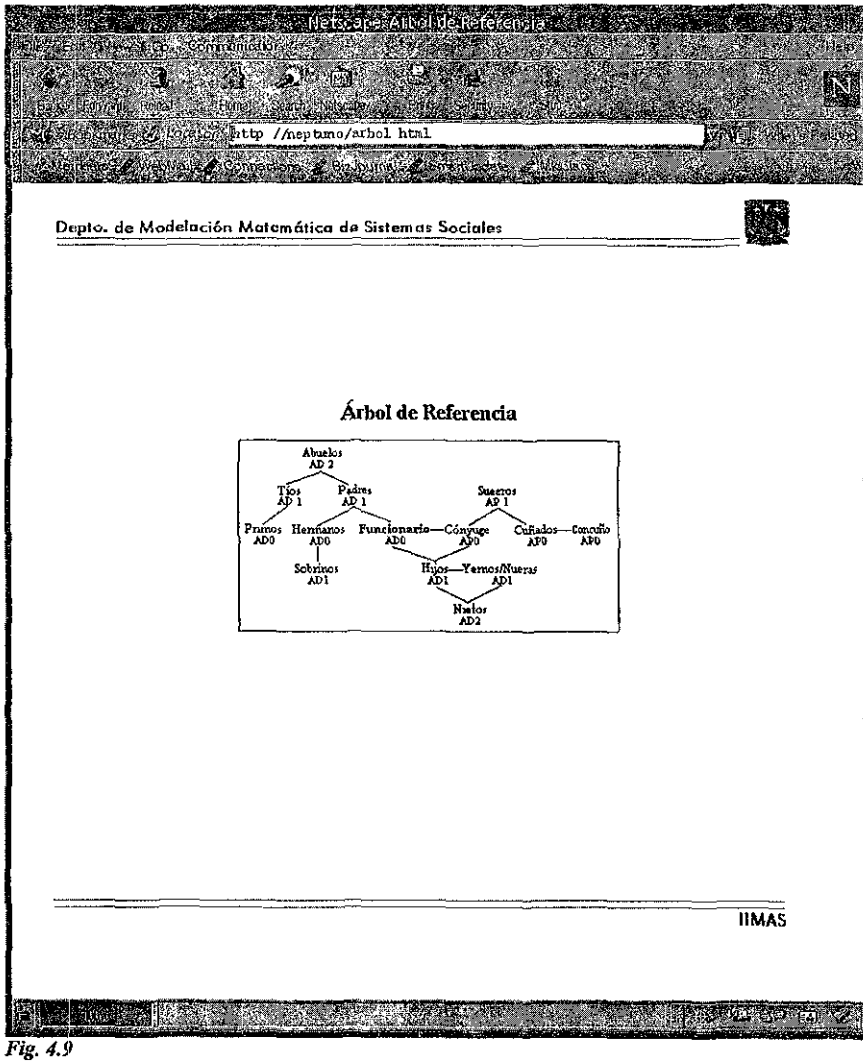

Netscape: Arbol de Referencia
http //neptuno/arbol html
Depto. de Modelación Matemática de Sistemas Sociales
Árbol de Referencia
Abuelos AD 2
Tíos AD 1
Padres AD 1
Suegros AP 1
Primos AD0
Hermanos AD0
Funcionario AD0
Cónyuge AP0
Cuñados AP0
Concuño AP0
Sobrinos AD1
Hijos AD1
Yernos/Nueras AD1
Nietos AD2
IIMAS

