UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES
ACATLAN

"ESTUDIO COMPARATIVO DE LOS DIFERENTES MODELOS DE BASE DE DATOS EN REDES DE PC'S"

T E S I S

PARA OBTENER EL TITULO DE

LIC. EN MATEMATICAS APLICADAS Y COMPUTACION

P R E S E N T A :

ALEJANDRA PEÑA BAUTISTA

Asesor: Lic. Sara Camacho C.

Acatlán, Edo. de México 1994

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

UNIVERSIDAD NACIONAL AVENTMA DE MEXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES "ACATLAN"

DIVISION DE MATEMATICAS E INGENIERIA
PROGRAMA DE ACTUARIA Y M.A.C.

SRITA. ALEJANDRA PEÑA BAUTISTA
Alumno de la carrera de Matemáticas
Aplicadas y Computación,
P r e s e n t e.

Por acuerdo a su solicitud presentada con fecha 24 de febrero de 1994, me complace notificarle que esta Jefatura tuvo a bien asignarle el siguiente tema de Tesis "ESTUDIO COMPARATIVO DE LOS DIFERENTES MODELOS DE - BASE DE DATOS EN REDES DE PC'S", el cual se desarrollará como sigue:

INTRODUCCION
CAP. I Redes de Microcomputadoras
CAP. II Análisis de las diferentes tecnologías de redes locales (Ventajas y Desventajas)
CAP. III Fundamentos de las bases de datos.
CAP. IV Tendencias en el Mercado
CONCLUSIONES
GLOSARIO
BIBLIOGRAFIA

Asimismo, fué designando como Asesor de Tesis la LIC. SARA CAMACHO CANSINO, profesor de esta Escuela.

Ruego a usted tomar nota que en cumplimiento de lo especificado en la Ley de Profesiones, deberá presentar servicio social durante un tiempo mínimo de seis meses como requisito básico para sustentar examen profesional así como de la disposición de la Coordinación de la Administración Escolar en el sentido de que se imprima en lugar visible de los ejemplares de la Tesis el título del trabajo realizado. Esta comunicación deberá imprimirse en el interior de la tesis.

A T E N T A M E N T E
"POR MI RAZA HABLARA EL ESPIRITU"
Acatlán, Edo. Méx. octubre 31 de 1994.

ACT. LAURA MARGARITA [illegible]
Jefe del Programa de Actuaria
y M.A.C.

cg'

ESTUDIO COMPARATIVO DE LOS DIFERENTES MODELOS DE BASE DE DATOS EN REDES DE PC'S.

ESTUDIO COMPARATIVO DE LOS DIFERENTES MODELOS DE BASE DE DATOS EN REDES DE PC'S.

INTRODUCCION

El surgimiento de las microcomputadoras a fines de la década de los 70, atrajo la atención de millones de usuarios que vieron en ellas la oportunidad de obtener capacidad de cómputo a bajo costo, con autonomía y versatilidad. El mejoramiento de sus características, velocidad y capacidad del microprocesador, memoria principal y secundaria, permitieron, no sólo capturar el mercado de estudiantes y usuarios particulares, sino también competir favorablemente contra minicomputadoras y equipos grandes en aplicaciones de negocios, en empresas de todos tamaños, al grado de desplazarlos paulatinamente de las salas de cómputo.

Para alcanzar la amplia aceptación que han tenido las microcomputadoras, se han tenido que superar algunas deficiencias que habían persistido pese al mejoramiento de sus capacidades individuales. Por ejemplo: resulta ineficiente compartir información, paquetes o herramientas de programación entre varios equipos, ya que ello implica duplicarlos. Es altamente incosteable conectar de manera dedicada cierto tipo de periféricos a cada microcomputadora de que se dispone.

La solución de éstos y otros inconvenientes fue la estructuración de redes de microcomputadoras, es decir, la interconexión y operación de este tipo de equipos en un ambiente homogéneo, que les permitiera compartir recursos e información de manera eficiente. Las redes además han traído consigo ventajas adicionales respecto a los otros sistemas computacionales.

El concepto de red se ha ampliado enormemente, incluyendo equipos, minicomputadores y equipos grandes, rebasando el ámbito de una sala de cómputo hasta alcanzar con sus nodos diversos puntos de una ciudad o de un país; utilizando como medio de conexión, desde líneas telefónicas convencionales hasta las comunicaciones vía satélite.

Actualmente la Tecnología de Redes ha creado sus propias herramientas de programación y comunicaciones o bien ha adaptado las de otras computadoras (mainframes y minis) en la rama de la computación con mayor desarrollo.

Por tal razón en el primer capítulo de este trabajo se dará un panorama general de las redes de microcomputadoras. Primeramente se definirá y explicará brevemente el concepto de red y de los componentes que la conforman, además de su función dentro de ella. Por otro lado se analizarán los medios de transmisión de datos físicos que existen, tales como: par trenzado, fibra óptica y cable coaxial. Se tratarán ya que en la actualidad estos medios de transmisión de datos físicos son los que tienen mayor uso debido a su precio ó calidad de transmisión, tal es el caso de la fibra óptica, la cual permite enviar información a gran distancia sufriendo poca deformación de la señal, es decir, el material de la fibra óptica es poco sensible al medio ambiente, así como al ruido. También la fibra óptica tiene la característica de poner poca resistencia al paso de la información, es decir, tiene poca atenuación. Así como se describe la fibra óptica, se describirán también los medios de transmisión antes mencionados.

Así mismo se mencionarán los medios de acceso al medio tal como el CSMA/CD, Token y otros, que como se mencionó anteriormente son los que se utilizan con mayor frecuencia. También se exponen los estándares que requiere la construcción de una red local para que esta pueda convivir con otras redes de protocolos de comunicación diferente, lo que implica mencionar el modelo OSI, que juega un papel importantísimo en el roll de las redes de microcomputadoras.

En el capítulo 2 se tratarán los tres tipos básicos de topologías, es decir, las diferentes maneras de distribuir físicamente las estaciones de trabajo, servidores, cableado y demás equipo que pertenece a la red, así mismo, se mencionarán las ventajas y desventajas de las diferentes

topologías, dejando al lector la posibilidad de decidir por la que más le convenga.

En el capítulo 3 se mencionarán los fundamentos de bases de datos, por ejemplo, los modelos de bases de datos existentes en la actualidad así como los que existierón. Se pondrá más enfásis al modelo relacional debido a que en la actualidad es el que predomina en las empresas.

Así mismo se expondrá la diferencia en cuanto a la seguridad e integridad de los datos y el optimizador de consultas que hay entre un DBMS's distribuido y un centralizado.

Se analizarán bases de datos distribuidas, tipos de distribución de datos, reglas que debe cumplir un manejador de bases de datos distribuidas para considerarlo como manejador de bases de datos distribuidas y protocolos de manejo de transacciones distribuidas.

Un sistema distribuido de base de datos consiste en un conjunto de localidades, cada una de las cuales puede paticipar en la ejecución de transacciones que accesen datos de una o varias localidades, la diferencia principal entre los sistemas de bases de datos centralizados y distribuidos es que, en los primeros, los datos residen en una sola localidad mientras que en los últimos, se encuentran en diferentes localidades. Como se verá en este capítulo es la causa de muchos problemas.

Como los medios de almacenamiento de datos son muy veloces permiten pensar en sistemas distribuidos, de esto se desprende la idea de bases de datos distribuidas.

En el último capítulo se discutirá el estado actual de los DBMS's, comunicaciones y tendencias de los sistemas informáticos, basandose en bibliografía actual de libros y revistas. En cuanto a los DBMS's se

expondrá la tendencia que hay de DBMS's centralizados a distribuidos así como lo que conlleva a tener bases de datos distribuidas.

1. REDES DE MICROCOMPUTADORAS.

1.1 DEFINICION

Una red es un conjunto de dispositivos interconectados en un ambiente computacional, para compartir información y recursos informáticos. Los dispositivos pueden incluir computadoras de diversas configuraciones, terminales y periféricos tales como impresoras, lectoras de cinta magnética, digitalizadores y graficadores.
Como parte de la red se cuentan los medios de transmisión de datos como cables, satélites, modems y programas de comunicaciones; en cuanto a las funciones, además de las que pueden desarrollar sus componentes por separado, las redes permiten la transferencia de archivos, el envío y recepción de mensajes (correo electrónico), la operación simultánea de aplicaciones desde varias estaciones de trabajo, entre otras.

El objetivo primordial de una red local es: Compartir recursos materiales (equipos, periféricos) y recursos informáticos (archivos de datos y programas), actualizándolos, organizándolos y explotándolos.

Otra forma de definir a una red de área local es: "Un sistema de comunicación de datos que permite que un número de dispositivos de tratamiento de información independientes como son PC's, minis o mainframes e impresoras se comuniquen entre ellos en un área geográfica limitada que va de aproximadamente 3 metros a 10 km." [1]

Ahora bien, de acuerdo a su distribución geográfica, las redes se clasifican en redes de cobertura amplia o WAN's (Wide Area Networks) y en redes locales o LAN's (Local Area Networks). Las redes locales (figura 1.1) son las que se encuentran instaladas dentro de un edificio o en un conjunto de edificios cercanos entre sí, mientras que las redes de

[1] Teleinformática y Redes de Cómputo. Antonio Alcabau Muñoz.

cobertura amplia o extendida (fig. 1.2) pueden unir puntos de varias ciudades, países e incluso continentes.

RED LOCAL

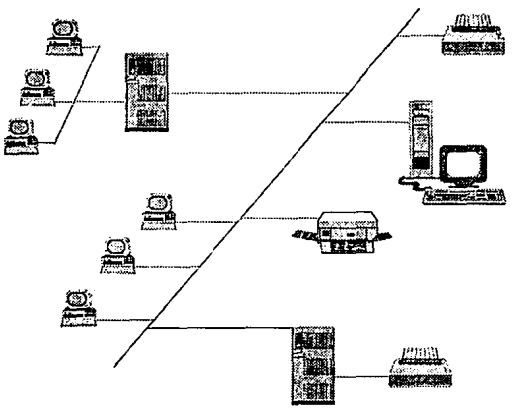

Fig. 1.1

RED DE COBERTURA AMPLIA

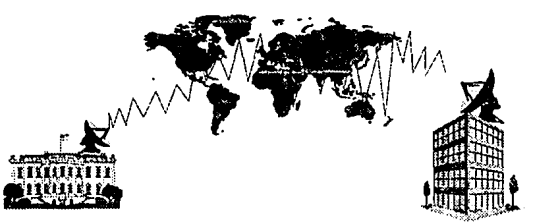

Fig. 1.2

Es una red local dada la cercanía entre los nodos, es costeable su interconexión mediante cables que permiten altas velocidades de transmisión. Además, es muy probable que los equipos tengan las mismas características y que las aplicaciones estén desarrolladas con las mismas herramientas de programación, lo cual facilita la comunicación entre los dispositivos interconectados. En cambio, las redes de cobertura amplia se construyen sobre la infraestructura de telecomunicaciones convencional, esto es, utilizando líneas telefónicas, microondas, comunicaciones vía sátelite que trabajan generalmente a velocidades más reducidas y con la ayuda de equipo que adapte los medios de transmisión de voz para la transferencia de datos. Los equipos que en ella operan tienden a ser más heterogéneos entre sí, minis y macrocomputadoras de distinto fabricante y microcomputadoras que pueden ser o no compatibles con el estándar de IBM.

A este respecto, existen empresas que ofrecen servicios de telecomunicaciones que pueden cubrir todo un país o ciudades importantes en todo el mundo, proporcionando el medio de transmisión para que los usuarios construyan sus propias redes. Dichos servicios incluyen en ocasiones otros beneficios como correo electrónico, consulta a bases de datos, noticias, y otros. Cuando estas redes son propiedad de empresas privadas se les denomina **redes comerciales**. Cuando son propiedad del gobierno del país donde operan se llaman **redes públicas**. Entre las primeras, las más conocidas son TYMNET, TELENET, UNICOL E INFONET. En México, la Red Pública de Transmisión de Datos (TELEPAC), depende de la Secretaría de Comunicaciones y Transportes.

1.2 DEFINICION Y ANALISIS COMPARATIVO DE LOS COMPONENTES DE UNA RED LOCAL

Los elementos con los que cuenta una red pueden variar dependiendo de las funciones específicas para las que esté diseñada. Sin embargo, existen elementos comunes a todas ellas que son los que veremos a continuación:

1.- **Estación de trabajo:** son las computadoras o terminales, desde las cuales el usuario puede accesar a la red.

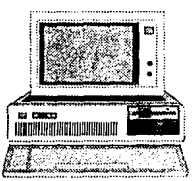

Fig. 1.3

2.- **Servidor (server):** Es un dispositivo que proporciona una función especial a todos los usuarios de la red.

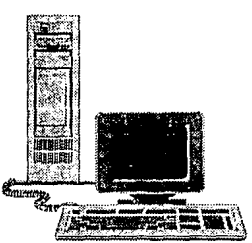

Fig. 1.4

Entre los tipos de servidores más importantes se encuentran los siguientes:

- **Servidor de archivos:** Provee área de almacenamiento y acceso a programas y archivos de datos compartidos.
- **Servidor de impresión:** Controla las colas de impresión y da acceso a la o las impresoras conectadas a él.
- **Servidor de base de datos:** Equipo dedicado al almacenamiento y organización de base de datos y a la recuperación de los datos solicitados en las consultas.
- **Servidor de comunicaciones:** Equipo dedicado para atender las comunicaciones entre estaciones de trabajo remotas y los demás dispositivos de la red.

En el server se carga el sistema operativo de red, además se conectan los dispositivos que van a compartir las estaciones de trabajo tales como impresoras, modems, graficadores y otros.

En realidad los equipos que operan como servidores son computadoras semejantes a las que se utilizan como estaciones de trabajo aunque generalmente con mayor capacidad, en las cuales se ejecuta un programa que les permite desempeñar la función especial que desarrollan. Cuando el equipo únicamente desempeña esa función se le llama **servidor dedicado.** Cuando además se utiliza como estación de trabajo se le llama **servidor no dedicado.**

3.- Tarjeta de red: Tanto las estaciones de trabajo como los servidores se encuentran conectados entre sí mediante cables especiales. Cada equipo cuenta con una tarjeta o adaptador de red, a la que se conectan dichos cables y está diseñada para controlar las comunicaciones de la estación de trabajo con los demás puntos de la red.

Entre las tarjetas más populares estan: la Ethernet, Token-ring y Arcnet.

- **Tarjeta Ethernet:** Esta tarjeta combina el método de acceso CSMA/CD (que se explica en el inciso 1.5) y la topología de bus (que se explicará en el capítulo 2); trabaja a una velocidad de transmisión de 10 millones de bits por segundo (Mbits/seg). Este tipo de tarjeta surgió a mediados de la década de los 70 y después de algunas modificaciones hechas por ISO/OSI, se convirtió en estándar. En sus conexiones se utiliza un cable coaxial de doble blindaje. Existen varios tipos de tarjetas Ethernet, dependiendo la elección del procesador que utilice la microcomputadora donde se va a instalar (de 8 bits para equipos 8086, de 16 bits para equipos 80286 y 80386, tarjeta de 32 bits para equipos 80486). La tarjeta Ethernet tiene como principales ventajas su relativo bajo costo y su conectividad con equipos grandes.

- **Tarjeta Token-Ring:** Esta tarjeta combina la topología de anillo (véase capítulo 2) con el método Token-Passing (que se explica en el inciso 1.5). Cuando salió al mercado en 1987, operaba a una velocidad de 4 Mbits/seg., actualmente opera a 16 Mbits/seg. Su diseño se debe a IBM, por lo que se convirtió en estándar desde su aparición. La tarjeta Token Ring se caracteriza por su mayor alcance y porque su rendimiento no se degrada al aumentar el número de nodos de la red. Sin embargo su costo es considerablemente alto, así como el equipo y programas de comunicación requeridos para su operación.

