104
2EJ

UNIVERSIDAD NACIONAL
AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

BASES DE DATOS RELACIONALES Y DISTRIBUIDAS

FALLA DE ORIGEN

T E S I S

QUE PARA OBTENER LOS TITULOS DE

INGENIERO EN COMPUTACIÓN,

P R E S E N T A N

WILHELM STREVEL GRACE

Director de Tesis
ING JUAN CARREÓN G

Ciudad Universitaria, México D F 1995

INDICE

CAPÍTULO 3. ESTRUCTURAS DE ALMACENAMIENTO Y ORGANIZACION FISICA DE LOS DATOS.

3.1 ESTRUCTURAS DE ALMACENAMIENTO DE DATOS

3.2 DISPOSITIVOS DE ALMACENAMIENTO DE DATOS

3.3 CONTROL DE ENTRADAS Y SALIDAS

3.3.1 Procesamiento de entradas y salidas

3.4 ORGANIZACION DE LOS ARCHIVOS

3.4.1 Comparacion de las organizaciones de archivos basicas

3.4.2 Modos de acceso a los registros

3.5 ESTRUCTURAS DE ALMACENAMIENTO UTILIZADAS EN INGRES

3.5.1 Estructura de almacenamiento HEAP (Secuencial)

3.5.2 Estructura de almacenamiento HASH

3.5.3 Estructura de almacenamiento ISAM

3.5.4 Estructura de almacenamiento BTREE (Árboles balanceados)

3.5.5 Paginas de datos asociadas

CAPÍTULO 4. EL ENFOQUE RELACIONAL Y DISTRIBUIDO.

4.1 EL MODELO ENTIDAD-RELACION (E-R)

4.1.1 Relaciones y conjuntos de relaciones

4.1.2 Limitantes de mapeo

4.1.3 Llaves primarias

4.1.4 El modelo Entidad-Relacion (E-R)

OBJETIVO

El objetivo de la tesis es revisar la teoria basica sobre bases de datos relacionales y distribuidas. Se analiza un caso real de transferencia de información entre dos bases de datos que se encuentran en entornos diferentes.

CAPITULO 1
INTRODUCCIÓN

Los sistemas de bases de datos son considerados como una de las areas de la computacion y de la informacion que ha tenido el mas rapido desarrollo. Este crecimiento fue propiciado por la evolucion del hardware y software por un lado y por la creciente necesidad de procesamiento de datos por parte de los usuarios, por el otro. La historia de las bases de datos puede ser dividida en cuatro generaciones, las cuales son

La primera generación fue la de los 50's. En esos días, la tarea principal de una computadora era procesar datos bajo el control de un programa. Su uso era limitado, se empleaba para realizar funciones sencillas como contar, sumar, etc. Cada programa era suministrado con el conjunto de datos que iba a operar. En algunos casos, el programa leía los datos necesarios de la memoria secundaria a la memoria principal de la computadora, los procesaba y los regresaba a la memoria secundaria, que consistía de tarjetas perforadas o cintas magneticas, las cuales sólo permiten procesamiento secuencial. Este fue el primer sistema de archivos de acceso secuencial.

Para la segunda generación, las computadoras se podían usar en modo línea y modo lote. El desarrollo de discos magnéticos como memoria secundaria, propició el surgimiento de sistemas de archivos más sofisticados, los cuales permitían accesos múltiples. Un archivo de acceso directo permite accesar un registro directamente por su dirección (en el disco), sin tener que leer todos los registros que le anteceden.

La tercera generacion se caracteriza por una distincion entre la informacion a nivel logico y fisico y por la necesidad de administrar grandes cantidades de datos. Durante este tiempo se usaron por primera vez los modelos de datos para describir las estructuras fisicas desde un punto de vista logico.

En la cuarta generacion se hace una clara distincion entre el modelo fisico y logico de los datos. Se comercializan varias bases de datos y se invade el mercado en la decada de los 80's. El aspecto principal radica en que el almacenamiento fisico de los datos es transparente a los usuarios, de esta manera los cambios en el modelo fisico o en el logico no deben afectar al otro modelo. Se populariza el uso de bases de datos relacionales.

Podemos decir que muchas empresas u organizaciones dependen de la operacion continua y eficaz de un sistema de bases de datos. Un sistema de bases de datos es un sistema de mantenimiento de registros basados en computadoras, es decir, un sistema cuyo proposito general es registrar y mantener la informacion. Tal informacion puede estar relacionada con cualquier cosa que sea significativa para la empresa donde el sistema opera.

En el corazon de todo sistema de manejo de informacion existe una base de datos. Un archivador metalico con registros de clientes, un lote de tarjetas con nombres y numeros de telefono o un cuaderno con una lista de un inventario de un almacen escrita a mano pueden ser considerados como bases de datos. Sin embargo, el archivador o el cuaderno no constituyen en si mismos la base de datos, lo que los convierte en bases de datos es la forma en que se organiza la informacion en ellos.

En una base de datos la información normalmente se organiza y se mantiene en una tabla compuesta por renglones y columnas. Los renglones en una tabla de base de datos se llaman registros o tuplas, a las columnas se les llaman campos o atributos, es capaz de manejar grandes volumenes de información

La presente tesis esta dividida en ocho capitulos. El capitulo uno es la introduccion. Los primeros capitulos que son del dos al cinco abarcan los conceptos basicos acerca de las bases de datos relacionales y distribuidas orientados a la base de datos INGRES.

En el capitulo dos se proporciona un panorama general de la estructura general de una base de datos, asi como los conceptos basicos y algunos terminos comunmente utilizados en el ambiente de bases de datos.

En el capitulo tres se presentan las estructuras de almacenamiento de datos, las distintas organizaciones de archivos y los diferentes metodos de acceso a los registros. Se finaliza presentando las estructuras de almacenamiento de datos que utiliza la base de datos INGRES mostrando sus ventajas y desventajas.

En el capitulo cuatro se plantea el enfoque relacional y distribuido, utilizando el modelo entidad relacion para comprender mejor el concepto de base de datos relacional. Se presenta la teoria basica, algebra relacional y cálculo relacional, de los lenguajes de consulta formales y algunas de las operaciones que utilizan. Posteriormente, se revisan dos lenguajes de consulta comerciales SQL y QUEL, basados en los conceptos anteriores. Se analiza el enfoque de las bases de datos distribuidas y su entorno, asi como las opciones para distribuir una base de datos. Finalmente, se incluye una breve descripción de las bases de datos orientadas a objetos.

En el capitulo cinco se exponen las consideraciones necesarias para diseñar e implantar un sistema de base de datos.

En el dos capítulos siguientes se desarrolla la parte práctica de la tesis. Se incluyen los criterios prácticos de selección que se deben tener en cuenta para elegir una base de datos del conjunto que existen en el mercado y se analiza un caso real de los sistemas de bases de datos.

En el capítulo seis se presentan los criterios para seleccionar una base de datos comercial y las pruebas por realizar para asegurarnos de que el producto cumpla con nuestros requerimientos. Las pruebas se realizaron con varios productos del mercado tales como ORACLE, INGRES, INFORMIX y SYBASE. Finalmente, se muestra una tabla comparativa de los resultados obtenidos y de las razones por las que se escogió INGRES.

En el capítulo siete analiza un caso práctico de los sistemas de bases de datos, el cual consiste en la transferencia de información entre dos bases de datos distintas que operan en diferentes plataformas de trabajo. Se hace una propuesta real, se presenta el esquema general de la transferencia y los pasos que se siguieron para realizar dicha transferencia, así como los problemas que surgieron al implantar dicho esquema.

En el capítulo ocho se describen las conclusiones a las que llegamos al realizar esta tesis.

CAPITULO 2.
CONCEPTOS SOBRE BASES DE DATOS

2.1. DEFINICION DE BASE DE DATOS

Una base de datos es una coleccion de datos relacionados entre si, los cuales estan almacenados juntos para servir a las necesidades de los usuarios.

Una base de datos es un repositorio de datos almacenados que es integrada y compartida. Por integrada debemos entender que la base de datos se considera como una unificacion de varias fuentes de datos independientes, donde se elimina parcial o totalmente cualquier redundancia entre los mismos. Por compartida se entiende que partes independientes de la base de datos pueden utilizarse entre varios usuarios, en el sentido de que cada uno de ellos puede accesar la misma parte de la base de datos (o, utilizarla con propositos diferentes) al mismo tiempo.

Una base de datos es un conjunto de datos de operacion almacenados y utilizados por los sistemas de aplicacion de una empresa determinada.

C. J. Date

Una base de datos es una coleccion de datos interrelacionados almacenados juntos sin redundancia innecesaria, para servir a multiples funciones.

James Martin

En resumen, podemos definir a una base de datos como un conjunto de datos los cuales están relacionados entre sí, se almacenan juntos y tienen la finalidad de servir a aplicaciones específicas de una empresa

2.2 OBJETIVOS DE LAS BASES DE DATOS

¿ Por que una empresa debe optar por almacenar sus datos en una base de datos ?

La mejor respuesta a la pregunta anterior es la siguiente el tener un sistema de base de datos proporciona a cualquier empresa un control centralizado de su información la cual constituye uno de sus activos más valiosos Los principales objetivos que se persiguen al tener un control centralizado de los datos se mencionan a continuación

• Reducir la redundancia.

En empresas que no utilizan sistemas con bases de datos o con bases de datos mal diseñadas hay una gran cantidad de datos duplicados o redundantes lo que origina que se eleve el costo de almacenamiento y de acceso, además de incrementar la posibilidad de que exista inconsistencia en la información es decir, que las distintas copias de la misma información no concuerden entre sí Con las bases de datos se pretende eliminar la redundancia aunque a veces se prefiere tener una redundancia controlada

mínima con el objeto de minimizar tiempos de acceso o simplificar cuestiones de direccionamiento. Podemos decir entonces que, con una base de datos bien diseñada, se evita la redundancia perjudicial o innecesaria.

Al tener redundancia controlada obtenemos cuatro beneficios que son:

- Reducción de errores
- Datos consistentes
- Se optimiza el espacio de almacenamiento físico
- Se reducen los requerimientos de almacenamiento para respaldos

- Evitar la inconsistencia de los datos.

Se puede decir que al eliminar o controlar la redundancia, la inconsistencia en los datos no ocurrirá. Si la redundancia no se elimina, pero se controla (enterando de esto al sistema), entonces se puede garantizar que la base de datos nunca sea inconsistente, ya que cualquier cambio hecho en los datos duplicados actualizara automaticamente a los mismos en las bases de datos.

- Compartir los datos.

Esto no sólo significa que las aplicaciones existentes pueden compartir la información en la base de datos, sino que también es factible desarrollar nuevas aplicaciones que operen con los mismo datos almacenados; es decir, las necesidades de datos de las nuevas aplicaciones pueden atenderse sin tener que crear nuevos archivos almacenados.

Los beneficios que se logran son:

- Los datos fuentes pueden ser usados concurrentemente por multiples aplicaciones

- Todas las aplicaciones usaran las versiones mas actualizadas de los datos

- **Cumplir normas establecidas.**

Con un control centralizado de la base de datos se puede garantizar que se cumplan todas las formas aplicables a la representacion de los datos las cuales pueden comprender lo siguiente normas de la compañia de instalacion departamentales industriales nacionales e internacionales Es deseable unificar los formatos de los datos almacenados como ayuda para la migracion e intercambio de datos entre sistemas

- **Aplicar restricciones de seguridad.**

No es recomendable que todos los usuarios de un sistema de bases de datos puedan tener acceso a toda la informacion Se debe asegurar que el unico medio para accesar la base de datos sea a través de los canales establecidos y por tanto, definir controles de autorizacion para que se apliquen cada vez que se intente el acceso a datos esenciales Pueden establecerse diferentes controles para cada tipo de acceso a la base de datos recuperacion modificacion supresion etc

- **Conservar la integridad.**

El objetivo de la integridad es garantizar que la informacion almacenada en la base de datos sea exacta o mejor dicho consistente La inconsistencia entre dos entradas que representan un mismo hecho, es un

ejemplo de falta de integridad. La integridad de los datos combinada con una redundancia controlada nos da la habilidad de reconstruir completamente los datos si tenemos fallas en el hardware o en el software. Debemos incluir procedimientos de chequeo que aseguren que los valores de los datos se ajusten a ciertas reglas establecidas.

- Abstracción de la información.

Uno de los objetivos principales de un sistema de bases de datos es proporcionar a los usuarios una visión abstracta de la información, esto es, el sistema oculta ciertos detalles relativos a la forma en como los datos se almacenan y mantienen. Sin embargo, para que el sistema sea útil la información debe recuperarse en forma eficiente.

Los niveles de abstracción en los que puede observarse la base de datos son:

⇒ **Nivel físico o interno (Modelo Físico)** : Este es el nivel mas bajo de abstracción, el cual tiene por objetivo la representación y distribución física de los datos y la organización de estos en las unidades de almacenamiento. En este nivel se describen en detalle las estructuras de datos del nivel mas bajo.

⇒ **Nivel conceptual (Modelo Conceptual)** : En este nivel se describen los datos reales que están almacenados en la base de datos y las relaciones existentes entre los mismos. Podemos decir que es la vista lógica de los datos, completamente diferente de la organización física.

⇒ **Nivel de visión o externo (Modelo Lógico)** : El nivel de visión o externo es el mas cercano a los usuarios, es decir, el que atañe a la manera en que cada usuario ve los datos. Este nivel sirve para

simplificar la interaccion entre usuarios y el sistema el cual puede proporcionar muchas vistas diferentes de la misma base de datos

En la figura 2.1 se muestra la interrelacion entre los tres niveles de abstraccion

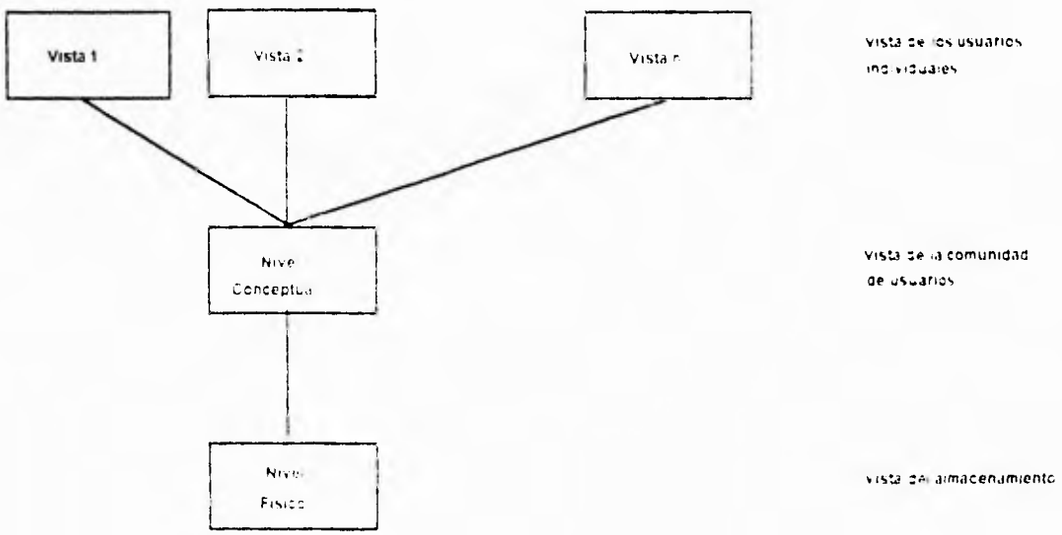


Figura 2.1 Niveles de abstracción

Podemos decir que el nivel conceptual es un nivel de mediacion entre los otros dos niveles Puede considerarse como el que define una vista de la comunidad de usuarios En otras palabras habra muchas vistas "externas", cada una compuesta por una representacion abstracta de alguna parte de la base de datos y habra solo una vista conceptual compuesta por una representacion abstracta de la base de datos en su totalidad

Así mismo, habra una sola vista física o interna, que representa la base de datos total tal como está almacenada.

- **Independencia de los datos.**

Para definir el concepto de independencia de los datos, revisaremos primero dos conceptos ligados al anterior. El primer concepto es el esquema de una base de datos, el cual se define como la descripción lógica de la base de datos o sea un diagrama de los tipos de datos que se usan y sus relaciones. El diseño general de la base de datos es un esquema de la misma. Los esquemas muy raras veces se modifican o alteran. El segundo concepto es la instancia de la base de datos, este concepto se define como el conjunto de información almacenado en la base de datos en un momento determinado.

El concepto de independencia de los datos se puede definir como sigue: es la capacidad de modificar una definición de esquema en un nivel sin afectar la definición del esquema en el nivel inmediato superior. Existen dos niveles de tal independencia.

⇒ Independencia física: es la capacidad de modificar el esquema físico de la base de datos sin obligar a reescribir los programas de aplicaciones. En algunas ocasiones son necesarios los cambios en el nivel físico para mejorar el rendimiento.

= Independencia lógica: es la capacidad de modificar el esquema conceptual sin obligar a reescribir los programas de aplicaciones. Las modificaciones en el nivel conceptual son necesarias siempre que se altera la estructura lógica de la base de datos.

Los beneficios que se obtienen al contar con independencia de los datos son los siguientes:

- Hacer cambios a los programas de aplicación y modificaciones a la base de datos es fácil.
- Dado que es fácil hacer cambios, la productividad de los programadores se incrementa.

2.3. MODELO DE DATOS.

Para describir la estructura de una base de datos, es necesario describir el concepto de modelo de datos, éste es un conjunto de herramientas conceptuales que nos ayudan a describir en forma concisa los datos, sus relaciones, su semántica y sus limitantes. Los modelos de datos se dividen en tres grandes grupos: los modelos lógicos basados en objetos, modelos lógicos basados en registros y los modelos físicos de los datos. A continuación se mencionan brevemente los diferentes modelos de datos.

2.3.1. Modelos lógicos basados en objetos

Estos modelos se utilizan para describir los datos en los niveles conceptual y de vision. Sus principales caracteristicas son las siguientes: permiten una estructuracion bastante flexible y permiten especificar claramente las limitantes de los datos. Algunos de los modelos mas conocidos son: el modelo entidad relacion, el modelo binario, el modelo semantico de los datos y el modelo infologico.

2.3.2. Modelos lógicos basados en registros.

Estos modelos se utilizan para describir los datos en los niveles conceptual y de vision. La diferencia entre estos y los modelos logicos basados en objetos, es que sirven para especificar la estructura lógica general de la base de datos como una descripcion en el nivel más alto de la implantacion.

Estos modelos no permiten especificar de manera precisa las limitantes de los datos. En la pagina siguiente se da una breve descripción de los modelos de datos de mas aceptacion.

- Modelo relacional.

En el cual los datos y las relaciones entre datos se representan por medio de tablas, cada una de las cuales tiene varias columnas con nombres unicos. Los datos son representados como una tabla de dos dimensiones. Las tablas están constituidas por columnas y renglones.

- Modelo de red.

Los datos se representan por medio de conjuntos de registros y las relaciones entre los datos se representan por medio de ligas que se pueden considerar apuntadores. En el modelo de red se permiten relaciones entre entidades 1:1(uno a uno), 1:N(uno a muchos), M:N(muchos a muchos). La ventaja del modelo de red es que refleja el mundo real y hace mas sencillo definir las estructuras de datos requeridas y sus limitantes. Su desventaja es que se pueden volver demasiado complejos.

- Modelo jerárquico.

En este modelo los datos y las relaciones entre los datos se representan por medio de registros y ligas respectivamente, de la misma forma que en el modelo de red, pero a diferencia de aquel los registros están organizados como un conjunto de arboles en vez de graficas arbitrarias. Este modelo permite usar relaciones entre los datos de 1:1 y 1:N, pero no permite usar relaciones M:N. La ventaja del modelo jerárquico radica en su sencillez y la facilidad de entenderlo. Una de sus desventajas es que no vivimos en un mundo jerárquico.

2.3.3. Modelos físicos de datos.

Los modelos físicos de datos sirven para describir los datos en el nivel mas bajo y en la actualidad no son muy utilizados. Algunos de los mas conocidos son: el modelo unificador y la memoria de cuadros.

2.4. CONCEPTOS GENERALES

2.4.1. Manejador de bases de datos.

Un manejador de bases de datos (en ingles DBMS, Data Base Management System) es un conjunto de programas que administran y controlan los datos que estan almacenados en la base de datos. Se puede decir que es una interface entre los datos de bajo nivel almacenados en la base de datos y los programas de aplicación y las consultas hechas al sistema. El manejador de base de datos debe cumplir con las siguientes tareas:

- Interacción con el manejador de archivos.

Los datos sin procesar se almacenan en el disco mediante el sistema de archivos proporcionado normalmente por un sistema operativo convencional. El manejador de base de datos traduce las diferentes proposiciones en DML (en ingles Data Manipulation Language, ver seccion 2.4.3) a comandos del sistema de archivos de bajo nivel. Así el manejador de base de datos se encarga del almacenamiento recuperacion y actualización de los datos en la base de datos.

- Implantación de la integridad.

Los valores de los datos almacenados en la base de datos deben cumplir ciertas *limitantes de consistencia*. Por ejemplo el saldo de una cuenta bancaria no debe bajar de un mínimo previamente

especificado. Estas limitantes deben especificarse en forma explícita, entonces el manejador de la base de datos puede verificar si las actualizaciones a la base de datos resultan en la violación de cualquiera de estas limitantes y si así es, podrá realizar la acción apropiada.

- Puesta en práctica de la seguridad.

No es necesario que todos los usuarios de la base de datos tengan acceso a todo su contenido. Es labor del manejador de la base de datos hacer que se cumpla los requisitos de seguridad.

- Respaldo y recuperación.

Un sistema de cómputo, como cualquier otro dispositivo mecánico o eléctrico, está sujeto a fallas, entre ellas las caídas de las cabezas lectoras del disco, la interrupción del suministro de energía y los errores de programas. En cada uno de estos casos se pierde la información de la base de datos. Es responsabilidad del manejador de la base de datos detectar estas fallas y restaurar la base de datos al estado que existía antes de presentar la falla. Esto se logra normalmente iniciando diversos procedimientos de respaldo y recuperación.

- Control de concurrencia.

Cuando varios usuarios actualizan la base de datos en forma concurrente, es posible que no se conserve la consistencia de los datos. Es necesario que el sistema controle la interacción entre los usuarios concurrentes. Lograr dicho control es una de las tareas del manejador de la base de datos.

2.4.2. Lenguaje de definicion de datos

Un esquema de base de datos se especifica por medio de una serie de definiciones que se expresan en un lenguaje especial llamado lenguaje de definicion de datos (en ingles DDL Data Definition Language). El resultado de la compilacion de las proposiciones en DDL es un conjunto de tablas que se almacenan en un archivo especial llamado diccionario de datos. La estructura de almacenamiento y los metodos de acceso empleado por el sistema de base de datos tambien se especifican por medio de este lenguaje de almacenamiento y definicion de los datos. El resultado de la compilacion de estas definiciones es una serie de instrucciones que especifican los detalles de implantacion de los esquemas de base de datos que normalmente no pueden ver los usuarios.

2.4.3. Lenguaje de manejo de datos.

Los niveles de abstraccion que se mencionaron en la seccion 2.2 no solamente se aplican a la definicion o estructuracion de los datos, sino tambien al manejo de los datos. esta manipulacion consiste en:

- La recuperacion de la informacion almacenada en la base de datos.
- La insercion de informacion nueva en la base de datos.
- La eliminacion de informacion de la base de datos.

En el nivel físico, deben definirse algoritmos que permitan tener acceso a los datos en forma eficiente. En los niveles de abstracción más altos lo importante es la facilidad de uso. El objetivo es lograr una interacción eficiente entre las personas y el sistema.

Un lenguaje de manejo de datos (en inglés, DML: Data Manipulation Language) permite a los usuarios manejar o tener acceso a los datos que estén organizados por medio del modelo apropiado. Existen dos tipos de DML:

- De procedimientos, necesitan que el usuario especifique *cuales* datos quiere y *como* deben obtenerse.
- Sin procedimientos, requieren que el usuario especifique *cuáles* datos quiere sin especificar como obtenerlos.

Los DML sin procedimientos son, por lo general, más fáciles de aprender y utilizar que los de procedimientos.

Una consulta (en inglés, query) es una proposición que solicita la recuperación de información. La parte de un DML que implica la recuperación de información se conoce como lenguaje de consultas.

2.4.4. Diccionario de datos.

Un diccionario de datos es un archivo que contiene "metadatos", esto es, "datos acerca de los datos". Este archivo se consulta antes de leer o modificar los datos reales en la base de datos. Un diccionario de datos completo incluye referencias cruzadas que indican, por ejemplo, que parte de datos utiliza cada

programa, entre otros. De hecho el diccionario puede integrarse incluso en la base de datos que describe y por tanto, incluir su propia descripción.

2.4.5. Administrador de la base de datos.

El administrador de la base de datos (en ingles DBA, Data Base Administrator) es la persona encargada de mantener el control de los datos que estan almacenados en la base de datos. Las funciones del DBA son las siguientes:

- **Definición de esquema**: es decir, la creacion del esquema original de la base de datos. Esto se logra escribiendo una serie de definiciones que el compilador DDL traduce a un conjunto de tablas que se almacenan permanentemente en el diccionario de datos.

- **Definición de la estructura de almacenamiento y el metodo de acceso**: es decir, la creacion de las estructuras de almacenamiento y metodos de acceso apropiados. Esto se lleva a cabo escribiendo una serie de definiciones que posteriormente son traducidas por el compilador del lenguaje de almacenamiento y definición de datos.