Fig. 4.9

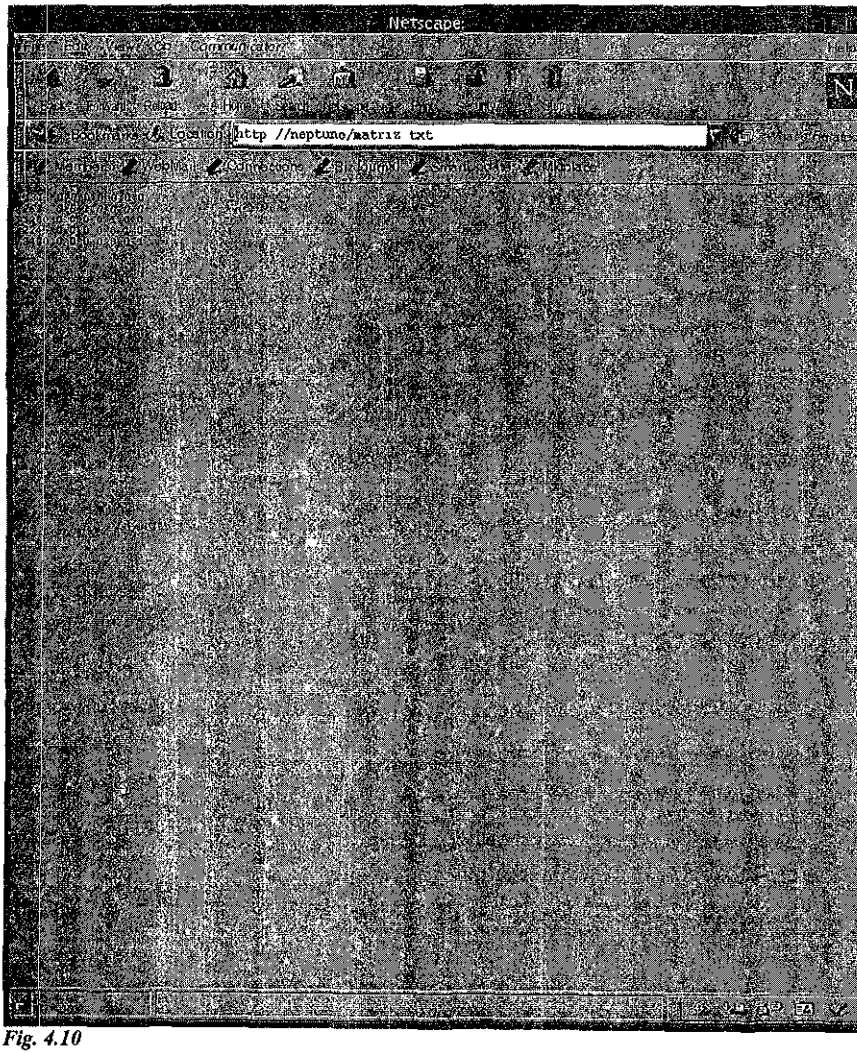


Fig. 4.10

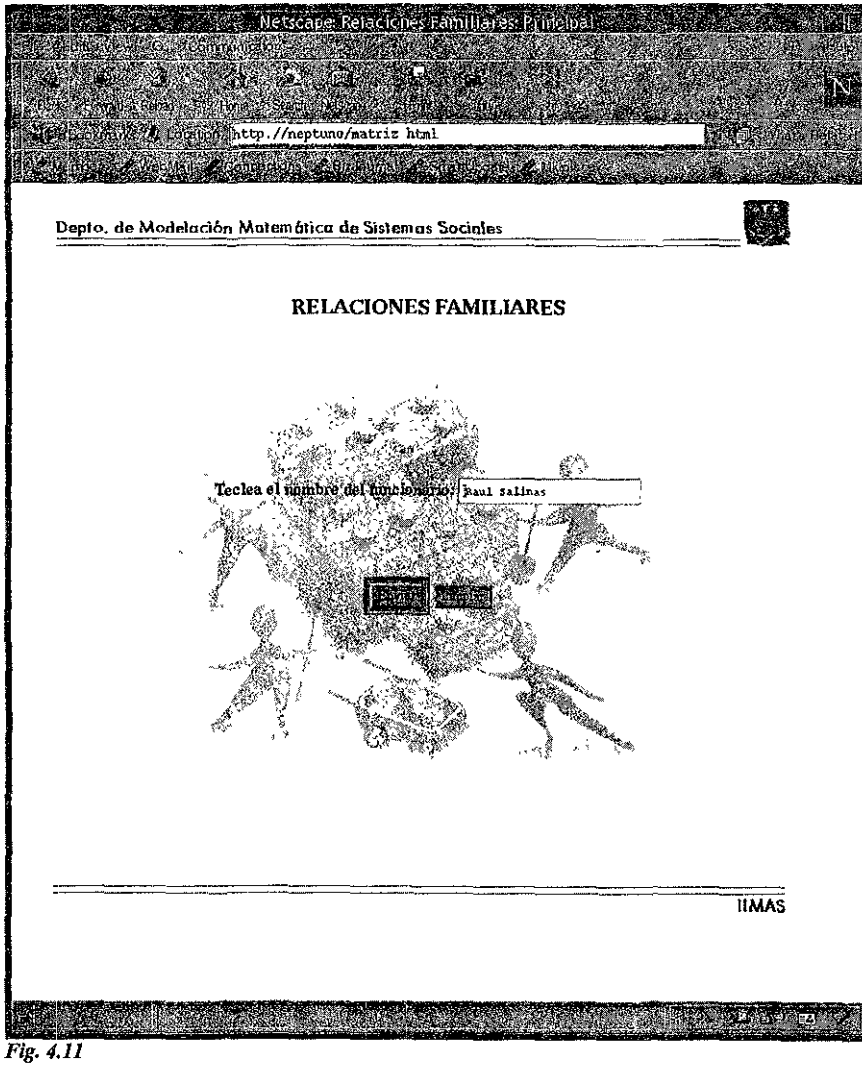

Netscape: Relaciones Familiares: Principal
http://neptuno/matriz.html
Depto. de Modelación Matemática de Sistemas Sociales
RELACIONES FAMILIARES
Teclea el nombre del funcionario:
raul salinas
IIMAS

Fig. 4.11

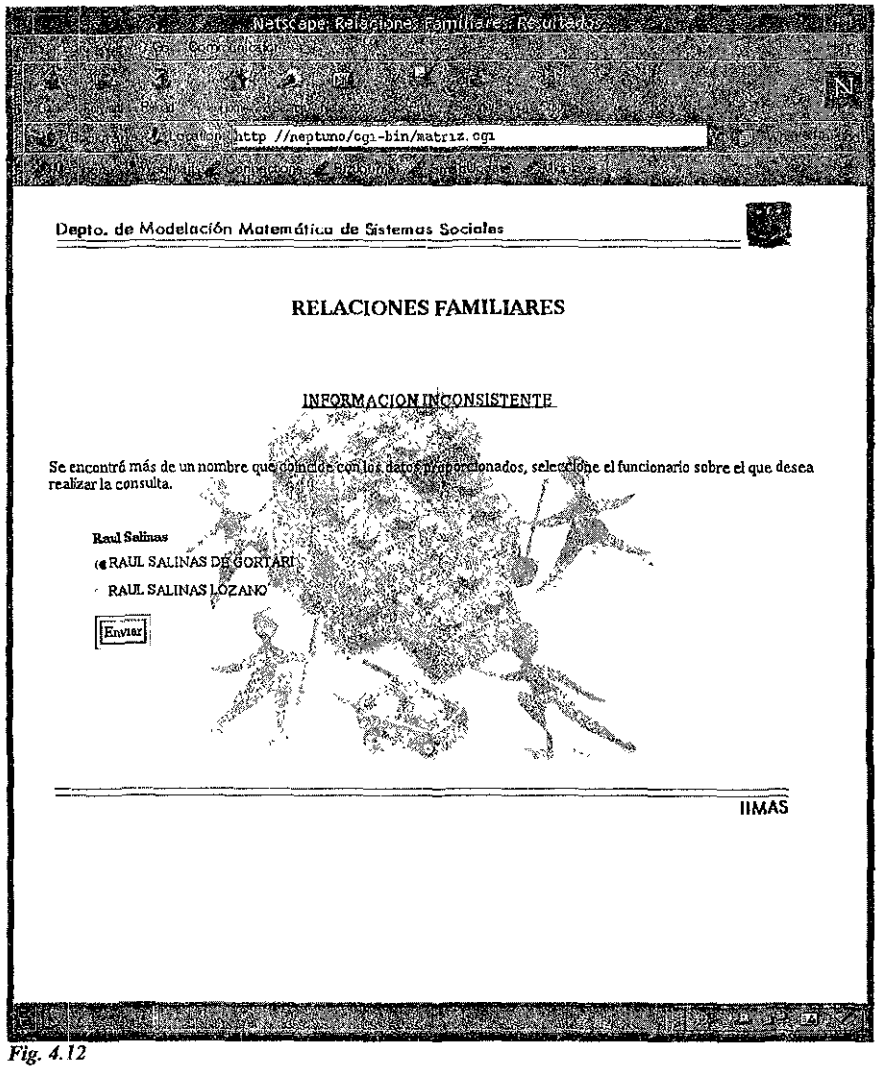

Netscape: Relaciones Familiares: Resultado
http //neptuno/cgi-bin/matriz.cgi
Depto. de Modelación Matemática de Sistemas Sociales
RELACIONES FAMILIARES
INFORMACION INCONSISTENTE
Se encontró más de un nombre que coincide con los datos proporcionados, seleccione el funcionario sobre el que desea realizar la consulta.
Raul Salinas
RAUL SALINAS DE GORTARI
RAUL SALINAS LOZANO
Enviar
IIMAS