- **Tarjeta Arcnet:** Esta tarjeta utiliza una topología de árbol, método de acceso Token passing (que se explica en el inciso 1.5) y transmite a una velocidad de 2.5 Mbits/seg. Su sistema de cableado utiliza cable coaxial y requiere que en cada "rama" del árbol se conecten repetidores para mantener la señal en una intensidad adecuada. Fue creada por Data Point Corporation y su popularidad se incrementó a partir de 1986, siendo a la fecha una de las tarjetas más utilizadas en el mundo. Arcnet tiene una gran flexibilidad en el cableado, lo que le permite, en el caso de redes grandes, formar

redes pequeñas, de 20 nodos, que se unen mediante puentes. Una de sus principales desventajas es su baja velocidad de transmisión.

La elección de la tecnología de red más adecuada dependerá de las condiciones en que se desee instalar la red. La tabla I contiene algunos de los elementos que pueden ayudar a esta elección.

	PAR TRENZADO	COAXIAL	COAXIAL BANDA ANCHA	FIBRA OPTICA
TOPOLOGIAS	ANILLO ESTRELLA LINEAL ARBOL	ANILLO LINEAL ARBOL	LINEAL ARBOL	ANILLO ESTRELLA ARBOL
NODOS MAXIMOS	< = 1,024	< = 1,024	< = 25,000	< = 1,024
LONGITUD MAXIMA	3 Km.	10 Km.	50 Km.	10 Km.
INMUNIDAD AL RUIDO	BAJA	MEDIA	ALTA	MUY ALTA

Tabla I

4.- **Protocolo:** Es el conjunto de reglas que se siguen para empacar la información que va a ser enviada por las estaciones de trabajo y los servidores de red. Se considera como el lenguaje de la red. Entre los ejemplos más importantes de los protocolos de red se encuentran los siguientes:

- **TCP/IP (TRANSMISSION CONTROL PROTOCOL/INTERNET PROTOCOL):** Desarrollado por el Departamento de Defensa de Estados Unidos, se convirtió en un estándar "de facto" debido a la posición que guarda su desarrollador como demandante de equipo de cómputo, por lo que se encuentra disponible para casi todas la plataformas.

- **NetBIOS (NETWORK BASIC INPUT-OUTPUT SYSTEM):** Protocolo estándar distribuido por IBM, basado en uno similar de la empresa Microsoft. Se considera como un protocolo genérico de red ya que es compatible con muchos sistemas operativos.

- **IPX/SPX (INTERNET PACKET EXCHANGE/SEQUENCED PACKET EXCHANGE):** Desarrollado por Novell Inc., empresa que tiene el mayor número de sistemas operativos de red instalados en el mundo, por lo que este protocolo está ampliamente aceptado.

1.3 ANALISIS DE LOS MEDIOS DE TRANSMISION

Sistema de cableado: Dentro de éste, se incluyen los cables y elementos adicionales asociados a ellos como cajas de conexiones y conectores especiales, que se utilizan en la interconexión de los puntos de la red. La forma de conexión en los equipos (topología), está en función de la tarjeta que se haya seleccionado. Los tipos de cable más utilizados son básicamente tres.

1.- **Par trenzado (Twisted-pair):** Cable utilizado comunmente en las instalaciones telefónicas domésticas (fig. 1.5), que se forma de dos alambres de cobre aislados que se tuercen entre sí. Pese a su popularidad, introduce distorsión y ruido en la línea cuando aumenta la distancia o la velocidad de transferencia. Estas deficiencias se han corregido en parte, agregándole un blindaje, aunque lo ha hecho más costoso. Sus principales características son:

- Un canal puede transportar de 12 a 24 canales de 300-3kHz.
- Son válidos en topologías: anillo, bus y estrella.
- Bajo costo.
- Alta tasa de error a grandes velocidades.

- Alta interferencia.
- Alcance de hasta 3 Kms.
- Se utilizan repetidores para ampliar las distancias.

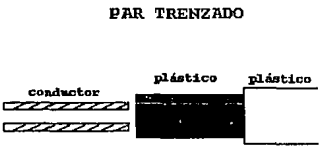

Fig. 1.5

2.- **Cable coaxial:** Alambre sólido protegido por un aislante y una malla de metal (fig. 1.6). De mayor calidad que el par trenzado, pero más caro. Existen dos tipos de cables coaxiales cuyas características son:

Cable coaxial de banda base

- No hay modulación de frecuencia.
- Uso de enchufes especiales para la conexión física.
- Se usa una tarjeta de interconexión de red.
- Se usa generalmente en topología bus.
- Alcance de 1 a 10 km.
- Ancho de banda de 10 Mbps.
- Bajo costo.
- Simple de instalar.
- Poca inmunidad al ruido.

Cable coaxial de banda ancha.

- Se combina voz, datos y video simultáneamente.
- Los datos son modulados antes de la transmisión.
- Instalación más difícil en comparación con el de banda base.
- Se utilizan amplificadores (para largas distancias).
- Alcance hasta 5 kms.
- Topología soportada: bus.
- Ancho de banda máximo 400 kHz.
- Mayor inmunidad al ruido.
- Alto costo.

CABLE COAXIAL

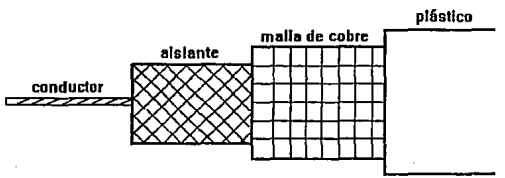

Fig. 1.6

3.- **Fibra óptica:** Cable hecho de fibra de vidrio o plástico (fig. 1.7), a través del cual se emite un haz de luz luminoso que se va reflejando dentro del cable debido a los diferentes índices de refracción entre éste y su cubierta. Su instalación es considerablemente más costosa que en los casos anteriores, ya que requiere de cuidados y mantenimiento especial, aunque en condiciones adecuadas permite velocidades de transmisión más elevadas y a mayores distancias. Se espera que con los avances

técnicos se convierta en el medio de transmisión del futuro. Sus características son las siguientes:

- La señal no se modula.
- No es afectada por la interferencia eléctrica, ruido, temporales, radiación o agentes químicos.
- Ancho de banda de 50 Mbps.
- Se puede transmitir datos, voz y video.
- Muy poca pérdida de señal.
- Topologías: anillo y estrella.
- Actualmente muy cara.

FIBRA OPTICA

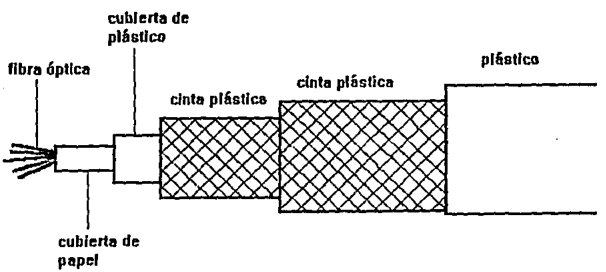

Fig. 1.7

1.4 MODELO OSI

La competencia entre los distintos fabricantes de equipo, desarrolladores de paquetes y sistemas, ha llevado a una alta especialización. Sólo las grandes empresas como IBM, Hewlett Packard, UNISYS, Digital, WANG, etc., son capaces de cubrir una amplia gama de campos dentro de la computación, mientras que la gran mayoría se dedica a productos específicos, lo que ha dado como resultado que ningún vendedor, por sí sólo, puede proporcionar los elementos para la solución integral a los requerimientos de cómputo de una entidad determinada, sobre todo considerando la relación costo-efectividad como un parámetro fundamental en la decisión del cliente.

Ello llevó en un principio, a que los productos de diferentes proveedores fueran de características distintas de fabricación, con protocolos de comunicación particulares, lo que hacía que fuera extremadamente difícil ponerlos a trabajar juntos.

El primer antecedente lo constituyó la Agencia de Proyectos de Investigación Avanzada (ARPA) del Departamento de Defensa de Estados Unidos, la cual formuló, a fines de los años 60, un conjunto de estándares denominado ARPANET. Poco tiempo después surgió el estándar BSD (Berckeley Software Distribution) propuesto por la Universidad de California. Ambos fueron ampliamente aceptados, aunque sólo resolvían parte de los problemas de interconectividad de las redes y no garantizaban la adecuada comunicación entre equipos de distinto fabricante, aun y cuando siguieran dichos estándares.

En 1972, el Instituto Nacional de Estándares de Estados Unidos (ANSI), desarrolló el Modelo de Referencia para Sistemas Distribuidos, el cual fue tomado en 1978 como base por la Organización Internacional de Estándares (ISO), para diseñar el Modelo de Referencia para la Interconexión de Sistemas Abiertos (OSI), el cual es, a la fecha, el más aceptado por los fabricantes.

El modelo OSI, publicado en 1983, divide los aspectos internos de los sistemas de comunicación en siete partes denominadas niveles "layers", los cuales proporcionan una representación consistente de las comunicaciones en una red. Los niveles del 1 al 4 (tabla II) definen la manera en que deben de establecerse las comunicaciones entre computadoras, a fin de permitir que los datos pasen de la fuente a su destino, sin importar el formato en que se encuentren. En caso de que exista algún error en la transmisión, las funciones de estos niveles permiten que dicho error sea reportado y diagnosticado. Los niveles del 5 al 7 se refieren de manera específica a la aplicación que está operando. Los niveles 6 y 7 en particular, hacen posible que los datos que se intercambian tengan el sentido correcto.

NIVEL	DESCRIPCION
7 Aplicación	Este nivel proporciona protocolos para funciones o aplicaciones comunes de usuario final con transferencia de archivos, correo electrónico o acceso remoto a bases de datos.
6 Presentación	Proporciona mecanismos de traducción entre datos de distinto formato, así como de traducción de códigos y conversión de datos de archivos.
5 Sesión	Es responsable de establecer, mantener y terminar las sesiones de comunicación entre nodos. Además sincroniza y traduce nombres y direcciones de bases de datos.
4 Transporte	Asegura una alta confiabilidad en el envío-recepción de datos. Puede dividir los paquetes en otros más pequeños y después volverlos a unir para cumplir su función adecuadamente.
3 Red	Su función consiste en controlar el flujo de información entre los nodos de la red, proporcionando las trayectorias que deben seguir los paquetes de datos para llegar a su destino (su función es sencilla en redes locales donde las trayectorias entre los nodos son únicas y se complica en redes de cobertura amplia o con redes interconectadas). Sus unidades de referencia son los paquetes de datos.
2 Enlace de datos	Se encarga de mantener una comunicación confiable entre los nodos, a pesar de ruido en el canal físico. Organiza los mensajes en cadenas de bits (frames) con identificación de origen y destino. Su unidad de referencia es el "frame".
1 Físico	Es responsble de las especificaciones físicas y eléctricas para la transmisión de datos a través del medio físico (cables). En él se definen los tipos de conectores y de puertos, así como los rangos de corriente.

Tabla II

A partir de este modelo, se han desarrollado básicamente cuatro tipos de productos que operan en diferentes niveles y que resuelven los problemas de enlace entre redes. Ellos son: repetidores, puentes, ruteadores y traductores de protocolo (gateways).

1.5 ANALISIS DE LOS DIFERENTES METODOS DE ACCESO AL MEDIO

Esquemas de acceso: Es la forma en que estan organizadas las comunicaciones dentro de la red, a fin de sincronizar convenientemente el envío y recepción de los mensajes desde cada estación. Los esquemas más importantes que se conocen son el CSMA/CD, el Token-Passing y el esquema de "pooling".

- **Esquema CSMA/CD (Carrier-Sense Multiple Access/Collision Detection):** Mediante este esquema, cada estación debe esperar a que el canal de la red se encuentre sin transmisión para iniciar el envío de la información. Si se detecta que otro equipo también está realizando un envío, frenará la transmisión e intentará enviar nuevamente su mensaje cuando el canal esté desocupado. El CSMA/CD fue desarrollado por XEROX como parte de la red local Ethernet. Este método de acceso al medio está asociado con la topología bus. Su forma de trabajo se puede resumir de la forma siguiente:
  El protocolo CSMA/CD (Carrier Sense Multiple Access/Collision Detection) de acceso multiple por sensibilidad de portadora/detección de colisiones requiere de un dispositivo para "escuchar" antes de transmitir el mensaje. El dispositivo puede enviar el mensaje sólo cuando el medio esté libre de señal. En caso de que dos nodos comiencen a enviar el mensaje simultáneamente, se detectará la colisión y se detendrá la transmisión.

- **Esquema Token-Passing:** El "token" es una señal especial que circula por la red. Sólo puede enviar información la estación por donde va pasando el "token" en ese momento. En caso de que una estación no tenga información que enviar, simplemente dejará pasar el "token" a la siguiente estación. Este esquema se considera más eficiente que el anterior (fig. 1.8).

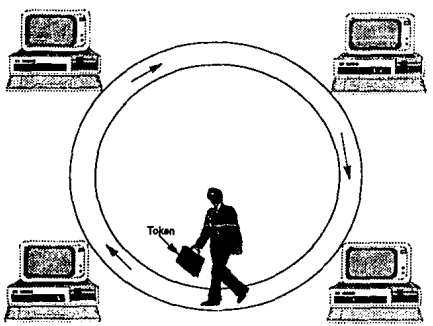

Fig. 1.8

- **Esquema de pooling:** Este esquema está asociado usualmente a la topología de estrella. Consiste en que, de manera periódica, el equipo ubicado en el centro de la red, un servidor de archivos o un procesador central, pregunta a cada uno de los nodos si tiene algún mensaje que enviar. Si es así, el mensaje es leído; en caso contrario, la pregunta se le hace al nodo siguiente. Lo mismo ocurre cuando el equipo central es quién desea enviar el mensaje. Este funcionamiento se asemeja al de un reloj con una manecilla. De esa forma elimina la posibilidad de que una estación de trabajo interfiera en las comunicaciones de otra (fig. 1.9).

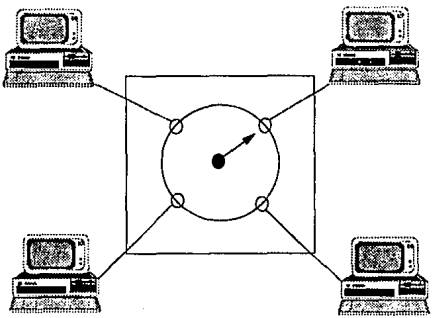

Fig. 1.9

2. ANALISIS DE TECNOLOGIAS DE REDES LOCALES (VENTAJAS Y DESVENTAJAS)

La manera de interconectar los distintos elementos de una red proporciona una primera visión de la estructura y comportamiento de ésta. A la configuración geométrica resultante se le denomina topología de red. Las características más importantes que definen a una red local son su topología y su esquema de acceso. La conjunción de ambas define la topología de una red específica.

Los nodos que aparecen en esta configuración pueden representar tanto elementos terminales de comunicación (estaciones de usuario o servidores de recursos comunes) como elementos de unión de los distintos ramales en que se divide la red.

La elección de la topología tiene un fuerte impacto en el comportamiento de la red. Aunque como más adelante veremos, el eficaz aprovechamiento de ésta dependerá de una serie de protocolos de comunicación entre sus distintos elementos, también la estructura topológica condiciona algunas características. Cabe citar entre las más relevantes:

- La mayor o menor flexibilidad para añadir o quitar estaciones.
- La repercusión que en el comportamiento de la red pueda tener el fallo en una de las estaciones.
- El flujo de información que pueda transitar por la red sin que se produzcan interferencias y los retardos mínimos que ésta introduzca.

Topología de red: Es la forma en que estan conectados entre sí los componentes de la red. Dicho de otra manera, este término se refiere a la distribución física de las estaciones de trabajo, servidores, impresoras y demás equipo que pertenece a la red. Esta definición quedará más clara con la revisión de los tipos de topología básicos:

2.1 Topología de estrella (star)

En una red en estrella todas las estaciones se comunican entre sí a través de un dispositivo central. En esta topología cada estación está conectada directamente a un equipo central, generalmente el servidor de la red (fig. 2.1).

El nodo central asume un papel muy importante debido a su protagonismo en todas las transferencias de información que se realicen en la red. Lo usual será que el nodo central realice todas las tareas de control y posea los recursos comunes de la red; para reducir su influencia puede optarse por localizar el control en alguno o algunos de los nodos periféricos, de manera que el nodo central actúe como una unidad de conmutación de mensajes entre todos los nodos periféricos.