- **Modificacion del esquema y de la organización fisica**, ya sea la modificacion del esquema de la base de datos o de la descripcion de la organizacion fisica del almacenamiento. Estos cambios, aunque son relativamente poco frecuentes, se logran escribiendo una serie de definiciones utilizadas por el

compilador de DDL o por el compilador del lenguaje de almacenamiento y definición de datos para generar modificaciones a las tablas internas apropiadas del sistema

- **Concesión de autorización para el acceso a los datos**, es decir conceder diferentes tipos de autorización para acceso a los datos a los distintos usuarios de la base de datos. Esto permite al administrador de la base de datos regular cuales son las partes de la base de datos a las que a van a tener acceso diversos usuarios

- **Especificación de las limitantes de integridad**, estas limitantes se conservan en una estructura especial del sistema que consulta el manejador de base de datos cada vez que se lleva a cabo una actualización en el sistema

- **Vincularse con los usuarios**, es responsabilidad del DBA vincularse con los usuarios, garantizar que los datos que requieran estén disponibles

- **Definir una estrategia de respaldo y recuperación**, en caso de daño de alguna parte de la base de datos, es esencial poder reparar los datos pertinentes a la mayor brevedad y reduciendo al mínimo las repercusiones en el resto del sistema. El DBA debe definir y poner en marcha una estrategia de respaldo y recuperación adecuada de la información

2.4.6. Usuario de la base de datos.

Uno de los objetivos primordiales de la base de datos es crear un ambiente para la recuperacion de informacion y para almacenar informacion nueva en la base de datos. Existen tres tipos de diferentes de usuarios de un sistema de base de datos y se distinguen por el modo en que ellos esperan interactuar con el sistema.

- **Programador de aplicaciones.** Estos usuarios interactuan con el sistema mediante llamadas en DML las cuales estan incrustadas en un programa escrito en un lenguaje *huésped*. Estos programas se denominan programas de aplicación. Puesto que la sintaxis del DML es por lo comun muy diferente de la del lenguaje *huesped*, las llamadas en DML van precedidas, generalmente de un caracter especial para que un preprocesador especial llamado *precompilador* de DML pueda generar el codigo adecuado.

- **Usuarios casuales.** Son usuarios que interactuan con el sistema sin escribir programas. En cambio escriben sus consultas en un lenguaje de consulta de base de datos. Cada una de las consultas se maneja a través de un procesador de consultas cuya funcion es tomar una proposicion en DML, descomponerla en instrucciones que pueda entender el DBMS.

2.4.6. Usuario de la base de datos.

Uno de los objetivos primordiales de la base de datos es crear un ambiente para la recuperación de información y para almacenar información nueva en la base de datos. Existen tres tipos de diferentes de usuarios de un sistema de base de **datos** y se distinguen por el modo en que ellos esperan interactuar con el sistema.

- **Programador de aplicaciones.** Estos usuarios interactuan con el sistema mediante llamadas en DML las cuales estan incrustadas en un programa escrito en un lenguaje *huésped*. Estos programas se denominan programas de aplicación. Puesto que la sintaxis del DML es por lo comun muy diferente de la del lenguaje *huespeo* las llamadas en DML van precedidas generalmente de un caracter especial para que un preprocesador especial, llamado *precompilador* de DML pueda generar el codigo adecuado.

- **Usuarios casuales.** Son usuarios que interactuan con el sistema sin escribir programas. En cambio escriben sus consultas en un lenguaje de consulta de base de datos. Cada una de las consultas se maneja a través de un procesador de consultas cuya función es tomar una proposición en DML y descomponerla en instrucciones que pueda entender el DBMS.

- **Usuarios ingenuos.** Son usuarios poco complejos que interactuan con el sistema llamando alguno de los programas de aplicaciones permanentes existentes.

2.4.7. Estructura general de un sistema de base de datos.

Un sistema de base de datos se divide en módulos que se encargan de cada una de las tareas del sistema general. Un sistema de base de datos consiste en varios componentes funcionales entre los que se cuentan:

- **El manejador de archivos**, encargado de asignar espacio en el disco y de las estructuras de datos que se van a utilizar para representar la información almacenada en disco.

- **El manejador de la base de datos**, que constituye la interface 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.

- **El procesador de consultas**, que traduce las proposiciones en lenguaje de consulta a instrucciones de bajo nivel que puede entender el manejador de la base de datos.

- **El precompilador de DML**, que convierte las proposiciones en DML incrustadas en un programa de aplicaciones en llamadas normales a procedimientos en el lenguaje huesped. El precompilador debe interactuar con el procesador de consultas para generar el código apropiado

- **El compilador de DDL**, que convierte las proposiciones en DDL en un conjunto de tablas que contienen **metadatos**. Dichas tablas se almacenan despues en el diccionario de datos

Además se requieren varias estructuras de datos como parte de la implantacion del sistema físico incluyendo

- **Archivos de datos**, que guardan la base de datos

- **Diccionario de datos**, que almacenan la información relativa a la estructura de la base de datos. Se utiliza constantemente, por lo que se debe desarrollar e implantar de manera eficiente

- **Índices**, que permiten el acceso rápido a elementos de información que contienen valores determinados

La figura 2.2 de la pagina siguiente muestra los componentes antes mencionados y las conexiones entre ellos

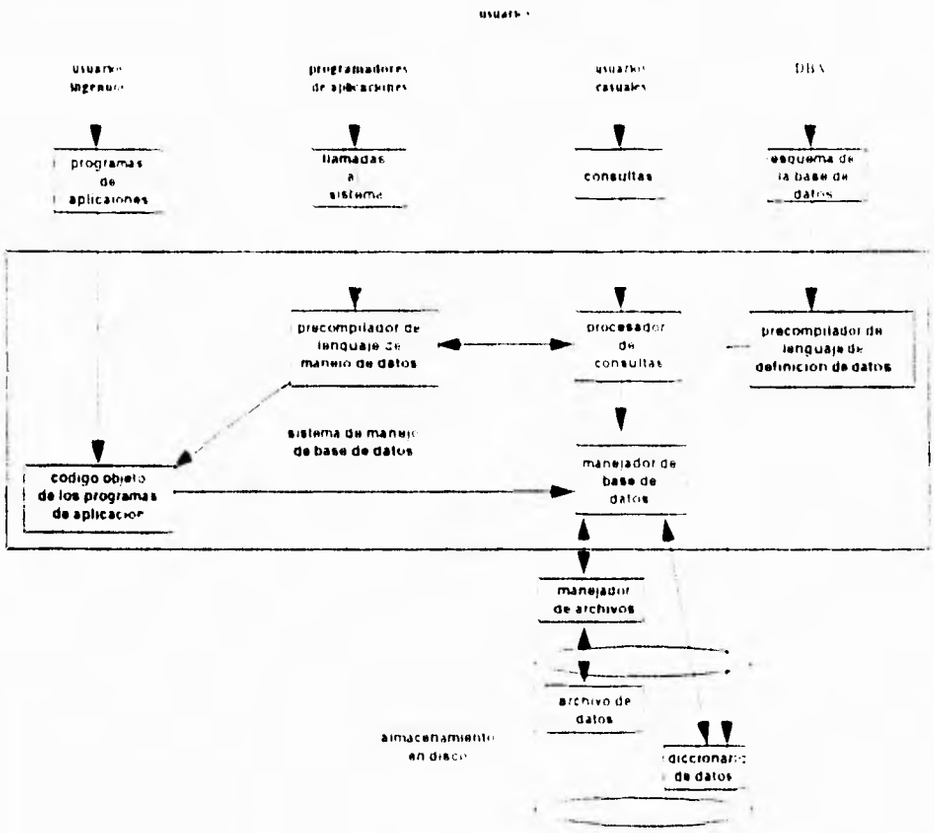


Figura 2.2 Estructura de un sistema de bases de dat.

CAPITULO 3
ESTRUCTURAS DE ALMACENAMIENTO DE DATOS Y ORGANIZACIÓN FÍSICA DE LOS DATOS.

3.1. ESTRUCTURAS DE ALMACENAMIENTO DE DATOS

El propósito de este capitulo es proporcionar una introduccion a la manera en que se pueden organizar los datos en almacenamiento secundario. Por *almacenamiento secundario* o *almacenamiento fisico de datos* se entiende el conjunto de medios actuales de acceso directo tales como paquetes de discos, tambores etc. No se consideran medios puramente secuenciales como las cintas que se utilizan en los sistemas de bases de datos como medio para las operaciones de registro de eventos de respaldo y recuperacion. Sin embargo, las cintas no se usan por lo general para almacenar la base de datos de operacion en s

Las operaciones del usuario se expresan (por medio del DML) en terminos de registros externos y el DBMS debe convertirlas en las operaciones correspondientes sobre los registros internos o almacenados. Estas últimas operaciones deben convertirse a su vez en operaciones al nivel real del hardware es decir en operaciones sobre registros fisicos o bloques. La componente responsable de esta conversion interna/física se llama *método de acceso*. El metodo de acceso se compone de un conjunto de rutinas cuya función es ocultar al DBMS todos los detalles dependientes de los dispositivos y presentarle una interfaz de registros almacenados.

3.2. DISPOSITIVOS DE ALMACENAMIENTO DE ACCESO DIRECTO.

Una base de datos puede considerarse a nivel fisico como una coleccion de archivos, registros almacenados.

Las operaciones tipicas que se realizan son

- Recuperar un registro del archivo
- Insertar un registro en el archivo
- Modificar un registro del archivo
- Leer el archivo entero
- Leer el siguiente registro del archivo
- Borrar un registro del archivo
- Reorganizar el archivo

Al seleccionar una determinada organización física de los datos el diseñador del sistema debe considerar un número importante de factores como son las características físicas de los dispositivos de almacenamiento secundario que se utilicen el sistema operativo y los métodos de acceso disponibles las necesidades del usuario con respecto a acceso y almacenamiento de la información

Los métodos de almacenamiento físico que consideraremos guardan la información en archivos que están almacenados en discos magnéticos En la actualidad los discos magnéticos son los dispositivos de acceso directo de mayor importancia y que más se utilizan para el almacenamiento secundario por esta razón describiremos las principales características de estos dispositivos

La unidad de disco que puede estar constituida por varios discos magnéticos es accesada por unas cabezas de lectura/escritura que están sujetas a un mecanismo El mecanismo controla el movimiento de éstas ya sea hacia dentro o hacia afuera con relación al centro de los discos y posiciona cada una de las cabezas de lectura/escritura en una pista específica de cada disco Una pista (en inglés track) es un area

de almacenamiento en la superficie de un disco que tiene forma circular y gira debajo de una cabeza de lectura/escritura. Todas las cabezas se mueven conjuntamente pero solamente una cabeza lee o escribe datos en un momento dado.

Un cilindro consiste en una serie de pistas que pueden ser leídas sin que el mecanismo de lectura/escritura tenga que desplazarse. En la mayoría de las unidades de disco, un cilindro está formado por un conjunto de pistas que tienen el mismo radio, lo cual implica que los discos están alineados verticalmente.

Un bloque de almacenamiento físico es la menor unidad de datos a la que puede asignársele una dirección en un disco. A esta unidad la llamaremos simplemente bloque. Normalmente cada pista está subdividida o contiene cierto número de bloques. En algunas unidades de disco el tamaño del bloque está delimitado por el hardware, mientras que en otras el tamaño del bloque se puede ajustar vía software. En este último caso se utiliza el mismo tamaño del bloque para un archivo entero. El encabezado del bloque (en inglés: block header) contiene información que permite al sistema identificar y localizar el bloque, la cual generalmente incluye un número de bloque único y la longitud de sus sub-bloques.

Los sub-bloques de datos están formados por los registros almacenados dentro de bloques. Cada registro almacenado contiene datos del usuario y datos de cabecera (en inglés: overhead) que relacionan lógicamente un registro con otros registros almacenados. Un sub-bloque de datos es la unidad de información que se transmite entre la memoria principal y la unidad de almacenamiento físico en respuesta a un comando de entrada o salida. Esto significa que el encabezado de bloque no se transmite y sirve solamente para identificar sub-bloques.

La direccion de un bloque de almacenamiento esta dada por tres factores: numero de cilindro, numero de pista y numero de bloque. La combinacion de estos numeros se conoce como direccion fisica del disco (en ingles: physical disk address). Para localizar un registro especifico en el disco, la unidad controladora del disco pasa la dirección fisica a la unidad de disco. Sin embargo, si el archivo se reorganiza o mueve a otra area entonces la direccion fisica de los bloques debe cambiarse, por esta razon conviene utilizar direcciones relativas. En este caso, cada direccion especifica una posicion del bloque de almacenamiento relativa al inicio del archivo. De esta manera, se considera que el primer bloque del archivo tiene asignada la dirección 0, el segundo bloque la dirección 1, y así sucesivamente. Asumiremos que el manejador de archivos de la computadora utiliza el directorio del disco para convertir las direcciones relativas en las direcciones fisicas requeridas.

Al número de registros de información que estan contenidos en un sub-bloque se le llama factor de bloqueaje (en inglés: blocking factor). Como se muestra en la figura 3.*, tenemos tres opciones: (a) registros sin bloqueo, (b) registros con bloqueo y (c) registros segmentados.

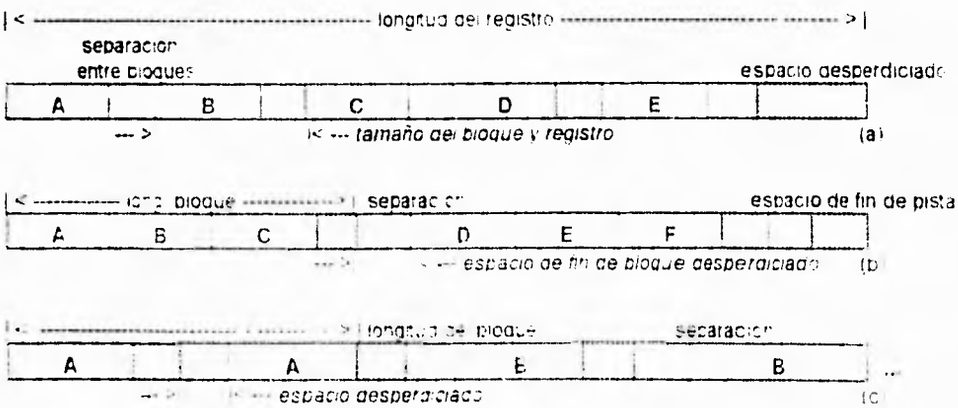


Figura 3.*

- **Registros sin bloqueo**: Cada sub-bloque corresponde a un registro de información. Si la longitud del sub-bloque es la misma que la del registro, no existirá espacio de desperdicio en cada sub-bloque. En cambio, si no son iguales las longitudes habrá espacio desperdiciado al final de cada pista.

- **Registros con bloqueo**: Varios registros son agrupados dentro de cada sub-bloque. Dependiendo de la longitud del bloque en relación con la longitud del registro existirá o no espacio desperdiciado dentro de cada sub-bloque. Así mismo se presentará espacio desperdiciado al final de la pista. El diseñador debe determinar el tamaño del bloque y el factor de bloqueaje para evitar el desperdicio excesivo de espacio.

- **Registros segmentados** (en inglés spanned records): Cada registro se divide en dos o más segmentos, cada segmento se almacena en un sub-bloque separado. Solamente se utilizan cuando la longitud del registro es mayor de la longitud prefijada del sub-bloque. Debido a las complicadas operaciones de entrada/salida, no son ampliamente usados en aplicaciones de bases de datos.

En algunos formatos de datos se utilizan sectores, que son subdivisiones del tamaño predefinido de las pistas (un tamaño típico de sector es de 512 bytes). Cada sector es direccionable. Un bloque puede consistir de uno o más sectores ligados entre sí.

Tenemos componentes de *hardware* y *software* en relación a las operaciones de entrada y salida (I/O). Los principales componentes de *hardware* de una computadora con dispositivos de almacenamiento de discos magnéticos se muestran en la figura 3.2.

3.3. CONTROL DE ENTRADAS Y SALIDAS.

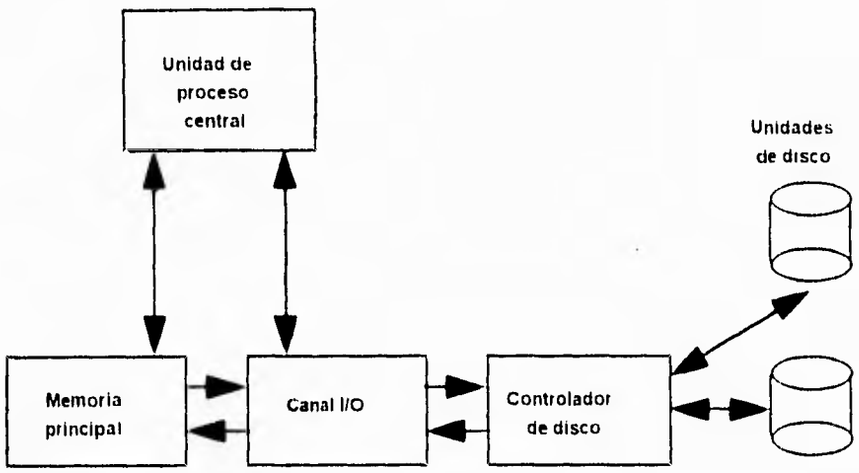


Figura 3.2. Principales componentes de hardware

La **unidad de proceso central** (CPU) que ejecuta los cálculos, operaciones lógicas y supervisa todas las operaciones con los componentes restantes de la computadora.

La **memoria principal** que consiste en un gran número de localidades de almacenamiento direccionables en donde se guardan las instrucciones de los programas y los datos. La información que está almacenada en los discos debe ser transferida a memoria principal antes de que pueda ser procesada por los programas de los usuarios.

El **procesador de entradas/salidas** (o **canal**) es esencialmente una pequeña computadora que ejecuta operaciones de entrada y de salida bajo el control del CPU dejando a éste libre para realizar otras tareas.

El **controlador de disco** controla la operación de una o mas unidades de disco. Administra a cada unidad de disco de acuerdo a sus caracteristicas especificas, como los son las instrucciones para lectura de pistas y cilindros. La mayoría de los controladores de disco que se utilizan en la actualidad son dispositivos programables que ayudan a optimizar el funcionamiento de los discos y realizan funciones de recuperación de errores.

En cuanto al software se refiere los principales componentes de un manejador de archivos en disco (ver sección 2.4.7) son

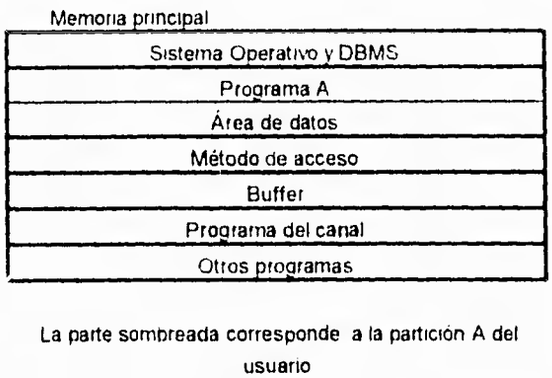


Figura 3.3 Componentes de un manejador de archivos

El **sistema operativo** es el programa que controla a la computadora y supervisa todas las actividades. El sistema operativo reserva memoria, controla la ejecucion de tareas y proporciona una gran variedad de funciones.

La mayoria de las computadoras en la actualidad usan sistemas operativos que ejecutan tareas multiples de tal forma que varios programas están en memoria principal al mismo tiempo dando la impresión de que son ejecutados simultaneamente (en realidad son ejecutados consecutivamente) Cuando un programa requiere de una operación de entrada o salida, el CPU detiene la ejecucion del programa y cede la tarea al procesador de entradas/salidas Mientras tanto, el CPU ejecuta otro programa hasta que éste requiera a su vez de una operacion de I/O Al efectuar tareas múltiples se mejora la productividad de la computadora debido a que mientras se realizan las operaciones de entrada y salida que son lentas, el CPU sigue procesando tareas

Los **programas de aplicación** son programas que realizan tareas de procesamiento para los usuarios Un programa de aplicacion puede insertar nuevos registros en la base de dato, modificar o borrar registros existentes.

El **buffer** es el área de memoria que recibe los bloques de registros desde un dispositivo de almacenamiento o transmite un bloque de registros al mismo dispositivo El buffer debe ser los suficientemente grande para poder recibir los sub-bloques de datos que se encuentran almacenados en el dispositivo de almacenamiento secundario

El **método de acceso** es un programa del manejador de archivos del sistema operativo Cuando el CPU está ejecutando un programa de aplicación y se encuentra con una instrucción de lectura (READ) o escritura (WRITE) el CPU cede el control al método de acceso Una copia del método de acceso

utilizada por el programa se conservara en la partición reservada al usuario o en la librería del sistema y estará ligada a los programas del usuario

El **programa del canal** es un programa especial que proporciona el sistema operativo y que el procesador de entradas y salidas (I/O) ejecuta. Al ejecutarse se transmite un sub-bloque de datos entre la memoria secundaria y el buffer

Cuando el espacio en memoria (partición) asignado a un programa de aplicación es insuficiente, se puede utilizar el concepto de almacenamiento virtual. Con el almacenamiento virtual, un programa (sus rutinas de acceso y áreas de datos asociadas) residen en una partición de almacenamiento externa, que se conoce como página de almacenamiento externa y que puede ser una unidad de disco "high-performance" o un dispositivo semiconductor de almacenamiento masivo. Los programas y archivos de datos se dividen en segmentos llamados páginas (un tamaño típico de página es de 4000 bytes). En un momento dado, solamente aquellas páginas que se requieren para la ejecución de un programa se encuentran en la memoria principal. Cuando el CPU requiere una página que no está en memoria, el sistema operativo virtual envía esta página a memoria principal. Mientras una nueva página del programa es leída del almacenamiento externo, el sistema operativo permite al CPU ejecutar la página de otro programa

3.3.1. Procesamiento de entradas y salidas.

Cuando el DBMS tiene que recuperar información para un programa de aplicacion, primero revisa el buffer para determinar si los datos solicitados se encuentran en memoria principal como resultado de un acceso anterior. Si el DBMS encuentra la información en el buffer, la transfiere al programa de aplicación

Un programa puede solicitar un registro que no se encuentra en memoria, debido a que

- El bloque que contiene al registro no ha sido previamente accesado por el programa

- El bloque que contiene el registro estuvo en el buffer en un momento dado pero fue removido para dejar espacio para otro bloque

En este caso, el DBMS llama a las rutinas del procesador de entradas/salidas para obtener del disco, el bloque de datos requerido. Para traer un bloque, la unidad de disco posiciona la cabeza de lectura/escritura para leer la información del disco y a través de una conexión eléctrica llamada canal, se transfiere una copia del contenido del bloque a memoria. Una vez en memoria, el DBMS envía los datos requeridos al programa de aplicación. A este acceso a disco se le conoce como entrada/salida física, debido a que la información es transferida físicamente del almacenamiento secundario a memoria principal. Cuando el registro se obtiene directamente del buffer, se le conoce como entrada/salida lógica

Una entrada/salida física requiere más tiempo del que requiere una entrada/salida lógica o del CPU. Así mismo, consume recursos al ocupar el canal que podría utilizarse para otros requerimientos

3.4 ORGANIZACIÓN DE ARCHIVOS.

La organización de archivos es una técnica para ordenar los registros de un archivo en un dispositivo de almacenamiento secundario. En la figura 3.4 se muestra un esquema de las organizaciones de archivos básicas: secuencial, indexada y directa.

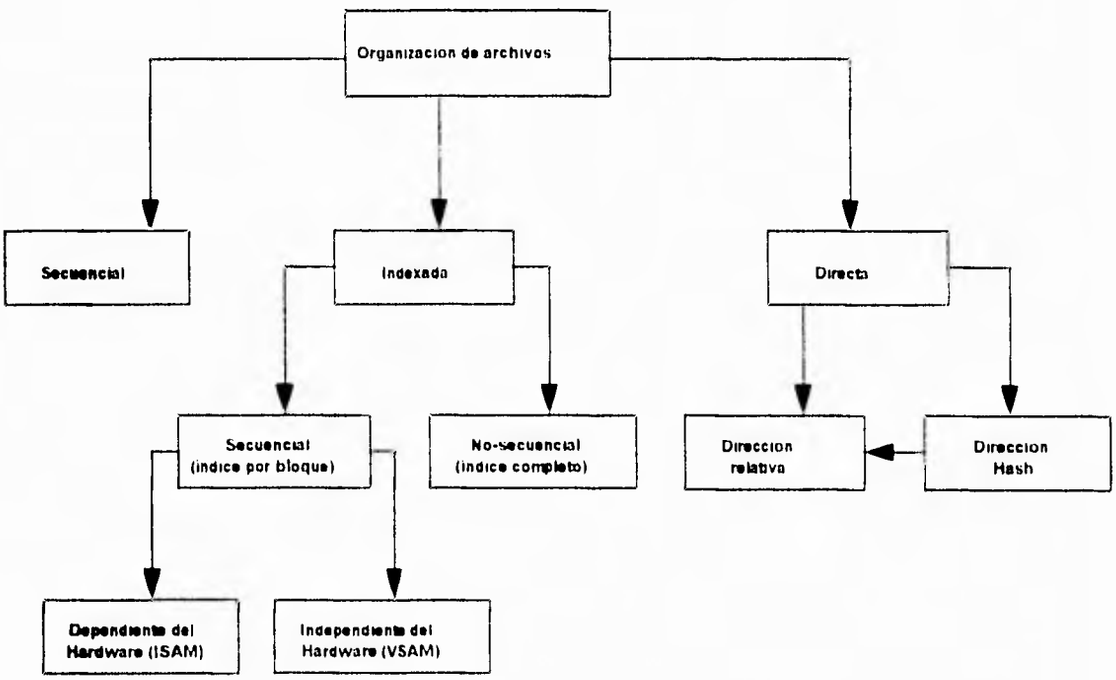


Figura 3.4 Organizaciones de archivos

En una organizacion de archivos indexados los registros pueden ser secuenciales (en este caso se utiliza indice por bloque), o no-secuencial (en este caso se requiere del indice completo)

En una organizacion directa de archivos, se utilizan con frecuencia dos esquemas dirección relativa y dirección *hash*. Cuando se usa direccionamiento tipo hash, un algoritmo de direccionamiento genera una dirección relativa (esto se indica por la flecha del direccionamiento hash al relativo de la figura 3.4)

3.4.1. Comparación de las organizaciones de archivos básicas.

En una organización de archivos secuencial el orden físico de los registros en el archivo es el mismo en el que los registros fueron escritos en el archivo. Normalmente, éste es en secuencia ascendente de la llave primaria. Sólo puede obtenerse un registro específico accesando en orden cada uno de los registros hasta encontrar el deseado

En una organización indexada secuencial, los registros también son almacenados en secuencia física de acuerdo a la llave primaria. El manejador de archivos o el método de acceso construye un índice separado de los registros de datos, que contiene valores de la llave y apuntadores a los registros. Este índice permite accesar individualmente registros sin tener que accesar todos los registros previos. Así mismo, el archivo entero puede ser accesado secuencialmente

En la organización directa con direccionamiento relativo, cada registro puede obtenerse especificando el número relativo del registro. El valor de este número es de 0 a *n*, y nos da la posición del registro relativa al

principio del archivo. El archivo puede procesarse secuencialmente, debido a que con frecuencia se ordenan los registros por la llave primaria. Por otro lado, los registros también pueden accesarse aleatoriamente cuando se usa la organización relativa conjuntamente con el direccionamiento hash.