Fig. 4.12

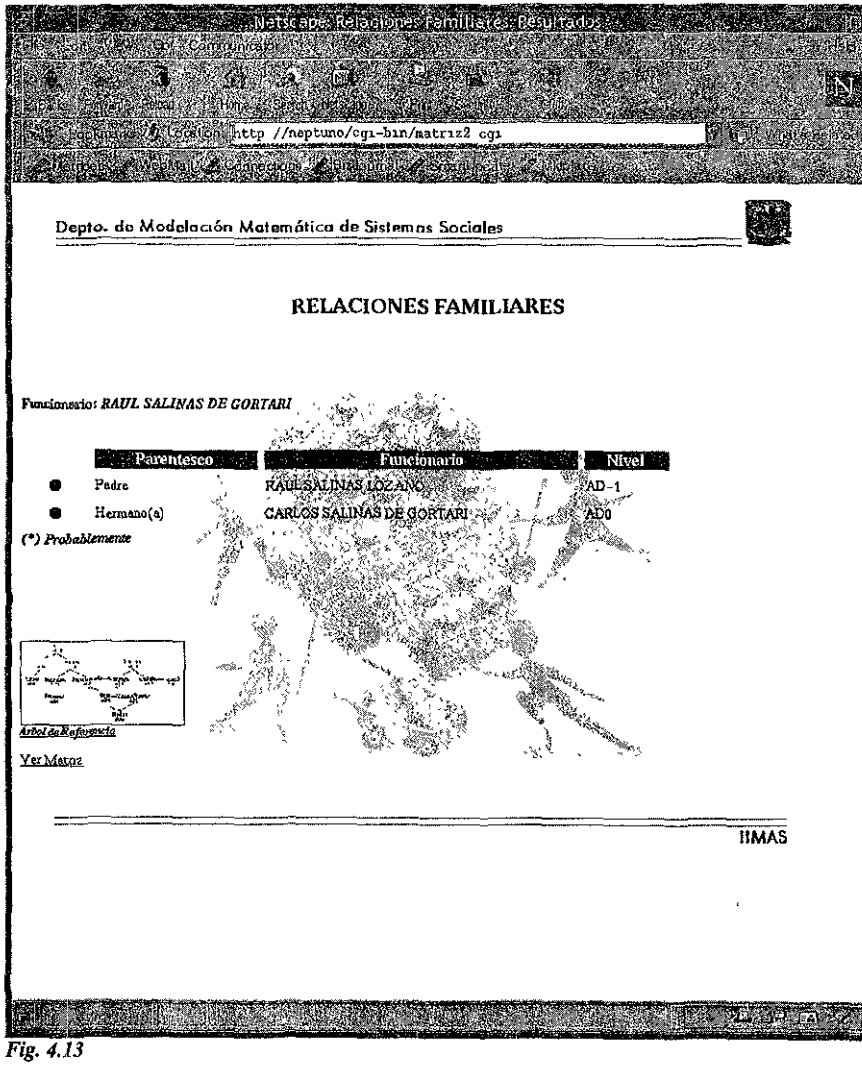

Netscape: Relaciones familiares: Resultado
http //neptuno/cgi-bin/matriz2 cgi
Depto. de Modelación Matemática de Sistemas Sociales
RELACIONES FAMILIARES
Funcionario: RAUL SALINAS DE GORTARI
Parentesco
Funcionario
Nivel
Padre
RAUL SALINAS LOZANO
AD-1
Hermano(a)
CARLOS SALINAS DE GORTARI
AD0
(*) Probablemente
Arbol de Referencia
Ver Matriz
IIMAS

Fig. 4.13

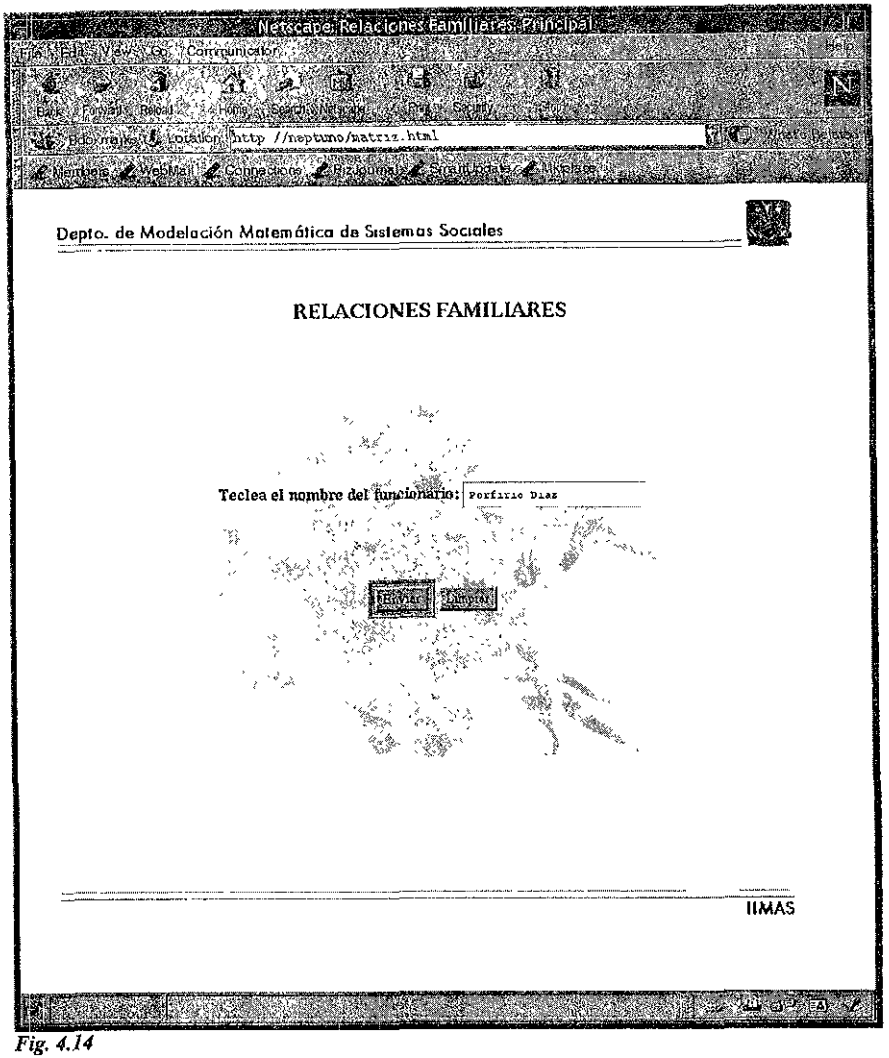

Netscape: Relaciones Familiares Principal
Location: http //neptuno/matriz.html
Depto. de Modelación Matemática de Sistemas Sociales
RELACIONES FAMILIARES
Teclea el nombre del funcionario: Porfirio Diaz
Enviar
Limpiar
IIMAS

Fig. 4.14

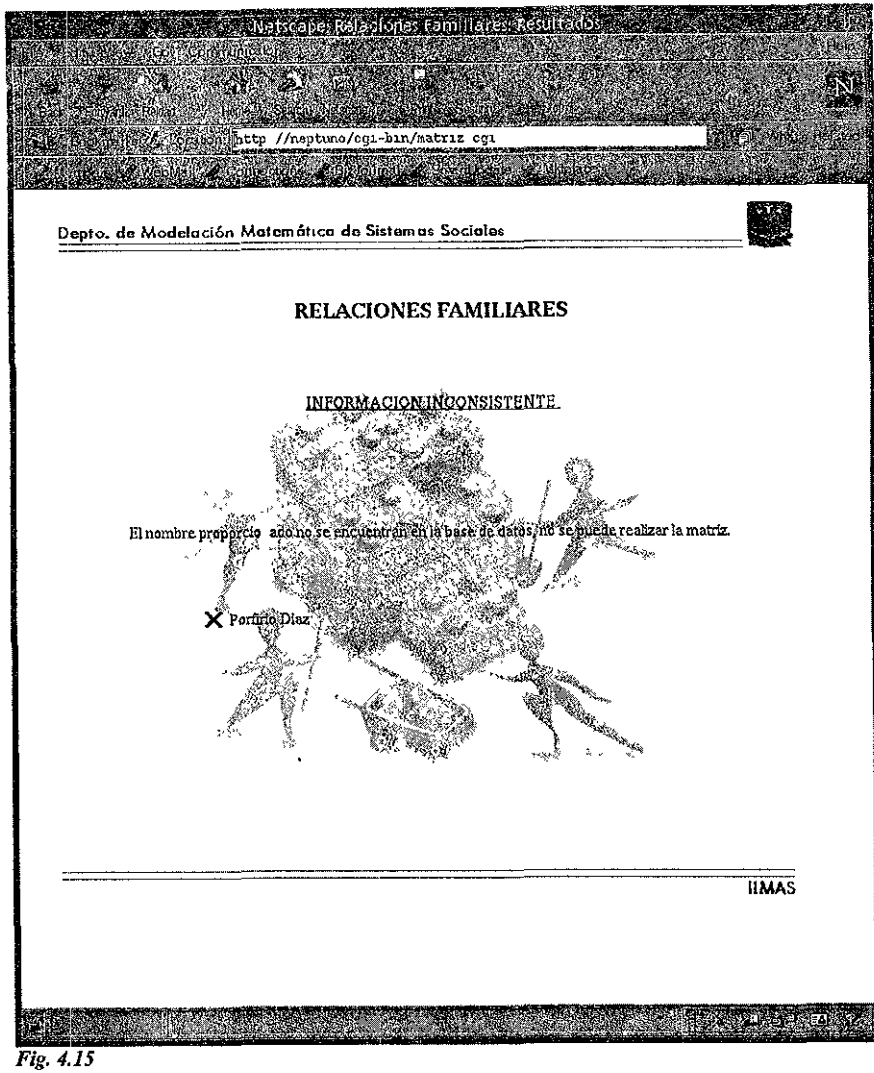

Netscape: Relaciones Familiares: Resultados
http //neptuno/cgi-bin/matriz cgi
Depto. de Modelación Matemática de Sistemas Sociales
RELACIONES FAMILIARES
INFORMACION INCONSISTENTE
El nombre proporcio ado no se encuentran en la base de datos, no se puede realizar la matriz.
Porfirio Diaz
IIMAS