Esta configuración presenta buena flexibilidad para incrementar o reducir el número de estaciones, debido a que estas modificaciones no representan ninguna alteración de su estructura y están localizadas en el nodo central.

La repercusión de un problema en uno de los nodos periféricos en el comportamiento global de la red es muy baja y sólo afectaría al tráfico relacionado con ese nodo. Sin embargo, si el problema se produjece en el nodo central, el resultado podría ser catastrófico y afectaría a todas las estaciones de trabajo.

El flujo de información puede ser elevado y los retardos introducidos por la red, pequeños, si la mayor parte de la información fluye entre el nodo central y los nodos periféricos. En caso que las comunicaciones se produzcan entre estaciones, el sistema se vería restringido por la posible congestión del dispositivo central.

El protocolo que utiliza es el token. Las redes estrella fueron las primeras redes en desarrollarse debido a su estructura sencilla. Las desventajas son:

- En caso de fallar el FILE SERVER, todo el sistema deja de funcionar.
- La red puede crecer solo hasta alcanzar la capacidad del FILE SERVER.
- Resulta costosa esta configuración por la cantidad de cable a utilizar.

Las ventajas son:

- Si un nodo deja de funcionar la red sigue funcionando, así mismo la flexibilidad es buena permitiendo adicionar o suprimir con sencillez estaciones de trabajo.

TOPOLOGIA ESTRELLA

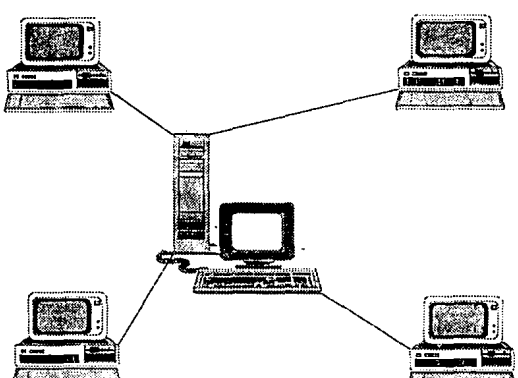

Fig. 2.1

Es la topología de más uso en las configuraciones de minis y equipos macro con terminales. Para redes de micros tiene algunas desventajas como es el uso excesivo de cable y la dependencia que existe, en caso de falla, del equipo que queda en el centro de la red.

2.2 Topología de anillo (ring)

En este caso, el cable pasa a través de cada estación y los servidores, hasta formar un anillo, de modo que cada estación tiene conexiones con otras dos (fig. 2.2). Los mensajes viajan por el anillo de nodo en nodo y en una única dirección, de manera que toda la información pasa por todos los módulos de comunicación de las estaciones de trabajo.

Cada nodo debe ser capaz de reconocer los mensajes dirigidos a él y actuar como retransmisor de los mensajes que van dirigidos a otras estaciones de trabajo.

Esta topología permite incrementar y disminuir el número de estaciones sin gran dificultad. Debido a que cada estación de trabajo está obligada a retransmitir cada mensaje, en caso de existir un número elevado de estaciones de trabajo el retardo introducido puede ser demasiado grande para ciertas aplicaciones.

En la estructura de anillo, una falla en cualquier parte de la vía de comunicación deja bloqueada a la red en su totalidad. En el caso de una configuración en estrella sólo quedaría fuera de servicio la estación de trabajo afectada y en la configuración lineal (bus), como se verá a continuación solo quedarían afectados algunos nodos. Si la falla se produce en una de las estaciones del anillo, la repercusión en el resto de la red será diferente dependiendo de si se avería o no el módulo de retransmisión, en caso de que la estación quede fuera de funcionamiento pero el módulo de retransmisión siga funcionando

normalmente, la avería sólo afecta la estación en cuestión. Pero si lo que falla es el módulo de comunicaciones, el anillo quedará cortado y la red bloqueada.

La forma de evitar estos riesgos consiste en el uso de concentradores en la configuración de una red en anillo. El concentrador es un dispositivo, fabricado con un alto nivel de fiabilidad, al que se conectan las estaciones de la red. Donde el anillo lógico se encuentra dentro del concentrador, y cuando un nodo deja de funcionar, se cortocircuita la entrada hacia la estación en el propio concentrador, restableciéndose el anillo. Como el número de estaciones conectables a un concentrador es limitado, se puede recurrir a concatenar varios concentradores para conseguir redes en anillos con más nodos periféricos.

En esta topología la información viaja en un solo sentido a través de un solo cable coaxial u otro medio. La información pasa de un nodo a otro.

En la actualidad no existen verdaderas topologías de anillo en el mercado, ya que tiene una desventaja fundamental. Si un nodo (estación de trabajo) falla, toda la red deja de funcionar.

Otro problema propio de la topología anillo es que a medida que se transmiten los mensajes puede disminuir la velocidad de la red. La ventaja de la red anillo es que se requiere de un mínimo de inteligencia, siendo de un costo menor en comparación con las otras.

TOPOLOGIA ANILLO

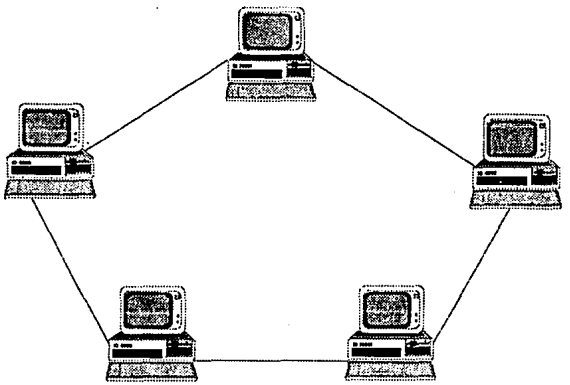

Fig. 2.2

2.3 Topología lineal (bus)

Tanto los servidores como las estaciones de trabajo se conectan a un cable (o bus) que atraviesa la red (fig. 2.3), es decir, todos los nodos están conectados a un único canal de comunicación. Es la más utilizada de las topologías en redes locales por su flexibilidad y confiabilidad, ya que las fallas en un punto no tienen efecto en la operación global. La red tipo bus permite que los mensajes sean transmitidos en todo el canal, cuando un nodo reconoce que el mensaje es para él lo saca del canal. A consecuencia de que en el canal sólo puede viajar un solo mensaje el bus requiere que cada nodo pueda transmitir, y recibir datos además de resolver colisiones. La topología bus utiliza el protocolo de contención CSMA/CD (Carrier Sense Multiple Access/Collision Detection).

TOPOLOGIA BUS

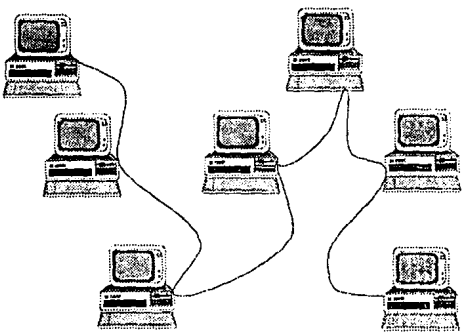

Fig. 2.3

A diferencia de la topología de anillo, cada nodo no ha de actuar como repetidor de los mensajes, sino que simplemente ha de reconocer su propia dirección para captar aquellos mensajes que viajan por el bus y van dirigidos a él. Cuando una estación de trabajo deposita un mensaje en la red, esta información es difundida a través del bus y todas las estaciones estarían capacitadas para recibirla. Debido al hecho de compartir el medio, antes de transmitir un mensaje cada nodo debe averiguar si el bus esta disponible para él.

Las redes en bus son sencillas de instalar y se adaptan con facilidad a las características del terreno o local. Presentan una gran flexibilidad en lo referente a reducir o aumentar el número de estaciones de la red, ello, unido a su buena fiabilidad, hace que esta topología haya sido la elegida por numerosos proveedores.

La falla en una estación aislada solo repercutirá en los mensajes dirigidos a ella, siendo su efecto nulo en el resto de la red. Una ruptura

en el bus, en cambio, deja a la red dividida en dos o inutilizada totalmente, según esté concebido el control.

El hecho de que exista un bus común al que acceden todas las estaciones le proporciona parte de las ventajas antes referidas, pero obliga a que el control de acceso a la red sea más delicado que en el caso de las topologías en estrella o anillo.

3. FUNDAMENTOS DE BASES DE DATOS

3.1 DEFINICION

A continuación se presentan algunas definiciones dadas por varios autores respecto de lo que es una base de datos:

Una **base datos** es un conjunto de ocurrencias de varios tipos de registros en el cual tanto los tipos de registros como sus ocurrencias están interrelacionados mediante **relaciones** específicas.

Una **base de datos** es un conjunto de archivos lógicos interrelacionados entre sí, de manera que reuniendo diversos elementos de los diferentes archivos se obtengan estructuras de información no redundantes y adecuadas a distintos procesos en el cual puedan estar entremezclados registros diferentes de diferentes tipos de archivos en un mismo o varios soportes físicos.

Una **base de datos** es una colección de datos relacionados acerca de una organización, con múltiples usos. En una base de datos las definiciones de los datos y las relaciones entre ellos están separadas de las declaraciones de procedimientos de un programa.

3.2 COMPONENTES

¿Qué significa **manejo** o **administración de datos**? Es un término muy general y amplio que se refiere a la tecnología de computadoras requerida para organizar, almacenar, recopilar y manipular datos. La unidad más pequeña de datos que se considera por lo general es el **dato elemental**, llamado también **campo** o **atributo**; por ejemplo el número de un empleado. Un conjunto de datos elementales constituye un **registro lógico** o **entidad**. Un tipo de registro lógico es un registro con una constitución particular de datos elementales; por ejemplo un registro de empleado constituido por un número de empleado, un

nombre y una dirección. Un **archivo** es una colección de ocurrencias de un mismo tipo de registros; por ejemplo un archivo de registros de empleados.

¿Por qué se tiene la necesidad de las bases de datos?

La pregunta que se debe plantear aquí es: ¿cuál es la principal diferencia entre una base de datos y un archivo de datos? Una base de datos puede tener más de un uso, y los múltiples usos pueden satisfacer múltiples **"enfoques"** de los datos almacenados. Un archivo de datos puede tener más de un uso, pero sólo se puede satisfacer un "enfoque" de datos almacenados. Los diferentes enfoques de un archivo sólo se pueden satisfacer después de clasificar los datos.

En una base de datos se concentran y se interrelacionan una gran diversidad de datos para otros tantos usuarios, a diferencia de los archivos convencionales, que usan unas cuantas personas o incluso una sola. Cualquier aplicación o usuario individual estará involucrado únicamente con una pequeña porción de la base de datos. Las porciones de diferentes usuarios pueden traslaparse; esto es, pueden tenerse en común partes de las porciones como se muestra en la figura 3.1. La habilidad para compartir y usar los datos permite que se reduzca al mínimo el espacio redundante de almacenamiento.

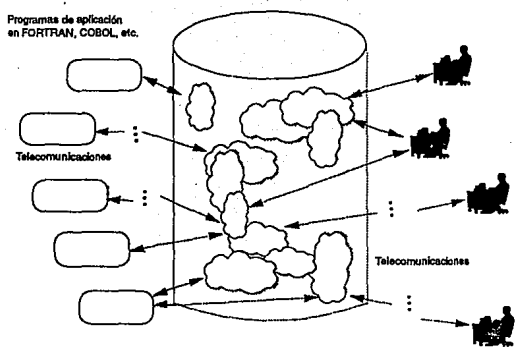

Fig. 3.1 Esquema de un sistema de base de datos integrada.

Un sistema de manejo (o administración) de bases de datos **DBMS** es, en su forma más general, un sistema de programas o software capaz de dar soporte y manejar una base de datos integrada como la que se describe esquemáticamente en la figura 3.1. Los programas de aplicaciones elaborados con los lenguajes convencionales de programación por procedimientos, como COBOL, PASCAL, C y otros, pueden tener acceso a partes específicas de la base de datos. Los usuarios individuales también pueden tener acceso en línea a determinadas partes de la base, vía un lenguaje especial para consultas (**SQL**, del inglés **Structured Query Language**) si existe alguno. Una porción de la base de datos podría considerarse similar a un archivo convencional, o una base en sí que estuviera constituida por varios tipos de registros y relaciones.

En resumen, un **DBMS** es un software que:

- almacena, recupera y modifica datos
- guarda la consistencia de los datos
- resuelve problemas de concurrencia

- permite una interface universal a los datos
- regula el acceso a los datos

Algunos usuarios o programas de aplicaciones podrían tener comunicación con la base mediante un sistema de telecomunicaciones desde localidades remotas. Puede tenerse acceso a la base de datos con el propósito de recopilar, adicionar, eliminar o actualizar datos.

a) DDL, DML y DCL.

Un RDBMS cuenta con los siguientes componentes: DDL (Data Definition Language), DML (Data Management Language), DCL (Data Control Language) y el optimizador de consultas. Estos se describen a continuación.

Un lenguaje de definición de datos (**Data Definition Language, DDL**), permite especificar el esquema de la base de datos. El resultado de la compilación de las proposiciones en el **DDL** son datos sobre la estructura de los archivos, los cuales se almacenan en el diccionario de datos, por lo que el contenido del diccionario de datos, son datos de los datos (es decir, metadatos).

Lenguaje de Definición de Datos (DDL)

- Sintáxis dependiente del DBMS.

- Permite definir:

  * nombre de la RELACION.
  * ATRIBUTOS que la componen.
  * DOMINIOS asociados a cada atributo.
  * LLAVE de la relación.
  * VALORES NULOS.

* RESTRICCIONES DE INTEGRIDAD.

Un lenguaje de manipulación de datos **(Data Management Language, DML)**, tiene como tarea actualizar y recuperar datos.

Existen dos tipos de **DML**: los procedurales y los no procedurales.

En el DML procedural el usuario especifica cuales datos quiere y cómo debe obtenerlos, en el DML no procedural el usuario indica qué datos desea sin decir como obtenerlos.

Otra característica de los no procedurales es que no contienen ciclos iterativos como son "for", "while" y otros.

Lenguaje de Manipulación de Datos (DML)

1. Modificación de la estructura:

   - agregar atributos
   - suprimir atributos
   - modificar sus características (nombre, tipo, longitud,...)

2. Modificación del contenido (en forma unitaria o de conjuntos):

   - almacenar nuevas tuplas
   - suprimir tuplas
   - modificar el valor de los atributos

Un lenguaje de control de datos (**Data Control Language, DCL**), permite especificar a aquellos usuarios que podrán accesar a la información así como sus privilegios para el control de la base de datos

b) El optimizador de consultas

Para el caso de los manejadores que utilizan el modelo relacional y como lenguaje de consultas al SQL, la petición de datos se puede hacer de diferentes maneras; esto es que la consulta puede expresarse de diferentes formas, cada una de las formas de expresar la consulta sugiere una **estrategia** para encontrar la respuesta. Estas estrategias son llamadas **trayectorias de acceso**. El procesamiento de recursos que son necesarios para recorrer las diferentes trayectorias de acceso para la recuperación de datos puede variar. Consecuentemente el optimizador es el componente del **DBMS** responsable de transformar la pregunta en una forma equivalente que pueda resolverse más eficientemente.

El optimizador juega un papel muy importante en el rendimiento **(performance)** del DBMS .

Los fabricantes de DBMS's no difunden la tecnología de sus optimizadores, sin embargo, es posible para los usuarios considerar algunos productos y evaluar el optimizador utilizando información propia.

El optimizador se puede considerar como un sistema que incorpora los conocimientos del administrador de bases de datos de determinada empresa y la experiencia de muchos investigadores de las ciencias del acceso a las bases de datos. Estos conocimientos toman la forma de estadísticas que describen a la base de datos, tamaño de la base de datos, índices, descripción de campos y otros. Entonces antes que se pueda optimizar una pregunta se debe traducir a una forma interna reconocible al manejador de consultas, a continuación, se verifica la sintáxis de la consulta del usuario y se checa que los nombres en la consulta sean nombres registrados en la base de datos.

Una vez que se ha traducido la consulta a una forma interna tal como el álgebra relacional, comenzará el proceso de optimización el cual está dividido en las siguientes fases:

a) encontrar una estrategia detallada para procesar la consulta (orden en que se ejecutarán los comandos).

b) encontrar una estrategia que utilice menos accesos al disco.