En una organización con direccionamiento hash, el valor de la llave primaria se convierte a través de un algoritmo (en inglés, hashing routine) en un número relativo de registro. Un registro es localizado como en la organización relativa. Después de haberse aplicado el algoritmo, los registros no terminan ordenados por la llave primaria.

3.4.2. Modos de acceso a los registros.

Existen dos modos o formas básicas para accesar registros: acceso secuencial y aleatorio.

- En el **acceso secuencial**, el almacenamiento o la recuperación de información inicia en un punto específico del archivo (generalmente el principio) y continúa en secuencia lineal hasta el fin del archivo. Un registro puede recuperarse solamente recuperando todos los registros que le preceden. Normalmente, se utiliza para copiar archivos y para el procesamiento en lote (en inglés, batch) de los registros.

- En el **acceso aleatorio**, un registro dado es accesado directamente sin accesar otros registros. A diferencia del acceso secuencial, el aleatorio no sigue ningún patrón predefinido. Generalmente, se utiliza para actualización en línea y recuperación de registros. Cuando se crea un archivo se opta por una organización y rara vez se cambia. Sin embargo el modo de acceso puede cambiar cada vez que se use

el archivo, un archivo puede procesarse utilizando el acceso secuencial una vez y acceso aleatorio a la siguiente (de hecho, el modo de acceso puede modificarse de un registro al otro). Por lo tanto, es de gran importancia escoger una organización de archivos que permita un acceso eficiente de acuerdo al modo de acceso requerido. La tabla de la figura 3.5 de la página siguiente muestra las combinaciones que son permitidas en la mayoría de los sistemas.

Organización de archivos	Modos de acceso a los registros	
	Secuencial	Aleatoria
Secuencial	si	no (impractico)
Indexada secuencial	si	si
Directa-relativa	si	si
Directa-hash	no (impráctico)	si

Figura 3.5 Métodos de acceso

Como se muestra en la tabla de la figura 3.5, todas las organizaciones de archivos excepto la secuencial permiten acceso aleatorio. Todas las organizaciones excepto hash permiten acceso secuencial (a pesar de que es técnicamente posible, no es práctico debido a que los registros no están ordenados secuencialmente). Las organizaciones indexada secuencial y directa con direccionamiento relativo permiten ambos accesos.

el archivo, un archivo puede procesarse utilizando el acceso secuencial una vez y acceso aleatorio a la siguiente (de hecho, el modo de acceso puede modificarse de un registro al otro). Por lo tanto, es de gran importancia escoger una organización de archivos que permita un acceso eficiente de acuerdo al modo de acceso requerido. La tabla de la figura 3.5 de la página siguiente muestra las combinaciones que son permitidas en la mayoría de los sistemas.

Organización de archivos	Modos de acceso a los registros	
	Secuencial	Aleatoria
Secuencial	sí	no (impráctico)
Indexada secuencial	sí	sí
Directa-relativa	sí	sí
Directa-hash	no (impráctico)	sí

Figura 3.5 Métodos de acceso

Como se muestra en la tabla de la figura 3.5, todas las organizaciones de archivos excepto la secuencial permiten acceso aleatorio. Todas las organizaciones excepto hash permiten acceso secuencial (a pesar de que es técnicamente posible, no es práctico debido a que los registros no están ordenados secuencialmente). Las organizaciones indexada secuencial y directa con direccionamiento relativo permiten ambos accesos.

3.5. ESTRUCTURAS DE ALMACENAMIENTO UTILIZADAS EN INGRES.

A continuación se presentan las estructuras de almacenamiento que utiliza la base de datos relacional Ingres . Esta base de datos fue desarrollada por un grupo de investigadores de la Universidad de Berkeley en California. Ingres se desarrollo en una computadora PDP-11 con el sistema operativo UNIX. Las estructuras básicas de almacenamiento que se pueden utilizar en Ingres son las que se muestran en la siguiente tabla

TIPO DE ESTRUCTURA	DESCRIPCION
HEAP	Estructura de almacenamiento por default Acceso y entrada de datos en forma secuencial
HASH	Direccion escogida algoritmicamente basada en el valor del dato llave
ISAM	Los datos son ordenados por el valor de la columna llave para rápidos accesos en recuperaciones de valores exactos y rangos El indice es estatico y necesita modificarse cada vez que la tabla aumenta su tamaño
BTREE	Los datos son ordenados por el valor de la columna llave para rapidos acceso en recuperaciones de valores exactos y rangos El indice es dinamico y crece al parejo de la tabla

Figura 3 6 Estructuras de almacenamiento usadas en INGRES

Ingres utiliza páginas de datos las tablas se almacenan en archivos y a su vez los archivos están divididos en páginas Una página de Ingres consta de 2048 bytes, de los cuales 2008 son utilizados para los datos del usuario, 38 bytes son utilizados para los datos de cabecera (en ingles overhead) por Ingres Las páginas se dividen en registros los cuales no segmentan las páginas. la longitud del registro es igual al ancho del renglón + 2 bytes de los datos de cabecera Las páginas son muy importantes porque Ingres accesa las páginas de datos a un tiempo solo se requiere un acceso de I/O al disco para recuperar la

tabla completa y las estructuras de almacenamiento que utilizan llaves permiten el acceso directo a una página.

Las páginas principales son aquellas que originalmente estan asignadas a una estructura para tener registros. Estas páginas pueden ser accesadas directamente durante una consulta. Las páginas de desborde (en inglés overflow pages) son aquellas que se "pegan" a la tabla cuando se tiene que añadir un registro a la página principal, pero ésta ya está llena. Estas páginas se pueden generar por llaves duplicadas y causan que el tiempo de procesamiento sea mas lento

3.5.1. Estructura de almacenamiento HEAP (Secuencial).

Cuando se crea una tabla en Ingres y no se especifica que tipo de estructura es, la estructura asignada será Heap. Heap significa que no hay una llave en la tabla, son solo datos apilados. Cuando se añade un renglón a la tabla, este es añadido al final del Heap. Esto hace que esta estructura sea la más rápida cuando se añaden una gran cantidad de datos. Sin embargo, cuando se quiere obtener un renglón particular de una tabla Heap, se debe buscar a través de toda la tabla buscando al renglón correspondiente. Esto hace que esta estructura sea relativamente lenta para obtener datos en tablas que contienen muchas páginas a menos que las consultas sean siempre a todos los renglones de la tabla. Una tabla Heap consiste de una cadena de páginas de datos. La figura 3.7 muestra una tabla con estructura de tipo Heap.

empno	nombre	edad	sueldo	comentarios
17	Shigio	29	26000.00	
9	Blumberg	33	32000.00	
26	Stover	38	35000.00	
1	Mandic	46	43000.00	
16	Giller	47	46000.00	
10	Ming	23	22000.00	
27	Curry	34	32000.00	
2	Ross	50	55000.00	
19	McTigue	44	41000.00	
11	Robinson	64	60000.00	
28	Kay	41	38000.00	
3	Stein	44	40000.00	
20	Cameron	37	35000.00	
12	Saxena	24	22000.00	
29	Ramos	31	30000.00	
4	Stannich	36	33000.00	
21	Huber	35	32000.00	
13	Clark	43	40000.00	
30	Brodie	43	40000.00	
5	Verducci	55	55000.00	
22	Zimmerman	26	25000.00	
14	Kreseski	25	24000.00	
31	Smith	20	10000.00	
6	Aitken	49	50000.00	
23	Gordon	28	27000.00	
15	Green	27	26000.00	
7	Curan	20	30000.00	Despedido
24	Sabel	49	210000.0	
16	Gregori	32	31000.00	
8	McShane	22	22000.00	
25	Sullivan	28	35000.00	

Figura 3.7 Tabla con estructura Heap

Las tablas de tipo Heap no están ordenadas, no eliminan renglones duplicados, por lo que Ingres debe recorrer toda la tabla para cualquier consulta que se realice. Si se van a cargar datos en tablas vacías, esta estructura es rápida y eficiente. Si las tablas de tipo Heap ocupan de una a cinco páginas, podemos trabajar con esta estructura, de lo contrario se debe modificar a otro tipo de estructura, las cuales veremos más adelante.

La estructura Heap no hace uso del espacio que queda libre cuando se borra un renglon (excepto en la última página de datos de la tabla). Se tiene que modificar y reorganizar la tabla nuevamente para utilizar ese espacio.

Cada página se trata de llenar completamente según sea el ancho de los renglones de la tabla. esto se conoce como factor de llenado(en inglés:fillfactor) del 100%. También se puede utilizar para tablas muy grandes con índices secundarios, esto es útil en una situación en la cual una tabla es tan grande que no puede ser modificada, pero un método de acceso acelerado es necesario

Ventajas de la estructura Heap.

- Buena para tablas pequeñas(menos de 5 páginas)
- Rápidas para carga de datos
- Recomendables para producto cartesiano(joins) entre de tablas de igual tamaño
- Empaqueta la mayor parte de los datos en un espacio dado

Desventajas de la estructura Heap.

- Son lentas para las recuperaciones, ya que deben barrer siempre la tabla completa
- No remueve renglones duplicados, al menos que al crear la tabla no se permita la duplicidad de renglones.
- Los datos no están ordenados
- Sólo la página principal se considera una pagina primaria, las demás páginas son consideradas páginas de desborde(overflow pages)

- Restaurar la base de datos puede ser muy lento en algunos casos

3.5.2. Estructura de almacenamiento HASH.

Al crear una tabla con estructura Hash en Ingres se debe especificar una llave. Si la llave no se especifica, el primer campo de la tabla será usado como llave. Una llave es el campo o campos a los cuales la tabla está indexada. especificando esta llave se tiene un acceso más rápido al renglón o renglones que se están buscando. Utilizando correctamente la estructura Hash es el método de acceso más rápido para realizar consultas. Sin embargo la estructura HASH es más limitada en los tipos de consulta que puede realizar. dado que el algoritmo de HASHing no es muy poderoso para manejar rangos y valores, restricciones parciales de llaves y coincidencia de patrones. La tabla debe de ser leída completamente para este tipo de consultas.

Si una tabla de cualquier otro tipo de estructura se modifica a una estructura tipo Hash. Ingres debe realizar ciertos cálculos. Toma primero el número de renglones actuales de la tabla y calcula cuántos renglones pueden caber en una página de 2000 bytes. Ingres calcula cuántas páginas principales son necesarias, entonces el algoritmo de Hash decide en que página principal los datos van a residir calculando su dirección de Hash. Las paginas principales son páginas de datos (donde los renglones están almacenados) y son diferenciadas de las páginas de desborde(en inglés overflow pages).

Ejemplo :

Supongamos que tenemos una tabla la cual cuenta con 31 renglones. de 500 byte cada uno

Ingres calcula el número de página principales necesarias El número escogido es siempre al menos siete, no importa que pequeña sea la tabla El numero de páginas principales escogido es aproximadamente el doble del número de páginas si la estructura de la tabla fuera Heap (Hash utiliza un factor de llenado(fillfactor) del 50%) Si el ancho del renglon es mayor que 1000 el factor de llenado es del 100 %.

La fórmula utilizada es la siguiente

Páginas_principales = 2 * (renglones_de_la_tabla / renglones_por_página).

Para nuestro ejemplo, aplicando la formula.

Páginas_principales = 2 * (31 renglones_de_la_tabla / 4 renglones_por_página)

Páginas_principales = 16

Este cálculo es revisado por Ingres contra los valores del número mínimo y máximo de páginas(en INGRES se conocen como minpages y maxpages)

Supongamos que la tabla Heap 3 7 se modifica a Hash en el campo de edad Como el algoritmo de Hash que utiliza Ingres es muy complejo utilizaremos la funcion MODULO como algoritmo de HASHing

Entonces :Pág _principal = Llave MOD Págs _principales

Consideraremos que Págs _principales = 10

Ross, Edad 50 50 MOD 10 = 0 la dirección de hasn es la página 0

McShane, Edad 22 22 MOD 10 = 2 la direccion de hash es la pagina 2

Una vez terminado este proceso de hashing la tabla queda como se ilustra en la figura 3 8

Página 0	50 20 30 20	Ross Smith Curan Sabel	55000 0 0 10000 0 0 30000 0 0 21000 0 0	
Página 1				
Página 2	22 32 42	McShane Gregori Brodie	22000 0 0 31000 0 0 40000 0 0	Cadena de desborde p/página 3
Página 3	33 43 23 43	Blumberg Clark Ming Kay	32000 0 0 40000 0 0 22000 0 0 38000 0 0	23 Ramos 30000 00 53 McTigue 41000 00
Página 4	24 34 44 64	Saxena Curry Stein Robinson	22000 0 0 32000 0 0 40000 0 0 80000 0 0	

Página 5	55 35 25	Verducci Huber Kresesky	55000.0 0 32000.0 0 24000.0 0
Página 6	26 46 36	Zimmerman Mandic Stannich	25000.0 0 43000.0 0 33000.0 0
Página 7	37 47 27	Cameron Giller Green	35000.0 0 46000.0 0 26000.0 0
Página 8	38 38 28	Stover Sullivan Gordon	35000.0 0 35000.0 0 27000.0 0
Página 9	49 29	Aitken Shigio	50000.0 0 28000.0 0

Figura 3.8 Tabla con estructura HASH

Ventajas de la estructura Hash.

- Recuperaciones rápidas para coincidencia exacta de patrones

- Es recomendable para llaves anchas y llaves multicolumnas dado que la profundidad del índice no esta en función del ancho de la llave

- Mejor que la estructura Isam para llaves secuenciales a menos que las consultas involucren rangos o coincidencias parciales

Desventajas de la estructura Hash.

- Las consultas deben utilizar la llave completa para hacer uso de la estructura

- Requiere más espacio que cualquier otra estructura (50% completo por default y un factor de llenado(en inglés "fillfactor) = 100% es riesgoso)

- Es difícil predecir desborde (en ingles overflow)

- Los datos no estan ordenados

- Es muy lenta al modificarse por ser tablas muy grandes y no estar los datos ordenados

3.6.3. Estructura de almacenamiento ISAM.

Al crear una estructura isam en Ingres se debe especificar una llave de otra forma el primer campo de la tabla será utilizado como llave. Utilizando esta llave se puede restringir o realizar el producto cartesiano(join) con otra tabla se puede limitar el numero de paginas que se deben barrer para los renglones que en un momento dado califican. Isam es mas versátil que Hash. Soporta coincidencia de patrones, búsqueda de rangos y especificacion parcial de llaves multicolumnas. Esta utiliza un indice estático, el cual apunta a un numero estatico de páginas principales similar a Hash. Si una tabla de cualquier tipo de estructura la modificamos a Hash la tabla es ordenada por las llaves especificadas construyendo un indice esparcido. Este indice contiene rangos de llaves y apuntadores a las páginas índices o a la página de datos donde los renglones con ese rango de la llaves se encuentran. En la figura 3.9. se muestra una tabla con tipo de estructura Isam

Páginas	de	índices	empno	nombre	edad	sueldo	
		<= 4	1	Mandic	46	43000.00	
		= Pagina	2	Ross	50	55000.00	Pagina
		1	3	Stein	44	40000.00	de
	<= 4		4	Stannich	36	33000.00	datos 1

<= 8	> = 5	> 4 y <= 8 = Pagina 2	5 6 7 8	Verducci Aitken Curan McShane	55 49 30 22	55000 00 50000 00 30000 00 22000 00	Pagina de datos 2
> 8	<= 12	> 8 y <=12 = Pagina 3	9 10 11 12	Blumberg Ming Robinson Saxena	33 23 64 24	32000 00 22000 00 80000 00 22000 00	Pagina de datos 3
	>= 13	> 12 y <=16 = Pagina 4	13 14 15 16	Clark Kresesky Green Gregori	43 25 27 32	40000 00 24000 00 26000 00 31000 00	Pagina de datos 4
	<= 20 > 20	> 16 y <= 20 = Pagina 5	17 18 19 20	Shigio Giller McTigue Cameron	35 47 44 37	32000 00 46000 00 41000 00 35000 00	Pagina de datos 5
<= 24 > 24		> 20 y <= 24 = Pagina 6	21 22 23 24	Huber Zimmerman Gordon Sabel	35 26 28 31	35000 00 25000 00 27000 00 21000 00	Pagina de datos 6
	<= 28	> 24 y <= 28 = Pagina 7	25 26 27 28	Sullivan Stover Curry Kay	38 38 34 41	35000.00 35000 00 32000 00 38000 00	Pagina de datos 7
	> 28	> 28 = Pagina 8	29 30 31	Ramos Brodie Smith	31 42 20	30000.00 40000.00 10000 00	Pagina de datos 8

Figura 3.9 Tabla con estructura ISAM

Como dijimos anteriormente esta estructura es muy versátil porque soporta coincidencia de patrones y búsqueda de rangos. Los índices que maneja son estáticos, así como sus páginas principales. Es conveniente utilizarla cuando la tabla es relativamente estática. Esta estructura no es buena cuando la tabla crece rápidamente, la tabla es muy grande para modificarse, la llave es secuencial, esto es, cada número llave es más grande que el último y los datos no son estáticos. Cada uno de estos casos causa que se generen páginas de desborde(en inglés overflow pages)

Ventajas de la estructura Isam.

- Es buena para tablas estáticas y si las consultas involucran coincidencias parciales o busqueda de rangos

- Mejores que Btree para tablas pequeñas (menos de 10-15 páginas), usan menos espacio de disco

Desventajas de la estructura Isam.

- Los índices son estáticos y se necesitan modificar frecuentemente

- Un poco más lenta que Hash si las dos no cuentan con desborde (overflow)

- No son recomendables cuando se utilizan llaves multicolumnas

3.5.4. Estructura de almacenamiento BTREE (Árboles balanceados).

Es la estructura más versátil de Ingres. Permite accesos por llave y soporta búsqueda de rangos y coincidencia de patrones. El índice que Btree maneja es dinámico, esto es, crece conforme la tabla crece. Esto elimina problemas de desborde(overflow), que presentan las estructuras Isam y Hash cuando crecen. Los BTREES de Ingres permiten uso máximo concurrente de una tabla.

Su diseño incorpora un índice disperso, el cual apunta a las páginas en el nivel de ramas. El nivel de rama es un índice denso, el cual contiene una llave y un par de identificadores de tupla o renglón (en INGRES se les conoce como tids) apuntando a los renglones en las páginas de datos de la tabla.

Los BTREES pueden ser considerados en cuatro partes separadas que son

- Una página de encabezado, la cual es utilizada para seguir la pista de las páginas asignadas que no están siendo utilizadas
- Una o más páginas de indices
- Una o más páginas de ramas
- Una o más páginas de datos, donde los datos del usuario se encuentran almacenados

Las páginas de índices contienen una llave y un apuntador (conocido como TID), al nivel más bajo de las páginas de índice o en las páginas de ramas donde el par llave-tid se halla. Las páginas de datos contienen los renglones de datos y no contienen apuntadores a otras páginas. El Btree más pequeño siempre contendrá al menos cuatro páginas, una de cada tipo.

El número de páginas de índices es dependiente del ancho de la llave y el no de páginas de ramas, porque eventualmente las páginas de índices apuntan a una página de ramas particular. El nivel de índices es similar al de Isam, excepto que el índice de Isam apunta a las páginas de datos, mientras que el índice de Btree apunta a las páginas de ramas.

En la tabla 3.10 se muestra una tabla con tipo de estructura Btree.

RAIZ
<= McShane

Nivel de Índices

Llave	Rama	Pagina de
	<= Giller	1
	<= Giller <= McShane	2

LLave	Rama	Pagina de
	> McShane <= Shigio	3
	> Shigio	4

Nivel de Páginas de Ramas

Pagina de Ramas 1	tids	Pagina de Ramas 2	tids	Pagina de Ramas 3	tids	Pagina de Ramas 4	tids
			3,0	Mctigue	5,0		
Aitken	1,0	Gordon	3,3	Ming	5,1	Smith	7,0
Blumberg	1,1	Green	3,2	Ramos	5,2	Stannich	7,1
Brodie	1,3	Gregori	3,1	Robinson	5,3	Stein	7,2
Cameron	1,2	Huber	4,0	Ross	6,0	Stover	7,3
Clark	2,0	Kay	4,1	Sabel	6,1	Sullivan	8,0
Curan	2,1	Kreseski	4,2	Saxena	6,2	Verducci	8,1
Curry	2,2	Mandic	4,3	Shigio	6,3	Zimmerman	8,2
Giller	2,3	McShane					

Nivel de páginas de datos

Página 1	0	Aitken	1	49	50000.00	
	1	Blumberg	2	33	32000.00	
	2	Cameron	4	37	35000.00	
	3	Brodie	3	42	40000.00	
Página 2	0	Clark	5	43	40000.00	Página de datos asociada a
	1	Curan	6	30	30000.00	la
	2	Curry	7	34	32000.00	Página de Ramas 1
	3	Giller	8	47	46000.00	
Página 3	0	Gordon	9	28	27000.00	
	1	Huber	12	35	32000.00	
	2	Green	11	32	31000.00	
	3	Gregori	10	27	26000.00	
Página 4	0	Kay	13	41	38000.00	Página de datos asociada a
	1	Kreseski	14	25	24000.00	la
	2	Mandic	15	46	43000.00	Página de datos
	3	McShane	16	22	22000.00	

Tabla 3.10 Tabla con estructura BTREE

Ventajas de la estructura Btree.

- Es buena para tablas dinámicas

- No tiene problemas de desborde(overflow)

- No se necesita una clausula de ordenamiento al hacer consultas

- Buena para llaves secuenciales porque no tiene problemas de desborde(overflow)

- No necesita buscar páginas de datos cuando se buscan renglones duplicados

- Los renglones de datos no se mueven cuando la llave cambia

Desventajas de la estructura Btree.

- No es buena en tablas estáticas

- El barrido o lectura de la tabla es mas lento que en isam

- No es buena para tablas pequeñas que contengan menos de 10-15 páginas

3.5.5 Páginas de datos asociadas.

Cada página de ramas contiene una página de datos asociada. Esta página de datos es donde se añaden nuevos renglones. Una página de ramas puede apuntar a varias páginas de datos, pero los datos nuevos siempre se añaden a la página de datos asociada. Cuando una página de datos asociada se llena, una nueva página de datos asociada se "pega" a la página de ramas. Si se borran renglones en la página de datos asociada, el espacio borrado es reutilizado. Si se tiene más de un página de datos asociada por páginas de rama, entonces se cuenta con la posibilidad de que los renglones con rangos de llave similares existan en la misma pagina de datos.

CAPÍTULO 4.
EL ENFOQUE RELACIONAL Y DISTRIBUIDO.

En este capitulo se presentaran las principales caracteristicas definiciones y conceptos de las bases de datos relacionales y distribuidas. Se presenta el modelo entidad-relación para comprender algunos conceptos que nos ayudarán a entender mejor las bases de datos relacionales

4.1 EL MODELO ENTIDAD-RELACION.

El modelo de datos entidad-relacion (E-R) se basa en una percepción de un mundo real el cual consiste de objetos básicos llamados entidades y de relaciones entre estos objetos

Una entidad se puede definir como cualquier objeto que existe es distinguible y se puede representar en la base de datos. Una entidad puede ser un objeto tangible por ejemplo un alumno un articulo o un lugar pero también puede ser un objeto intangible tal como un suceso el número de cuenta de un alumno un nombre de tarea, etc

Un conjunto de entidades es un grupo de entidades del mismo tipo, por ejemplo, el conjunto de empleados de una empresa.

Las entidades se describen o representan por medio de atributos. Los posibles atributos del conjunto de entidades "empleado" son nombre numero de empleado dirección y ciudad.

Para cada atributo existe un rango de valores permitidos llamado dominio del atributo. Por ejemplo el dominio del atributo número de empleado de la entidad "empleado" podria ser el conjunto de todos los enteros positivos.

Formalmente, un atributo es una funcion que mapea un conjunto de entidades a un dominio Asi, cada entidad se describe por medio de un conjunto de parejas (atributo valor del dato) una pareja para cada atributo del conjunto de entidades

La figura 4.1 muestra parte de una base de datos de una institución bancaria, la cual esta compuesta por dos conjuntos de entidades cliente y cuenta

Ortiz	100345	Taine #56	Distrito Federal
Hernández	234577	Guerrero #343	Tijuana
Martinez	784532	Taine # 1987	Distrito Federal
Pedroza	213379	Edison # 345	Guadalajara

ENTIDAD CLIENTE

259	1000
630	2000
401	1500
700	1500

ENTIDAD CUENTA

Figura 4.1 Entidades cliente y cuenta

La entidad cliente cuenta con los siguientes atributos nombre, número de cliente, dirección y ciudad donde vive

La entidad cuenta tiene los siguientes atributos numero de cuenta y saldo

4.1.1. Relaciones y conjuntos de relaciones.

Una relación es una asociación entre varias entidades. Por ejemplo, es posible determinar una relacion que asocia al cliente "Hernández" con la cuenta 630.