Fig. 4.15

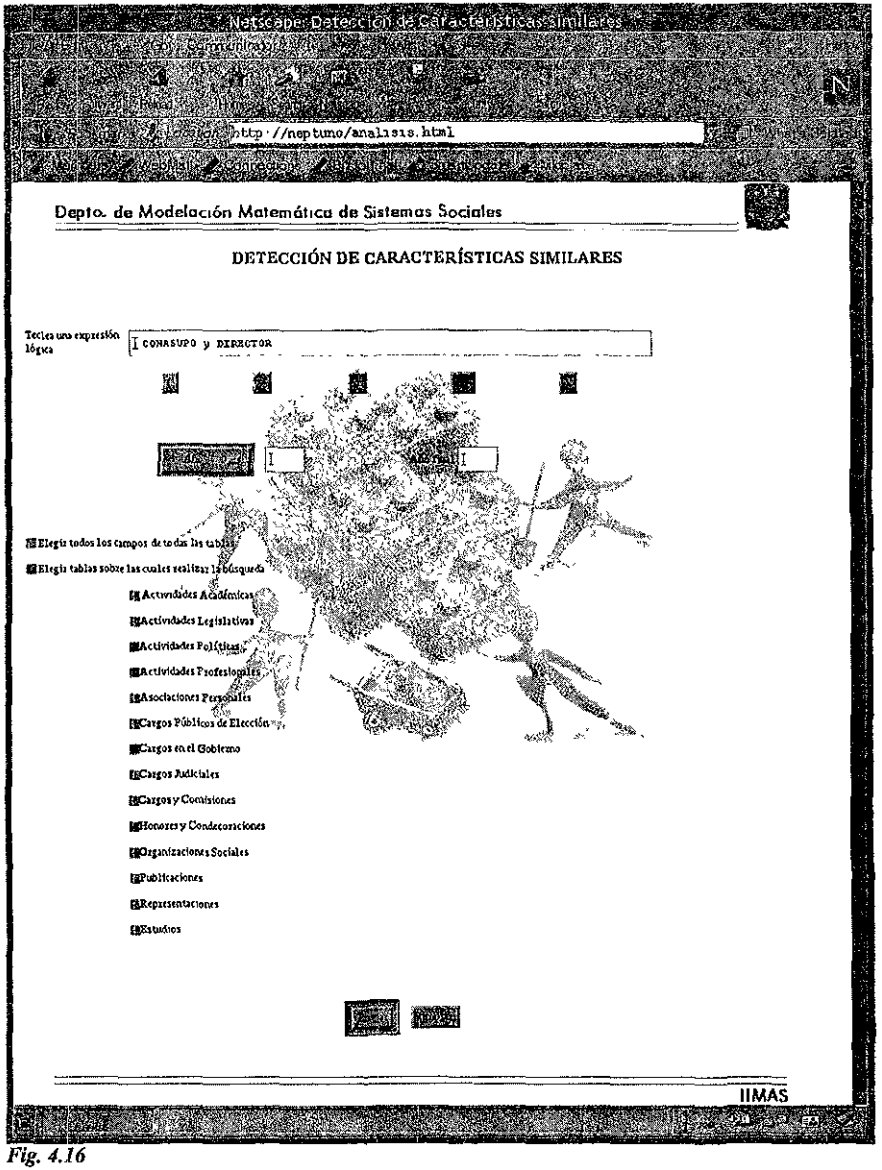

Netscape: Detección de Características Similares
http://neptuno/analisis.html
Depto. de Modelación Matemática de Sistemas Sociales
DETECCIÓN DE CARACTERÍSTICAS SIMILARES
Teclea una expresión lógica
CONASUPO y DIRECTOR
Elegir todos los campos de todas las tablas
Elegir tablas sobre las cuales realizar la búsqueda
Actividades Académicas
Actividades Legislativas
Actividades Políticas
Actividades Profesionales
Asociaciones Personales
Cargos Públicos de Elección
Cargos en el Gobierno
Cargos Judiciales
Cargos y Comisiones
Honores y Condecoraciones
Organizaciones Sociales
Publicaciones
Representaciones
Estudios
IIMAS