Como la información cambia constantemente, ya que se hacen modificaciones a las relaciones, y dado que los DBMS's no actualizan la información del diccionario cada vez que se realiza una modificación a las tablas, la información que toma el optimizador no es del todo exacta por lo que la estrategia tomada puede no ser la mejor.

La información estadística es muy importante cuando se tienen índices en las relaciones. Los índices permiten tener acceso rápido a las túplas que cumplen con un valor determinado en la llave de indización, además existen índices que son agrupados ("**clustered**")* ; lo que permiten estos es que el rendimiento sea mejor; además permiten que los registros de un archivo se puedan leer en un orden que corresponda aproximadamente al orden físico por lo que permiten leer las túplas en bloques.

Como es de pensarse se utiliza tiempo para leer los índices que se encuentran en el disco, por lo que es necesario considerar estos accesos cuando se estima el costo de la consulta tomando en cuenta índices.

* El "performance" (rendimiento) de un RDBMS se degrada considerablemente conforme el número de renglones se incrementa en una tabla. Los índices agrupados "clustered"(una variación sofisticada de índices B-tree) contienen tanto a los índices como a los datos clasificados en secuencia. Los índices "clustered" permiten que los datos sean almacenados al nivel de hoja índice, de forma tal que los índices y los datos puedan ser físicamente recuperados en una sola operación de I/O, en lugar de múltiples I/Os.

c) SQL: el lenguaje relacional estándar

Cuando E.F. Codd introdujo el concepto de una base de datos relacional en 1970, declaró: **"la adopción de un modelo de datos relacional...permite el desarrollo de un sublenguaje de datos universal basado en un cálculo de predicados aplicado."** No obstante que él indicó los requerimientos y las ventajas de un lenguaje tal, no intentó en ese momento crear uno. En un artículo posterior, discutió el concepto de completitud relacional.

La aceptación de la idea relacional fue relativamente lenta (en comparación con la usual rápidez de los avances técnicos en el campo de la computación). Por lo tanto, no fue sino hasta 1974 que Chamberlin y Boyce publicaron un artículo sugiriendo la forma de un lenguaje de peticiones estructurado, el cual en ese momento fue llamado **SEQUEL** (de aquí surge la actual pronunciación de SQL). Al año siguiente, Boyce, Chamberlin, King y Hammer publicaron un artículo dando a conocer el sublenguaje **SQUARE** el cual era muy parecido al **SEQUEL** con la excepción de que **SQUARE** usaba expresiones matemáticas en lugar de términos en simple inglés como **SEQUEL.** Ambos lenguajes son mostrados por los autores como completamente relacionales en el sentido descrito por Codd en sus artículos de 1970 y 1971. "Completamente Relacional" en este contexto significa "al menos tan poderoso como el cálculo relacional de túplas."

El artículo sobre **SQUARE** fue seguido por otro artículo por Chamberlin y otros en 1976 cuando el nombre fue cambiado a **SEQUEL 2,** y fue usado como el lenguaje de consultas para la base de datos experimental de IBM **System R.**

Para el momento en que Chamberlin escribió en 1980, el resúmen de su experiencia como usuario del lenguaje, el nombre había sido cambiado a su forma actual: **SQL**, denotando un Lenguaje de

Consultas Estructurado (**Structured Query Language**). Una prueba de **SQL** por un amplio grupo de usuarios dio como resultado varias adiciones al lenguaje, incluyendo la adición de juntas externas "**outer joins**" a las capacidades de SQL, lo cual Codd ya había sugerido en su artículo de 1979.

Desarrollos adicionales reportados en otros artículos dieron como resultado el SQL como se conoce hoy.

Durante la última década, las bases de datos relacionales han emergido y se han hecho cada vez más populares que las bases de datos jerárquicas y de red que las precedieron. Esta tendencia parece estar acelerándose. Para inyectar algo de orden en la creciente literatura sobre bases de datos relacionales, Codd en 1985 estableció doce principios, de los cuales **seis al menos deben ser satisfechos para que una base de datos se considere relacional.**

El hecho de que los desarrolladores supieran con anticipación lo que SQL debía ser y lo que se requería hacer proporcionó un fuerte fundamento teórico. Esto es probablemente algo único en el desarrollo de lenguajes de computadora ya que la mayoría de los lenguajes de programación en uso actualmente son el resultado de una idea básica suplementada por una gran cantidad de ajustes ad hoc para enfrentar los problemas conforme estos surgían. Este hecho - especificando la necesidad para SQL antes de desarrollar la mecánica del mismo- dio origen a un lenguaje elegantemente parsimonioso, que consiste de unos cuantos comandos (relativamente) que pueden ser usados para satisfacer la mayoría de la necesidades de una base de datos muy compleja. Su simplicidad hace al SQL conveniente para el usuario casual así como también para el desarrollador sofisticado. Puede ser usado para "queries" (consultas o peticiones) ad hoc, y también puede ser intercalado en un programa del lenguaje anfitrión.

En este momento, hay varios lenguajes estructurados en existencia que están siendo usados para consultar bases de datos relacionales. Sin embargo, SQL parece ser uno de los más ampliamente adoptados para uso comercial.

El Instituto Nacional de Estándares de Estados Unidos (American National Standards Institute, **ANSI** por sus siglas en inglés) ha publicado un estándar dando a conocer la sintáxis y la semántica para SQL. El estándar ANSI usa la notación sintáctica **BNF** (Backus-Naur Form).

A través de los años, han sido introducidos algunos cambios al SQL como resultado de pruebas al producto, aportes del usuario, y desarrollos adicionales.

Notación

Sintáxis General de un Query

- Al usar SQL se puede consultar una base de datos en un número indefinido de maneras.
- No obstante que SQL es altamente flexible, aún tiene una sintáxis muy específica.
- Es posible elaborar una consulta con comandos SQL que sea equivocada, y terminar ya sea sin resultado alguno, o con respuestas a una pregunta que no se deseaba formular.

- La sintáxis global es muy simple:

SELECT...
FROM...
(WHERE...)
GROUP BY...
(HAVING...)
(ORDER BY...)

La siguiente notación es generalmente aceptada para propósitos de una discusión:

1. Todos los comandos SQL serán escritos con mayúsculas, por ejemplo: **CREATE TABLE.**

2. Aquello que deba ser proporcionado por el usuario, será escrito en minúsculas, como el nombre de la tabla. Así, la siguiente expresión:

   CREATE TABLE nombre_tabla

   indica que el comando SQL **CREATE TABLE** debe ser seguido por un nombre de tabla elegido por el usuario.

3. Un asterisco (*) será usado para indicar "todas las columnas que cumplan la descripción," como en:

```
SELECT    *
FROM      nombre_tabla
WHERE     Ciudad='México';
```

   lo cual es interpretado como: "SELECT (selecciona) todas las columnas (*) FROM (de) la tabla nombrada WHERE (donde) el valor en la columna Ciudad es "México."

   Las comillas sencillas que aparecen al principio y al final de la palabra México en la claúsula WHERE, se deben a que México es el valor de una columna, i.e., un dato. Cualquier cadena de caracteres que representa el valor de una columna debe estar entre comillas sencillas.

4. Las elipsis (puntos suspensivos) serán usados para indicar la continuación de una serie, tal como

columna1, columna2,...,columnaN

lo cual debe ser interpretado como: "columna1, columna2, y cualesquiera otras columnas, hasta el final de la lista de columnas."

5. Para que el lenguaje anfitrión (el lenguaje que reside en el equipo que se trabaja) las interprete como una sola cadena de caracteres, dos o más palabras adyacentes deberán estar conectadas por un subgulón siempre que vayan a ser desplegadas como encabezado de una tabla o columna:

   nombre_empleado

6. Toda proposición de SQL debe ser concluida con un símbolo de terminación. La mayoría de los manejadores de bases de datos utilizan punto y coma (;) para tal fin.

7. La lista deseada de columnas en un query (lista de columnas que aparece después del comando **SELECT**) será llamada la lista objeto (**target**). Esto con el propósito de mantener el uso que hizo Codd de ese término en el cálculo relacional de túplas.

8. Las inclusiones opcionales en una proposición SQL serán indicadas colocándolas dentro de **(corchetes)**. Por ejemplo,

   CREATE VIEW nombre_vista **(nombres_columnas)**

   significa que la inclusión de los nombres de las columnas en el ejemplo mostrado son opcionales.

Definiciones

De acuerdo con el hecho de que SQL está adjunto a un sistema de administración de bases de datos relacionales (RDBMS), los siguientes términos se definen dentro del contexto relacional:

En 1971, Codd introdujo la idea de usar "tablas" como la principal estructura de una base de datos. Describió una tabla como un arreglo rectangular con las siguientes propiedades:

P1: Sus columnas son homogéneas; en otras palabras, en cualquier columna seleccionada todos los elementos son del mismo tipo, mientras que los elementos en diferentes columnas no necesariamente son del mismo tipo.

P2: Cada elemento es un número simple o una cadena de caracteres (así, por ejemplo, si buscamos el elemento en cualesquiera columna y renglón específicados, no necesitamos encontrar un conjunto de números o un grupo).

P3: Todos los renglones de una tabla deben ser distintos (los renglones duplicados no son permitidos).

P4: El ordenamiento de los renglones dentro de una tabla es irrelevante.

P5: Las columnas de una tabla tienen asignados nombres distintos, y el orden de las mismas dentro de una tabla es irrelevante.

Este tipo de tabla es llamada una relación. Si tiene n columnas, entonces es llamada una "relación de orden n".

Los nombres de las columnas son llamados **atributos**. El conjunto de todos los nombres de las columnas para una tabla es llamado un

esquema de relación; y la tabla es una **"relación sobre el esquema de relación."** En general, una base de datos consiste de más de una tabla, cada una teniendo su propio conjunto de atributos o nombres de columnas. El esquema de la base de datos es la colección de todos los esquemas de relación de las tablas en la base de datos. (Dos esquemas de relación distintos pueden tener algunos nombres de columnas atributos en común). Una tabla es una instancia de una relación sobre un esquema de relación; por ejemplo, si cambiamos los datos en la tabla en cualquier forma tenemos una relación diferente con el mismo esquema de relación.

Las siguientes líneas resumen estas definiciones y las relacionan a los términos usados en este texto,

- Una relación denotará una tabla.

- Un atributo será llamado una columna.

- Un sólo registro será llamado un renglón.

- Un valor individual en la intersección de cualquier renglón y columna será llamado un dato.

Propiedades de las Tablas

Un nombre de tabla puede ser precedido por el nombre de la persona que la creó, seguido por un punto. Por ejemplo, una tabla llamada **"Tarifa"** creada por **Marco** puede ser referida como **"Marco.Tarifa"** o simplemente por el nombre "Tarifa". Pero el nombre de una tabla no puede ser una claúsula (o palabra reservada) de SQL.

Existen dos tipos de tablas en una base de datos relacional: **tablas base** y **tablas virtuales**. Una tabla base cumple con la definición de una tabla dada anteriormente. Una tabla virtual, también llamada una

vista, existe sólo como una definición en el catálogo de la base de datos. Cada vez que una vista es accesada, la definición es recuperada del catálogo y el "query" en la definición es ejecutado, creando una tabla con las columnas en la definición de la vista. **Una vista es llamada una tabla virtual** debido a que no existe como tal en la base de datos, en la forma como una tabla lo está. En su lugar, una vista es reconstruida de los datos de su(s) tabla(s) base original(es) siempre que la vista sea requerida

Mientras que en su forma más simple una vista puede ser una porción de una tabla base, una vista también puede ser el resultado de unir una porción de una o más tablas. Una vista puede ser también una porción de una vista, con ésta última siendo una porción de una tabla base.

En general, las tablas base son llamadas simplemente tablas y las tablas virtuales son llamadas vistas.

El Valor Nulo (NULL)

SQL soporta el concepto de valor **NULO (NULL)** para indicar la idea de información incompleta o no disponible. Se puede pensar en este valor simplemente como el de un contenedor (o depósito) en el dominio de un atributo.

Existen diferentes opiniones con respecto a la utilidad o de la lógica detrás del concepto de los valores **NULL**.

Hasta el momento, no obstante, los NULL's son una parte importante de SQL y lo más probable es que sean modificados y embellecidos que eliminados de su sintáxis en el futuro. Tienen como propósito el de llenar un espacio en blanco en el arreglo de datos, pero deben ser usados con precaución. Las siguientes reglas son aplicables, otros usos

especiales y advertencias con respecto a los valores NULL deberán ser consultados en los manuales o documentos donde ocurran:

- Un valor **NULL** no es lo mismo que cero.
- Un valor **NULL** no necesariamente es igual a otro valor **NULL**.
- Un valor **NULL** no puede ser usado en una proposición **SELECT**.
- El tratamiento de los **NULLs** por cada una de las funciones no es uniforme.

SQL en el trabajo

Un gran número de manejadores de bases de datos comerciales incorporan ahora SQL como su sublenguaje de datos.

¿Por qué cambiar a un DBMS?

Después de haber descrito qué es una base de datos, para qué sirve y el lenguaje de consultas SQL. Nos preguntamos ¿por qué cambiar a un DBMS, por que utilizar el modelo de datos relacional, y como lenguaje de consultas al SQL?. La respuesta se explica a continuación.

Un sistema de bases de datos proporciona a las empresas un control de centralizado de los datos de operación, en contraste con la situación en la que cada aplicación tiene sus propios archivos (y algunas veces sus discos particulares). De modo que los datos se hallan muy dispersos y por lo tanto son difíciles de controlar.

La centralización de la información da la ventaja de evitar la redundancia de información, los datos pueden compartirse, puede aplicarse seguridad, y se puede conservar la integridad.

¿Por qué utilizar el modelo relacional?

Cualquier estructura jerárquica se puede convertir en estructura de red, y a su vez cualquier estructura de red puede transformarse a estructura tipo tabla.

¿Por qué SQL?

Los estándares son una necesidad primordial en el mundo de las computadoras, así como en las comunicaciones.
La estandarización del lenguaje entre las bases de datos relacionales es de vital importancia, ya que esto permite la interconexión de bases de datos que residan en mainframes, minis y micros. Obteniendo así una interconectividad entre diferentes ambientes y el proceso distribuido de información.
Para evitar que cada RDBMS hable su propio lenguaje se introduce al SQL como un estándar.
Por primera vez IBM nombro al SQL como un estándar para los RDBMS, desafortunadamente no existe un sólo SQL estándar.
El estándar oficial lo define el American National Standar Insitute (ANSI), pero se considera que tiene limitaciones de uso. Por lo que el estándar más usado es el de IBM.

3.3 OBJETIVOS DE LA TECNOLOGIA DE BASES DE DATOS

En la siguiente lista se presentan los principales objetivos de la tecnología de bases de datos. Los DBMS´s que permiten lograr tales objetivos constituyen una herramienta valiosa y esencial para desarrollar sistemas de información modernos e integrados y proporcionarles el apoyo necesario.

- **Independencia de los datos**
- **Habilidad de compartir datos**
- **Irredundancia de los datos almacenados**
- **Habilidad para relacionar**
- **Integridad**
- **Flexibilidad de acceso**
- **Seguridad**
- **Rendimiento y eficiencia**
- **Control y administración**

Independencia de los datos

El concepto de **independencia de los datos** es básico en el enfoque de bases de datos. Se refiere a la independencia o al aislamiento de los programas de aplicaciones y los usuarios, para protegerlos de cambios que pueda haber en la organización específica de la base a nivel lógico y físico, así como de criterios relativos al almacenamiento de la base en la computadora. La independencia de datos **física** es la propiedad que permite aislar las aplicaciones de los cambios en la organización de los datos empleados en aquéllas; por ejemplo, cambios en la localización de los datos (dispositivo x vs. dispositivo y), encadenamientos internos entre datos, estrategias de ordenamiento internas, rutas de acceso existentes (llave del método básico de acceso, llaves invertidas entre otras.) y la colocación de los datos en los dispositivos de almacenamiento. La independencia de datos **lógica** es

la capacidad de aislar las aplicaciones de los cambios que se hagan en la organización lógica de la base de datos.