Un conjunto de relaciones es un grupo de relaciones del mismo tipo. Formalmente es una relacion matemática de n >= 2 conjuntos de entidades. Si $E_1, E_2, \ldots, E_n$ son un conjunto de entidades, entonces un conjunto de relación R es un subconjunto de

$$(e_1, e_2, \ldots, e_n)$$

donde

$$\{(e_1, e_2, \ldots, e_n) \mid (e_1 \in E_1, e_2 \in E_2, \ldots, e_n \in E_n)\}$$

Para ilustrar lo anterior, considerense dos conjuntos de entidades, clientes y cuenta de la figura 4.1. Se definirá el conjunto de relaciones "ctecta" para denotar la asociación entre los clientes y las cuentas bancarias que tienen. Esta asociación se representa en la figura 4.2.

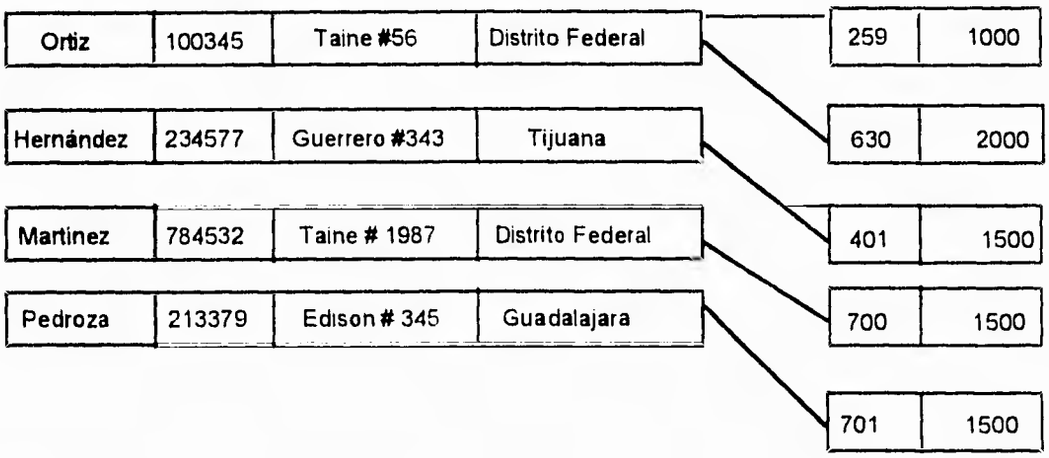


Figura 4.2 Conjunto de relaciones entre las entidades cliente y cuenta

Esta relación es un ejemplo de una relacion binaria, es decir, que involucra a dos conjuntos de entidades. La mayor parte de las relaciones en una base de datos son binarias, pero en ocasiones existen conjuntos de relaciones que incluyen a más de dos conjuntos de entidades.

Las relaciones tambien pueden tener atributos descriptivos. Por ejemplo, fecha podria ser un atributo descriptivo del conjunto de relaciones "ctecta". Esto podria especificar la ultima vez que el cliente tuvo un acceso a su cuenta.

Para ilustrar esto, podemos considerar la relacion (Hernandez, 401) se puede describir con {(fecha, 13/Abril/1993)}.

4.1.2. Limitantes de mapeo.

El modelo entidad relación puede definir ciertas limitantes con las que deben cumplir los datos contenidos en la base de datos. La cardinalidad de mapeo es una limitante que expresa el número de entidades con las que puede asociarse otra entidad mediante una relación.

Para ilustrar esta limitante utilizaremos los conjuntos de relaciones binarias, aunque pueden existir relaciones del tipo n-arias. Para un conjunto binario de relaciones ***R*** entre los conjuntos de entidades ***A*** y ***B***, la cardinalidad de mapeo debe ser una de las siguientes.

- Una a una, en donde una entidad en ***A*** está asociada con solo una entidad en ***B*** y una entidad en ***B*** está asociada con sólo una entidad en ***A*** Ver figura 4.3

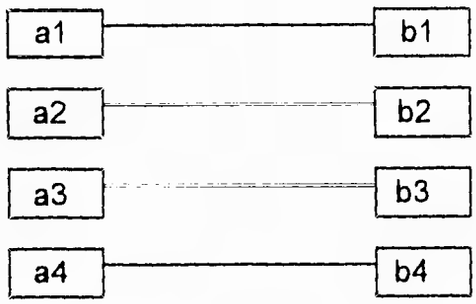


Figura 4.3 Relación una a una

- Una a muchas, en donde una entidad en ***A*** está asociada con cualquier número de entidades en ***B***, pero una entidad en ***B*** solo puede estar asociada con una entidad en ***A*** Ver figura 4.4

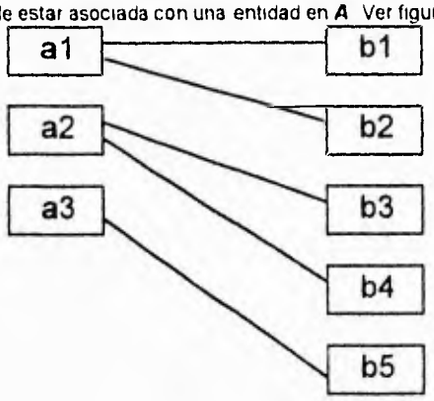


Figura 4.4 Relación una a muchas

-Muchas a una, una entidad en ***A*** está relacionada unicamente con una entidad en ***B*** , pero una entidad en ***B*** puede relacionarse con cualquier número de entidades en ***A*** Ver figura 4 5

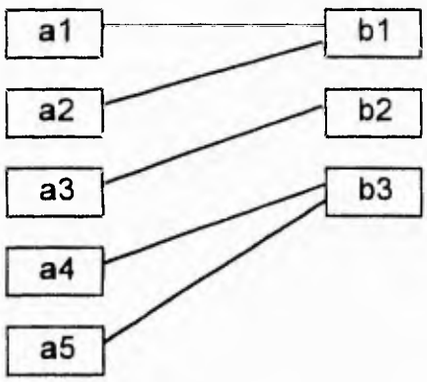


Figura 4 5 Relación muchas a muchas

-Muchas a muchas, en donde una entidad en ***A*** está asociada con cualquier numero de entidades en ***B*** y cualquier entidad en ***B*** puede estar asociada con cualquier numero de entidades en ***A*** Ver figura 4 6

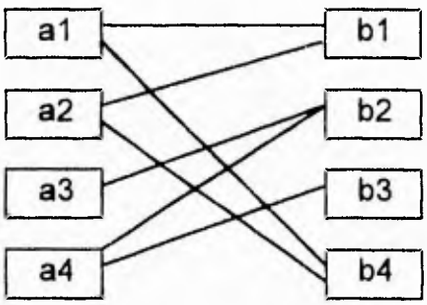


Figura 4 6 Relación muchas a muchas

Las dependencias de existencia constituyen otra clase de limitantes. Específicamente si la existencia de una entidad ***X*** depende de la entidad ***Z*** se dice que ***X*** es dependiente por existencia de ***Z***. Esto quiere decir que si se elimina ***Z***, también se elimina ***X***. Se dice que la entidad ***Z*** es dominante y que la entidad ***X*** es una entidad subordinada.

4.1.3. Llaves primarias.

Una tarea importante al modelar bases de datos es especificar como se van a distinguir las entidades y las relaciones.

La diferencia entre entidades y relaciones desde el punto de vista de bases de datos debe expresarse en términos de sus atributos.

Para realizar estas distinciones, se asigna una superllave a cada conjunto de entidades. La superllave es un conjunto de uno o más atributos, que juntos permiten identificar en forma única a una entidad dentro del conjunto de entidades.

El concepto de superllave no es suficiente para el modelado de la base de datos, ya que una superllave puede incluir atributos ajenos, esto es, si ***k*** es una superllave, entonces cualquier subconjunto ***k*** lo sería. Lo óptimo es buscar que la superllave sea lo más pequeña posible, en donde ningún subconjunto propio sea una superllave. A estas superllaves se les llama llaves candidato. El término llave primaria se utiliza para referirse a la llave candidato con la cual se identificará unívocamente a las entidades dentro de un conjunto de éstas. La llave primaria es el identificador de identidad formado por uno o mas atributos. Cada

relación tendrá alguna combinacion de atributos que tomados en conjunto, tienen las propiedad de la identificación única

4.1.4. Diagrama Entidad-Relación (E-R).

La estructura lógica general de una base de datos se puede expresar gráficamente mediante un diagrama entidad-relación, el cual consiste de los siguientes componentes

- Rectángulos Representan conjuntos de entidades
- Elipses. Representan atributos
- Rombos. Representan conjuntos de relaciones
- Lineas. Conectan los atributos a los conjuntos de entidades y los conjuntos de entidades a los conjuntos de relaciones. La figura 4 7 ejemplifica estos conceptos

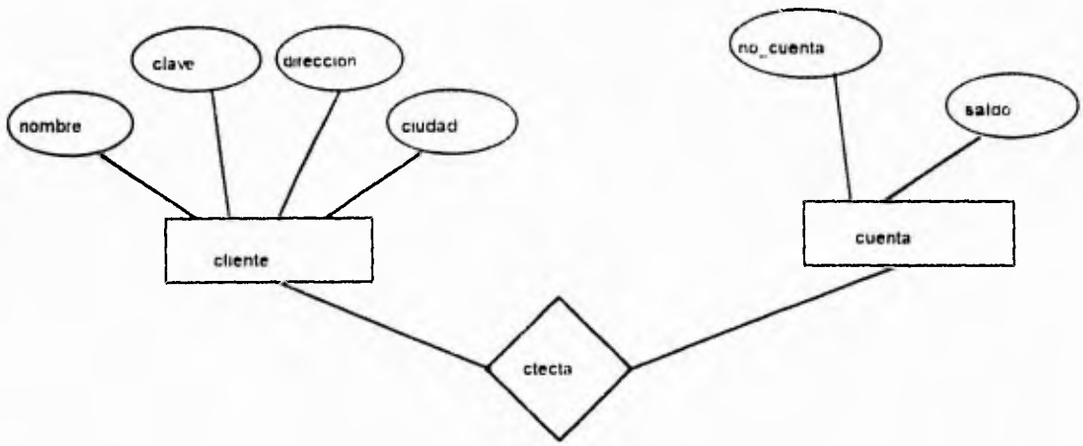


Figura 4.7. Diagrama de Entidad-Relación.

Los diagramas de entidad-relación pueden representarse por medio de un conjunto de tablas. Para cada conjunto de entidades y de relaciones en la base de datos existe una tabla única que recibe el nombre del conjunto de entidades o relaciones correspondientes. Cada tabla tiene un numero de columnas las cuales también tienen un nombre único. Utilizando la figura 4.7 obtenemos lo siguiente

NOMBRE	CLAVE	DIRECCIÓN	CIUDAD
Ortiz	100345	Taine #56	Distrito Federal
Hernández	234577	Guerrero #343	Tijuana
Martinez	784532	Taine #1987	Distrito Federal
Pedroza	213379	Edison #345	Guadalajara

ENTIDAD CLIENTE

NO_CUENTA	SALDO
259	1000
630	2000
401	1500
701	1500

ENTIDAD

CUENTA

Figura 4.8 Las tablas cliente y cuenta

4.2. EL ENFOQUE RELACIONAL.

Una base de datos relacional consiste de un conjunto de tablas. La figura 4.9 muestra un pequeño ejemplo de una base de datos relacional

Empleado

Nombre	Edad	No_depto
Torres	41	7
Strevel	31	7
Hernández	19	4

Departamento

No_depto	Gerente	Telefono
7	Torres	2936
4	Sánchez	4068

Figura 4.9 Una base de datos relacional

Una columna de una tabla representa una relación entre un conjunto de valores. Puesto que una tabla es un conjunto de estas relaciones, existe una correspondencia entre el concepto de tabla y el concepto matemático de relación, del cual recibe su nombre el modelo de datos relacional

Una de la razones por la cual el modelo de datos relacional es ampliamente aceptado es porque está basado en un modelo conceptual muy fácil de entender. Este modelo se puede resumir como sigue, una

base de datos relacional consiste de tablas Cada tabla contiene el nombre de una relacion y contiene renglones y columnas Podemos definir un esquema de relaciones como un conjunto de nombres de atributos para una relación Se pueden crear nuevas tablas a partir de las ya existentes El proceso de construir nuevas tablas en el modelo relacional está gobernado por las operaciones del álgebra relacional

En bases de datos relacionales a las tablas se les denomina relaciones. los renglones de las tablas se denominan tuplas y a las columnas se les denomina atributos Para ilustrar esto examinemos la figura 4.10 en la página siguiente. en donde vemos representados los términos anteriores

No_cuenta	Nombre_alumno	Direccion	Edad	Sexo
8340258-8	Strevel	Guerrero #343	26	M
8249486-5	Ordaz	Luis Enrique Erro #54	30	M
8721894-0	Soto	Recite # 705	24	F
8234569-9	Toledo	Samuel # 32	26	F
9076298-1	González	Bosques de Asia # 13	21	M
8834219-4	Del Razo	Rivero #21 int 4	23	M
8543256-2	Grace	Zarco # 345	22	F

Figura 4 10 Relación alumno

La tabla 4.10 tiene cinco atributos. no_cuenta, nombre_alumno. dirección. edad y sexo Para cada atributo existe un conjunto de valores permitidos, llamado dominio También tenemos una relación llamada "alumno" y cuenta con siete tuplas o renglones

Para el atributo no_cuenta, el dominio sería el conjunto de todos los números de cuenta existentes Sea D_1 este conjunto y sea D_2 el conjunto de todos los nombres de alumnos. D_3 el conjunto de todas las direcciones, D_4 el conjunto de todas las edades y D_5 el conjunto del tipo de sexo Cada una de las

columnas de la relación "alumno" debe componerse de una tupla de cinco(V_1 V_2 V_3 V_4 V_5) donde V_1 es un número de cuenta V_2 es un nombre de alumno V_3 es una direccion V_4 es una edad y V_5 es un tipo de sexo.

En general , la relacion alumno va a contener solamente un subconjunto del conjunto de todas las columnas posibles Por tanto la relacion alumno es un subconjunto de

$$\overset{n}{\underset{i=1}{X}} di$$

En general, una tabla de n columnas debe ser un subconjunto de

$$\overset{n}{\underset{i=1}{X}} di$$

En la teoría matemática se define a una relación como un subconjunto de un producto cartesiano de un listado de dominios. Esto es similar a la definición de tablas que hemos visto. la diferencia es que nosotros asignamos nombres a los atributos y la teoria matemática se basa en nombre numericos. usando el entero 1 para denotar el atributo cuyo dominio aparece primero en el listado de dominios. 2 para el siguiente y asi sucesivamente

En la relación alumno de la figura 4 10 hay siete tuplas Sea la variable de tupla t la primera tupla de la relación. Utilizaremos la notacion t[no_cuenta] para denotar el valor del atributo nombre_sucursal Asi t[no_cuenta] = 8340258-8

El Esquema_alumno se utiliza para indicar el esquema de la relación alumno. De esta forma

Esquema_alumno = (no_cuenta, nombre_alumno, direccion, edad, sexo)

El esquema de una relación es una lista de atributos y sus correspondientes dominios. El hecho de que alumno es una relación con el esquema Esquema_alumno, se expresa al escribir

Alumno(esquema_alumno)

Si se desea definir los dominios se utiliza la notación

(no_cuenta:cadena, nombre_alumno cadena, dirección cadena, edad entero, sexo cadena)

para definir el esquema de relación de la relación alumno

Una característica importante de la estructura de datos relacional es que las asociaciones entre tuplas se representan únicamente por valores de datos en columnas sacadas de un dominio comun

Ocupar atributos comunes es una forma de relacionar las tuplas de relaciones distintas.

4.2.1. Lenguajes de consulta formales.

Los lenguajes de consulta se utilizan para que el usuario obtenga información de la base de datos. Estos lenguajes se pueden clasificar en lenguajes de procedimientos o sin procedimientos

En un lenguaje de procedimientos el usuario le ordena al sistema que realice una serie de operaciones con la base de datos para obtener el resultado deseado, en cambio, en un lenguaje sin procedimientos el usuario describe la informacion que desea sin indicar un procedimiento especifico para obtenerla

Veremos los conceptos basicos del álgebra y cálculo relacional y algunos lenguajes de consulta comerciales como SQL, QUEL

4.2.2. El álgebra relacional.

El álgebra relacional es un lenguaje de procedimientos el cual consiste en un conjunto de operaciones sobre las relaciones. Cada operación toma una o más relaciones como su(s) operando(s) y produce otra relación como resultado.

Este lenguaje es de procedimientos. Existen cinco operaciones fundamentales en el álgebra relacional que son : selección, proyección, producto cartesiano, unión y diferencia de conjuntos. Todas estas operaciones producen como resultado otra relación.

Las operaciones de selección y proyección se llaman operaciones unitarias, ya que actúan sobre una sola relación, las otras tres operaciones se efectuan sobre parejas de relaciones por lo que se llaman operaciones binarias.

Examinaremos las operaciones básicas del álgebra relacional, utilizando para esto la relación cliente_dirección que se muestra a continuación.

Nombre	#cliente	Calle	Ciudad
Hughes	1712	Sepulveda	Culver
Aamco	2487	Washington	Venice
9MGM	1464	La Brea	Hollywood
Gucci	3577	Rodeo	Beverly Hills
SAG	2667	Stone Canyon	Beverly Hills

4.11 Relación cliente_dirección

4.2.2.1 Operación de selección.

La operación de selección involucra tomar una relación y seleccionar las tupias que satisfagan un predicado.

Por ejemplo, podemos seleccionar todos los clientes que estén en la ciudad de Beverly Hills de la relación cliente_dirección.

El operador de selección toma una tabla (figura 4 11) y un predicado (en este caso ciudad = Beverly Hills) como entrada y regresa otra tabla como salida

Nombre	cliente#	Calle	Ciudad
Gucci	3577	Rodeo	Beverly Hills
SAG	2667	Stone Canyon	Beverly Hills

Se pueden utilizar las comparaciones utilizando los símbolos (=,<>,<=,>=,<,>) en el predicado de la selección. Además pueden combinarse varios predicados para formar un predicado mayor utilizando los conectivos "o" e "y". (Λ y V respectivamente).

4.2.2.2. Operación de proyección.

La operación de proyección remueve ciertas columnas de una tabla. esto es. se obtiene una relación al seleccionar los atributos especificados en un orden especificado de izquierda a derecha y eliminando

luego las tuplas repetidas de los atributos seleccionados El operador de proyeccion toma una tabla y un conjunto de campos como entrada y regresa otra tabla como salida Por ejemplo

Proyectar nombre, cliente#, ciudad(cliente_dirección)

produce la siguiente tabla donde solo las columnas referidas en la proyección son incluidas La tabla resultante tiene el mismo numero de renglones que la tabla 4 11 pero menos columnas

Nombre	#cliente	Ciudad
Hughes	1712	Culver
Aamco	2487	Venice
MGM	1464	Hollywood
Gucci	3577	Beverly Hills
SAG	2667	Beverly Hills

4.2.2.3. Operación producto cartesiano.

Esta operación es binaria y, basicamente realiza la multiplicacion de dos tablas Si una tabla contiene N renglones e I columnas, y otra tabla contiene M renglones y J columnas, la tabla resultante del producto cartesiano contendrá (N*M) renglones con (I+J) columnas

Por ejemplo, si realizamos el producto cartesiano de las dos siguientes tablas

Cliente_dirección

Nombre	cliente#	Ciudad
Gucci	3577	Beverly Hills
SAG	2667	Beverly Hills

Cliente_crédito

Nombre	Crédito
Gucci	Good
SAG	Average

obtendremos una tabla de 2 * 2 = 4 renglones y 3+2 = 5 columnas, donde las primeras tres columnas son de la primera tabla y las ultimas dos columnas son de la segunda tabla

Nombre	cliente#	Ciudad	Nombre	Crédito
Gucci	3577	Beverly Hills	Gucci	Good
SAG	2667	Beverly Hills	SAG	Average
Gucci	3577	Beverly Hills	Gucci	Good
SAG	2667	Beverly Hills	SAG	Average

Al observar la tabla anterior nos damos cuenta que existen dos registros duplicados por lo que los eliminaremos quedando solo los dos primeros registros de la tabla anterior, tambien existe una columna repetida, por lo que podemos realizar una proyección para quitar una de las columnas nombre y obtenemos.

Nombre	cliente#	Ciudad	Crédito
Gucci	3577	Beverly Hills	Good
SAG	2667	Beverly Hills	Average

4.2.2.4. Operaciones de unión y diferencia.

La unión de dos tablas con N y M renglones respectivamente se obtiene concatenándolas en una tabla con un total de N+M renglones. La union entre tablas tiene sentido solo si los esquemas de las dos tablas son iguales, esto es, si tienen el mismo numero de atributos y si estos son del mismo tipo

La operación de diferencia puede ser usado para encontrar tuplas que se encuentran en una relación pero no en otra. La diferencia entre A y B se denota como A- B

Como ejemplo de diferencia podemos encontrar todas las partes en la tabla productos(ver figura 4 12 en la página siguiente) que son parte de "bulbo" pero no de "adaptador" especificando la siguiente consulta

Proyectar nombre_parte((Seleccionar producto = 'bulbo'(productos) -

(Seleccionar producto = 'adaptador'(productos))

Productos

Producto	nombre_parte	#parte	cantidad
bulbo	lever	2021	1
bulbo	sprocket	2197	3
bulbo	cog	2876	4
bulbo	spring	2346	6
adaptador	spring	2346	5
adaptador	pulley	2477	5
adaptador	rivet	2498	21
transformador	pulley	2477	3
transformador	lever	2021	1
transformador	cam	2655	3
transformador	rivet	2498	12
procesador	cpull	9876	1
procesador	8k-chip	9801	4
procesador	led	9701	4

Figura 4 12 Relación productos

El resultado es la siguiente tabla

#parte
Lever
Sprocket
Cog

4.2.2.5. Operadores adicionales.

Los operadores fundamentales anteriormente vistos, son suficientes para expresar cualquier consulta en álgebra relacional. Si utilizamos solamente estos cinco operadores básicos algunas consultas pueden producir expresiones muy largas. Es por esto que se definen operadores adicionales, los cuales simplifican algunas consultas comunes pero no hacen más potente al álgebra relacional.

El primer operador que se definirá va a ser la intersección de conjuntos. Se puede definir como sigue: $A \cap B = A-(A-B)$, donde A y B son relaciones, da como resultado una relación donde tenemos las tuplas que están en A y también están en B.

El segundo operador es el llamado producto theta, la cual es una operación binaria que permite combinar la selección y el producto cartesiano en una sola operación. Este operador forma el producto cartesiano de sus dos argumentos y después lleva a cabo una seleccion mediante el predicado dado.

4.2.3. El cálculo relacional.

El cálculo relacional es un lenguaje sin procedimientos, donde se da la descripción formal de la información deseada sin especificar como obtenerla

Existen dos formas de cálculo relacional el cálculo relacional de tuplas y el cálculo relacional de dominios, en el primero las variables representan tuplas y en el segundo las variables representan dominios Los dos tipos son muy similares por lo que solamente examinaremos el cálculo relacional de tuplas

Una variable de tupla es una variable que varia sobre alguna relacion con nombre es decir es una variable cuyos únicos valores permitidos son tuplas de esa relacion

Las expresiones del cálculo de tuplas se construyen a partir de los elementos siguientes

- Variable de tupla T,U,V,...etc Cada variable de tupla se restringe a variar sobre alguna relación con nombre. Si la variable de tupla T representa a la tupla t(en algún instante dado) entonces la expresión T A representa el componente A de t(en ese instante), donde A es un atributo de la relación sobre la cual varia T.

-Condiciones de la forma x*y, donde * es cualquiera de los simbolos (= <>,< >,<=,>=) y al menos una de entre x e y es una expresión de la forma T A y la otra es una expresión semejante o una constante.

- Fórmulas bien formadas (FBF's) Una FBF se construye a partir de condiciones, operadores booleanos(AND,OR,NOT) y cuantificadores (∀,∃), de acuerdo a las siguientes reglas

1 Toda condición es una FBF

2 Si f es una FBF entonces tambien lo son (f) y NOT(f)

3 Si f y g son FBF's, tambien lo son (f AND g) y (f OR g)

4 Si f es una FBF en la cual T aparece como variable libre (vease a cont.) entonces ∃T(f) y ∀T(f) son FBF's.

5 Ninguna otra cosa es una FBF

4.2.3.1 Variables libres y acotadas.

Cada ocurrencia de una variable de tupla dentro de una FBF es libre o acotada [Una ocurrencia de una variable de tupla es una aparición del nombre de la variable dentro de la hilera de simbolos que constituye la FBF bajo consideración. Una variable de tupla ocurre dentro de una FBF en el contexto de una expresión de la forma T.A (donde T es una variable de tupla y A es un atributo de la relación asociada), o como la variable que sigue a uno de los simbolos de cuantificación (∀,∃)]

1 Dentro de una condición, todas la ocurrencias de las variables de tuplas son libres

2 Las ocurrencias de las variables de tupla en las FBF's (f), NOT(f) son libres/acotadas segun sean libres/acotadas en f. Las ocurrencias de las variables de tupla en las FBF's (f AND g), (f OR g) son libres/acotadas segun sean libres/acotadas en f o g(cualquiera de las dos en donde aparezcan)

3. Las ocurrencias de T que sean libres en f son acotadas en las FBF's ∃T(f) y ∀T(f)s. Otras ocurrencias de variables de tupla en f son libres/acotadas en estas FBF's segun sean libres /acotadas en f.

El cuantificador existencial ∃ se lee "existe un(a). El cuantificador ∀ se lee "para todo un(a) "

A continuación se dan algunos ejemplos de consultas utilizando cálculo relacional con base en las relaciones que se muestran en la figura 4.13

Relación proveedor

S#	Nombres	Estado	Ciudad
S1	Salazar	20	Londres
S2	Jaramillo	10	Paris
S3	Bernal	30	Paris

Relación parte_proveedor

S#	P#	Cantidad
S1	P1	300
S1	P2	200
S1	P3	400
S2	P1	300
S2	P2	400
S3	P2	200

P#	Nombre_p	Color	Peso	Ciudad
P1	Tuerca	Rojo	12	Londres
P2	Perno	Verde	17	Paris
P3	Tornillo	Azul	17	Roma
P4	Tornillo	Rojo	14	Londres

Relación parte

Figura 4.13. Relaciones proveedor, parte y parte_proveedor

En los siguientes ejemplos utilizaremos las variables de tupla SX, SY, ...etc. para la relación proveedor, PX, PY...,etc. para la relación parte y SPX, SPY, ... etc. para la relación parte_proveedor

Algunas condiciones válidas en el cálculo relacional son

- SX.S# = 'S1'

- SX.S# = SPY.S#

FBF'S válidas.