Fig. 4.16

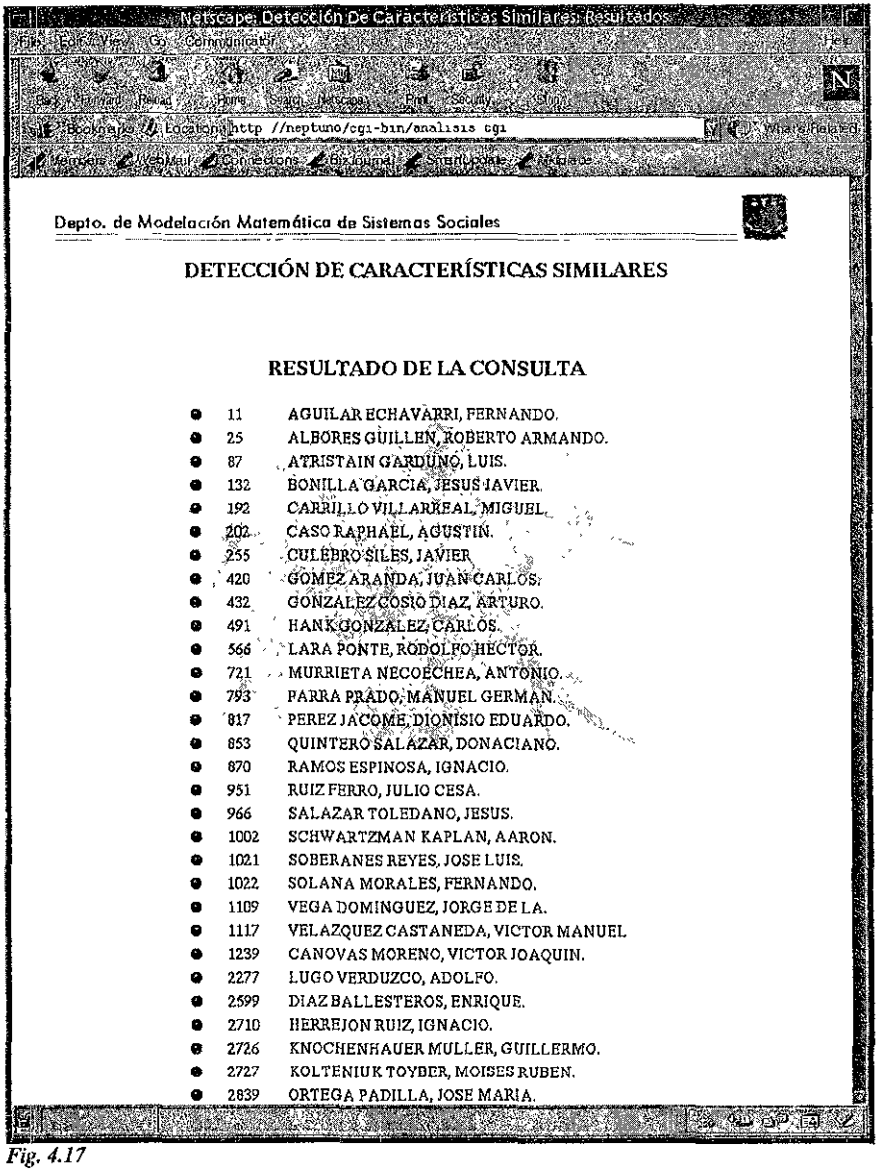

Netscape: Detección De Características Similares: Resultados
File Edit View Go Communication Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Location: http //neptuno/cgi-bin/analisis cgi What's Related
WebMail Connections SmartUpdate Mktplace
Depto. de Modelación Matemática de Sistemas Sociales
DETECCIÓN DE CARACTERÍSTICAS SIMILARES
RESULTADO DE LA CONSULTA
11 AGUILAR ECHAVARRI, FERNANDO.
25 ALBORES GUILLEN, ROBERTO ARMANDO.
87 ATRISTAIN GARDUNO, LUIS.
132 BONILLA GARCIA, JESUS JAVIER.
192 CARRILLO VILLARREAL, MIGUEL.
202 CASO RAPHAEL, AGUSTIN.
255 CULEBRO SILES, JAVIER.
420 GOMEZ ARANDA, JUAN CARLOS.
432 GONZALEZ COSIO DIAZ, ARTURO.
491 HANK GONZALEZ, CARLOS.
566 LARA PONTE, RODOLFO HECTOR.
721 MURRIETA NECOECHEA, ANTONIO.
793 PARRA PRADO, MANUEL GERMAN.
817 PEREZ JACOME, DIONISIO EDUARDO.
853 QUINTERO SALAZAR, DONACIANO.
870 RAMOS ESPINOSA, IGNACIO.
951 RUIZ FERRO, JULIO CESA.
966 SALAZAR TOLEDANO, JESUS.
1002 SCHWARTZMAN KAPLAN, AARON.
1021 SOBERANES REYES, JOSE LUIS.
1022 SOLANA MORALES, FERNANDO.
1109 VEGA DOMINGUEZ, JORGE DE LA.
1117 VELAZQUEZ CASTANEDA, VICTOR MANUEL
1239 CANOVAS MORENO, VICTOR JOAQUIN.
2277 LUGO VERDUZCO, ADOLFO.
2599 DIAZ BALLESTEROS, ENRIQUE.
2710 HERREJON RUIZ, IGNACIO.
2726 KNOCHENHAUER MULLER, GUILLERMO.
2727 KOLTENIUK TOYBER, MOISES RUBEN.
2839 ORTEGA PADILLA, JOSE MARIA.

Fig. 4.17

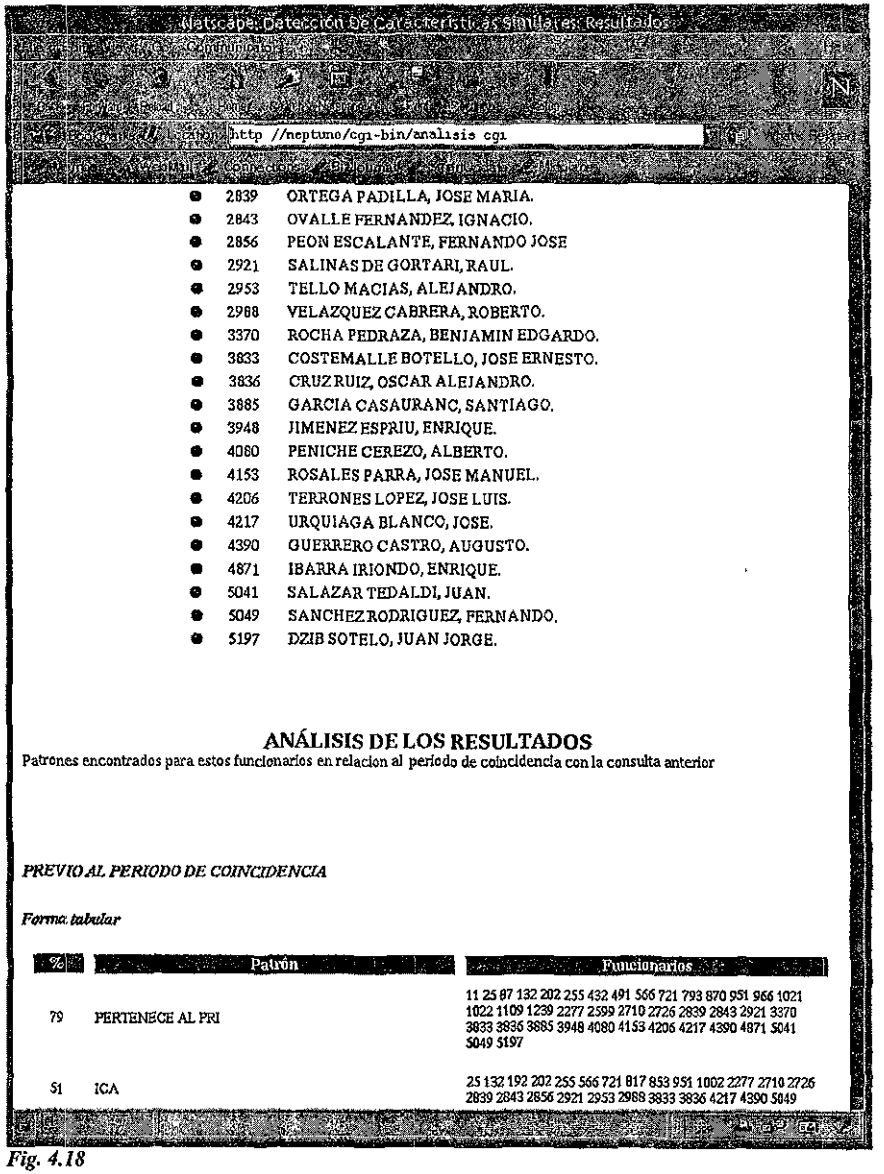

Netscape: Detección de Características Similares: Resultados
http //neptuno/cgi-bin/analisis cgi
2839 ORTEGA PADILLA, JOSE MARIA.
2843 OVALLE FERNANDEZ, IGNACIO.
2856 PEON ESCALANTE, FERNANDO JOSE
2921 SALINAS DE GORTARI, RAUL.
2953 TELLO MACIAS, ALEJANDRO.
2988 VELAZQUEZ CABRERA, ROBERTO.
3370 ROCHA PEDRAZA, BENJAMIN EDGARDO.
3833 COSTEMALLE BOTELLO, JOSE ERNESTO.
3836 CRUZ RUIZ, OSCAR ALEJANDRO.
3885 GARCIA CASAURANC, SANTIAGO.
3948 JIMENEZ ESPRIU, ENRIQUE.
4080 PENICHE CEREZO, ALBERTO.
4153 ROSALES PARRA, JOSE MANUEL.
4206 TERRONES LOPEZ, JOSE LUIS.
4217 URQUIAGA BLANCO, JOSE.
4390 GUERRERO CASTRO, AUGUSTO.
4871 IBARRA IRIONDO, ENRIQUE.
5041 SALAZAR TEDALDI, JUAN.
5049 SANCHEZ RODRIGUEZ, FERNANDO.
5197 DZIB SOTELO, JUAN JORGE.
ANÁLISIS DE LOS RESULTADOS
Patrones encontrados para estos funcionarios en relacion al periodo de coincidencia con la consulta anterior
PREVIO AL PERIODO DE COINCIDENCIA
Forma tabular
% Patrón Funcionarios
79 PERTENECE AL PRI 11 25 87 132 202 255 432 491 566 721 793 870 951 966 1021 1022 1109 1239 2277 2599 2710 2726 2839 2843 2921 3370 3833 3836 3885 3948 4080 4153 4206 4217 4390 4871 5041 5049 5197
51 ICA 25 132 192 202 255 566 721 817 853 951 1002 2277 2710 2726 2839 2843 2856 2921 2953 2988 3833 3836 4217 4390 5049