Habilidad de compartir datos e irredundancia de los que se almacenen

El objetivo es permitir a las aplicaciones el compartir una base de datos integrada que contenga todos los datos requeridos por las aplicaciones, y eliminar así, en la medida posible, la necesidad de almacenar datos en forma redundante. Las aplicaciones requieren la facilidad de operar sin percatarse de la existencia de las demás. Deben proporcionarse facilidades como: permitir visualizaciones concurrentes de los mismos datos, el control sobre el acceso, el control sobre los interefectos de programas independientes, el acceso eficiente a diferentes subconjuntos de los datos y una gran cantidad de requerimientos relacionados.

La eliminación de la redundancia conduce a la habilidad de compartir. Sin embargo, puede ser necesaria cierta redundancia para mejorar el rendimiento de la base de datos en términos de tiempo de acceso, o para la conveniencia de la visión lógica del usuario.

Habilidad de relacionar (o capacidad de relacionar)

La habilidad de relacionar (o "relacionabilidad") es precisamente la habilidad para definir relaciones entre registros o entidades a nivel lógico, de manera conveniente, tal como se hace para definir a los registros mismos. Las relaciones son tan importantes y tan susceptibles de identificación como cualquier registro o atributo de los datos, y deben poder definirse y manejarse sin ambigüedades por el sistema de base de datos. Considérese la base de datos de la figura 3.2 que muestra cuatro archivos interrelacionados, cada uno de los cuales contiene información sobre un tipo dado de entidad. Por ejemplo, un PEDIDO se asocia o conecta con el correspondiente registro de PROVEEDOR (el cual contiene información sobre el vendedor o

proveedor a quien se envía la orden o pedido) y con los registros correspondientes de INVENTARIO DE MATERIA PRIMA (que contienen información interna de inventario sobre los artículos que se ordenan en el pedido). Las relaciones que ligan o encadenan a estas entidades, se representan con flechas conectoras en el diagrama esquemático que representa a la base de datos.

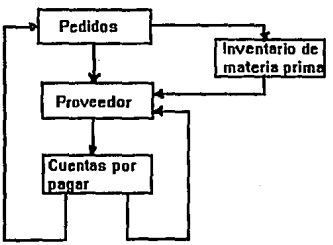

Fig. 3.2 Ejemplo de una base de datos.

Integridad

El término **integridad** se refiere a diversas tareas; entre las principales se tienen: la coordinación del acceso a los datos que realizan las diferentes aplicaciones; la propagación de los valores actualizados a otras copias y valores dependientes; la preservación de un alto grado de consistencia y validez de los datos. Al tenerse muchos usuarios distintos que comparten diversas partes de la base, es imposible que cada uno de ellos sea responsable de la consistencia de los valores de la base y del mantenimiento de las relaciones entre los datos del usuario y el resto de los datos, algunos de los cuales pueden serle desconocidos o estarle vedados para acceso. Un objetivo principal de un sistema de bases de datos es mantener control y preservar la integridad de la base.

Flexibilidad de acceso

La **flexibilidad de acceso** es la capacidad de lograr acceso a cualquier parte de la base de datos, en base a cualesquier llave(s) de acceso y calificación lógica, mediante un lenguaje de alto nivel para consultas, no por procedimientos, que permita curiosear en la base, o mediante instrucciones de entrada/salida desde un programa escrito en algún lenguaje convencional de programación por procedimientos. La flexibilidad de acceso a que se aspira, está más allá del alcance limitado proporcionado por los lenguajes de programación, que hacen uso de los métodos básicos de acceso (manejo de archivos) de los sistemas operativos convencionales.

Seguridad

Deben existir los mecanismos apropiados para asignar, controlar y revocar los derechos de acceso (leer, insertar, borrar, cambiar) de cualesquiera usuarios a cualesquier datos o subconjuntos definidos de la base. Al aumentar la cantidad de datos compartidos y el número de usuarios, aumenta también la tarea del DBMS para garantizar tal seguridad. Una pieza de información o dato elemental debe protegerse completamente de intromisión no autorizada, ya sea accidental o intencionada.

Rendimiento y eficiencia

Debido al gran tamaño de las bases de datos y de las exigencias de los requerimientos de acceso a la base, el buen funcionamiento y la eficiencia son requisitos mayores. Entre más grandes sean la base y el número de usuarios, mayor es la posibilidad de que menores porcentajes de datos sean relevantes para un usuario determinado. Resulta intolerable la búsqueda clásica, ineficiente y exhaustiva, que

hace uso únicamente de las facilidades convencionales de manejo de archivos de los sistemas operativos. La viabilidad de una base de datos integral, es altamente dependiente de un rendimiento adecuado y de su eficiencia.

Administración y control

En el medio de las bases de datos están involucrados muchos usuarios. Es absolutamente necesaria una función que pueda analizar las diferentes necesidades y resolver conflictos de intereses. Se debe establecer una función administrativa permanente para coordinar y llevar a cabo todos los pasos de diseño, implantación y mantenimiento de una base de datos integrada.

Así, la administración de la base de datos es una función compuesta por gente responsable de proteger un valioso recurso: **los datos.** En el medio convencional del procesamiento de datos, un programador de aplicaciones "posee" un archivo de datos. Los usuarios "protegen" sus datos para evitar que otros los usen, forzándolos a recopilar los mismos datos. La era de la base de datos ha eliminado la idea de "propiedad" individual. A la persona encargada de la función de administración de la base de datos se le llama **administrador de la base de datos (ABD).** El administrador de la base de datos no es el "propietario" de los datos sino el "protector" de ellos. En una organización moderna (gubernamental o privada) la generación de información y los procesos de decisión se complican a medida que aumenta la gama de funciones. Ya que al programador de aplicaciones se le **"quita"** el control directo sobre los datos, él o ella pierden el sentimiento de contacto personal y responsabilidad por éstos. Esta falta de contacto fuerza a la organización a desarrollar procedimientos para asegurar que la integridad de los datos no quede comprometida. Este objetivo deberá estar coordinado con la función de administración de la base de datos. La administración de la base de datos es una función que proporciona servicios a los usuarios de la base de datos. Se podría

establecer una analogía entre el administrador de la base de datos (ABD) y el contralor de una organización. El contralor protege el recurso de la organización llamado "dinero", y el ABD protege el recurso llamado "datos". En muchas organizaciones se considera a la función del ABD solamente como la de una persona técnicamente calificada. Este apelativo desvirtúa el objetivo de un ABD. La función del ABD deberá estar situada lo bastante alto en la jerarquía de la organización para tener autoridad, así como responsabilidad, sobre las estructuras de datos y su acceso. La persona a cargo de la función de ABD debe también conocer la forma en que trabaja la organización y cómo usan los datos. Aunque es aconsejable que el ABD sea técnicamente competente, es más importante que conozca la organización y que tenga la capacidad suficiente para interactuar con la gente y para proponer alternativas a los procedimientos normales. De otra manera, la función del ABD es mucho menos efectiva.

Aunque es benéfico que la función del ABD posea autoridad, la ubicación del administrador de la base de datos puede variar de una organización a otra. El primer factor importante para determinar su ubicación lo constituye la importancia que la base de datos tenga para la supervivencia de la organización. El segundo factor es la complejidad de la organización, tanto en el procesamiento de datos como en la naturaleza de sus operaciones. En el medio actual de las bases de datos, la función del ABD frecuentemente permanece en las areas de procesamiento de datos, y casi siempre dentro de un área de aplicaciones de este departamento.

El ABD tiene que coordinar las funciones de recopilación de información acerca de los datos y diseñar, implantar y mantener la base de datos y su seguridad. La misión del ABD no termina con la implantación de la base de datos. La integración de nuevas funciones en la base de datos hace al ABD más necesario que nunca. Una de las principales funciones del ABD consiste en poner atención tanto a las futuras como a las presentes necesidades de información de la

organización. Para llevar a cabo esta función, el diseño de la base de datos debera ser tan flexible, o independiente de los datos, como sea posible.

La función del ABD es claramente significativa para las operaciones de un sistema de base de datos multiusuarios. Sin embargo, no debemos descartar la posibilidad de que algunas de estas funciones algún día se automaticen. Tampoco debemos olvidar la importancia de las bases de datos "personales" sobre las cuales el usuario tiene el control total. Aún con un solo usuario, es posible tener diferentes "enfoques" de los datos o hacer consultas no previstas en el procedimiento. Pero en el medio actual de las bases de datos y en el futuro, el administrador de la base de datos desempeña el papel principal.

3.4 ANALISIS DE LOS DIFERENTES MODELOS DE BASES DE DATOS

a) arquitectura de un sistema de manejo de bases de datos

En la figura 3.3 se muestran los ámbitos de la organización de una base de datos. En el nivel superior se observa el modelo conceptual de la base de datos en la forma como la visualiza un individuo. En el siguiente nivel se tiene el modelo equivalente a la visión de los datos que tiene el usuario, sólo que organizado de acuerdo a las reglas y estructuras lógicas de un modelo particular de sistema de base de datos. En el tercero y último nivel está el ámbito de la estructura de la base de datos física, que representa todos los directorios de definición de datos, las rutas de acceso, las organizaciones secundarias de archivos, los métodos básicos de acceso, la distribución del almacén y los datos mismos almacenados en los dispositivos masivos para tal efecto. Existen tres tipos principales de modelos de bases de datos para definir registros o entidades y las relaciones que guardan, esto es, las estructuras de las bases de datos lógicas.

. 1 El modelo jerárquico.
2 El modelo de red.
3 El modelo relacional

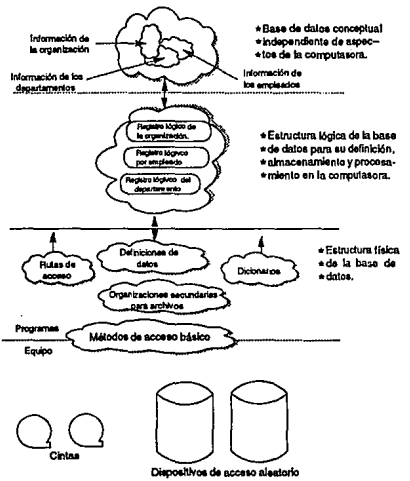

Fig. 3.3 Los ámbitos de la organización de la base de datos

3.4.1 Modelo Jerárquico

El modelo jerárquico no es más que una colección de arboles. Un árbol es una gráfica en la que el número de nodos es igual al número de arcos más uno, es decir el número de arcos que llegan al nodo hijo no puede tener más de un padre, el nodo superior se llama "raíz" y los nodos que no tienen sucesor se les llama hojas.
En las bases de datos jerárquicas se considera a un nodo como un registro y al arco como una liga que une a dos registros.

Las relaciones de mapeo pueden ser solo 1 a 1 o 1 a M es decir, uno a uno y uno a muchos, pero no puede ser N a M (muchos a muchos). En otras palabras un hijo sólo puede tener un padre, un padre puede tener varios hijos (1 a M), así mismo, un padre puede tener un solo hijo (1 a 1).
En este modelo el contenido de un registro puede repetirse en varios lugares, por ejemplo, en el sistema de proveedores y partes, una parte puede ser surtida por varios proveedores, la información correspondientes al proveedor se tendrá que repetir para cada pieza que surta.
Esto trae una gran desventaja, al actualizar la información puede suceder alguna inconsistencia de la información y por supuesto un desperdicio de espacio en el almacenador.

- JERARQUICO 1960's
un programador sólo puede especificar una relación un-padre-múltiples-hijos.

3.4.2 Modelo de red

Los diagramas que representan los modelos de red están constituídos por cuadros que representan a los registros y líneas que corresponden a las ligas, por lo que se puede decir que el modelo jerárquico y el de red se representan igual. La diferencia esta en que la estructura de red es más general que la jerárquica porque una ocurrencia de registro específico puede tener cualquier número de antecesores y también cualquier antecesor puede tener varios sucesores de esta manera se tendra un mapeo N a M, M a 1, 1 a M y M a N.

- RED 1970's
un programador puede especificar relaciones de múltiples-padres-múltiples-hijos.

3.4.3 EL MODELO RELACIONAL

El desarrollo de las bases de datos relacionales comenzó en la década de los 70's, cuando el matemático E.F. Codd publicó en junio de 1970

un documento llamado "A Relational Model of Data for Large Shared Data Bank". Codd estableció 12 conceptos (reglas) en los que se basa el modelo relacional (los cuales se mencionan posteriormente).

- **RELACIONAL 1980'S**
  el sistema automáticamente establece todas las posibles relaciones.
  se origina con el modelo ALPHA de IBM (EF Codd, 1970).

  ORACLE V2, primer RDBMS comercial (1979).

La investigación realizada por Codd influyó para el desarrollo de un lenguaje prototipo que concretaba las características del modelo relacional.

Las doce reglas definidas por Codd, se utilizan como parámetros para verificar que tanto se apega un DBMS a los conceptos originales y con esto establecer su grado como manejador relacional.

Un DBMS relacional se caracteriza por:

1. Proveer una interface con un lenguaje, garantizando **DINAMICAMENTE** la definición, la interrogación, la actualización y el control de un conjunto de relaciones.

2. Asegurar la independencia de los datos (relaciones) con respecto a:

   - las estructuras físicas de almacenamiento.
   - los caminos de acceso a la información.

3. Asegurar la integridad de los datos de acuerdo a la **SEMANTICA** de las aplicaciones y también:

- controlar los accesos.
- administrar la concurrencia.
- tomar en consideración los efectos de caídas para recuperar la base a un estado coherente.

Las siguientes definiciones se consideran de importancia para la comprensión del contexto:

Una **relación** es un subconjunto del producto cartesiano de una lista de **dominios.**

Un **dominio** es simplemente un conjunto de valores. Por ejemplo, el conjunto de números enteros es un dominio, el conjunto de caracteres (A,...,Z) es un dominio

Un **modelo relacional** consta de tablas bidimensionales, las cuales contienen columnas y renglones.

El concepto de **tabla** tiene una similitud con el concepto de relación, ya que dados los dominios en la tabla (atributos), su producto cartesiano forma una relación por lo que los renglones son llamados túpias.

b) características de un RDBMS (Relational DataBase Management System).

- Representación de los datos en forma de tabla.

- Lenguaje de cuarta generación (4GL).
  Sintáxis no procedural

- Capacidades totalmente relacionales.
  Navegación automática

Todos los operadores relacionales

- Flexibilidad
  Datos fácilmente modificables
  Estructura de la base de datos fácilmente cambiable

- Diccionario de datos integrado.

Estos son los enfoques alternativos para visualizar y manipular datos a un nivel lógico, independientemente de cualesquiera estructuras físicas de soporte en que se basen. En los DBMS's comerciales actuales existen aún los modelos jerárquico y de red. El modelo relacional es el enfoque más atractivo y promisorio que se sigue investigando hoy en día y se considera para DBMS's. Cualquier estructura de una base de datos a nivel lógico puede definirse en cualquiera de los cuatro modelos.

En la figura 3.4 se presenta la arquitectura de un sistema de manejo de bases de datos.

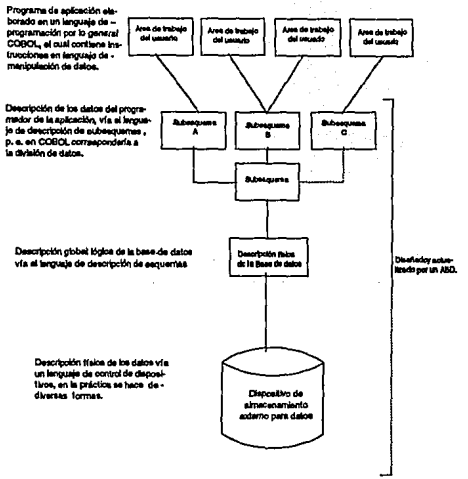

Fig. 3.4 Arquitectura de un sistema de manejo de bases de datos

c) ¿por qué el modelo relacional para los sistemas manejadores de bases de datos?

Hasta hace algunos años el hecho de pensar en un manejador relacional en un ambiente de microcomputadoras, era considerado tan sólo eso, un sueño; actualmente debido al desarrollo de la tecnología aplicada en la fabricación de los microcomponentes utilizados en los equipos microcomputadores hace posible la incorporación de manejadores de bases de datos relacionales en estos equipos. Pero no es sólo esta característica la que hace atractivo a un manejador relacional RDBMS (Relational Data Base Management System) en estos ambientes.