- NOT(SX.CIUDAD = 'Londres')

- ∃ SPX(SPX.S# = SX.S# AND SPX.P# = 'P2')

Este último ejemplo se puede leer como sigue existe una tupla de SP con el valor de S# igual al componente S# de SX [cualquiera que este sea] y el valor de P# igual P2

- ∀ PZ(PZ.COLOR = 'Rojo')

Esta FBF se puede leer como sigue para toda las tuplas de P el color es rojo

En las dos FBF's anteriores SX, SPY y PZ son libres En la tercer FBF SX es libre y SPX es acotada

Expresiones del cálculo de tuplas.

- SX.S#. Denota el conjunto de todos los números de proveedor de la relación S

- SX.S# WHERE SX.CIUDAD = 'Londres' Denota el subconjunto de los números de proveedor para los cuales la ciudad es Londres.

- SX.S#,SX.CIUDAD WHERE ∃ SPX(SPX.S# = SX.S# AND SPX.P# = 'P2'). Denota la siguiente consulta obtenga los números de proveedor y las ciudades de los proveedores que suministran la parte P2

4.2.4. Lenguajes de consulta comerciales.

En esta sección se estudiarán las operaciones básicas de dos lenguajes de consulta comerciales que son SQL, QUEL. Estos lenguajes no solo sirven como lenguajes de consulta, sino que pueden realizar otras funciones, tales como, definir la estructura de los datos, crear las tablas, modificar los datos y especificar las restricciones de seguridad de la base de datos

El SQL esta basado en el álgebra relacional y el QUEL en el calculo relacional de tuplas, que se examinaron anteriormente

4.2.4.1. SQL

Este lenguaje se introdujo durante la mitad de los 70's por IBM, para el SYSTEM R

La estructura básica de una expresión SQL se compone de tres cláusulas

SELECT atributo1, atributo2, ..., atributoN

FROM relación1, relación2, ..., relaciónN

WHERE predicado

- La cláusula SELECT corresponde a la operación de proyección del álgebra relacional. Sirve para listar todos los atributos que se desean en el resultado de una consulta

En esta notación, la lista de atributos se puede reemplazar con un asterisco para seleccionar todos los atributos de todas las relaciones que aparecen en la cláusula FROM

- La cláusula FROM es una lista de relaciones que se van a examinar

- La cláusula WHERE corresponde al predicado de la seleccion del algebra relacional. Se compone de un predicado que incluye atributos de las relaciones que aparecen en la clausula FROM. Si omitimos la cláusula WHERE el predicado es verdadero

SQL actúa como sigue

- Forma el producto cartesiano de las relaciones que se nombran en la clausula FROM
- Realiza una seleccion del algebra relacional utilizando el predicado de la clausula WHERE
- Proyecta el resultado de los atributos de la clausula SELECT

El SQL incluye las operaciones de unión(UNION), intersección(INTERSEC) y diferencia(MINUS) y los operadores lógicos AND, OR y NOT

El SQL ofrece al usuario cierto control sobre el orden en el que se van a mostrar las tuplas de una relacion con la cláusula ORDER BY(ordenar por) la cual hace que las tuplas en el resultado de una consulta aparezca en algún orden determinado(escogiendo un atributo de una relacion), tambien permite calcular funciones de grupos de tuplas utilizando la clausula GROUP BY(agrupar por) el atributo que se da en esta cláusula sirve para formar grupos las tuplas que tengan el mismo valor para este atributo se colocan en un grupo.

- La cláusula FROM es una lista de relaciones que se van a examinar

- La cláusula WHERE corresponde al predicado de la selección del algebra relacional. Se compone de un predicado que incluye atributos de las relaciones que aparecen en la clausula FROM. Si omitimos la cláusula WHERE el predicado es verdadero

SQL actúa como sigue

- Forma el producto cartesiano de las relaciones que se nombran en la cláusula FROM
- Realiza una seleccion del álgebra relacional utilizando el predicado de la clausula WHERE
- Proyecta el resultado de los atributos de la clausula SELECT

El SQL incluye las operaciones de unión(UNION), intersección(INTERSEC) y diferencia(MINUS) y los operadores lógicos AND, OR y NOT

El SQL ofrece al usuario cierto control sobre el orden en el que se van a mostrar las tuplas de una relación con la cláusula ORDER BY(ordenar por), la cual hace que las tuplas en el resultado de una consulta aparezca en algún orden determinado(escogiendo un atributo de una relacion), tambien permite calcular funciones de grupos de tuplas utilizando la clausula GROUP BY(agrupar por), el atributo que se da en esta cláusula sirve para formar grupos, las tuplas que tengan el mismo valor para este atributo se colocan en un grupo.

El SQL incluye diversas funciones que permiten realizar ciertos cálculos

- Promedio : AVG

- Mínimo : MIN

- Máximo : MAX

- Total : SUM

- Número de tuplas : COUNT

También cuenta con operaciones de DML, las cuales no se examinarán a detalle, estas operaciones son

Insertar(INSERT), la cual permite insertar registros a la base de datos, Actualizar(UPDATE), la cual permite actualizar los registros de la base de datos que puedan cambiar y, Borra(DELETE) la cual permite borrar registros de la base de datos

4.2.4.2. QUEL.

El QUEL se introdujo como lenguaje de consulta para la base de datos INGRES y su estructura sigue muy de cerca la del cálculo relacional

Gran parte de las consultas en QUEL se expresan mediante tres tipos de cláusulas : RANGE OF(rango de), RETRIEVE(obtener) y WHERE(donde)

- Cada una de las variables de tupla se declara en una cláusula RANGE OF Se expresa RANGE OF t IS r, para declarar que t es una variable de tupla que solamente puede tomar los valores de las tuplas en r

- La cláusula RETRIEVE tiene una función similar a la cláusula SELECT del SQL(listar los atributos que se desean en el resultado de la consulta)

- La cláusula WHERE contiene el predicado de la selección

Una consulta común en QUEL tiene la forma siguiente

RANGE OF t1 IS r1

RANGE OF t2 IS r2

RANGE OF tm IS rm

RETRIEVE (ti1.Aj1, ti2.Aj2,... tin..Ajk)

WHERE p(predicado)

las ti son variables de tuplas, las ri son las relaciones y las Ajk son atributos QUEL utiliza la notación t.A para expresar el valor de la variable de tupla t en el atributo A

El QUEL no incluye operaciones del álgebra relacional como son la intersección, unión o diferencia Además, no permite consultas anidadas Se pueden utilizar los operadores lógicos AND, OR y NOT, el comando APPEND permite realizar uniones en QUEL

El QUEL cuenta con las siguientes operaciones para realizar cálculos

- Cuenta : COUNT

- Total : SUM

- Promedio : AVG

- Máximo : MAX

- Mínimo :MIN

- ANY : hace un COUNT y si la cuenta es >0 se obtiene un 1, de lo contrario se obtiene un 0

EL QUEL también cuenta con operaciones que permiten modificar la base de datos como insertar, borrar y actualizar las cuales no se analizarán.

4.3. BASES DE DATOS DISTRIBUIDAS

Cuando una organización está geográficamente dispersa, puede escoger almacenar sus bases de datos en una computadora central o distribuirlos en computadoras locales (o una combinación de ambas) Una base de datos distribuida es una base de datos a nivel lógico que está repartida en varias computadoras que se encuentran en diferentes lugares y que están interconectadas por una red de comunicaciones La red debe permitir a los usuarios compartir la información, de tal manera que un usuario (o programa) en un lugar A puede accesar y tal vez modificar información en un lugar B

Es importante distinguir bases de datos distribuidas de descentralizadas Una base de datos descentralizada también se encuentra almacenada en computadoras en distintos lugares Sin embargo, las computadoras no se encuentran interconectadas por una red, por lo tanto la información no puede ser compartida por los usuarios en los distintos sitios (en ingles *sites)* De esta manera una base de datos

descentralizada más bien es una coleccion de bases de datos independientes y no una base de datos individual distribuida geograficamente

El objetivo principal de una base de datos distribuida es proporcionar a los usuarios un fácil acceso a la información en diferentes lugares Para conseguir este objetivo, el sistema de base de datos distribuido debe ser transparente, es decir que el usuario o programa que solicita información no necesita saber en que sitio se encuentra esta Cualquier requerimiento que se haga para accesar o modificar informacion en un sitio que no sea el local, es automáticamente direccionado por el sistema al sitio correspondiente

En comparación con las bases de datos centralizadas, existen varias ventajas en las distribuidas Estas ventajas se enlistan a continuación

-Incrementar la disponibilidad. Cuando un sistema centralizado falla la base de datos no esta disponible para los usuarios Sin embargo un sistema distribuido seguirá funcionando (a un nivel reducido) aún cuando un componente falle La contabilidad y disponibilidad dependeran, entre otras cosas, en cómo la información está distribuida

-Control local. Al distribuir la informacion es necesario que las entidades locales tengan un mayor control de sus datos Esto mejora la integridad y administración de la informacion Al mismo tiempo, los usuarios pueden accesar información de otro lugar cuando sea necesario

-Crecimiento modular. Frecuentemente es más fácil y economico incluir una computadora local y la información asociada en la red distribuida, que conectarla a una computadora central

-Disminuir costos en las comunicaciones. Con un sistema distribuido, la información puede ser almacenada en el lugar donde mas se use. Esto puede reducir costos en las comunicaciones comparado con un sistema central

-Rapidez en la respuesta. Al distribuir una base de datos, se debe procurar que la información requerida por los usuarios de un cierto sitio resida en éste. Esto aumenta la velocidad de procesamiento debido a que se minimizan los retrasos en la comunicación. También es posible dividir consultas (*queries*) complejos en subconsultas (*subqueries*) que pueden ser procesados en paralelo en varios sitios, y así obtener una respuesta más rápida

-Bases de datos muy grandes. Permite tener bases de datos extremadamente grandes que nos se podrían manejar centralmente

Algunas de las desventajas de una base de datos distribuida son

-Costo y complejidad del Software. Se necesita un software más complejo (especialmente el DBMS) para un entorno de bases de datos distribuidas

-Mayor procesamiento. Todos los sitios deben intercambiar mensajes y efectuar cálculos adicionales para obtener una coordinación apropiada entre ellos

-Integridad de los datos. Una consecuencia de la mayor complejidad y coordinacion es la posibilidad de modificaciones imprevistas y problemas con la integridad de los datos

-Lentitud en la respuesta. Si la información no se distribuye correctamente de acuerdo a su uso o las consultas no son óptimas, la repuesta a un requerimiento de información puede ser muy lenta

4.3.1. Opciones para distribuir una base de datos.

Mencionamos que una base de datos distribuida es aquella repartida en varias computadoras en distintos lugares que están interconectados por una red. Como se muestra en la figura 4.14 existen diferentes maneras de configurar la red.

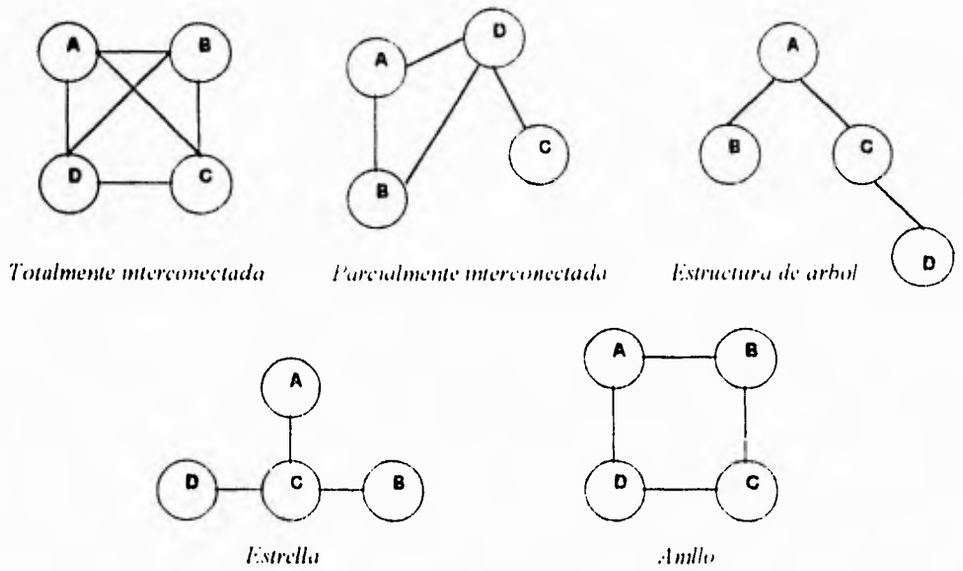


Figura 4.14 Configuraciones de red

Una red **totalmente interconectada** es aquella en la que cada sitio o computadora está físicamente conectada con cada una de las demás. Nos proporciona la mayor confiabilidad y flexibilidad, pero es la opción más costosa de instalar.

Una red **parcialmente interconectada** está conectada con algunos (pero no todos) de los sitios. Las conexiones se hacen entre los sitios que tienen mayor tráfico entre ellos.

Una red con **estructura de árbol** es un arreglo jerárquico de nodos. Usualmente se utiliza en organizaciones con una estructura jerárquica que corresponde a la red.

Una red en **estrella** conecta numerosas computadoras satélite con una central. Este tipo de red es utilizado en compañías cuyas sucursales tienen que comunicarse con una computadora central.

Una red tipo **anillo** interconecta computadoras en una malla cerrada. Con frecuencia se utiliza para conectar computadoras personales formando una red de área local (LAN por sus siglas en inglés Local Area Network).

Existen cuatro maneras básicas de distribuir una base de datos.

1. Repetición de datos.
2. Partición horizontal.
3. Partición vertical.
4. Combinación de las anteriores.

Para ilustrar las maneras de distribuir la informacion vamos a utilizar el siguiente ejemplo. supongamos que tenemos un banco con dos sucursales en el estado. La llave primaria en esta relacion es el numero de cuenta (NUM-CUENTA). SUCURSAL es el nombre de la sucursal en donde los clientes abrieron sus cuentas y en donde realizan la mayoria de sus transacciones

NUM-CUENTA	CLIENTE	SUCURSAL	SALDO
200	Jones	Lakeview	1000
324	Smith	Valley	250
153	Gray	Valley	38
426	Dorman	Lakeview	796
500	Green	Valley	168
683	McIntyre	Lakeview	1500
252	Elmore	Lakeview	330

Figura 4.15

1. **Repetición de datos.** Una opcion para distribuir informacion consiste en almacenar copias de la base de datos en dos o más sitios. De esta manera, la relación CLIENTES de la tabla de la figura 4.15 puede estar almacenada en dos o más sitios como Lakeview o Valley. Si la copia está almacenada en cada uno de los sitios tenemos *repetición total*

Tenemos dos ventajas en la repetición de datos

- *Confiabilidad*. Si alguno de los sitios falla, se dispone de una copia de la relación o base de datos en algún otro sitio

- *Respuesta rápida* Cada sitio que tenga una copia completa puede procesar transacciones localmente y por lo tanto, rápidamente

Las desventajas en la repeticion de datos son

- *Requerimientos de almacenamiento* Cada uno de los sitios que tenga una copia completa debe tener la misma capacidad de almacenamiento que se requiere para guardarlos en una computadora central

- *Costo y complejidad de la actualización* Cada vez que se haga una actualización debe realizarse en cada uno de los sitios que tenga copia, esto requiere de una cuidadosa coordinación

Por estas razones, la repetición de datos es adecuada para transacciones de lectura básicamente, donde los cambios son poco frecuentes Por ejemplo catálogos directorios telefonicos, horarios de trenes etc La tecnologia de almacenamiento de CD-ROM se está convirtiendo en un medio económico para la repetición de bases de datos

2. Partición horizontal. En una partición horizontal, algunos registros de una tabla se almacenan en un lugar y los otros en un lugar distinto, de acuerdo a cierta relación En general, los registros de una relación se distribuyen en varios sitios En las tablas de la figura 4 16 se muestra el resultado de aplicar particiones horizontales a la relación CLIENTES

NUM-CUENTA	CLIENTE	SUCURSAL	SALDO
200	Jones	Lakeview	1000
426	Dorman	Lakeview	796
683	McIntyre	Lakeview	1500
252	Elmore	Lakeview	330
Sucursal		Lakeview	

NUM-CUENTA	CLIENTE	SUCURSAL	SALDO
324	Smith	Valley	250
153	Gray	Valley	38
500	Green	Valley	168
Sucursal		Valley	

Figura 4 16

Cada uno de los registros se encuentra en la sucursal que le corresponde. Si cada cliente realiza la mayoría de las operaciones en su sucursal, éstas serán procesadas localmente y los tiempos de repuesta serán mínimos. Cuando el cliente necesita realizar una transaccion en otra sucursal, ésta tiene que transmitirse a la sucursal sede para ser procesada y la respuesta se envía de regreso a la sucursal inicial (este es el proceso de los cajeros automáticos). Si el patron de comportamiento del cliente cambia (tal vez se muda a otro lugar), el sistema debe detectar este cambio y dinamicamente mover su registro hacia donde la mayoría de las transacciones son originadas.

Las particiones horizontales que se muestran en la figura 4.16 se forman utilizando operaciones de álgebra o cálculo relacional en la relación CLIENTES original. La relación original puede ser reconstruida aplicando la operacion de union a todas sus partes.

En resumen, las particiones horizontales de una base de datos distribuidas presentan tres ventajas principales:

- Eficiencia. Los datos se almacenan en donde serán utilizados y están separados de los datos que utilizan otros usuarios o aplicaciones

- Optimización local Se pueden optimizar los accesos locales

- Seguridad Cada sitio sólo puede accesar la información que le corresponde

Se recomienda utilizar partición horizontal en una organización con funciones distribuidas, donde a cada sitio le corresponde un subconjunto de la información de acuerdo a la distribución geográfica

Las principales desventajas son

- Cuando se requiere información de varias particiones el tiempo de acceso puede aumentar significativamente con respecto al tiempo de acceso local

- Debido a que no se repite la información, cuando ocurre una falla en un sitio no puede obtenerse una copia de otro lado. La información puede perderse si no se realizan respaldos adecuados en cada sitio.

3. **Partición vertical.** Con la partición vertical la tabla se divide por columnas de acuerdo a cierta relación y éstas se guardan en distintos *sitios*. Las partes resultantes deben compartir un dominio común, de tal manera que la tabla original pueda ser reconstruida

Para ilustrar la partición vertical vamos a utilizar el ejemplo que se muestra en la figura 4.17

PARTE #	IDENTIFICADOR #
P2	123-7
P7	621-0
P3	174-3
P1	416-2

PARTE #	NOMBRE	COSTO	CANTIDAD
P2	WIDGIT	100	20
P7	GIZMO	550	100
P3	THING	48	0
P1	WHATSIT	220	16

Figura 4.17

Se utiliza a PARTE # como llave primaria en ambas relaciones. Algunos de los datos son utilizados por el departamento de manufactura mientras que los otros por ingeniería. La información está distribuida en las computadoras de estos departamentos utilizando partición vertical.

En general, las ventajas y desventajas de la partición vertical son idénticas a las de la partición horizontal. Sin embargo, en las particiones horizontales se distribuye información del mismo tipo por zonas y en las particiones verticales se distribuye información distinta con base en las funciones que realizan los departamentos de una organización.

4. Combinación de las anteriores. Se pueden realizar múltiples combinaciones de las estrategias anteriores. Parte de la información se puede almacenar centralmente, mientras otra parte puede ser repetida en varios sitios. Para una relación dada, se puede distribuir la información utilizando particiones verticales y horizontales.

El principio fundamental del diseño de bases de datos distribuidas es que la información debe estar almacenada en el lugar donde vaya a ser accesada con más frecuencia (otras consideraciones como seguridad, integridad de la información y costo también son de suma importancia). El DBA desempeña

un papel clave en organizar una base de datos para que no sea descentralizada sino realmente distribuida.

4.3.2. El entorno de las bases de datos distribuidas.

Para tener una base de datos distribuida necesitamos un sistema administrador de bases de datos distribuidas (DBMS distribuido) que coordine el acceso a la información en los distintos nodos. Es decir cada sitio requiere de un DBMS para el manejo de la base de datos local y además, un DBMS distribuido para realizar las siguientes funciones

1. Determinar de que lugar se desea obtener información
2. Si es necesario, traducir el requerimiento del DBMS de un nodo local, en el requerimiento apropiado para otro nodo que utiliza diferentes DBMS y modelo de datos
3. Administrar varias funciones como la seguridad, concurrencia y control de abrazos mortales (en inglés: *deadlocks)*, la optimización de los accesos y recuperación de fallas del sistema

Cada sitio o nodo en un sistema distribuido está sujeto al mismo tipo de fallas de un sistema centralizado Sin embargo, existe el riesgo adicional de problemas con la comunicación Para que un sistema sea robusto debe poder delectar una falla, reconfigurar el sistema para continuar con el procesamiento y recuperar la información, cuando se presenten problemas con un procesador o con la comunicación

El DBMS distribuido es el responsable de recuperar la base de datos cuando se presenta una falla. En cada sitio tiene un componente llamado manejador de transacciones que realiza las siguientes funciones:

1. Lleva un registro de las transacciones
2. Controla la concurrencia de las transacciones cuando se realizan procesamientos en paralelo, para asegurar la integridad de los datos

En transacciones globales, cada uno de los manejadores de transacciones interviene para que todas las operaciones de actualización se realicen en sincronía. Para asegurar la integridad de los datos durante la actualización, los manejadores de transacciones ejecutan un protocolo de confirmación. Este protocolo es un procedimiento que involucra intercambio de mensajes y que nos da la seguridad de que la transacción global se realiza exitosamente en cada sitio o es abortada. Un protocolo muy utilizado es el de confirmación en dos fases. El origen envía un requerimiento para procesar información a los demás sitios y cada uno procesa la parte de la transacción que le corresponde pero no actualiza su base de datos local. Cuando todos los sitios notifican al origen que ya terminaron su respectiva parte, se inicia el protocolo de confirmación en dos fases, se envía un mensaje de confirmación a todos los sitios que participaron, si todos responden que la operación se realizó con éxito (OK) se procede a la actualización de las bases de datos locales. Si alguna responde que no fue (OK) se aborta la transacción.

Los manejadores de transacciones en cada sitio deben controlar la concurrencia de las transacciones, esto se realiza a través del bloqueo o del registro del tiempo, hora exacta y sitio en que se realiza una transacción. Existen dos tipos de bloqueo, compartido y exclusivo. Cuando una transacción bloquea un

registro de modo compartido, puede leer el registro pero no escribir en el y otras transacciones pueden accesarlo al mismo tiempo. Cuando una transacción bloquea un registro de modo exclusivo puede leer y escribir en el registro pero otras transacciones no pueden accesar el registro mientras esta bloqueado

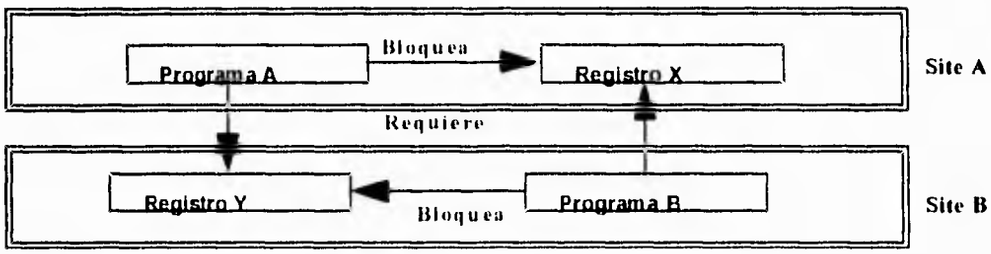


Figura 4.18

Antes de realizar una modificación, la transacción bloquea de modo exclusivo cada copia del registro a actualizar. En un entorno distribuido se incrementa la posibilidad de ocurrencia de abrazos mortales. En la figura 4.18 se muestra un ejemplo de abrazos mortales. Durante el procesamiento de una transacción el programa A en el sitio A bloquea de modo exclusivo el registro X y requiere bloquear el registro Y del sitio B. Al mismo tiempo, el programa B en el sitio B ha bloqueado de modo exclusivo el registro Y requiere bloquear el registro X. Cuando esto sucede se presenta un abrazo mortal y se detiene el procesamiento hasta que sea eliminado. Para poder detectar y resolver un abrazo mortal, debe tenerse un mecanismo de detección global que recopile información de los sitios involucrados

La segunda opción, consiste en asignar un identificador del sitio y hora exacta a cada transacción. El propósito de este registro de tiempo es asegurarse que las transacciones sean procesadas en orden serial, evitando bloqueos y abrazos mortales. Cada registro de la base de datos guarda el identificador de tiempo de la última transaccion que lo modificó. Si una nueva transaccion trata de actualizar un registro y su identificador de tiempo es anterior al que guarda el registro, se le asigna un nuevo identificador a la transacción y se inicia nuevamente. De esta manera, una transacción no puede procesar un registro hasta que su identificador de tiempo sea posterior al del registro y por lo tanto no puede interferir con otra transacción. La ventaja principal es eliminar bloqueos y abrazos mortales, la desventaja es que las transacciones que entran en conflicto deben ser canceladas y reinicializadas, lo que aumenta el tiempo de procesamiento y respuesta del sistema.

La administración de bases de datos distribuidas es una necesidad en la comunicacion de datos de cara al futuro. Adelantos recientes en DBMS para computadoras personales que conectan la base de datos de la PC, con una base de datos asociada a una mainframe, están marcando una nueva dimension en la bases de datos distribuidas. De cualquier manera, la distribucion de bases de datos será necesaria para gran número de entornos que no son del todo eficientes con el procesamiento centralizado de información. Algunos DBMS actualmente a la venta, tienen versiones distribuidas que proporcionan algunas de las facilidades que hemos mencionado. Con la tecnologia actual, el mismo tipo de DBMS debe estar instalado en cada nodo de la red para usarse en forma distribuida, pero se está buscando permitir el uso de un conjunto heterogéneo de productos.