Fig. 4.18

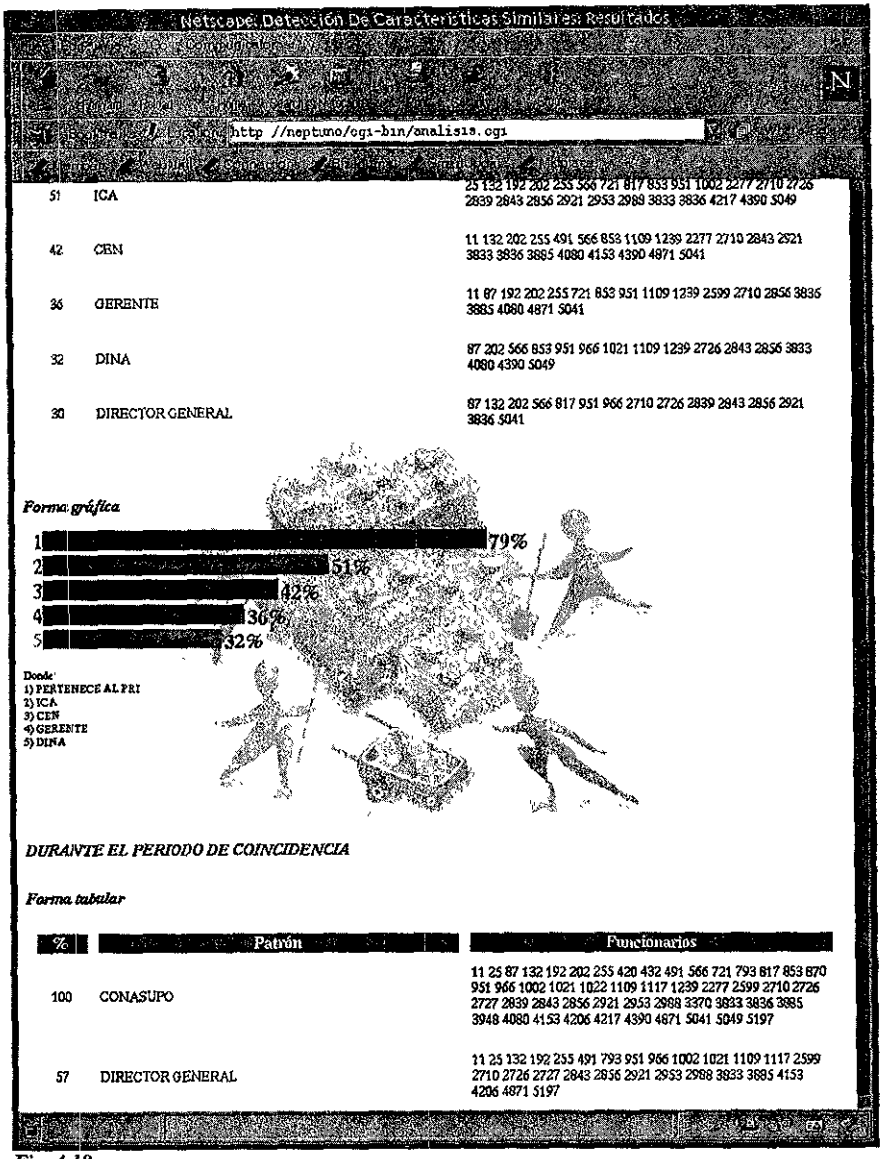

Netscape: Detección De Características Similares: Resultados
http //neptuno/cgi-bin/analisis.cgi
51 ICA
25 132 192 202 255 566 721 817 853 951 1002 2277 2710 2726 2839 2843 2856 2921 2953 2988 3833 3836 4217 4390 5049
42 CEN
11 132 202 255 491 566 853 1109 1239 2277 2710 2843 2921 3833 3836 3885 4080 4153 4390 4871 5041
36 GERENTE
11 87 192 202 255 721 853 951 1109 1239 2599 2710 2856 3836 3885 4080 4871 5041
32 DINA
87 202 566 853 951 966 1021 1109 1239 2726 2843 2856 3833 4080 4390 5049
30 DIRECTOR GENERAL
87 132 202 566 817 951 966 2710 2726 2839 2843 2856 2921 3836 5041
Forma gráfica
1 79%
2 51%
3 42%
4 36%
5 32%
Donde:
1) PERTENECE AL PRI
2) ICA
3) CEN
4) GERENTE
5) DINA
DURANTE EL PERIODO DE COINCIDENCIA
Forma tabular
% Patrón Funcionarios
100 CONASUPO
11 25 87 132 192 202 255 420 432 491 566 721 793 817 853 870 951 966 1002 1021 1022 1109 1117 1239 2277 2599 2710 2726 2727 2839 2843 2856 2921 2953 2988 3370 3833 3836 3885 3948 4080 4153 4206 4217 4390 4871 5041 5049 5197
57 DIRECTOR GENERAL
11 25 132 192 255 491 793 951 966 1002 1021 1109 1117 2599 2710 2726 2727 2843 2856 2921 2953 2988 3833 3885 4153 4206 4871 5197