Como se mencionó anteriormente, un sistema de bases de datos proporciona a las organizaciones un control centralizado de los datos de operación, en contraste con la situación en la que cada aplicación tiene sus propios archivos. De modo que los datos se hallan muy dispersos y por lo tanto son difíciles de controlar.
La centralización de la información da la ventaja de evitar la redundancia de información, los datos pueden compartirse, puede aplicarse seguridad y se puede conservar la integridad.

¿por qué utilizar el modelo relacional?

Cualquier estructura jerárquica se puede convertir en estructura de red, y a su vez cualquier estructura de red se puede transformar a una estructura tipo tabla. Como se muestra en la figura 3.5 donde se transforma la estructura jerarquica a red y de red a tabla.

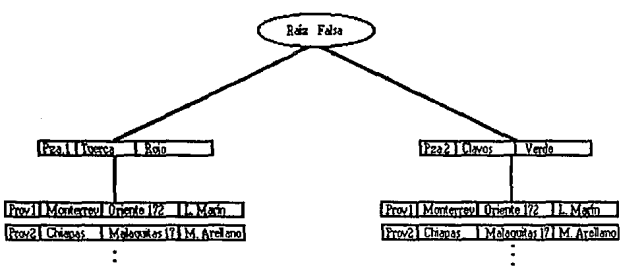

Fig. 3.5 (a) Arbol (Jerárquica)

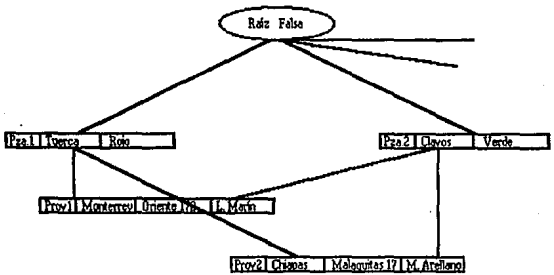

Figura 3.5 (b) Red

Tabla proveedor

Nom-Prov	Ciudad	Dirección	Nombre-Repres
Pza.1	Monterrey	Oriente 172	L. Marín
Pza.2	Chiapas	Malaquitas 17	M. Arellano
.	.	.	.
.	.	.	.
.	.	.	.

Tabla Partes

Num-Parte	Nombre	Color
Pza.1	Tuerca	Rojo
Pza.2	Clavo	Verde
Pza.3	Tornillo	Azul
.	.	.
.	.	.
.	.	.

Tabla Embarques

Num-Prov	Num-Parte
Prov.1	Pa.1
Prov.1	Pa.2
.	.
.	.
.	.
Prov.2	Pa.1
.	.
.	.
.	.

Figura 3.5 (c) Tabla

d) Las reglas de E. F. Codd para calificar un DBMS relacional.

1. **Regla de Información.**- Toda la información en una base de datos (BD) relacional es representada explícitamente al nivel lógico y solamente de una manera (por valores en tablas relacionales).

2. **Regla de Acceso.**- Todos y cada uno de los datos (valores atómicos) en una BD relacional tienen la garantía de ser lógicamente accesados por una combinación de nombre de la relación, valor de la llave primaria y nombre de atributo (columna).

3. **Tratamiento del Caracter Nulo.**- Un DBMS totalmente relacional, soporta indicadores (distintos del caracter vacío, cuerda de caracteres de blancos y diferente de cero o cualquier otro número) para representar a nivel lógico la **INFORMACION FALTANTE** de manera sistemática independientemente del tipo de dato. Además de la representación lógica, el DBMS debe soportar funciones de manipulación de estos indicadores y deben ser independientes del tipo de dato de la información faltante.

4. **Catálogo en Línea.**- La descripción de la DB es representada a nivel lógico como datos ordinarios de manera que usuarios autorizados puedan aplicar el mismo lenguaje relacional para interrogarla como si se aplicara a datos normales.

5. **Regla de Sublenguajes.**- Un DBMS relacional (no importa cuantos lenguajes y modos soporte) debe ofrecer al menos un lenguaje:

   a) cuyos estatutos son expresables en alguna sintáxis bien definida.
   b) que comprenda los siguientes aspectos:
      - definición de datos
      - definición de vistas
      - manipulación de datos

- restricciones de integridad
- autorización
- límites de la transacción (**begin, commit** y **rollback**)

6. **Regla de Actualización de Vistas.**- El DBMS incluye un algoritmo para determinar (en el momento de la definición de las vistas) cuando una vista permite la inserción o supresión de túplas y cuando cada una de sus columnas es actualizable. Registra además el resultado de esta investigación en el **CATALOGO.**

7. **Alto Nivel de Inserción, Modificación y Borrado.**- La capacidad de manejar relaciones de base o relaciones derivadas a través de un operando simple que se aplique no solo a la recuperación de datos, sino también a la inserción, actualización y supresión de los datos.

8. **Independencia Física de los Datos.**- Los programas de aplicación permanecen lógicamente inalterados aún si se efectúan cambios en las estructuras de almacenamiento o los métodos de acceso.

9. **Independencia Lógica de los Datos.**- Los programas de aplicación permanecen lógicamente inalterados cuando se realizan cambios que modifican la estructura de las relaciones base.

10. **Regla de Integridad.**- Las restricciones de integridad específicas a una BD relacional específica, deben ser definibles en el sublenguaje de datos relacional y almacenables en el **CATALOGO** (**NO** en los programas de aplicación).

11. **Independencia de Distribución.**- Un DBMS relacional posee independencia en la distribución.

12. Si un sistema relacional posee un lenguaje de bajo nivel (acceso a un registro a la vez), ese bajo nivel no puede ser usado para violar

las reglas de integridad y restricciones expresadas en el lenguaje relacional de alto nivel (múltiples registros a la vez).

VI. TENDENCIAS DEL MERCADO (LAS NUEVAS PERSPECTIVAS)

a) bases de datos distribuidas (Distributed Database Management Systems) DDBMSs

Después de muchos años de investigación y desarrollo, las bases de datos distribuidas están al fin siendo una realidad. Sin embargo, será hasta dentro de varios años, en que todos los problemas técnicos y de manejo que rodean esta compleja tecnología hayan sido resueltos, se puede, no obstante, sacar ventaja de las facilidades de las bases de datos distribuidas que están disponibles en muchos de los productos en la actualidad. Es muy importante comprender lo que se puede y lo que no se puede hacer con estos productos. Sólo con esta comprensión se puede establecer el nivel correcto de manejo y expectativas del usuario final. En este inciso del capítulo se presentan los diferentes tipos de aplicación distribuida, y se definen los tipos de capacidad de la base de datos distribuida requerida por cada una de ellas. El principal objetivo es documentar un conjunto de criterios que se puedan utilizar para determinar las capacidades de los productos de diferentes proveedores de bases de datos.

¿Qué se entiende por una base de datos distribuida, o DDB (Distributed Data Base)? La definición más simple establece que una base de datos distribuidas es una colección de datos distribuidos sobre diferentes computadoras en una red, y que no obstante estar físicamente dispersos, los datos aparecen a los programas y a los usuarios como una sóla base de datos. Este es llamado algunas veces el **modelo de imágen simple** o **de una sóla imágen.**

Hace varios años, Chris Date se puso a la tarea de proponer y explicar las doce reglas que un sistema de bases de datos distribuidas debían cumplir. Esta reglas, resumidas en la siguiente tabla, se han convertido en la definición de trabajo común de las bases de datos distribuidas:

1. **Autonomía local.** Los datos locales son administrados independientemente de otros sitios.
   Control local de la información, seguridad y almacenamiento, salvo cuando la integridad lo requiera. Ningún local X depende de algún local Y para su funcionamiento.

2. **Independencia de un local central.** Ningún sitio con un DBMS es más necesario que otro.

   Evita: diccionario central, control central de concurrencia, control central de recuperación, proceso central de consultas, cuellos de botella.

   Provee: menor vulnerabilidad, mayor modularidad.

3. **Operación continua.** Ninguna actividad planeada debe requerir un cese de las actividades.

   La operación de un DDBMS (Distributed Database Management System) no debe detenerse o verse afectada por circunstancias tales como: actualizaciones de versión, salida de un equipo de la red, creación/supresión de relaciones.

4. **Independencia de ubicación.** Ni los usuarios ni los programas necesitan saber con respecto a la localización de los datos.

   El usuario no debe saber donde se encuentra ubicada físicamente la información. Esto implica manejar un esquema distribuido de nombres.

5. **Independencia de fragmentación.** Una tabla que ha sido fragmentada aparecerá como una sóla tabla a usuarios y programas.

Una relación puede particionarse por consideraciones técnicas de almacenamiento. El usuario no necesita saber acerca de dicha fragmentación.

6. **Independencia de replicación.** Los datos redundantes serán transparentemente actualizados.

   Una relación puede representarse por medio de copias almacenadas en distintos lugares. El usuario no necesita saber acerca de las réplicas.

7. **Proceso distribuido de consultas.** Las consultas (queries) deben ser optimizadas para la base de datos distribuida.

   Cuando sea apropiado, el proceso para satisfacer una consulta debe ser distribuido; al igual que el proceso de optimización de la consulta.

8. **Manejo distribuido de transacciones.** Las transacciones que actualizan en múltiples sitios, deben ser ejecutadas como si fueran transacciones en serie (control de concurrencia) y deben dejar la base de datos distribuida en un estado lógicamente consistente por si se presentase una falla (control de recuperación).

   Una transacción puede dividirse en varios procesos, cada uno de los cuales puede llevarse a cabo en una localidad diferente. Dichos procesos deberán coordinarse mutuamente para preservar la integridad.

9. **Independencia de hardware.** Las bases de datos distribuidas deben ser capaces de correr en diferentes tipos de hardware, con todas las máquinas con la capacidad para participar de igual manera.

Un DDBMS debe poder distribuir la base de datos en diferentes equipos, sin que esto afecte su participación en el sistema global.

10. **Independencia del sistema operativo.** Las bases de datos distribuidas deben ser capaces de correr en diferentes tipos de sistemas operativos.
    Es deseable que un DDBMS sea soportado por lo menos por los siguientes sistemas operativos: **HP.**- MPE, UNIX; **IBM.**- MVS/XA, MVS/ESA, VM/CMS; **DEC.**- VAX/VMS, ULTRIX; **PC.**- DOS, OS/2, XENIX.

11. **Independencia de red.** Las bases de datos distribuidas deben ser capaces de trabajar con diferentes tipos de redes.

    Un DDBMS debe poder platicar utilizando diferentes tipos y arquitecturas de red, como pueden ser: ETHERNET, TOKEN-RING, ARCNET, SNA, DECNET.

12. **Independencia de bases de datos.** Las bases de datos distribuidas deben ser capaces de trabajar con diferentes tipos de manejadores de bases de datos, con tal de que tengan las mismas interfaces.

    Un DDBMS en una máquina, debe poder accesar información en otra máquina bajo otro DDBMS.

¿Por qué un DDBMS?

Para comprender mejor lo que está involucrado, se necesita examinar por qué la gente quiere usar un DDBMS en lugar de la alternativa, el acceso remoto a una base de datos centralizada.

La principal ventaja es la capacidad para accesar datos localizados en múltiples sitios de una manera transparente mientras se mantienen la mayoría de los datos locales a los sitios que los utilizan.

Por ejemplo, una consulta a una base de datos distribuida para un sistema de librería, informaría que no obstante el libro no se encuentra en una sucursal, podría encontrarse en otras sucursales. Existe, por supuesto, algo de **"overhead"**(tiempo empleado por un sistema de computación que no contribuye directamente al progreso de las tareas de usuario del sistema) en la comunicación cuando el sistema lleva a cabo la consulta a la base de datos en otras sucursales. Sin embargo, cuando se verifica la existencia de un libro que se encuentra en la misma sucursal, sólo la base de datos local requiere ser consultada; no hay actividad en la red, o alguna otra interacción con sitios remotos.

Este ejemplo ilustra el importante principio de la autonomía local. La regla 1 de Date. Cada sitio en la red puede correr aplicaciones locales independientemente de otros sitios, o globalmente sobre datos en sitios remotos.

El acceso a datos locales es importante porque proporciona a los usuarios un acceso más rápido a datos usados más frecuentemente. Debido a que el procesamiento local puede efectuarse en paralelo entre los diferentes sitios en una red, el rendimiento específico global **(overall throughput)** es probable que sea mejor que depender de un sitio central. Además, los sistemas locales usualmente serán más sensibles a las necesidades locales que el proceso de datos en forma remota. Finalmente, por supuesto, un sistema de bases de datos distribuidas, hace disponibles los datos de otros sitios como si estos fueran locales (con la probable excepción del tiempo de respuesta). Existen otros beneficios potenciales de las bases de datos distribuidas. Cada nodo de la base de datos puede ser apropiadamente hecho a la medida de la cantidad de datos, la complejidad de los requerimientos del usuario, y el número de usuarios. Cuando el crecimiento de la base de datos distribuida demanda una actualización de hardware, éste tenderá a ser en incrementos más

pequeños que con una base de datos centralizada, ya que las demandas adicionales se pueden satisfacer por cambios más pequeños en los nodos existentes o agregando nuevos nodos a la red.

Terminología

La gente usa diferentes terminos para describir las capacidades de un sistema de bases de datos distribuidas. Por lo tanto, se sumarizan las características de un sistema tal usando la terminología a través de éste capítulo.

Tipos de Acceso de las Bases de Datos Distribuidas

El objetivo de un sistema de bases de datos distribuidas es dar a los usuarios de aplicaciones acceso tanto a datos locales como remotos. Existen cuatro tipos básicos de acceso a bases de datos distribuidas (ver figura 4.1):

- **Petición remota**: permite a una **única** petición ser procesada en una **única** localidad remota.

- **Transacción remota**: permite a una transacción consistente de **múltiples** peticiones ser procesada en una **única** localidad remota.

- **Transacción distribuida**: permite a una transacción consistente de **múltiples** peticiones ser procesada en **múltiples** localidades (locales o remotas). Cada petición puede ser procesada solamente en una **única** localidad, pero diferentes peticiones dentro de la misma transacción pueden ser procesadas en diferentes localidades.

- **Petición distribuida**: permite a una transacción consistente de **múltiples** peticiones ser procesada en **múltiples** localidades (locales o remotas). Cada petición puede ser procesada en **múltiples** localidades. El procesamiento distribuido de peticiones permite, por ejemplo, tablas de múltiples localidades ser accesadas haciendo uso de una operación relacional **"join"** o **unión.**

De los cuatro tipos, sólo el procesamiento distribuido de una petición puede ser considerado como capaz de soportar totalmente el concepto de una base de datos distribuida. Es este el único tipo de procesamiento el que permite a los usuarios distribuir datos a través de múltiples localidades sin que la aplicación tenga que saber donde están los datos localizados físicamente. Los otros tres tipos de procesamiento imponen restricciones sobre lo que puede ser hecho por la aplicación, y a veces requieren que la aplicación sepa acerca de la localización física de los datos. No obstante, estos tres tipos, permiten el acceso a datos remotos, y permiten a los usuarios ejecutar procesamiento de aplicaciones (procesamiento del **cliente**) en una localidad diferente del procesamiento de la base de datos (procesamiento del **servidor**), i.e., todos soportan una forma de procesamiento cliente/servidor. Se puede ver entonces que se tienen diferentes opciones cuando se desarrollan aplicaciones distribuidas.

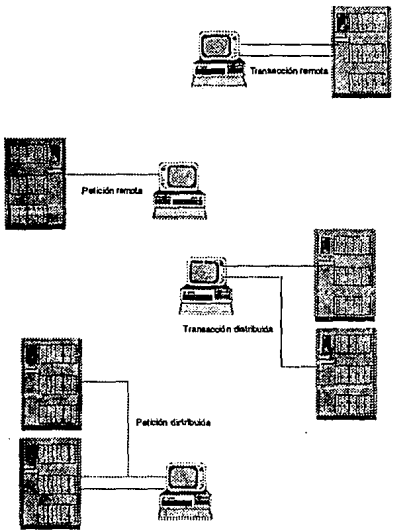

Fig. 4.1 Tipos de acceso a datos distribuidos

Tipos de Distribución de Datos

Los datos en un medio ambiente distribuido se pueden distribuir en una de varias formas:

- **Extracción manual:** donde un usuario proporciona una petición que causa que los datos sean copiados de una localidad y cargados en una o mas tablas en otra localidad. Las peticiones remotas o el procesamiento remoto de transacciones pudiera ser usado para ejecutar una extracción manual.