4.4. BASES DE DATOS ORIENTADAS A OBJETOS.

En los últimos años se ha desarrollado una nueva tendencia en relacion a los modelos de datos y sistemas de bases de datos. Desde esta perspectiva, una base de datos es considerada como una colección de objetos, donde cada objeto representa una entidad física, concepto, idea, evento o algun aspecto de interés para una aplicación de bases de datos. El termino "orientado a objetos" (siglas en inglés: **O-O**) es utilizado en varias disciplinas, incluyendo lenguajes de programación, bases de datos, inteligencia artificial y sistemas de computo en general. La diferencia principal entre los lenguajes y bases de datos orientadas a objetos es que éstas requieren del almacenamiento permanente de los objetos en memoria secundaria, mientras que los objetos en un lenguaje existen solamente durante la ejecución del programa.

En la actualidad, existen varias propuestas de modelos de datos y sistemas de bases de datos orientados a objetos cuyas principales caracteristicas son

- *Abstracción de datos y encapsulado*. Se refiere a la habilidad para definir un conjunto de operaciones, llamadas **métodos** en la terminología O-O, que pueden ser aplicadas a los objetos de una **clase** en particular (tipo de objeto). Todos los accesos a objetos se realizan a través de los métodos predefinidos. Cada objeto tiene dos partes: la *implantación* que es privada, almacena el estado del objeto y puede cambiarse sin afectar a la otra parte, que es la *interface*. Todos los accesos a un objeto son via su interface que es pública, es decir, es reconocida por otros objetos y por los usuarios del sistema. Muchos lenguajes de programacion permiten definir tipos abstractos de datos

proporcionando construcciones especiales para el encapsulado de estructuras y operaciones. De esta manera, el encapsulado es el principio de modelar los datos y las operaciones al mismo tiempo.

- *Identidad de los objetos*. Es la propiedad que distingue a un objeto de cualquier otro del sistema. Cada objeto tiene una identidad, que es independiente de su valor y de sus **atributos** (propiedades).

- *Herencia*. Los objetos que pertenecen a una subclase heredan los atributos y los métodos de las clases superiores del mismo tipo. De esta manera, la herencia permite que objetos con distintas estructuras compartan operaciones de acuerdo a su parte común.

- *Objetos complejos*. Es la habilidad de definir nuevos objetos compuestos, a partir de objetos previamente definidos, ya sea de forma jerárquica o anidada.

- *Intercambio de mensajes*. Los objetos se comunican y ejecutan operaciones de obtención de datos, cálculos y actualizaciones, intercambiando mensajes. Un **mensaje** es una petición dirigida a un receptor para que cambie su estado, regrese un resultado o ambos. La forma en que un objeto responde a un mensaje es a través del método o procedimiento que llama cuando recibe el mensaje.

- *Sobreposición de operadores*. Se refiere a la característica de utilizar el mismo nombre para realizar distintas operaciones con diferentes tipos de objetos. La sobreposición se resuelve cuando se determinan las propiedades específicas del objeto al que se va a aplicar el operador.

Las ventajas principales del enfoque orientado a objetos son:

- *Amplitud*. Los tipos de objetos y sus métodos pueden ser modificados cuando sea necesario. Es más sencillo identificar el tipo de objeto y realizar los cambios necesarios, mientras que en los sistemas basados en registros, muchos tipos de registros pueden resultar afectados. Además, nuevas clases de objetos y sus métodos pueden incorporarse en el sistema.

- *Comportamiento restringido*. Debido al encapsulado, el comportamiento de cada tipo de objeto está determinado por un conjunto de métodos predefinidos. De esta manera, las operaciones en la base de datos están restringidas a este comportamiento específico.

- *Flexibilidad en la definición de tipos de datos*. El usuario no está limitado a cierto modelo de datos, puede definir muchos tipos de datos cada uno con propiedades únicas.

- *Modelado*. La herencia de atributos y métodos es una herramienta muy poderosa en cuanto al modelaje de datos se refiere.

Las principales desventajas del enfoque orientado a objetos son:

- *Pérdida de asociaciones*. Las asociaciones o relaciones entre los distintos tipos de objetos se generan indirectamente por medio de referencias inter-objetos. Es una debilidad inherente al enfoque O-O, debido a que trata a cada objeto como unidad autosuficiente de información.

- *Rigidez en el comportamiento* El hecho de determinar todas las operaciones a traves de un conjunto predefinido de métodos, es una limitante a la naturaleza dinámica y al desarrollo tecnologico de las bases de datos

- *Ausencia de DBMS de alto nivel* No existen DBMS de alto nivel para los modelos de datos O-O actuales. El poder y la elegancia de los lenguajes relacionales de acceso a bases de datos no tiene su contraparte en los sistemas O-O

CAPÍTULO 5.
CONSIDERACIONES PARA EL DISEÑO Y LA IMPLANTACIÓN.

5.1. CONSIDERACIONES PARA EL DISEÑO Y LA IMPLANTACIÓN.

El diseño de bases de datos es un proceso de modelado complejo y dependiente del entorno, cuya finalidad es reducir la redundancia en la información y proveer estabilidad, flexibilidad, funcionamiento aceptable y facilidad de uso al sistema de bases de datos.

El proceso de diseño e implantación de bases de datos consiste en seis pasos, estos son:

1. Analizar las funciones del negocio y determinar los requerimientos de datos.
2. Definir las entidades principales de los sistemas.
3. Definir las relaciones entre las entidades.
4. Definir los atributos de las entidades.
5. Definir el flujo de información y la ruta de acceso de las entidades.
6. Diseño físico de la base de datos.

En el diseño de sistemas y la implantación de bases de datos, el término "modelo de datos" se utiliza de manera diferente. En la literatura sobre la tecnología de DBMS, en ocasiones se hace referencia a las estructuras de datos como modelos de datos, el modelo de red, el modelo jerárquico y el modelo relacional. Sin embargo, vamos a hacer uso de este término con un significado orientado al diseño de bases de datos. Un modelo de datos es un modelo de las entidades y de sus relaciones, que son objeto de interés para una organización.

En el desarrollo de sistemas de información, se pueden utilizar otros modelos de datos para facilitar el análisis. Vamos a describir brevemente algunos de estos modelos, sin entrar al detalle debido a que no forman parte de los objetivos de este capítulo

Modelo Corporativo de Datos (CDM)

Este modelo describe de manera general la naturaleza y las relaciones entre los datos sin considerar aspectos propios de la computación. El modelo describe la importancia de los distintos objetos del negocio, identifica cuáles son fundamentales para el buen desempeño de este y define las relaciones entre los objetos.

Modelo de Datos por Agrupación de Sistemas (SGDM)

Es un subconjunto del Modelo Corporativo que agrupa a las entidades de datos y a las relaciones que son comunes para un grupo de aplicaciones

Modelo de Datos de Proyectos (PDM)

Es un subconjunto del Modelo por Agrupación de Sistemas que agrupa a aquellos objetos del negocio, relaciones, atributos llave y datos que se aplican a un proyecto específico

Modelo de Datos de las Funciones del Negocio (BFDM)

Es un subconjunto del Modelo de Proyectos que incluye a aquellos objetos del negocio, relaciones, atributos llave y de datos, requeridos para realizar una función específica.

Vista Lógica de los Datos

Describe a los elementos de datos, registros y relaciones en términos de un modelo de bases de datos específico (red, jerárquico o relacional) o de la estructura de archivos utilizada

5.1.1. Analizar las funciones del negocio y determinar los requerimientos de datos.

Una función de negocio es un grupo de actividades relacionadas entre sí, que se realizan para lograr un fin dentro de una organización

El área de compras dentro una organización es un ejemplo de función de negocio, en esta se realizan ciertas actividades como lo son

- Mantener un registro actualizado de los productos, servicios, precios y calidad de los proveedores, procurando tener buenas bases para la selección de un proveedor
- Definir objetivos a conseguir para cierto periodo de tiempo, márgenes de utilidad, descuentos, volúmenes, etc.
- Generar un reporte de evaluación de desempeño de los compradores y proveedores

Algunos métodos para recopilar información acerca de las funciones del negocio son realizar entrevistas con los usuarios, aplicar cuestionarios, revisar documentos, manuales de políticas y procedimientos, analizar sistemas existentes, observación directa, etc

Mediante este análisis se pretende obtener suficiente información del negocio y de los requerimientos del usuario para:

- Definir el esquema general del negocio
- Delimitar las aplicaciones de los sistemas de computo

- Determinar qué procesos del negocio serán realizados por computadoras
- Producir información de costo/beneficio

5.1.2. Definir las entidades principales de los sistemas.

El estudio de las funciones del negocio nos ayuda a obtener un mayor conocimiento de una organización y nos permite definir las principales entidades que interesan a ésta. La información requerida se identifica en el análisis de los datos, que se realiza con personal familiarizado con el entorno del negocio y se agrupa dentro de entidades. Las entidades son descripciones de personas, cosas o eventos de interes para una organización.

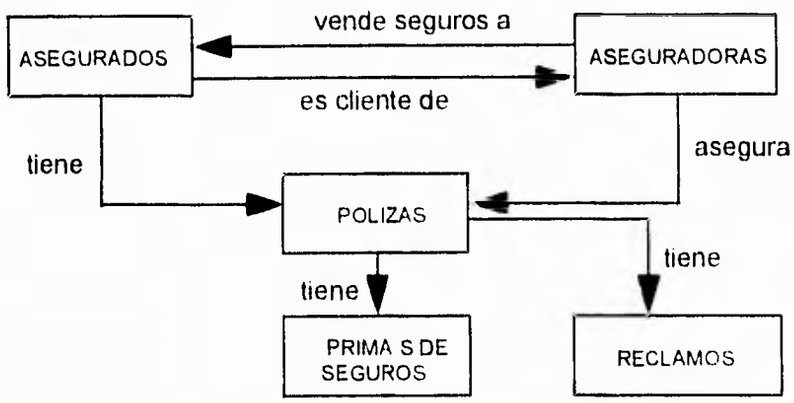
Se presenta un ejemplo de compañías de seguros, en donde se obtuvieron las siguientes agrupaciones de datos o entidades

ASEGURADORAS	ASEGURADOS	POLIZAS	PRIMA S DE SEGUROS	RECLAMOS

5.1.3. Definir las relaciones entre las entidades.

Una vez que se han identificado las entidades de la organización, definimos las relaciones entre ellas. Cuando consideramos las ligas entre entidades, incrementamos la información previa. Por ejemplo, la relación entre todos los asegurados que son clientes de una aseguradora nos da un mayor significado que cada entidad por sí sola. Como se puede notar, una relación es una asociación entre entidades.

Pueden existir relaciones múltiples entre las entidades. cada una representa una asociacion entre las entidades por razones distintas del negocio Un asegurado puede tener varias pólizas y ser cliente de distintas aseguradoras. Una aseguradora vende seguros a muchos clientes y asegura muchas pólizas Una póliza tiene varios reclamos y primas asociados a ésta



5.1.4. Definir los atributos de las entidades.

Los elementos de datos que describen a una entidad se conocen como atributos. Después de identificar las entidades y sus relaciones. debemos describir las entidades en términos de sus atributos Los atributos se obtienen en función de las entradas y salidas del sistema Por ejemplo. la entidad Aseguradoras tiene los siguientes valores

Atributos de la Entidad ASEGU RADORAS				
No. Aseguradora	**Dirección**	**Código**	**Tipo**	**Prima**
G164289	**110 Main Sl. San Francisco, CA**	**041**	**Auto Policy**	**100,000,000**
V a l o r e s				

Los elementos de datos de Asegurados. Pólizas. Reclamos y Primas se describen en forma similar En este punto se aplica la normalización

- Normalización.

Normalización es un proceso que reduce relaciones complejas en formas simples Nos permite agrupar los datos de la manera más simple posible. de tal manera que puedan hacerse cambios futuros en los requerimientos de procesamiento con un impacto mínimo en las estructuras de datos La razón principal de partir un registro (entidad) en uno o más registros es para evitar los problemas de mantenimiento que se presentan cuando hay entidades implícitas dentro de otras entidades Entrando a detalle, el proceso de normalización se lleva a cabo en las tres fases que se presentan a continuación.

I) Pnmera Forma Normal

Se pretende aislar en nuevas entidades a los atributos (o grupos de datos) que se repiten. partiendo la entidad en dos o más entidades Por ejemplo, vamos a considerar un sistema cuya finalidad es registrar

las horas que le toma a cada empleado realizar ciertas tareas especificas. La información requerida es el número del empleado, apellido, división a la que pertenece, gerente de la división, identificador de la tarea lugar de trabajo y horas invertidas. Estos datos están organizados de la forma siguiente

EMPLEADO				TAREA			TAREA		
NÚM.	APELLIDO	DIV.	GTE.	IDENT.	LUGAR	HORAS	IDENT.	LUGAR	HORAS

La llave = 

Notemos que los primero cuatro atributos tendrán datos que rara vez van a cambiar. Sin embargo los tres siguientes: IDENT., LUGAR y HORAS tienen datos que frecuentemente pueden hacerlo. Así mismo estos elementos se repiten para cada tarea, son grupos repetitivos que deberían separarse de la entidad EMPLEADO.

Si algún empleado estuviera asignado a más de dos tareas, tendríamos que modificar todos los registros de empleados para ajustarlos al cambio. Podrían utilizarse registros variables pero éstos requieren de procesamiento adicional. Para evitar este problema podemos reestructurar los datos empleando la primera forma normal y aislando los grupos repetitivos. Se obtienen dos archivos con registros distintos:

EMPLEADO

NÚM.	APELLIDO	DIV.	GTE.

EMPLEADO/TAREA

NÚM.	IDENT.	LUGAR	HORAS

Observemos que aislando los grupos repetitivos, se elimina el problema de modificar el tamaño de los registros. En vez de anexar campos (columnas) a los registros previos, solamente necesitamos dar de alta un nuevo registro.

Es importante resaltar que al construir el nuevo archivo EMPLEADO/TAREA, conservamos la llave NÚM De otra manera, no habría forma de relacionar los registros de este archivo con aquellos del archivo de EMPLEADO. De tal forma que el archivo utiliza una llave concatenada, que se formó de la combinación de dos atributos número de empleado e identificador de tarea

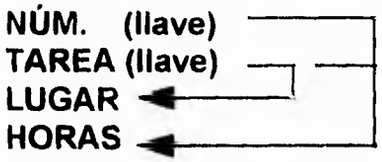
Las razones por las que se usa la primera forma normal son

- Los grupos repetitivos que son parte de un registro crean problemas El numero de ocurrencias tiende a crecer más allá de los límites originalmente esperados La nueva agrupación, disminuye el impacto de aumentar el tamaño de los registros
- Los DBMS no son receptivos de grupos repetitivos La solución frecuente es reservar un área grande en los registros y "esconder" los grupos repetitivos en esta área, lo que genera espacio de almacenamiento desperdiciado.
- Con frecuencia una entidad repetida posee un interés intrínseco para la compañía, y su aislamiento como entidad separada nos es útil

II) Segunda Forma Normal

Una vez que hemos aislado los grupos repetitivos dividiendo una relación compleja en la primera forma normal, podemos simplificar aún más la relación. Para pasar a la segunda forma normal, debemos considerar los atributos que no son llave y que son dependientes de una llave concatenada.

Observemos los archivos que resultaron de la primera forma normal. Como no existe una llave concatenada en el archivo EMPLEADO, todos los atributos están en función de la llave NÚM. (número de empleado). En el archivo EMPLEADO/TAREA, el atributo HORAS es dependiente de NUM. y de IDENT debido a que necesitamos ambos datos para determinar cuántas horas necesita un trabajador para desempeñar una tarea. El atributo LUGAR depende únicamente de una parte de la llave concatenada que es IDENT. Estas dependencias se muestran a continuación.



Notemos que HORAS depende de la combinación de NÚM. (llave) y de TAREA (llave) que es una llave concatenada. Sin embargo, LUGAR sólo de TAREA y no de NÚM.

Para la segunda forma normal, nos aseguramos de que *los atributos que no son llave de una entidad, son dependientes de todos los atributos llave de la entidad*. Si no se presenta esta situación, entonces dividimos los atributos en entidades más simples.

Para nuestro ejemplo, podemos observar que LUGAR no depende de la llave completa. Esto resulta en tres problemas:

1. Una nueva tarea no puede ser registrada hasta que se asigne un número de empleado. Se conoce como anomalía de inserción
2. Si un empleado, que es el único que desempeña cierta tarea deja de realizarla, entonces el detalle de la tarea se pierde. Se conoce como anomalía de borrado.
3. Si el detalle de una tarea cambia (p.e., lugar), entonces todos los registros con esta información deberán modificarse. Se conoce como anomalía de actualización.

Para solucionar los problemas, creamos nuevas entidades para cada atributo que no es la llave y que no depende de la llave concatenada. Separamos al atributo LUGAR y lo dejamos en otro archivo, su llave es IDENT. El resto de los atributos NÚM (llave), IDENT. (llave) y HORAS quedarán, después de aplicar la segunda forma normal, de la forma siguiente:

EMPLEADO

NÚM.	APELLIDO	DIV.	GTE.

EMPLEADO/TAREA

NÚM.	IDENT.	HORAS

TAREA

IDENT.	LUGAR

De esta manera, se resuelven los inconvenientes señalados con anterioridad.

Para nuestro ejemplo, podemos observar que LUGAR no depende de la llave completa. Esto resulta en tres problemas:

1. Una nueva tarea no puede ser registrada hasta que se asigne un número de empleado. Se conoce como anomalía de inserción.
2. Si un empleado, que es el único que desempeña cierta tarea deja de realizarla, entonces el detalle de la tarea se pierde. Se conoce como anomalía de borrado.
3. Si el detalle de una tarea cambia (p.e., lugar), entonces todos los registros con esta información deberán modificarse. Se conoce como anomalía de actualización.

Para solucionar los problemas, creamos nuevas entidades para cada atributo que no es la llave y que no depende de la llave concatenada. Separamos al atributo LUGAR y lo dejamos en otro archivo, su llave es IDENT. El resto de los atributos NÚM. (llave), IDENT. (llave) y HORAS quedarán, después de aplicar la segunda forma normal, de la forma siguiente:

EMPLEADO

NÚM.	APELLIDO	DIV.	GTE.

EMPLEADO/TAREA

NÚM.	IDENT.	HORAS

TAREA

IDENT.	LUGAR

De esta manera, se resuelven los inconvenientes señalados con anterioridad.

1. Podemos insertar tareas sin asignar a un empleado.

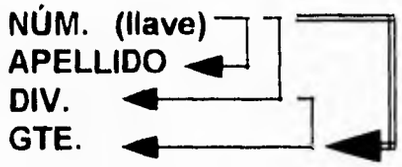
2. Si el archivo LUGAR cambia, solamente tenemos que cambiar un registro en el archivo de TAREA.

3. Si un empleado deja una tarea, la información de ésta continuará en el archivo TAREA.

III) Tercera Forma Normal.

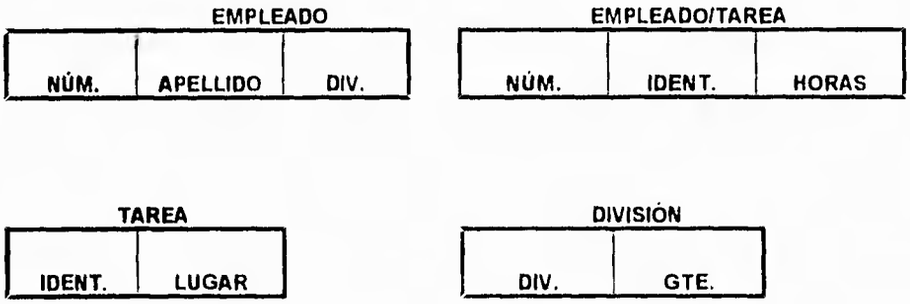
En la tercera forma normal nos aseguramos de que *no existen atributos que aparentemente dependen de la llave, pero que en realidad dependen de un atributo que no es la llave*. A esto se le conoce como "llave escondida".

Volviendo a nuestro ejemplo, en el archivo de EMPLEADO podemos observar que DIV. depende de NÚM. Sin embargo, GTE. no depende de NÚM. sino de DIV. Lo anterior se ilustra a continuación:

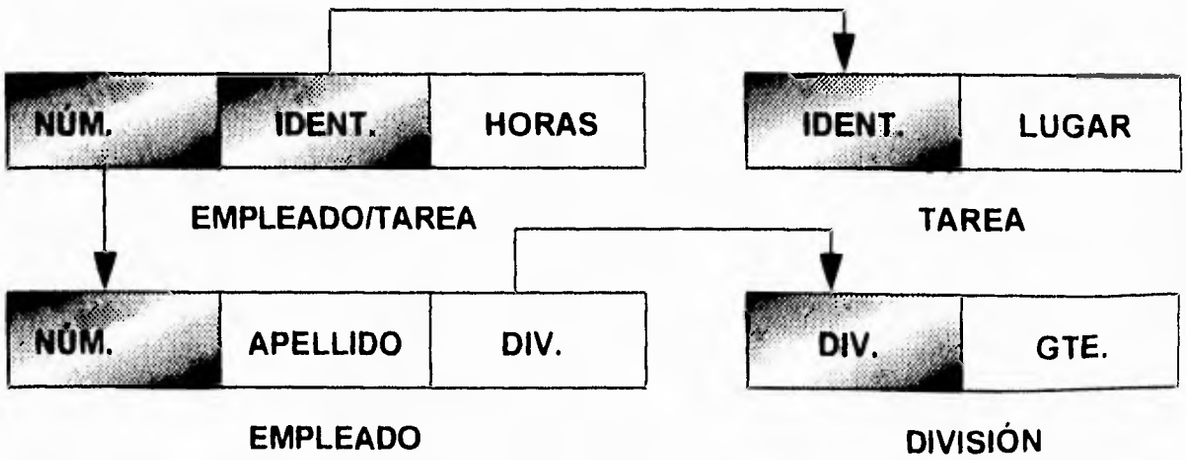


La línea doble indica que GTE. aparentemente depende de NÚM. (se conoce como dependencia transitiva). En realidad, GTE. depende directamente del atributo DIV. que no es la llave. Por esta razón, necesitamos aislar a GTE. junto con DIV. dentro de una nueva entidad.

Si no aislamos a GTE, podrían presentarse anomalías de inserción, borrado y actualización similares a las que se mencionaron con anterioridad. Estos problemas pueden evitarse aplicando la tercera forma normal, como se muestra a continuación.



Hemos creado un nuevo archivo al que llamamos DIVISIÓN. En el archivo de EMPLEADO tratamos de describir dos entidades, división y empleado, en la misma entidad. Como gerente corresponde únicamente a división, es conveniente ponerlo en una entidad separada.



Podemos notar que las ligas entre los archivos se establecieron a través de los atributos comunes. De esta manera, toda la información y las relaciones que forman parte del archivo sin normalizar, pueden reconstruirse a través de los atributos comunes.

En algunos casos se emplea la redundancia planificada de datos con la finalidad de mejorar el desempeño del sistema. Por ejemplo, si se desea un reporte mensual de las horas trabajadas por división: tendríamos que realizar los siguientes cálculos:

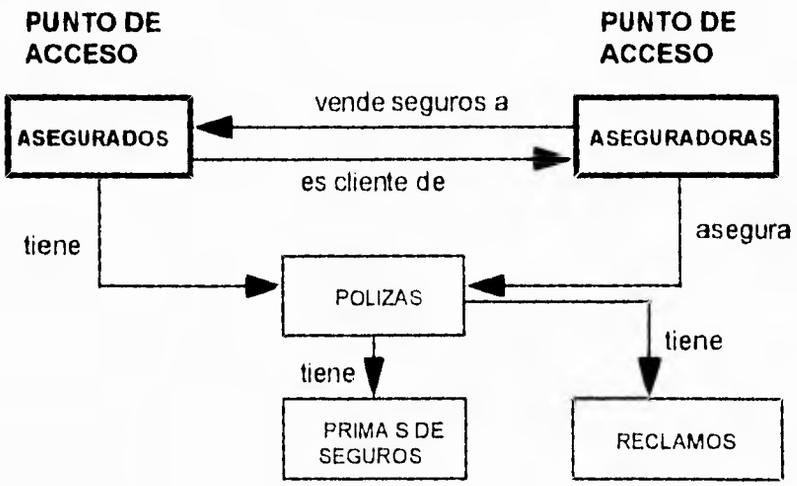
- Obtener el total de horas trabajadas por cada trabajador del archivo de EMPLEADO/TAREA
- Determinar la división a la que pertenece cada trabajador en el archivo de EMPLEADO
- Acumular el total de horas de cada trabajador en la respectiva división dentro del archivo de DIVISIÓN

Para disminuir el tiempo de procesamiento, podemos asignar un campo de "horas" al archivo de DIVISIÓN. De esta manera, las horas por división pueden acumularse cada vez que el archivo EMPLEADO/TAREA es actualizado.

Aunque la tercera forma normal proporciona una gran flexibilidad en el modelado de datos, con frecuencia conduce a un bajo desempeño. Algunos autores proponen una cuarta y quinta forma normal; sin embargo, el aplicar estas formas nos puede llevar a tener una base de datos demasiado subdividida, de desempeño pobre e impráctica. En la práctica, se combinan la primera y la tercera forma de tal manera que se obtiene el nivel deseado de normalización.

5.1.5. Definir el flujo de información y la ruta de acceso de las entidades.

Los puntos de acceso son los puntos de entrada a la base de datos, dependen del tipo de procesamiento y de cómo desea el usuario extraer información de la base de datos. Regresando a nuestro ejemplo, podemos asumir que serán requeridos reportes de Asegurados e información de las Aseguradoras. Estos serán los puntos de acceso naturales. Como las entidades Reclamos y Primas son inútiles sin la información adicional de Asegurado o Aseguradora, no son considerados como puntos de acceso Cualquier acceso a la base de datos debe realizarse vía Asegurado o Aseguradora, como se muestra en el diagrama siguiente:



5.1.6. Diseño físico de la base de datos.

El diseño físico de bases parte del diseño lógico del sistema y obtiene un diseño propio para la implantación en un DBMS. Esta transformación la realiza el diseñador de la base de datos. Es una tarea muy técnica que requiere de amplios conocimientos y experiencia en DBMS. El proceso de diseño físico de una base de datos representa un esfuerzo continuo para proveer un desempeño aceptable y satisfacer las necesidades de los usuarios.