Fig. 4.19

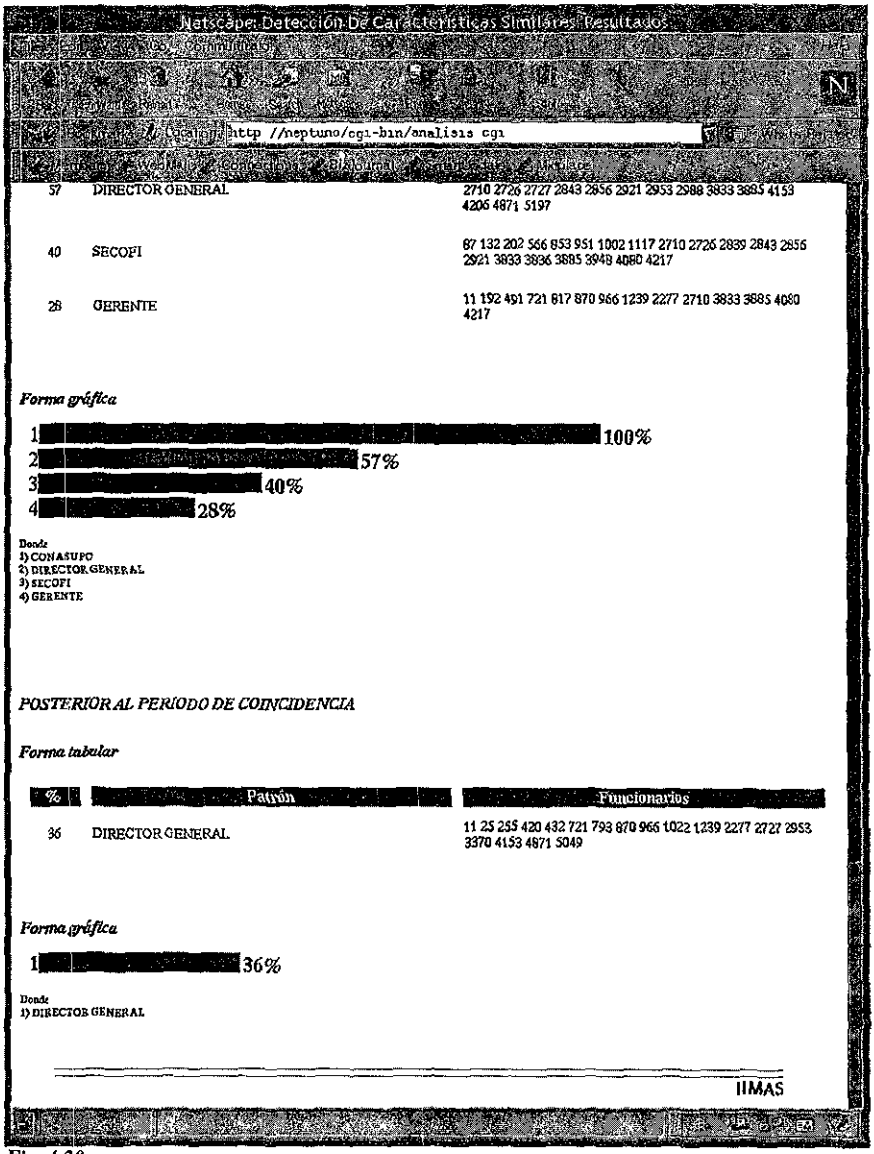

Netscape: Detección De Características Similares: Resultados
http //neptuno/cgi-bin/analisis cgi
57 DIRECTOR GENERAL 2710 2726 2727 2843 2856 2921 2953 2988 3833 3885 4153 4206 4871 5197
40 SECOFI 87 132 202 566 853 951 1002 1117 2710 2726 2839 2843 2856 2921 3833 3836 3885 3948 4080 4217
28 GERENTE 11 192 491 721 817 870 966 1239 2277 2710 3833 3885 4080 4217
Forma gráfica
1 100%
2 57%
3 40%
4 28%
Donde
1) CONASUPO
2) DIRECTOR GENERAL
3) SECOFI
4) GERENTE
POSTERIOR AL PERIODO DE COINCIDENCIA
Forma tabular
% Patrón Funcionarios
36 DIRECTOR GENERAL 11 25 255 420 432 721 793 870 966 1022 1239 2277 2727 2953 3370 4153 4871 5049
Forma gráfica
1 36%
Donde
1) DIRECTOR GENERAL
IIMAS

Fig. 4.20

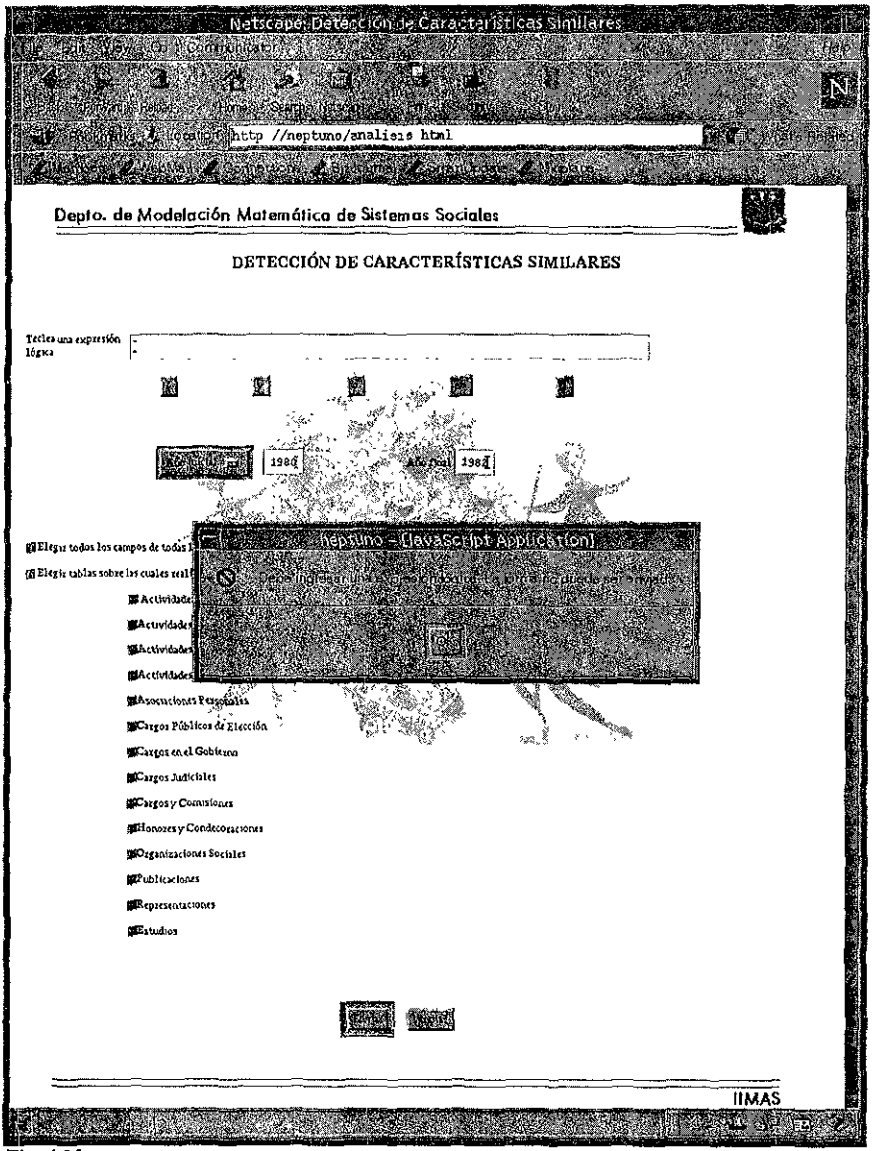

Netscape: Detección de Características similares
http //neptuno/analisis html
Depto. de Modelación Matemática de Sistemas Sociales
DETECCIÓN DE CARACTERÍSTICAS SIMILARES
Teclee una expresión lógica
1980
1984
neptuno - [JavaScript Application]
Actividades
Asociaciones Personales
Cargos Públicos de Elección
Cargos en el Gobierno
Cargos Judiciales
Cargos y Comisiones
Honores y Condecoraciones
Organizaciones Sociales
Publicaciones
Representaciones
Estudios
IIMAS

Fig. 4.21

CONCLUSIONES

CONCLUSIONES

El desarrollo del presente trabajo nos dio la oportunidad de conocer las principales técnicas existentes hoy en día para la Minería de Datos, así como de poder aplicar algunas de ellas a un problema real, las técnicas elegidas: los árboles de decisión y el análisis del carrito del supermercado, proporcionaron los resultados esperados, lo cual se debió también, en gran medida, a que los algoritmos fueron diseñados en forma específica para el problema planteado al inicio.

Actualmente el principal desarrollo de las aplicaciones de la Minería de Datos se ha dado en el ámbito comercial, por lo que, el desarrollo de una aplicación para la detección de posibles relaciones familiares y de características similares entre un grupo de personas, proporciona una nueva alternativa para el uso de la Minería de Datos.

Durante el desarrollo de los programas de aplicación se trabajó con una base de datos relacional cuyo objetivo inicial era tener disponible la información contenida en diversas ediciones del "Diccionario Biográfico del Gobierno Mexicano" en un medio electrónico, es decir, no fue diseñada para ser utilizada como base para la implantación de aplicaciones de Minería de Datos, sin embargo, aunque los datos no se encontraban en un formato adecuado, fue posible desarrollar una aplicación que funcionara eficientemente aún bajo esas condiciones.