- **Instantánea (snapshot)**: donde el DBMS periódicamente extrae datos de una localidad y los carga en una o mas tablas en otra localidad. El usuario define la frecuencia (diario a la medianoche, por ejemplo) con la cual el proceso de instantánea se debe llevar a cabo. Una instantánea es usualmente creada para un proceso de lectura solamente.

- **Réplica:** donde el DBMS mantiene múltiples copias de la misma tabla en múltiples localidades. La localización de los datos replicados debe ser transparente a las aplicaciones que los accesan. El mantenimiento de las copias puede ser ejecutado asíncronamente y síncronamente con el procesamiento de las aplicaciones que modifican los datos replicados.

- **Fragmentación:** donde una tabla es dividida en múltiples pedazos, y cada pedazo físicamente almacenado en una localidad diferente.

Las aplicaciones que accesan los datos fragmentados los perciben como una sola tabla. La fragmentación puede ser hecha verticalmente dividiendo las columnas de una tabla, u horizontalmente dividiendo los renglones de una tabla.

Arquitectura Distribuida

Un sistema de bases de datos distribuidas debe tener las siguientes características relativas a la arquitectura:

- **Transparencia de localización:** los usuarios no tienen que conocer acerca de la localización física de una tabla en el momento de llevar a cabo el procesamiento de una aplicación. Esta característica es particularmente importante cuando se hace

el procesamiento de peticiones distribuidas, y cuando se accesan tablas fragmentadas o replicadas.

- **Optimización global:** aquí, el optimizador del DBMS relacional toma en cuenta el costo de accesar datos remotos a la hora de determinar las trayectorias de acceso a los datos. Esta característica es importante para un buen rendimiento **"performance"** cuando se hace el procesamiento de peticiones distribuidas.

- **"Commit" distribuido:** aquí, el sistema usa un **protocolo de "commit" de dos fases** cuando actualiza datos en múltiples localidades. Esta característica es requerida para transacciones distribuidas y peticiones distribuidas en operaciones que efectúan modificación de datos.

- **Control de concurrencia distribuida:** aquí, el sistema utiliza un mecanismo global de concurrencia para controlar el acceso multiusuario a los datos en múltiples localidades. Este mecanismo debe por ejemplo, manejar a través de las diferentes localidades (y en forma global) los **"deadlocks"** (bloqueo). Esta facilidad es requerida cuando se lleva a cabo una transacción distribuida o una petición distribuida en el procesamiento de modificación de datos.

- **Control de integridad distribuida:** aquí, el sistema asegura que las reglas de integridad de las bases de datos distribuidas (restricciones referenciales, por ejemplo) son ejecutadas.

- **Administración distribuida:** aquí, las facilidades son proporcionadas para definir, crear y mantener las tablas en un medio ambiente distribuido. El DBMS debe contar también con las herramientas para monitorear y afinar el sistema de bases de datos distribuidas.

TIPOS DE PROCESAMIENTO DE APLICACIONES DISTRIBUIDAS

Habiendo definido los diferentes tipos de acceso a datos distribuidos y distribución de datos, se puede establecer como pueden estos ser usados por diferentes tipos de aplicación distribuida, y revisar las características de arquitectura de las bases de datos distribuidas requeridas por cada una.

Existen diferentes tipos de aplicación distribuida. Para mostrar algunos ejemplos típicos, se utilizarán los cuatro escenarios mostrados en la figura 4.2. Para propósitos de simplicidad y fácil identificación serán llamadas:

- Tipo 1 cliente/servidor
- Tipo 2 cliente/servidor
- Tipo 3 cliente/servidor
- Base de datos distribuida

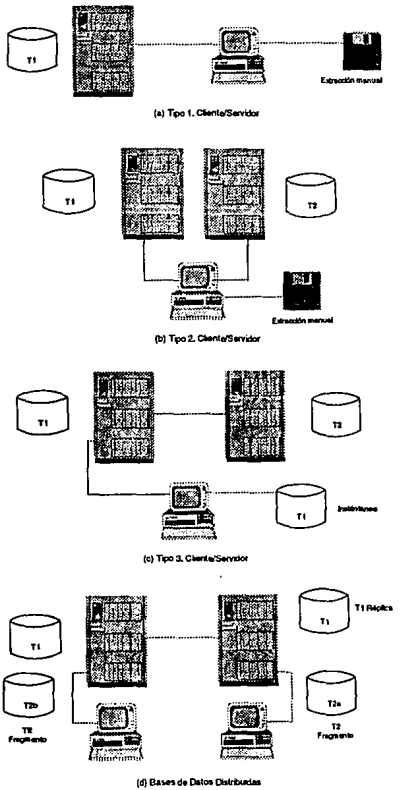

Fig. 4.2 Tipos de procesamiento de aplicaciones distribuidas

- Una **aplicación tipo 1 cliente/servidor** puede ser empleada por un usuario final para accesar datos operacionales almacenados en una sola localidad remota. El acceso dinámico de los datos puede ser hecho usando un procesamiento de peticiones remotas o un procesamiento de transacciones remotas. Este procesamiento será normalmente de sólo lectura, pero pudiera, bajo circunstancias excepcionales, ser usado para modificar datos remotos. La cantidad de acceso dinámico remoto se mantendría normalmente a un nivel bajo, debido al potencial impacto sobre el rendimiento en el sistema remoto, y también debido a que los datos operacionales no están frecuentemente en un formato útil (el usuario final pudiera, por ejemplo, querer ver información resumida o datos consolidados, en lugar de los datos a detalle). También, los datos pudieran ser inconsistentes mientras están siendo accesados por el usuario final, si las aplicaciones operacionales están manteniendo los datos al mismo tiempo. Usualmente, la mayor parte del procesamiento en este tipo de medio ambiente es hecho al accesar subconjuntos de los datos operacionales almacenados en el sistema del usuario **(cliente)**. Estos subconjuntos pueden ser creados por extracciones regulares en forma manual.

- Una **aplicación tipo 2 cliente/servidor** extiende el soporte previo agregando la capacidad para accesar múltiples localidades remotas usando procesamiento de transacciones distribuidas. Los datos accesados en cualquier localidad son usualmente independientes de los datos almacenados en otra. Debido a que las localidades remotas no están conectadas por un enlace de red, ninguna localidad puede actuar como una coordinadora en un protocolo de **"commit"** de dos fases. Por esta razón, este estilo de procesamiento sólo permite a una transacción **"cliente"** actualizar datos en una localidad remota. (Nota: si el producto proporciona soporte para que el **cliente** actúe como un coordinador, la actualización en múltiples localidades sería

entonces posible). Aún y cuando una transacción pudiera actualizar datos en una sola localidad, es posible que ocurran **"deadlocks"**, y por esta razón, son requeridos los controles de concurrencia distribuida para aplicaciones del tipo 2 cliente/servidor. Como antes, la distribución de los datos es manejada por extracciones de datos desde localidades remotas.

- Una **aplicación tipo 3 cliente/servidor** agrega soporte para un protocolo de **"commit"** de dos fases entre localidades y, por lo tanto, permite a las transacciones **cliente** actualizar datos almacenados en múltiples localidades remotas. Estas extensiones permiten que los datos almacenados en una localidad remota sean relacionados a los datos almacenados en otra localidad, proporcionando así los primeros elementos de una capacidad de bases de datos distribuidas. Por razones de rendimiento, la cantidad de acceso dinámico de datos a datos remotos se debe mantener bajo. La distribución de datos es llevada a cabo a través de instantáneas **(snapshots)**, más bien que por extracciones manuales.

Como en los dos primeros ejemplos de las aplicaciones mencionadas anteriormente, una aplicación del **tipo 3 cliente/servidor** puede ser usada por usuarios finales para accesar datos operacionales remotos. Este tipo de arquitectura es también idealmente apropiada para aplicaciones ingenieriles y científicas que necesitan accesar y mantener datos distribuidos. Los requerimientos funcionales y de rendimiento de este tipo de aplicación no son normalmente tan estrictos como aquellos para las aplicaciones operacionales.

El último ejemplo en la figura 4.2 muestra una verdadera aplicación de bases de datos distribuidas haciendo uso de un procesamiento de peticiones distribuido. En este ejemplo, el DBMS proporciona tanto la **fragmentación** como la **réplica**, permitiendo accesos más rápidos para operaciones de lectura; no obstante, existen aún implicaciones

de rendimiento para operaciones significativas de modificación de datos. Los requerimientos de la arquitectura de bases de datos distribuidas para este tipo de aplicación, se extienden para incluir transparencia de localización, optimización global, control de integridad distribuida y administración distribuida

La Tabla 2 resume los cuatro ejemplos de aplicaciones, y las características de la arquitectura para las bases de datos distribuidas requeridas por cada una.

	Tipo 1 Cliente/Servidor	Tipo 2 Cliente/Servidor	Tipo 3 Cliente/Servidor	Bases de datos distribuidas
Petición Remota	SI	SI	SI	SI
Transacción Remota	SI	SI	SI	SI
Transacción Distribuida	NO	SI	SI	SI
Petición Distribuida	NO	NO	NO	SI
Extracción manual	SI	SI	SI	SI
Instantánea	NO	NO	SI	SI
Réplica	NO	NO	NO	SI
Fragmentación	NO	NO	NO	SI
Transparencia de Localización	NO	NO	NO	SI
Optimización Global	NO	NO	NO	SI
"Commit" Distribuido	NO	NO	SI	SI
Concurrencia Distribuida	NO	SI	SI	SI
Integridad Distribuida	NO	NO	NO	SI
Administración Distribuida	NO	NO	NO	SI

Tabla 2. Requerimientos para procesamiento de aplicaciones distribuidas

Compuertas (gateways) para DBMSs Foráneos

En muchas instalaciones estarían complacidos al poder utilizar los tipos de aplicación descritos líneas arriba, en un medio ambiente de bases de datos heterogéneas. Como se muestra en la Tabla 2, esto se puede llevar a cabo usando una **compuerta** (gateway) al DBMS **foráneo.**

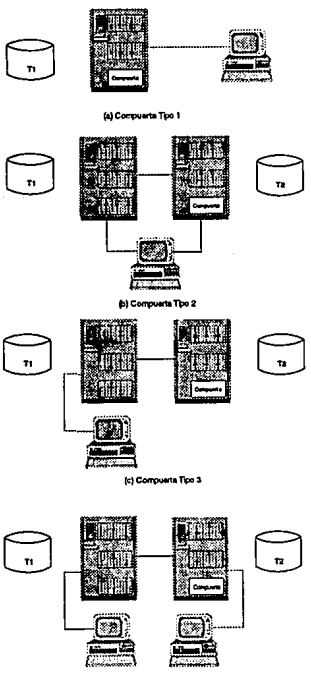

Fig. 4.3 Tipos de compuertas de DBMS foráneas

Las cuatro compuertas mostradas proporcionan las siguientes capacidades:

- **Compuerta tipo 1:** permite el acceso a un sólo DBMS foráneo en una localidad remota usando procesamiento de peticiones remotas o procesamiento de transacciones remotas.

- **Compuerta tipo 2:** permite el acceso a múltiples localidades remotas, una o más de las cuales puede estar corriendo un DBMS foráneo, haciendo uso de procesamiento de transacciones distribuidas. La modificación de datos está restringida a una sola localidad remota.

- **Compuerta tipo 3:** permite el acceso a múltiples localidades remotas, una o más de las cuales puede estar corriendo un DBMS foráneo, usando procesamiento distribuido de transacciones. La modificación de los datos es permitida en múltiples localidades remotas.

- **Compuerta tipo 4:** permite el acceso a múltiples localidades remotas, una o más de las cuales puede estar corriendo un DBMS foráneo, haciendo uso de procesamiento distribuido de peticiones. Este tipo de compuerta permitiría a una sola base de datos hacer una petición para efectuar una operación unión **(join)** de una tabla de un DBMS **nativo** con una tabla de un DBMS **foráneo.** Una compuerta tipo 4 puede ser una compuerta **unidireccional** o **bidireccional.** Una compuerta unidireccional, permite a una aplicación accesar un DBMS foráneo, pero no permite a aplicaciones foráneas utilizar la compuerta en sentido contrario. Una compuerta bidireccional soporta aplicaciones en ambos lados de la compuerta como se muestra en la tabla 2.

Cada uno de los tipos de compuerta soporta el correspondiente tipo de aplicación mostrado en la figura 4.2; una compuerta del tipo 2, por ejemplo, soporta el procesamiento de aplicaciones del **tipo 2 cliente/servidor.** Los requerimientos de la arquitectura de las bases de datos distribuidas mostrados en la Tabla 2 también se aplican cuando se usa una compuerta.

Una compuerta debe también enmascarar las diferencias sintácticas y semánticas de un sublenguaje de bases de datos, SQL, por ejemplo, entre los diferentes DBMSs. Una compuerta que no enmascara estas diferencias puede ser pensada como una compuerta **pass-thru;** esto es, simplemente pasa las peticiones de los datos al sistema DBMS foráneo sin hacer ningún cambio. Una compuerta debe manejar:

- La sintáxis del lenguaje de manipulación de datos y las diferencias funcionales.
- Las diferencias del lenguaje de definición de datos.
- Las diferencias entre los tipos de datos.
- Las diferencias entre los códigos de retorno.
- Las diferencias entre diccionarios de datos.
- Las diferencias del manejo de transacciones; **commit, rollback, locking, y otras.**
- Los accesos interactivos y programados.

Hay varios esfuerzos por parte de la industria de bases de datos para estandarizar las comunicaciones entre sistemas cliente y DBMSs foráneos. Estos incluyen:

- El estándar de acceso a datos remotos ANSI/ISO (RDA, Remote Data Access Standard).

- Arquitectura de Sistemas de Aplicación de IBM (SAA), SQL e Interface Común de Programación-Comunicaciones (CPI-C).

- Lenguaje de Conectividad Network Innovations CL/1.

- Relational Technology, Inc. Open Data Access; que consiste de una arquitectura de comunicaciones global (GCA) y de un SQL abierto.

- Sybase Open Client y Open Server Architecture.

En este inciso se han descrito brevemente las características de un sistema de bases de datos distribuidas y se han discutido diferentes tipos de aplicaciones distribuidas. Se ha mostrado también que no necesariamente se requiere de un soporte de bases de datos distribuidas completas para construir aplicaciones distribuidas. Las arquitecturas cliente/servidor proporcionan algunas alternativas viables. El procesamiento cliente/servidor puede imponer ciertas restricciones en lo que una aplicación puede hacer, y puede afectar la capacidad de la aplicación para soportar el cambio, sin embargo, no requiere algunas de las tecnologías más avanzadas que demandan las bases de datos distribuidas totales.

b) bases de datos orientadas a objetos

¿Qué es un objeto? Un objeto es como un bloque de software para construcción. Es una estructura de datos que está encapsulada con código que entiende la estructura de datos y proporciona los servicios deseados. Un objeto usualmente representa una abstracción de una

cosa útil del mundo real. Algunos objetos modelan cosas en el dominio de una aplicación, tales como ciudades y caminos en un sistema de información geográfica. Algunos objetos proporcionan interfaces hacia los recursos de cómputo, tales como modems y periféricos en una red de computadoras. Otros objetos modelan abstracciones de programación tales como arreglos, pilas (stacks) y árboles (trees).