CAPÍTULO 6
SELECCION DE UNA BASE DE DATOS

6.1.MARCO DE REFERENCIA.

- La evaluación fue realizada en los primeros meses de 1992

- El proyecto en su primera fase deberia estar terminado a fines del año por lo cual ademas del desempeño de la base de datos deberia considerarse la facilidad de aprendizaje de uso y las herramientas de desarrollo proporcionadas

- Se evaluaron cuatro manejadores de bases de datos de amplio uso comercial (OR=ORACLE SB=SYBASE IF=INFORMIX, IN=INGRES) Se considero la version disponible en esa fecha debido a que algunos proveedores trataron de anticipar las ventajas de sus nuevas versiones aun en desarrollo

- A cada uno de tos proveedores o representantes se le solicitó informacion tecnica, proyectos importantes y exitosos de su producto Para complementar esta información, se programó una visita a una institución que estuviera desarrollando alguna aplicación con el manejador de bases de datos del proveedor

- Las pruebas se realizaron en las instalaciones de la compañia A cada representante se le pidió que desarrollara la misma aplicación con un plazo limite de tiempo

- La evaluación del equipo de hardware se realizo en paralelo a la de los manejadores de bases de datos.

6.2. CRITERIOS PARA EVALUAR UNA BASE DE DATOS.

Los resultados obtenidos de la evaluacion se enlistan en la tabla siguiente

Caracteristicas técnicas	**OR**	**SB**	**IF**	**IN**
	ORACLE	SYBASE	INFORMIX	INGRES
Desempeño OLTP	0	2	1	2
Desempeño Query	0	1	1	2
Confiabilidad # errores	0	2	2	1
Conf. Ruta de actualizacion	0	2	1	2
Conf. Integridad y Seguridad	0	2	0	2
Tamaño de tablas	0	0	2	2
Núm. de Usuarios	0	1	1	2
Multiprocesamiento	0	1	1	2
Interoperabilidad Unisys	1	0	0	0
Total	1	11	9	15
Uso y herramientas de desarrollo	**OR**	**SB**	**IF**	**IN**
Facilidad de uso y aprendizaje	1	2	1	2
Control sobre el lenguaje	0	2	2	2
Lenguaje de cuarta generacion	0	2	2	2
Interface gráfica con el usuario	0	0	1	2
Facilidad de interface con C++	1	1	1	2
Aprovechamiento de prop. nativas del equipo	1	1	1	2
Total	3	8	8	11
Servicio al cliente	**OR**	**SB**	**IF**	**IN**
Servicio	0	0	0	0
Personal	2	1	1	0
Experiencia	2	0	1	0
Costo de Consultoria	1	1	2	1
Capacitacion	2	0	2	0

Servicio al cliente	OR	SB	IF	IN
Instalación	1	1	1	1
Organización	2	1	1	0
Total	10	4	8	2
Costos	+++	++	+	++++
Totales	15	25	27	28

2	muy bien
1	bien
0	mal

+	precio menor
++	precio medio
+++	precio alto
++++	precio mayor

Desempeño OLTP (del inglés On Line Transaction Processing) Se refiere al desempeño y al tiempo de respuesta del manejador de bases de datos en aplicaciones en linea

Desempeño Query. Se refiere al desempeño y al tiempo de respuesta del manejador cuando se hacen accesos a la base de datos para realizar consultas de información dar de alta modificar o borrar registros.

Confiabilidad # errores. Representa el nivel de confiabilidad en los accesos a la base de datos y el manejo de los errores

Conf. Ruta de actualización. Se refiere a la historia de actualizaciones(upgrades) del producto que tan frecuentemente son realizadas y que tan confiables son (menos errores o bugs) Otro aspecto importante es la transparencia o facilidad para la migración de una actualizacion a otro del producto

Conf. Integridad y Seguridad. Se refiere a las herramientas que proporciona el manejador para asegurar la integridad de la información el manejo de permisos y restricciones en los usuarios para respetar la confidencialidad de la información de la base de datos

Tamaño de tablas. Se refiere al tamaño máximo de las tablas de la base de datos que permite el manejador. Este punto es sumamente importante debido a que se pretende utilizar tablas que van a almacenar gigabytes de información

Núm. de Usuarios. Es el numero máximo de usuarios que puede atender el manejador de bases de datos.

Multiprocesamiento. Se evalua el comportamiento del manejador de bases de datos mientras realiza múltiples tareas al mismo tiempo o en paralelo Se evalua que tanto se aprovecha el hardware utilizando más de un CPU y que tan eficientemente distribuyen las tareas

Interoperabilidad Unisys. Debido a que se cuenta con equipo Unisys adquirido con anterioridad, se evaluó la capacidad del manejador de bases de datos para también operar con este equipo

Facilidad de uso y aprendizaje. Se considera que tan fácil es aprender y hacer uso del manejador de bases de datos a corto plazo y el nivel de conocimientos requeridos para la programación

Control sobre el lenguaje. Evalua que tan poderoso es el software del manejador y el nivel de control que se tiene sobre este para poder realizar distintas tareas en la base de datos accesos ajustes en el funcionamiento, modificaciones etc

Lenguaje de cuarta generación. Se considera si el manejador proporciona herramientas de cuarta generación para el rapido desarrollo de aplicaciones

Interface gráfica con el usuario. Se considera si el manejador proporciona una interface gráfica tipo Windows para el desarrollo de aplicaciones en linea

Facilidad de interface con C++. Ante la posibilidad de utilizar programacion orientada a objetos para el desarrollo de las aplicaciones, se tomo en cuenta si el manejador soportaba un lenguaje como C++

Aprovechamiento de prop. nativas del equipo. Se considera si el manejador de bases de datos aprovecha al máximo las caracteristicas del hardware de un equipo y el software del sistema operativo orientado al desempeño OLTP

Servicio. Se evalúa la calidad, eficiencia y prontitud en el servicio que proporciona el proveedor o representante

Personal. Se evalúa la cantidad de personas disponibles para el servicio la preparación, experiencia y habilidades del personal

Experiencia. Se considero el impacto la penetracion y antiguedad del producto en el mercado Proyectos importantes de caracteristicas similares que se hayan finalizado con exito

Costo de Consultoria. Se refiere al costo y a la calidad de la asesoria al cliente por parte del proveedor o representante

Capacitación. Se refiere a las instalaciones equipo cursos material didáctico instructores duracion etc de que dispone el proveedor o representante para capacitar al cliente en el conocimiento del producto

Instalación. Evalúa el equipo que se necesita el tiempo requerido y la facilidad para instalar el software al cliente

Organización. Se consideró la organizacion que respalda al producto a nivel nacional e internacional

6.3.CONCLUSIONES DE LA SELECCIÓN DE BASES DE DATOS.

De los resultados obtenidos de la evaluacion podemos señalar que

- Desde el punto de vista de las caracteristicas tecnicas el manejador de bases de datos mas robusto fue IN. El manejador OR obtuvo una pésima calificación debido a que demostro poco interes y cooperación en cumplir los requerimientos del concurso por lo que faltaron elementos para su evaluación.
- Desde el punto de vista de facilidad de uso y herramientas de desarrollo, el manejador de bases de datos que más ofreció fue IN
- En lo que corresponde al servicio al cliente el manejador de bases de datos que mejor calificacion obtuvo fue OR y el peor fue IN Hay que resaltar que en forma general el servicio al cliente fue deficiente por parte de los proveedores o representantes de estas bases de datos

Concluyendo, se eligió el manejador de bases de datos **IN = Ingres** como la mejor opcion debido a sus caracteristicas técnicas, facilidad de uso y a las herramientas de desarrollo que proporciona que fueron superiores a IF. Aunque Ingres fue el producto más caro se consideró más importante su desempeño que el precio, en la selección final de la base de datos La gran desventaja de SB fue que el tamaño máximo de las tablas que soportaba la base de datos era menor del esperado

CAPÍTULO 7
ANÁLISIS DE UN CASO REAL PARA LA BASE DE DATOS INGRES

7.1. SINOPSIS.

El caso que analizaremos, es la transferencia de información entre dos bases de datos distintas que estar en equipos de cómputo y plataformas de trabajo diferentes. Este caso corresponde a un proyecto realizado por una empresa comercializadora de cobertura nacional. Elegimos este proyecto para la tesis porque representa un ejemplo real y común del proceso de migración de información que requiere una empresa al cambiar de una plataforma a otra. Generalmente este cambio obedece a la necesidad de las empresas de modernizar su infraestructura tecnológica para ser más competitivos en el mercado.

Actualmente, la empresa comercializadora esta reemplazando gradualmente los sistemas que tiene operando en equipos UNYSIS de la serie "A" a equipos HP9000. Los sistemas que se encuentran en el equipo UNYSIS fueron desarrollados principalmente con lenguaje COBOL y LINC14 (lenguaje de cuarta generación de UNYSIS), mientras que los nuevos desarrollos utilizan la base de datos INGRES, el sistema operativo UNIX y lenguajes tales como WINDOWS4GL (lenguaje de cuarta generación de INGRES) y "C".

Uno de los sistemas que se encuentra actualmente en producción en UNYSIS es el sistema de COMPRAS, el cual es parte medular de la empresa, ya que con este sistema se realizan las operaciones estratégicas del negocio, como son las negociaciones con proveedores, los pedidos de artículos y su distribución a las diferentes sucursales de la República Mexicana. Este sistema tiene cerca de 10 años en producción y ya no responde a las necesidades del negocio, actualmente solo se le da mantenimiento. Por esta razón, se desarrolló un nuevo sistema de COMPRAS llamado SISNE (Sistema de Negociadores)

con la nueva tecnología que adquirió la empresa. Durante un año se realizo el análisis y el diseño de la base de datos, el desarrollo del front-end y de los enlaces o interfaces para realizar las transferencias de la información entre las dos bases de datos COMPRAS y SISNE.

En este capítulo plantearemos el mecanismo y las estrategias que se llevaron a cabo para realizar la migración de la información entre estas dos bases de datos se examinaran ejemplos representativos para no comprometer la información confidencial de la empresa.

7.2. ESQUEMA GENERAL DE LA TRANSFERENCIA DE INFORMACIÓN.

En el diagrama 7.1 se presenta el enfoque utilizado para realizar la transferencia de información entre las bases de datos.

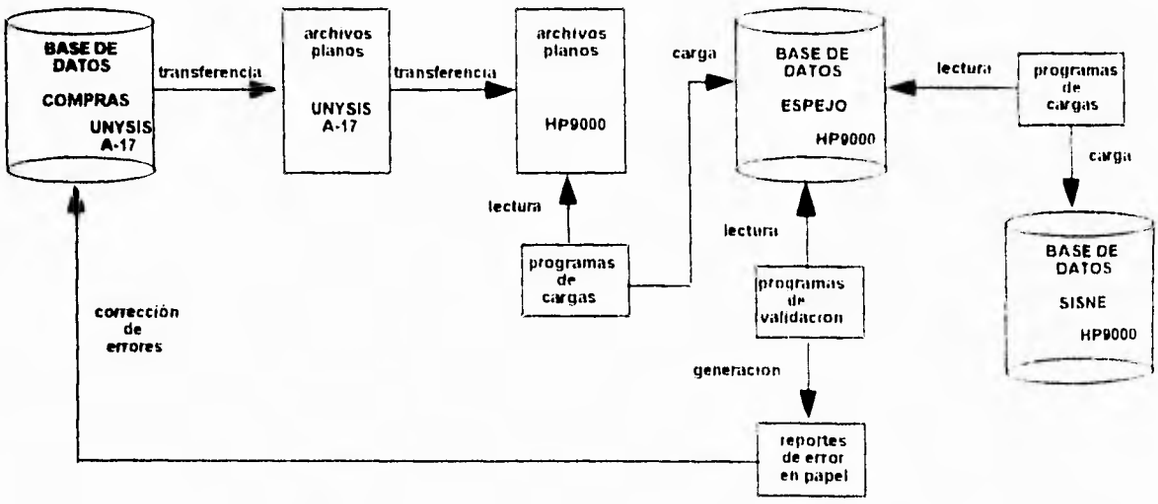


Figura 7.1 Diagrama del enfoque de transferencias

Se eligió este esquema de transferencia de información buscando aprovechar los recursos del equipo HP-9000 bajo UNIX, por otro lado el equipo encargado de realizar las transferencias desconocía las herramientas que se utilizan en UNYSIS y las personas involucradas en el sistema de COMPRAS en UNYSIS realizaban un mantenimiento continuo del sistema y no se pudo contar con la ayuda de estos recursos en la parte de las transferencias

La transferencia de información del equipo UNISYS A-17 a archivos planos la hizo el área de Soporte Técnico con una utilería para bajar la información de las tablas de COMPRAS a este tipo de archivos. Posteriormente, un operador realizó la transferencia de dichos archivos planos a la HP9000

El concepto de la base de datos espejo se refiere simplemente a la creación de las mismas tablas que existen en COMPRAS (UNYSIS) en la aplicación de SISNE (INGRES) para posteriormente llenarlas con la información con que se cuenta en COMPRAS en determinado momento. Este enfoque nos sirve para realizar ciertas validaciones, generar reportes de error y corregir la base de datos de COMPRAS para que cuando se haga la carga inicial de información en SISNE y entre en producción la información sea mas completa y consistente. En los siguientes puntos se explicará el proceso que se muestra en el diagrama anterior.

7.3. COMPRENSIÓN Y ANALISIS DE LA BASE DE DATOS DE COMPRAS.

En esta fase nos reunimos con gente de Compras para analizar a fondo cada una de las tablas de la base de datos de COMPRAS esto es, analizamos cada uno de los campos de todas las tablas que existen que dominios pueden tener y sus relaciones con otras tablas

Como ejemplo, podemos ver la siguiente tabla de la base de datos de COMPRAS la cual es el catalogo de compañias

Nombre de la tabla CP001 Descripcion Catalogo de compañias

Atributo	Tipo	Descripción
CP001-MAINT	CARACTER(1)	Para saber el status actual de la compañia Si es "D" esta dada de baja "C" que tuvo cambios y "A" es esta activa
CP001-CEDEMP	CARACTER(10)	Cedula de empadronamiento
CP001-DIRECC	CARACTER(35)	Direccion de la compañia
CP001-EJFISCA	CARACTER(4)	Año fiscal actual
CP001-NOMCIA	CARACTER(35)	Nombre de la compañia
CP001-POBLACIO	CARACTER(20)	Población en donde se encuentra la compañia
CP001-RFC	CARACTER(12)	RFC
CP001-CODPOST	ENTERO(5)	Codigo postal
CP001-NUMCIA	ENTERO(2)	Identificador único de la compañia
CP001-REGSECOF	CARACTER(10)	Registro SECOFI de la compañia

Figura 7.2 Tabla CP001 (Compañias)

También analizamos las relaciones existentes entre tablas como la que se muestra en la siguiente ejemplo

La tabla que se muestra es el catalogo de compradores el cual tiene una relacion uno a muchos con el catálogo de compañias

Nombre de la tabla CP004 Descripcion Catalogo de compradores

Atributo	Tipo	Descripcion
CP004-MAINT	CARACTER(1)	Indica el status del comprador. Al igual que el CP001-MAINT
CP004-CVECOMP	CARACTER(8)	Identificador unico del comprador. Esta compuesto por (CP004-NUMCIA2, CP004-NUMCOMP2, CP004-NUMSECC2)
CP004-NOMCOMP	CARACTER(25)	Nombre y apellidos del comprador
CP004-FECALTA	ENTERO(6)	Fecha en que fue dado de alta (AAMMDD)
CP004-NUMCIA2	ENTERO(2)	Identificador de la(s) compañia(s) que maneja el comprador
CP004-NUMCOMP2	ENTERO(3)	Clave del comprador
CP004-NUMSECC2	ENTERO(3)	Identificador único de las secciones que maneja el comprador

Figura 7.3 Tabla CP004(Compradores)

El atributo CP004-NUMCIA2 indica las compañias que el comprador puede manejar, por lo tanto es obvio que existe una relacion uno a muchos un comprador puede manejar varias compañias

Esta fase tuvo una duración de tres semanas durante las cuales se revisó toda la base de datos de COMPRAS.

7.4. CREACIÓN DE LA BASE DE DATOS ESPEJO.

Decidimos como parte de la estrategia de transferencia crear una base de datos en INGRES-HP9000 exactamente igual a la base de datos de COMPRAS a la que llamamos base de datos espejo para poder validar las tablas, sus atributos y posibles dominios. Esto se hizo con la finalidad de revisar la consistencia

e integridad de la información de la base de datos de COMPRAS, así como para generar reportes de los errores y posteriormente corregir la información de dicha base de datos

La creación de la base de datos espejo contiene los siguientes pasos

1. Bajar las tablas de la base de datos de COMPRAS a archivos planos y transferirlos al equipo HP9000

Esta parte la realizó el area de soporte técnico, utilizaron una utileria que baja toda la informacion que tiene una tabla a un archivo plano y posteriormente dichos archivos planos fueron transferidos a nuestros directorios en la HP9000

2. Escribir los *scripts* (texto) para crear las tablas y construir la base de datos ESPEJO

Las instrucciones para crear tablas son del DDL (Data Definition Language) de INGRES. En la figura 7.4 se muestra el script para crear la tabla de compañías de la base de datos espejo

```
CREATE TABLE CP001 /* nombre de la tabla*/
(       /*nombre del campo*/              /*tipo del campo*/
        maint                           char(1),
        cedemp                          char(10),
        direcc                          char(35),
        ejfisca                         char(4),
        nomcia                          char(35),
        poblacio                        char(20),
        rfc                             char(12),
        codpost                         integer4,
        numcia                          integer1,
        regsecof                        char(10)
                )
with noduplicates \plg /* no acepta duplicados*/
CREATE UNIQUE INDEX cias_idx  on compañia (numcia ASC) \plg /*crea un indice sobre la
tabla*/
```

Figura 7.4 Script para crear una tabla en INGRES

3. Crear y ejecutar los scripts de subida de informacion de archivos planos a la base de datos ESPEJO

En la figura 7.5 se muestra un script para copiar en la tabla CP001(compañias) el archivo plano CP001

```
COPY CP001 /* nombre de la tabla en la que se va a copiar*/
(       /*nombre del campo*/              /*tipo del campo*/
        maint           =               char(1),
        cedemp          =               char(10),
        direcc          =               char(35),
        ejfisca         =               char(4),
        nomcia          =               char(35),
        poblacio        =               char(20),
        rfc             =               char(12),
        codpost         =               char(5),
        numcia          =               char(2),
        regsecof        =               char(10)
                )
                FROM '/104q2/compra/CP001'   /*archivo plano*/
                WITH on_error = continue,       /* condicion de error */
               log = 'error/error.CP001' \p\g    /*archivo de errores */
```

Figura 7.5 Script para copiar de un archivo plano a una tabla de INGRES

Como se puede observar, en la parte del tipo del campo de la figura 7.5 el tipo char(2) para el campo numcia no es del mismo tipo que en la tabla 7.4 (donde se crea la tabla compañia), esto es debido a que en la base de datos de COMPRAS (ver figura 7.2) su tipo es entero de dos posiciones y en el archivo plano, en la parte que corresponde a numcia es de dos posiciones. Pero cuando se hace el COPY a la tabla compañia queda con tipo integer1 que es como se creó la tabla.

Los tres pasos mencionados anteriormente se realizaron para toda la base de datos de COMPRAS quedando finalmente la base de datos ESPEJO en el equipo HP9000.

7.5. CREACIÓN DE LOS PROGRAMAS DE VALIDACIÓN DE LA B.D. ESPEJO.

Los programas de validación se realizaron para revisar errores e inconsistencias en la base de datos de COMPRAS e imprimir un reporte en papel. Este reporte se entregara al area de Administracion de Compras y a los encargados del sistema de COMPRAS para que con base en los reportes de error generados se realice la depuracion y correccion de la base de datos.

Los programas se realizaron en lenguaje "Embedded C" el cual es "C" pero permite ejecutar instrucciones del SQL de INGRES tambien es conocido como SQL inmerso. Dichos programas se dividieron en :

- Programas de validación de catálogos en donde se valida principalmente todos los catalogos generales.
- Programas de validación de relaciones entre tablas donde se valida que las relaciones entre las tablas sea consistente.

En las figuras 7.6 y 7.7 se muestran dos ejemplos de las validaciones que se tienen que hacer en las tablas de la base de datos espejo.

Atributo	Tipo	Descripción	Dominio
CP001-MAINT	CARACTER(1)	Para saber el status actual de la compañia. Si es "D" esta dada de baja "C" que tuvo cambios y "A" es esta activa	"D" "A" "C"
CP001-CEDEMP	CARACTER(10)	Cedula de empadronamiento	diferente de espacios
CP001-DIRECC	CARACTER(35)	Direccion de la compañia	diferente de espacios
CP001-EJFISCA	CARACTER(4)	Año fiscal actual	Año actual [1994]

CP001-NOMCIA	CARACTER(35)	Nombre de la compañía	diferente de espacios
CP001-POBLACIO	CARACTER(20)	Poblacion en donde se encuentra la compañia	diferente de espacios
CP001-RFC	CARACTER(12)	RFC	diferente de espacios
CP001-CODPOST	ENTERO(5)	Código postal	diferente de CERO
CP001-NUMCIA	ENTERO(2)	Clave unica para identificar a la compañia	diferente de CERO y que no se repita
CP001-REGSECOF	CARACTER(10)	Registro SECOFI de la compañía	diferente de espacios

Figura 7 6 Validaciones de la tabla CP001 (compañia)

Atributo	Tipo	Descripción	Dominio
CP004-MAINT	CARACTER(1)	Indica el status del comprador Al igual que el CP001-MAINT	A = ESTA ACTIVO C = TUVO CAMBIOS D = DADA DE BAJA
CP004-CVECOMP	CARACTER(8)	Identificador único del comprador Está compuesto por (CP004-NUMCIA2 CP004-NUMCOMP2 CP004-NUMSECC2)	Diferente de espacios y que no se repita
CP004-NOMCOMP	CARACTER(25)	Nombre y apellidos del comprador	Diferente de espacios
CP004-FECALTA	ENTERO(6)	Fecha en que fue dado de alta (AAMMDD)	Diferente de CEROS y con formato (AAMMDD)
CP004-NUMCIA2	ENTERO(2)	Identificador de la(s) compañia(s) que maneja el comprador	diferente de CEROS y exista en la tabla compañia
CP004-NUMCOMP2	ENTERO(3)	Clave del comprador	diferente de CEROS
CP004-NUMSECC2	ENTERO(3)	Identificador único de las secciones que maneja el comprador	diferente de CEROS y exista en la tabla seccion

Figura 7 7 Validaciones de la tabla CP004 (comprador)

Este proceso de valioaciones se realizo para todas las tablas de la base de datos espejo

Una vez identificadas todas las validaciones se construyeron los programas de validacion de la base de datos ESPEJO en lenguaje "Embedded C" para todas las tablas Estos programas generan un archivo de errores donde se guardan los datos que no cumplen con las validaciones para que el area de Administración de Compras los revise y corrija Para cuando se realicen otras cargas de la base de datos de COMPRAS a la base de datos espejo parte de los errores ya deben haber sido corregidos

En la página siguiente se muestran partes de reportes de error generados para la tablas CP001 y CP004

```
********** CATÁLOGO DE COMPAÑIAS **********
**********
**********
El código postal de la compañia 01 viene en CERO.
**********
**********
El código postal de la compañia 14 viene en CERO
**********
**********
Número de registros con error : 2
_____
_____

********** CATÁLOGO DE COMPRADORES **********
**********
**********
Comprador 01, compañia 58, la compañia no existe en el catálogo de compañias(CP001).
**********
**********
Número de registros con error : 1
_____
_____
```

Figura 7 8 Reportes de error generados por los programas de validación

7.6. IDENTIFICACIÓN DE EQUIVALENCIAS ENTRE BASES DE DATOS.

Esta fase consistió en identificar las equivalencias entre las dos bases de datos involucradas, esto es se definió la correspondencia entre tablas de las dos bases de datos y sus campos. En las tablas de las figuras 7.9 y 7.10 se muestran ejemplos de las equivalencias encontradas

TABLA EN BD ESPEJO **CP001**	TABLA EN BD SISNE **compañia**	TIPO	DEFAULT
CP001-MAINT	CIA-STATUS	CARACTER(4)	x
CP001-CEDEMP	CIA-CEDEMP	CARACTER(10)	Espacios Si CP001-CEDEMP=" "
CP001-DIRECC	CIA-DOMICILIO	CARACTER(40)	Espacios Si CP001-CEDEMP=" "
CP001-EJFISCA	CIA-EJFISCAL	CARACTER(4)	Año actual
CP001-NOMCIA	CIA-NOMBRE	CARACTER(35)	Espacios Si CP001-CEDEMP=" "
CP001-POBLACIO	CIA-POBLACION	CARACTER(20)	Espacios Si CP001-CEDEMP=" "
CP001-RFC	CIA-RFC	CARACTER(13)	Espacios Si CP001-CEDEMP=" "
CP001-CODPOST	CIA-CP	ENTERO	CERO SI CP001-CODPOST =0
CP001-NUMCIA	CIA-COMPANIA	ENTERO	x
CP001-REGSECOF	CIA-REGSECOF	CARACTER(10)	Espacios si CP001-REGSECOF = " "
NO EXISTE	CIA-FECHA-ALTA	CARACTER(8)	Fecha del dia de carga
NO EXISTE	CIA-FECHA-BAJA	CARACTER(8)	Espacios Fecha del dia de carga si CP001-MAINT = "D"