Considerando la naturaleza de los programas y la magnitud de la base de datos se decidió aplicar un proceso de adaptación y normalización de los datos para las tablas utilizadas por el programa encargado de descubrir relaciones familiares entre funcionarios, mientras que para el programa que detecta características similares en un grupo de funcionarios se optó por mantener el formato original de las tablas utilizadas debido al número de registros involucrados en la obtención de los resultados. Al respecto pudimos observar que una mejor organización en los registros de la base de datos trae como resultado el hecho de poder hacer comparaciones directas entre los datos almacenados en ellos, esto queda de manifiesto en la operación del programa para el que se llevó a cabo la adaptación de los datos, en el otro programa esto no fue posible, por lo que dentro del algoritmo se tuvo que contemplar código que realizara esta tarea, determinando, además, la forma en que son presentados los resultados.

De acuerdo a la experiencia adquirida en el uso y administración del servidor de WWW así como de los clientes del mismo, se realizó un ajuste en los tiempos de respuesta permitidos por el servidor con el fin de evitar cortes en la conexión, y por lo tanto en la ejecución de los programas, derivados del tiempo que tarda la ejecución de los mismos, así como de cualquier otro problema presentado en la red.

Adicional a la configuración de los tiempos de respuesta, los clientes de WWW deben de cumplir con el requisito de poder interpretar programas escritos en Java Script, uno de los problemas que se encontraron es que las primeras versiones de los clientes de WWW no cumplen con esta característica, además la forma en que es interpretado el código difiere entre los dos clientes más utilizados: Internet Explorer y Netscape Navigator, sin embargo no se han presentado problemas al utilizar Internet Explorer versión 4.0 o Netscape Navigator versiones 4.0 o superiores.

Pudimos darnos cuenta que para el desarrollo de sistemas de Minería de Datos se requieren conocimientos sólidos de las técnicas de Minería de Datos así como del funcionamiento general de los Almacenes de Datos, asimismo se requiere experiencia en el diseño de bases de datos y en el uso y administración de al menos un manejador de bases de datos de tipo comercial, así como de codificación en algún lenguaje de programación. Para el caso particular de aplicaciones basadas en la WWW se requiere además del conocimiento de la operación de programas CGI que interactúen con bases de datos y de la administración de servidores WWW.

Nos pudimos dar cuenta durante el desarrollo del proyecto, de la gran importancia que ha adquirido en los últimos meses la tecnología de la Minería de Datos a nivel mundial. El número de libros así como el de documentos en Internet ha experimentado un importante incremento desde abril de 1997 (fecha en que se planeaba iniciar con el proyecto) a la fecha, además han surgido empresas dedicadas a la construcción de sistemas de Minería de Datos "a la medida" del cliente así como sistemas de propósito general

Para la interpretación de los resultados obtenidos en el Laboratorio de Redes del IIMAS, se requiere de la ayuda de politólogos, sociólogos y antropólogos este hecho no es una excepción para el sistema de Minería de Datos desarrollado, en este aspecto consideramos que el sistema es de fácil manejo y que los resultados devueltos se presentan en un formato que facilita su interpretación.

Considerando el funcionamiento de la aplicación, pensamos que es susceptible de recibir algunas mejoras en el futuro, por ejemplo, pueden desarrollarse funciones que lleven a cabo las operaciones que actualmente efectúan las herramientas de Unix, las cuales al ser compiladas como parte del programa se ejecuten de una manera más ágil, conservando su funcionalidad pero haciendo más rápida la ejecución de todo el programa, asimismo se piensa que en un futuro no muy lejano se cuente ya con una base de datos normalizada, con la cual el análisis de la información sería más rápido, los resultados obtenidos serían más consistentes y se minimizaría el uso de estas rutinas. Actualmente dentro de los resultados que entrega el programa encargado de obtener características similares, aparecen palabras que por sí solas no tienen ningún significado en el contexto de la aplicación, si, una vez aparecida una de estas palabras, se desea que no vuelva a formar parte de ninguno de los resultados posteriores, tiene que ser almacenada en forma manual en un archivo, consideramos que este problema debe desaparecer al contarse con una base de datos normalizada, sin embargo mientras no se cuente con ella, es posible mejorar esta operación permitiendo que el programa actualice automáticamente la información que contiene ese archivo a petición del usuario. Otra mejora que proponemos al sistema es que existen relaciones entre funcionarios que no se pueden obtener de la base de datos pero que los expertos en el tema conocen y que pueden ser puntos clave en las relaciones entre los funcionarios, en este punto se propone la creación de una nueva tabla en la base de datos que almacene tanto los resultados obtenidos en ejecuciones anteriores que hayan sido validadas por un experto, así como aquellas que no se encuentran en la base de datos, esta tabla sería consultada antes de efectuar cualquier otra operación, de tal forma que el programa, además, recuerde y pueda hacer uso de relaciones encontradas anteriormente.

Del desarrollo global de este trabajo pudimos darnos cuenta de la gran importancia y utilidad que tiene la implantación de sistemas de Minería de Datos en cualquier área, pero principalmente en el ámbito de los negocios, donde está difundida en su gran mayoría, afortunadamente en nuestro país estas técnicas se encuentran ya en sus inicios y comienzan a motivar a las empresas en su estudio y desarrollo y con ello lograr que los registros almacenados diariamente en sus bases de datos históricas les permitan obtener "oro" donde hoy sólo hay bits.

BIBLIOGRAFÍA

BIBLIOGRAFÍA

- BELL Davis & GRIMSON Jane
  Distribuited Database Systems
  UK, Addisson Wesley, 1992

- BERRY Michael J.A. & LINOFF Gordon
  Data Mining Techniques for marketing, sales and customer support
  USA, John Willey & Sons, 1997

- BONTEMPO Charles J. & MARO Saracco Cynthia
  Database management principles and products.
  USA, Prentice Hall, 1995

- BROWN, Alan W.
  Obect-Oriented Databases Applications in Software Engineering
  Mc Graw Hill, UK, 1981

- DANESH, Arman
  Aprendiendo Java Script en una Semana
  México, Prentice Hall, 1996

- DATE C.J.
  An Introduction to Database Systems
  5a. Edición Vol. I
  USA, Addisson Wesley, 1991

- FLANAGAN David
  JavaScript The definitive Guide
  2a Edición
  USA, O'Reilly & Associates, 1997

- HANSEN Gary W & HANSEN James V.
  Database management and design
  2a. Edición.
  USA, Prentice Hall, 1996

- HARJINDER S. Gill & PRAKASH C. Rao
  Data Warehousing
  USA, Prentice Hall, 1996

- HUGHES John G.
  Object Oriented Databases
  Prentice Hall, UK, 1991

- KORTH Henry F. & SILBERSCHATZ Abraham
  Databse System Concepts
  2a. Edición
  USA, Mc Graw Hill, 1991

- KROENKE David M.
  Database Processing. Fundamentals, Design and Implementation
  6a. Edición.
  USA, Prentice Hall, 1998

- LOOMIS, Mary E.S.
  Estructuras de Datos y Organización de archivos
  México, Prentice Hall, 1991

- PEEK Jerry, O'REILLY Tim, LOUKIDES Mike and others
  Unix Power Tools
  USA, O'Reilly & Associates, 1993

- SCHILDT, Herbert
  Turbo C/C++. Manual de referencia
  México, Mc Graw Hill, 1993

- STEFANO Ceri & PELAGATTI Giusepe
  Distribuited Databases Principles and Systems
  Singapur, Mc Graw Hill, 1985

- TENENBAUM Aaron M., LANGSAM Yedidyah y AUGENSTEIN Moshe A.
  Estructuras de datos en C
  México, Prentice Hall, 1993