Figura 7.9 Equivalencias entre base de datos ESPEJO y SISNE

Como se puede observar en la tabla de la figura 7.9 existen campos en la base de datos ESPEJO que no existen en la tabla equivalente de la base de datos de SISNE por lo que al cargarse se les pondrá un valor

por *default* Por ejemplo en el caso del campo CIA_FECHA_ALTA de SISNE no existe en la base de datos ESPEJO por lo que tendrá el valor de la fecha del dia de carga que da el sistema operativo También se presenta el caso de que los valores de los campos de la base de datos ESPEJO vienen en ceros o con espacios dependiendo su tipo en este caso al insertar en la base de datos SISNE se cargarán con el valor que traigan esperando que en las siguientes cargas estos errores ya hayan sido corregidos por la Administracion de Compras

En la tabla de la figura 7 11 se muestra la estructura de la tabla "compañia" de la base de datos SISNE

Atributo	Tipo	Descripción
CIA_COMPANIA	ENTERO	Identificador unico de la compañia
CIA_NOMBRE	CARACTER(35)	Nombre de la compañia
CIA_STATUS	CARACTER(4)	"D" = dada de baja "C"=Modificada "A"=Activa
CIA_DOMICILIO	CARACTER(40)	Dirección de la compañia
CIA_POBLACION	CARACTER(20)	Poblacion en la quese encuentra la compañia
CIA_CEDEMP	CARACTER(10)	Cédula de empadronamiento
CIA_RFC	CARACTER(13)	RFC de la compañia
CIA_REGSECOF	CARACTER(10)	Registro de SECOFI
CIA_CP	CARACTER(5)	Codigo Postal
CIA_FECHA_ALTA	CARACTER(8)	Fecha de alta de la compañia
CIA_FECHA_BAJA	CARACTER(8)	Fecha de baja de la compañia
CIA_EJERCICIO_FISCAL	CARACTER(4)	Año de ejercicio fiscal

Figura 7 11 Tabla compañia en base de datos SISNE

Una vez encontradas las equivalencias entre las dos bases de datos se realizaron los programas de carga a la base de datos de SISNE Estos programas de carga se realizaron en lenguaje "Embedded C" Dichos programas se dividieron en programas de carga de catalogos generales y programas de carga de

relaciones entre tablas. Los primeros leen una tabla de la base de datos ESPEJO e insertan en su equivalente en la base de datos SISNE. los segundos leen varias tablas de la base de datos espejo e insertan en sus tablas equivalentes de la base de datos SISNE

Una vez efectuados los pasos anteriormente descritos hicimos el plan de pruebas e implantación el cual consta de los siguientes puntos

- Ejecución de los programas de validación y generación de reportes
- Chequeo y corrección de los errores generados
- Ejecución de los programas de cargas en la base de datos SISNE
- Chequeo y corrección de los errores generados
- Una vez corregidos los errores se realizó un proceso en lotes (en inglés batch) el cual ejecuta todos los programas de validación y de cargas

Para las pruebas de este plan tuvimos que tomar en consideración algunos problemas los cuales se mencionan en el punto 7.7. Con los programas de carga a la base de datos SISNE se termina el ciclo del proceso de transferencia de información entre las dos bases de datos

7.7. PROBLEMAS.

Los problemas que encontramos en el proceso de cargas se mencionan a continuacion

- La base de datos de COMPRAS tiene movimientos diariamente a excepcion de los domingos por lo que se tiene que pedir la carga ese dia

- La transferencia de la base de datos de COMPRAS a archivos planos (en UNISYS) tarda alrededor de cuatro horas

- La transferencia de la base datos de COMPRAS (archivos planos UNISYS) a archivos planos en HP9000 tarda alrededor de 6 horas

- La subida de informacion a la base de datos espejo se tarda de 4 a 6 horas dependiendo de la carga de trabajo que tenga HP9000

- Los programas de validación de la base de datos ESPEJO, aún creando indices secundarios tardan alrededor de 10 horas (son 20 programas). El tiempo sin indices secundarios fue de alrededor de 18 horas. Hay que considerar que existen tablas muy grandes (la mayor es de 3'500'000 registros aproximadamente)

- Los programas de cargas a la base de datos SISNE tardan aproximadamente 12 horas

7.8. CONSIDERACIONES FINALES.

Para realizar la carga inicial que el sistema SISNE requiere para entrar en produccion se necesita que todos los procesos en lotes que se ejecutan actualmente en COMPRAS estén finalizados para que la información de la base de datos de COMPRAS ya no tenga movimientos y se transfiera a SISNE correctamente Ademas necesitamos que el sistema actual de COMPRAS no registre movimientos de ningún tipo durante aproximadamente una semana, que es el tiempo estimado para cargar todas las tablas en SISNE.

Para realizar la carga en la base de datos de SISNE todas las tablas tienen estructura de tipo HEAP (Secuencial), ya que es la forma más rapida de insertar registros en una tabla Posteriormente se modifica la estructura de cada tabla al tipo más apropiado

El plan que se propuso en el punto7 6 fue aprobado y al ponerse en marcha, resultó exitoso La Administración de Compras empezó a corregir y depurar la informacion de la base de datos de COMPRAS, y se repitió el proceso hasta que la informacion de COMPRAS quedó mas consistente y con menos errores

Finalmente, en diciembre de 1994 se realizo la primera carga de informacion en la base de datos de SISNE en un tiempo estimado de una semana En la primera semana de enero de 1995 los usuarios empezaron a trabajar con SISNE (el nuevo sistema de COMPRAS) con la información que fue

transferida del sistema anterior. Hasta el momento no se han presentado problemas con esta información, por lo tanto, podemos considerar que el enfoque seleccionado para realizar la transferencia de información fue adecuado y que el resultado de esta transferencia fue exitoso.

CAPÍTULO 8
CONCLUSIONES

En esta tesis hemos revisado la teoría básica sobre las bases de datos relacionales y distribuidas. Así mismo, se ha analizado un caso real de transferencia de información entre dos bases de datos que se encuentran en plataformas de trabajo distintas. Para completar este conjunto de conocimientos orientados a su aplicación en sistemas de bases de datos reales, vamos a formular algunas preguntas que tenemos que resolver en la práctica: ¿Como podemos identificar los datos que van a almacenarse en la base de datos? ¿Cómo determino las relaciones entre los datos? ¿Cuántas tablas necesitamos? ¿Cómo puedo controlar y reducir al mínimo la redundancia?

Las respuestas a estas preguntas se obtienen a partir del Modelaje de Datos. El Modelaje de Datos pretende obtener una visión coherente de la información que mantiene una empresa, de una manera independiente de las transformaciones que sufra por su empleo en las diversas actividades de la

empresa. Esta vision deberá identificar los diferentes tipos de entidades, atributos y relaciones de manera tal que la información este completamente normalizada. Podemos afirmar que los datos se encuentran en el centro del procesamiento de información moderno y que los procesos son secuencias de acciones controladas por condiciones, que se relacionan con los datos. Considerando lo anterior es de gran importancia en el desarrollo de sistemas de información y de bases de datos, que mediante el uso de una metodología apropiada se conjunten los datos, los procesos y la tecnología para obtener soluciones adecuadas a las necesidades de los usuarios.

A mediados de los 80's, James Martin desarrolla la Ingeniería de la Información. Esta se define como un conjunto de técnicas formales con las cuales se construyen modelos organizacionales de datos y de procesos que se utilizan para crear y mantener sistemas de procesamiento de datos. James Martin toma elementos de metodologías propuestas con anterioridad, como el Diseño Estructurado (Yourdon/Constantine), el Análisis Estructurado (Yourdon/De Marco, Gane/Sarson), las Técnicas de Bases de Datos (Codd) y elabora la Ingeniería de Información. Un acierto de J. Martin fue el de obtener un enfoque integral al combinar los modelos de datos y los modelos de procesos de los sistemas de información.

Tomando en consideración que los tipos de datos usados en una empresa no cambian mucho con el paso del tiempo, el elemento más importante de la Ingeniería de la Información son los datos. Su objetivo es crear una estructura estable donde los datos son almacenados y modificados por computadoras, para poder obtener la información requerida. Las fases que comprende la Ingeniería de Información son

1. **Planeación.** Determinar las metas y factores criticos de éxito del área de negocios así como el entendimiento general de la empresa, sus funciones, datos y necesidades de información. Corresponde al modelo conceptual.
2. **Análisis.** Determinar que procesos son ejecutados en un area de negocios especifico, así como la interrelación entre éstos y que datos se necesitan. Corresponde al modelo lógico.
3. **Diseño.** Definir como determinados procesos y datos en un area de negocios son implantados en procedimientos especificos y como estos procedimientos deben interactuar con el usuario. Corresponde al modelo logico.
4. **Construcción.** Instrumenta procedimientos con el uso de lenguajes de programacion, generadores de código y herramientas de construccion orientados al usuario final. Corresponde al modelo fisico.

Podemos resumir las ventajas que se derivan del empleo de la Ingenieria de Información en proyectos de desarrollo de sistemas en dos puntos:

1. Proveer lineamientos, herramientas y tecnicas que son utilizados en todos los proyectos de sistemas.
2. Liberar productos de mas alta calidad que satisfagan las expectativas de los usuarios, dentro de los calendarios y presupuestos propuestos.

Como ya se mencionó, las respuestas a las preguntas con que iniciamos este capítulo se obtienen a partir del Modelaje de Datos, que es la parte central de la Ingeniería de Información. A continuación se presenta un síntesis de las actividades y los aspectos a considerar para el desarrollo de aplicaciones de bases de datos y vamos a retomar el punto anterior. Se hace énfasis en el modelo de datos y se ignora el modelo de procesos, debido a que este último queda fuera del alcance de la tesis. Aplicando las fases de la Ingeniería de Información se obtiene el siguiente cuadro resumen (Figura 7.1)

FASES	MODELO DE DATOS	PASOS
Planeación	Entidad-Relacion Nivel Organizacional	Definición de Entidades Definición de Relaciones
Análisis	Entidad-Relacion	Definición de Entidades Definición de Relaciones Definición de Llaves
Diseño	Tablas relacionales o Scripts	Definicion de Atributos
Construcción	Bases de Datos y Archivos	

Figura 7.1 Cuadro resumen

NOTA. Se sugiere revisar el capítulo 5.2 Consideraciones para el diseño y la implantación

Después de analizar las funciones del negocio y determinar los requerimientos de datos, procedemos a la definición de las entidades principales del sistema, y es a partir de este momento, que podemos identificar

los datos que van a almacenarse en la base de datos y determinar las relaciones entre estos Con esta información construimos el modelo de entidad-relacion (E-R) y a continuación definimos las llaves y los atributos de las entidades El numero de tablas que necesitamos se obtienen directamente de los diagramas de entidad-relacion (E-R) Finalmente podemos controlar y reducir la redundancia aplicando el proceso de normalizacion

Para la realización exitosa de un proyecto es de primordial importancia la participacion de los usuarios en cada una de las fases todos los productos resultantes deben ser revisados por los usuarios involucrados en el desarrollo del proyecto y debemos obtener su aprobacion Por otro lado actualmente existen varias herramientas CASE (del inglés Computer Aided Systems Engineering) que facilitan significativamente el chequeo de errores la generacion de *scripts* (textos para crear bases de datos) la creacion y modificacion de los modelos de datos y procesos etc Algunas de las ventajas que se obtienen del uso de las herramientas CASE son mejorar la productividad y la calidad del software, rapidez en el proceso de desarrollo, reducir costos, automatizar el desarrollo y mantenimiento, automatizar la generación del código y de la documentación

Durante el desarrollo de una base de datos, una de las partes más importantes y que más cuidado requieren es el diseño lógico y físico de las tablas que conforman a nuestra base de datos En Ingres tenemos diferentes tipos de tablas, que podemos usar dependiendo de las condiciones que requiera nuestro sistema en especifico El mejor tipo de estructura es el que más se ajusta a los requerimientos de uso de la aplicaciones De la experiencia obtenida durante el desarrollo de aplicaciones con Ingres podemos concluir lo siguiente

Todas las tablas que se crean sin especificar el tipo son creadas como *Heap*, es decir, sin estructura alguna. Cuando se dan de alta registros, se insertan uno tras otro sin orden alguno. Heap resulta bastante útil es los siguientes casos

- Cuando se necesita hacer cargas de información muy grandes a una tabla, es mejor que primero se hagan con la tabla tipo Heap y despues cambiar el tipo de la tabla. Esto es mucho más rápido y ocupa menos recursos de la maquina
- Cuando se tienen tablas con muy pocas paginas
- Cuando siempre se consulta toda la información de una tabla sin importar el orden

El tipo de estructura *Hash* almacena la información a traves de la asignación de una localidad en disco para cada una de las "páginas principales" donde se insertan los registros. Esta asignación se hace con base en un cálculo matemático para cada llave. Hash tiene limitantes en cuanto al tipo de *querıes* que se pueden hacer, las principales son

- No se debe usar para búsquedas por rangos
- No se debe usar cuando solo se dispone de una parte de la llave

Hash es excelente cuando se necesita traer informacion por llaves únicas y de tamaño pequeño

La estructura tipo *Isam* contiene páginas de índices y de datos estáticos es bastante versátil debido a que permite búsquedas por rangos y por llave Sin embargo, Isam no debe usarse en los siguientes casos

- La tabla crece rápidamente
- La tabla es muy grande para modificarla
- Las llaves son secuenciales

La estructura tipo *Btree* es la mas versátil dentro de Ingres, permite búsquedas por rangos y por llave A diferencia de Isam, las páginas de índices y las de datos son dinámicas, es decir, crecen conforme crece la tabla. Btree no debe usarse en los siguientes casos

- La tabla es relativamente estática
- La tabla es pequeña, estática y tiene gran cantidad de accesos recurrentes

Si se desea manejar una fuerte concurrencia en Btree, el factor de llenado *(fill factor)* debe ser pequeño, pero obviamente esto hará que nuestra tabla se extienda ocupando más espacio en disco

Comparando Isam contra Btree podemos señalar que.

- Isam requiere menos operaciones a disco que Btree para accesar una pagina de datos
- Isam no hace bloqueos *(locks)* en las páginas de índices, mientras que Btree si los hace, por lo que el desempeño en accesos concurrentes en las páginas de índices de un Btree no es tan bueno como en un Isam

- Cuando se requieren accesos secuenciales sin un orden definido, Isam trae la información más rápido que Btree

Es importante que se revise continuamente la integridad de las tablas, debido a que cuando crecen se pueden crear *páginas de desbordamiento (overflow)*, lo que reduce considerablemente el desempeño de los *queries* que se realizan, puesto que se tienen que hacer más accesos a disco para traer información Después de liberar un sistema de bases de datos a producción, se deben revisar continuamente las tablas para poder diagnosticar si la estructura que se eligió fue adecuada

La definición de los índices secundarios debe ser realizada después de un análisis minucioso y preciso de los tipos de consultas que se tienen dentro de nuestro sistema, se deben tener en cuenta todos los procesos que afectan a la base de datos

Es importante remarcar el hecho de que Ingres utiliza siempre la política del "Left-Most", que quiere decir, si tenemos llaves que involucran más de una columna, Ingres siempre toma en cuenta el orden que tienen estas llaves. De esta forma, si se tiene una llave compuesta por tres campos, y nosotros tratamos de hacer una búsqueda sólo por los campos dos y tres, lo más seguro es que Ingres recorra la totalidad de la tabla para encontrar la información

Se debe tener un balance entre el número de índices secundarios y la velocidad de acceso, debido a que si se tienen numerosos índices secundarios, los procesos de escritura serán mas lentos y las actualizaciones ocuparan más espacio en el *log* de Ingres

Cuando se tienen indices secundarios de tipo Btree es importante no tener una profundidad mayor a tres

Para medir la profundidad de una estructura Btree, se utiliza la siguiente formula

profundidad = log(numero de registros) / log(llaves por pagina)

donde llaves por pagina = 2000 / tamaño de la llave

La profundidad excesiva en los indices puede desbordar el cache y aumentar los accesos de entrada/salida a disco(disk I/O)

Dentro del diseño fisico de una base de datos es sumamente importante el tipo de almacenamiento de las tablas y el ancho de banda para la transmisión de información Uno de los principales enemigos a vencer en el diseño fisico es el reducir al máximo los accesos de entrada/salida a disco(disk I/O)

Las bases de datos relacionales han dominado el mercado de las bases de datos en la ultima decada Si las ventajas de un sistema relacional como lo es INGRES pudieran ser resumidas en una sola palabra esta seria *simplicidad* para el usuario la cual se traduce en *productividad* Esto quiere decir que los usuarios finales pueden frecuentemente obtener resultados óptimos del sistema sin depender del departamento de sistemas La productividad se incrementa debido a que los usuarios y desarrolladores pueden ser más productivos en sus actividades diarias El modelo relacional expuesto en el CAPÍTULO 4, es notable por el numero reducido de conceptos que involucra Todo dato dentro de un sistema relacional es representado en solamente un sentido esto es, columnas de valores dentro de renglones de tablas Por esta misma razon, se necesitan realmente pocos operadores para las cuatro funciones basicas de

manipulación (consultar, modificar, insertar y borrar) y así mismo pocos operadores son necesarios para las demás funciones de un DBMS (definición de datos, seguridad, integridad, etc.)

La independencia de los datos significa que los usuarios y programadores no tienen que saber los detalles de cómo la información es accesada y almacenada. La independencia de datos y los lenguajes de alto nivel como el SQL, van de la mano. Es críticamente importante por cuando menos dos razones

1. Es importante para los diseñadores de aplicaciones porque sin ella, los cambios en la estructura de la base de datos implicarían cambios directos sobre la programación de las aplicaciones

2. Es importante para los usuarios ya que si no se tuviera independencia de los datos, el uso de la base de datos estaría muy restringido

Básicamente, INGRES provee independencia de datos física debido a su característica de navegación automática e independencia lógica en virtud de su mecanismo de vistas

En un sistema no relacional, el desempeño del sistema depende directamente de la calidad con la que fue programada la aplicación, en un sistema relacional, depende mucho más de la calidad del optimizador del sistema. De esta manera, un sistema relacional como INGRES puede superar por mucho los parámetros de desempeño de un sistema no relacional, debido a que el optimizador utiliza gran cantidad de información estadística tal como el tamaño de cada tabla, el número de valores de cada columna, etc

Como resultado, el optimizador es capaz de hacer mejoras viables en la estrategia escogida para realizar una tarea en particular, seleccionando la estrategia mas eficiente y óptima

El término de proceso distribuido se refiere a la habilidad de conectar múltiples computadoras en una red de comunicaciones En un sistema distribuido un usuario o programa operando en una sola computadora debe ser capaz de operar con datos almacenados en toro sitio Idealmente el proceso distribuido debería de ser "transparente" para el usuario esto es los usuarios no deberían de tener que hacer nada especial para accesar información remota, tal y como si ésta estuviera almacenada en forma local en su sitio. Hay cuando menos dos razones de peso para que esta transparencia sea deseable

1. Simplifica la lógica de las aplicaciones

2. Permite que los datos sean movidos de un sitio a otro sin tener que reprogramar la aplicación

INGRES/STAR fue el primer producto que salió al mercado buscando proveer verdadero soporte de bases de datos distribuidas. En INGRES existen cuatro niveles de proceso distribuido al momento, pero aún así existe espacio suficiente para seguir implantando mejoras sin lugar a dudas Estos niveles son

- INGRES/NET permite que frontends de INGRES en una sola máquina, interactuen con el backend de INGRES en otra máquina

- INGRES/PC LINK permite que la información sea cargada de un servidor de INGRES y la convierte en una forma propia para PC.

- INGRES/NET PC, es una versión de INGRES/NET que permite que los frontends de INGRES corran en PC, permitiendo que ésta funcione como estación de trabajo (Workstation) para el servidor de INGRES.

- INGRES/STAR permite que una colección arbitraria de tablas de varias bases de datos en diferentes sitios funcionen como una sola tabla para propósitos de manipulación de la información

Hemos de resaltar que los dos casos expuestos en la parte práctica de la tesis son muy representativos y corresponden a problemas actuales. En los últimos años gran número de empresas buscando consolidar su presencia en el mercado, han tenido que transferir su información de una base de datos a otra. Esta tendencia se debe entre otras razones, a que la base de datos anterior ha llegado a su límite de capacidad, es obsoleta o debido a que la empresa busca modernizar sus equipos de sistemas aprovechando las ventajas de las tecnologías actuales como los son: los sistemas abiertos, las arquitecturas cliente-servidor, los sistemas distribuidos, los sistemas orientados a objetos, etc. De esta manera, lo expuesto en esta tesis reviste de gran importancia. Primero, se expone un método para la transferencia de información que asegura la integridad de ésta y que ha probado ser exitoso. Y segundo, se definen los criterios prácticos que se deben evaluar para adquirir una base de datos. Con el surgimiento de nuevas tecnologías como los sistemas de bases de datos orientadas a objetos (OODBMS), la elección de una base de datos será aún más difícil. Como complemento a los criterios

expuestos en el CAPITULO 6 los desarrolladores de aplicaciones comerciales de bases de datos deben considerar los tres puntos siguientes

1. Escalabilidad, lo que significa que los sistemas de bases de datos deberan manejar un creciente número de usuarios concurrentes y enormes cantidades de información

2. Desempeño, que quiere decir que los sistemas de bases de datos que no proporcionen un desempeño eficiente con tiempos de respuesta adecuados (fracciones de segundo) serán rechazados

3. Confiabilidad, se refiere a que la información almacenada sea consistente y se conserve la integridad, lo que es absolutamente necesario para el manejo de la información crítica y estratégica de las empresas

Los sistemas de bases de datos orientadas a objetos (OODBMS) son consideradas como la nueva ola en la tecnología de la administración de la información Los OODBMS ofrecen las mismas características de las bases de datos relacionales como los son transacciones, respaldo y recuperación concurrencia, mecanismos de acceso y seguridad Pero además, proveen algunas ventajas en el manejo de aplicaciones complejas de computación distribuida y en el uso de tipos inusuales de datos para imágenes y multimedia. Así mismo, utilizan el lenguaje de desarrollo de las aplicaciones y la funcionalidad de la base de datos conjuntamente, en lugar de basarse en un lenguaje de bases de datos como el SQL Algunos proveedores de hardware como DEC, HP IBM, Sun Microsystems y de bases de datos relacionales como Sybase, ASK/Ingres y Oracle consideran la orientacion a objetos como clave en sus estrategias de

negocio de cara al futuro. Existen algunos proveedores especializados en OODBMS, siendo algunos de sus principales representantes Itasca Systems, Object Design Inc., Ontos Corp., Objectivity Inc. y Versant Object Technology Corp., quienes han orientado sus productos a distintos mercados de la computación. Sin embargo, los OODBMS representan sólo una pequeña porción del mercado de sistemas de bases de datos. El uso de bases de datos orientadas a objetos es limitado debido a que muchas empresas se encuentran en un proceso de integración e interoperabilidad de distintas bases de datos y por lo tanto, no es el momento para adoptar un nuevo modelo de datos. Además, los sistemas relacionales se encuentran en un proceso de asimilación, es decir, las compañías han hecho grandes inversiones en tecnología relacional y aún no han obtenido todo el provecho de ésta. Por un lado, las características operacionales de las bases de datos orientados a objetos, que en el dominio de CAD/CAM han probado ser notables, aún no han sido comprobadas exhaustivamente en el dominio de los sistemas de información en general. Por otro lado, la ausencia de estándares internacionales (por ejemplo ISO, ANSI, ECMA ...) producen un riesgo significativo para cualquier inversión en la nueva tecnología. Por estas razones, el uso de OODBMS se reduce actualmente a la generación de herramientas de desarrollo y repositorios de datos que son extensiones de los RDBMS actuales y a la administración de redes. Finalmente, a pesar de lo mencionado con anterioridad, se espera que los sistemas de bases de datos orientadas a objetos jueguen un papel importante en el paradigma del futuro de los sistemas de información.

BIBLIOGRAFÍA.

BIBLIOGRAFÍA.

ATRE, SHAKU (1992) Distributed databases, cooperative processing and networking Mc Graw Hill, New York, E.U.

CASAVANT T.L , SINGHAL M (1994) Distributed Computing Systems IEEE Computer Society Press, E.U.

DASH, J. (1993) Tomorrow's Information Technology (DB/Expo) *The Computer Conference Analysis Newsletter* n319 p6(1)

DATE, C.J. (1987) Bases de Datos *Una guía práctica* Addison-Wesley Iberoamericana, S A , 1a edn , México.

DATE, C.J. (1986) Introducción a los Sistemas de Bases de Datos Addison-Wesley Iberoamericana, S.A., México.

ELMASRI, RAMEZ (1989) Fundamentals of database systems The Benjamin/Cummings Publishing Company Inc., Redwood City, Cal , E U

GILLENSON, M. L., (1984) Strategic planning, systems analysis and database design. Wiley, New York, E.U.

HAWRYSZKIEWYCZ, I T. (1991) Database analysis and design. Macmillan, New York, E.U

INMON, W. H. (1990) Database machines and decision support systems. QED, Boston, E U

KORTH, H. F., SILBERSCHATZ A. (1988) Fundamentos de Bases de Datos. Mc Graw Hill, 1a. edn., México.

McClanahan, D. (1993) ODBMS development. *DBMS* v6 n12 p20(5)

Mc FADDEN F.R., HOFFER J A. (1988) Data Base Management.The Benjamin/Cummings Publishing Company Inc., E.U

TIMO, M. (1993) Putting object databases to work. *UNIX Review* v11 n7 p73(4)

TSAI, ALICE Y.H. (1990) Sistemas de bases de datos: administración y uso. Prentice-Hall Hispanoamericana, México.

VARMA, SANJEEV (1993) Objects and databases Where are we now? *Database Programming & Design* v6 n5 p60(5)

VOSSEN G. (1991) Data Models, Database Languages and Database Management Systems Addison-Wesley Publishing Company, Great Britain

WIEDERHOLD, Gio (1977) Database design, Mc Graw-Hill, New York, E.U

Designing INGRES Databases

Course Version 3/92

INGRES for Application Developers (Terminal-Based)

INGRES Release 6.3/03

Course Version 2/91