Un objeto es encapsulado de forma tal que sus usuarios no necesitan saber con respecto a su estructura interna o utilización. Cada objeto proporciona una interface bien especificada a sus usuarios. Un cliente usuario pudiera accesar un objeto con tan sólo proporcionar las peticiones para que el objeto ejecute sus servicios. Por ejemplo, un objeto llamado **Inventario** pudiera tener los siguientes servicios: **Verifica_Cantidad, Despacha_Parte** y **Elimina_Parte.** Un objeto llamado **Empleado** pudiera contar con servicios tales como **Nuevo, Cambio_Salario, Cambio_Departamento,** y **Retiro.** Un objeto llamado **Rectángulo** pudiera contar con servicios tales como **Rota, Agranda, Llena,** y **Trasiada.**

Cómo actúan e interactúan los objetos. Para hacer que un objeto ejecute uno de sus servicios, un cliente hace una petición. Una petición es una declaración que especifica un servicio a ser llevado a cabo por los objetos. Cada petición tiene un nombre que identifica a los objetos que van a proporcionar el servicio, y pueden tener argumentos y producir resultados. El cliente no necesita saber ninguno de los detalles de la operación del servicio; las operaciones están completamente encapsuladas dentro del objeto.

Una petición genérica pudiera ser solicitada a diferentes objetos que proporcionan servicios similares pero diferentes utilizaciones y posiblemente diferentes comportamientos. Por ejemplo, una petición genérica de impresión pudiera ser hecha a cualquier objeto imprimible tal como un documento o una hoja tabular. Cuando una petición genérica es emitida, un proceso de selección determina cual

código se va a ejecutar para proporcionar el servicio. Parte de ese código pudiera ser común a varios objetos y ser reutilizado en la aplicación de sus servicios.

Cómo están organizados los objetos. Los objetos pueden estar organizados jerárquicamente en términos del grado por el cual comparten una utilización común. Los objetos que comparten completamente una utilización común son instancias múltiples de la misma clase (algunas veces llamada "tipo"). Por ejemplo, todos los objetos **Empleado** pueden ser de la misma clase.

Los objetos pueden compartir parcialmente una utilización común. Esta noción es llamada **"herencia"**. Una utilización de clase puede ser definida en términos de otras utilizaciones de clase. La nueva utilización amplia las existentes agregando datos a la representación, nuevas operaciones, y/o cambiando la definición de las operaciones existentes. Por ejemplo, una clase llamada **TituloVentana** pudiera ser definido para heredar a partir de la clase **Ventana.** La clase **TituloVentana** agregaría una definición para los datos del título y los procedimientos a ser utilizados para los servicios **Título** y **PonTítulo.** La herencia múltiple permite más de una utilización de un objeto al definir uno nuevo. La herencia proporciona una forma sistemática de entender aún grandes y complejas utilizaciones de objetos para satisfacer nuevos requerimientos.

Dominios de la Tecnología de Objetos

Las bases de datos orientadas a objetos es uno de los mayores dominios técnicos de traslape, que en conjunto comprenden una solución orientada a objetos completa. Los otros dominios incluyen lenguajes de programación, medios ambientes, herramientas y metodologías.

Una base de datos orientada a objetos almacena y maneja bloques de construcción de datos y funcionamiento (o comportamiento), con las mismas nociones soportadas por los lenguajes de programación de objetos. Estas incluyen utilizaciones compartidas para los objetos de una clase dada, soporte para la herencia, y ejecución de interfaces explícitas. De hecho, un lenguaje de programación de objetos como **C++** muy bien pudiera ser uno de los lenguajes de interface soportados por un DBMS. Otro pudiera ser una versión orientada a objetos de SQL, o un 4GL (lenguaje de cuarta generación) orientado a objetos.

Un DBMS orientado a objetos difiere de los modelos jerárquico, de red y relacional en tres aspectos fundamentales:

- Soporta tipos de datos abstractos y no está restringido a nociones de registros o túplas.

- Soporta arbitrariamente operaciones complejas sobre los tipos de datos. Las peticiones no están restringidas a conjuntos predefinidos de manipuladores; por ejemplo, el **Select** relacional, **Project, Join** y derivados del **Union.**

- Los servicios para la utilización de código son almacenados en la base de datos y son activados por el DBMS objeto. La utilización de servicios son manejados como recursos compartidos.

De estas diferencias provienen los mayores beneficios de las bases de datos orientadas a objetos: la extensión de la tecnología de bases de datos en áreas de aplicación que no han estado bien atendidas por los DBMSs tradicionales, reutilización de código, confiabilidad a través de la encapsulación y oportunidades de optimización del rendimiento.

Evolución del Medio Ambiente de Aplicación

Las bases de datos orientadas a objetos proporcionan el siguiente paso en la evolución de los medios ambientes de aplicación. Anteriormente, las aplicaciones estaban generalmente basadas en archivos. En un medio ambiente basado en archivos cada programa de aplicación incluye su propio código para describir las estructuras de datos y las interfaces con el usuario, para la manipulación de datos, y la lógica de control, para llevar a cabo la interface con los servicios del sistema operativo. Algunos archivos son accesados por programas de diversas aplicaciones, pero esta compartición típicamente está limitada al paso de un archivo producido por un programa como entrada para otro. Las responsabilidades de la integridad de los datos, la manipulación de los mismos y la lógica residen en el programador de la aplicación

En un medio ambiente basado en datos, el DBMS toma la responsabilidad para el manejo de las descripciones de los datos y su utilización. También tiene la responsabilidad para proteger los datos, con todos los esperados servicios de respaldo y recuperación, manejo de transacciones, control de concurrencia y protección de la seguridad. Los datos se convierten en un recurso compartido y las aplicaciones son más independientes de los cambios en las estrucuturas de los datos y su utilización. El programador de aplicaciones retiene la responsabilidad para la manipulación de los datos y la corrección de la lógica.

En un medio ambiente basado en objetos, un sistema manejador de objetos es responsable del manejo de las descripciones y la utilización de los servicios objeto, y de proporcionar las funciones tradicionales de manejo de datos de un DBMS. El código específico de la aplicación en un programa se reduce eventualmente a secuencias de control que requieren los servicios de los objetos pertinentes.

Aplicabilidad de las Bases de Datos Orientadas a Objetos

Características de las bases de datos orientadas a objetos: El interés primario en las bases de datos orientadas a objetos surge de áreas de aplicación con necesidades de manejo de datos que no son fácilmente soportadas por el enfoque relacional. Estas aplicaciones pueden requerir lo siguiente:

- Estructuras de datos compuestas, las cuales incluyen colecciones, objetos anidados y objetos agregados. Esto motiva el interés en las bases de datos orientadas a objetos para soportar aplicaciones de oficina y sistemas de publicación, los cuales almacenan y manejan documentos, aplicaciones de ingeniería y manufactura, las cuales requieren datos de definición de productos compartidos.

- Búsquedas no basadas en valores. Esto motiva el interés de aplicaciones de tipo geográfico.

- Múltiples representaciones lógicas de los mismos datos. Esto motiva el interés de aplicaciones como: CAD (Computer Aided Design), CAM (Computer Aided Manufacturing), CAE (Computer Aided Engineering) y CASE (Computer Aided Software Engineering), las cuales integran varios modelos de la misma definición del producto y de las facilidades de manejo de aplicaciones, las cuales necesitan una gran variedad de representaciones de componentes encadenados.

Características de la aplicación. Un interés secundario en las bases de datos orientadas a objetos surge en áreas de aplicación que están comprometidas para hacer uso de otros aspectos de la tecnología de

objetos. Por ejemplo, las nuevas aplicaciones en telecomunicaciones cada vez más hacen uso de **C++** y están expresando un gran interés en las bases de datos orientadas a objetos. Algunos nuevos productos de CASE están incorporando análisis orientado a objetos y técnicas de diseño con el mismo propósito.

El actual interés en las bases de datos orientadas a objetos surge principalmente por parte de la comunidad técnica, la cual históricamente no ha sido bien servida por productos DBMS. La demanda se basará en el mercado potencialmente lucrativo de **MIS** (Management Information Systems), después de que se acepten las metodologías y herramientas de análisis y diseño orientadas a objetos. Una barrera significativa para la aceptación de soluciones completas orientadas a objetos para MIS será eliminada cuando esté disponible un COBOL orientado a objetos.

Aplicando Bases de Datos Orientadas a Objetos

Suponiendo que los procesos de análisis y diseño han dado como resultado un modelo de datos que representa la semántica del área de un problema. Resulta instructivo comparar varios aspectos del diseño de bases de datos bajo los ambientes relacional y orientado a objetos.

Herencia. El modelo de datos puede incluir estructuras hereditarias. Una entidad **Empleado**, por ejemplo, puede tener dos subentidades **Administrador** e **Ingeniero**. Existen varias opciones para implantar esta estructura con una base de datos relacional. Cada una tiene sus problemas.

Considérense tres posibilidades. (1) Si una tabla fuera usada por todos los tipos de empleados, con un campo extra para denotar el tipo de empleado, los valores nulos serían problemáticos y el DBMS no podría hacer cumplir cuales campos son válidamente no nulos para cuales

tipos de empleado. (2) Si una tabla fuera usada para cada una de las entidades (e.g., toda la información de los empleados almacenada en una tabla, toda la información adicional para administradores almacenada en una segunda tabla, y toda la información de los ingenieros almacenada en una tercera tabla), entonces las aplicaciones necesitarían saber cual tabla accesar para la información de administradores. (3) Si una tabla fuera usado para cada subentidad (e.g., toda la información de administradores almacenada en una tabla y toda la información de ingenieros almacenada en una segunda tabla), entonces las instancias de empleados estarían distribuidas entre las tablas, y las aplicaciones necesitarían saber cuales tablas accesar para la información de empleados.

En contraste, el aplicar la herencia con una base de datos orientada a objetos es más directo. Los subtipos **Administrador** e **Ingeniero** pueden heredar del tipo **Empleado** y las aplicaciones pueden utilizar abstracciones más naturales.

Encapsulación (o encapsulamiento). Considérese un caso relativamente simple en el cual se toma una decisión para cambiar una base de datos para almacenar el **Cumpleaños del Empleado** en lugar de su **Edad.** Si esta fuera una base de datos relacional, las aplicaciones que hacen uso de la **Edad** cesarían de operar apropiadamente a menos que fueran cambiadas para accesar el **Cumpleaños** y calcular la **Edad.** En contraste, con una base de datos orientada a objetos, la **Edad** aun sería útil y las aplicaciones no tendrían que ser cambiadas. En su lugar, la utilización del objeto **Empleado** sería cambiada para calcular la **Edad** a partir del **Cumpleaños** más bien que tan sólo recuperar la **Edad** de la base de datos.

Referencia a objetos. Considérese la situación de un negocio en el cual no hay nada único con respecto a **Proyectos;** tienen nombres e

información de fechas, pero no atributos que los identifiquen en forma única y natural. Para almacenar datos de un proyecto en una base de datos relacional, las llaves primarias deben ser inventadas a pesar de ser poco natural. Las llaves que no son significativas son difíciles de manejar. En contraste, con una base de datos orientada a objetos cada objeto recibe una referencia de objeto única, asignada por el sistema en el momento de su creación. Esta referencia del objeto es un identificador único que es independiente de los valores de los atributos. Las llaves basadas en los atributos son usadas sólo donde el problema es natural al mundo real.

Construyendo un DBMS Orientado a Objetos

Varios enfoques pueden ser tomados para construir un producto DBMS orientado a objetos. Tres de ellos se presentan a continuación.

Agregando persistencia al ambiente del lenguaje de programación. Un enfoque es el de la persistencia del lenguaje, en el cual el almacenamiento permanente es agregado a los objetos definidos en un lenguaje de programación orientado a objetos. Esto se está haciendo con **C++** por varias compañías fabricantes de DBMS's orientados a objetos, que incluyen a **ONTOLOGIC** y **Object Design.**

Smalltalk es la base no sólo del trabajo de **ParcPlace,** sino también del producto **Serviologic** de **Gemstone** y del producto **Vision** de **Insyte.** Tanto **Graphael** como **Symbolics** le han agregado persistencia de objetos a **Lisp.**

Extendiéndose a un DBMS relacional. Otro enfoque es agregar extensiones de objetos a DBMS's relacionales esto es típicamente llevado a cabo al relajar las restricciones sobre los tipos de datos impuestas por productos relacionales clásicos. Los tipos de datos resultantes son algunas veces llamados **"blobs".** Por ejemplo, los **DBMS's**

relacionales de Oracle e Informix se están moviendo en esta dirección. **Sybase** va un paso adelante en el producto **SQLServer** al agregar procedimientos almacenados, los cuales efectivamente ponen código dentro de la base de datos. Ninguno de estos sistemas cumplen dos de los primeros criterios para ser orientados a objetos: **soporte de la herencia y la ejecución de interfases explícitas**

Empezando. Un tercer enfoque es imponer un modelo de objetos sobre la fundamentación de las bases de datos. Surgen considerables variaciones en las características de los resultados. Por ejemplo, el prototipo **Iris** de Hewlett Packard utiliza un modelo de objetos en un almacenamiento de calidad de la producción y manejador de transacciones. El resultado es un verdadero DBMS orientado a objetos. Object Sciences Corporation, otro de los DBMS's objeto que empieza, también toma este enfoque. En contraste el enfoque de Data General utiliza una versión objeto de SQL en la parte superior de un DBMS relacional. El resultado toma ventaja de todas las capacidades del DBMS relacional subyacente, pero es a veces restringido por un paso de traducción intermedia a SQL.

Retos para las Bases de Datos Orientadas a Objetos

Varias áreas de reto deben ser enfocadas para alcanzar un potencial completo de la tecnología de bases de datos orientadas a objetos, como sigue.

Técnica. Las preguntas técnicas sobre bases de datos orientadas a objetos ciertamente no han sido todas respondidas. Aún existe una necesidad de investigación y de desarrollo innovativo. Las áreas desafiantes incluyen las siguientes:

- **Semántica sin sorpresas.** Debe ser eventualmente posible para un usuario visualizar el modelo objeto de su aplicación y ser capaz de predecir con precisión el comportamiento del sistema. Esto no es

posible hasta el presente. Por ejemplo, diferentes sistemas orientados a objetos utilizan herencia múltiple en forma diferente. Un elemento significativo es el de determinar las prioridades de herencia de múltiples objetos padres.

- **Versión y herencia.** La herencia por sí misma es bastante bien comprendida (excepto por la pregunta respecto a la herencia múltiple presentada líneas arriba). El manejo de la versión también ha sido bien investigado. Juntos sin embargo, estos dos conceptos crean preguntas difíciles de como manejar versiones múltiples de un objeto modelo cuando diferentes estructuras de herencia están involucradas.

- **Objetos agregados.** Los objetos agregados están compuestos de otros objetos. Algunas veces son llamados objetos complejos o compuestos. Los retos técnicos permanecen al utilizar estos objetos como extensiones de bases de datos relacionales y al especificar servicios que manipulen estos objetos.

- **Conexiones de lenguaje.** Las conexiones entre lenguajes de bases de datos orientadas a objetos y lenguajes de programación orientados a objetos han permitido prototipos, con el resultado de que objetos definidos en **C++** (por ejemplo), pueden ser almacenados en una base de datos de objetos. Típicamente las construcciones del lenguaje de interface a la base de datos de objetos no corresponden con las construcciones del lenguaje de programación orientado a objetos. Aún no es claro cual debe ser la sintáxis en común. Es menos claro aún lo que el programador debe de conocer acerca de si la manipulación del objeto proviene de la base de datos o de la programación

- **Objetos realmente grandes.** Muchas aplicaciones que pudieran beneficiarse de las bases de datos de objetos requieren del manejo de grandes objetos. Por ejemplo, los objetos que

representan imágenes, características geográficas, y estructuras de productos complejos pueden cada una requerir gigabytes de almacenamiento. La tecnología de los DBMSs actual comúnmente falla en el manejo de bases de datos de estos tamaños.

- **Rendimiento.** Como con muchas otras áreas de la tecnología, existen muchas oportunidades para mejorar el rendimiento de los productos DBMS que están surgiendo. La complejidad de este hecho se presenta porque no es aún claro como debe ser medido el rendimiento. Existen unos cuantos **"benchmarks"** (pruebas de rendimiento) publicados, pero caracterizan (como otros benchmarks) sólo ciertos tipos de actividades, los cuales pueden no ser útiles del todo para predecir el rendimiento de una aplicación particular. La comunidad de usuarios de tecnología de DBMS's relacionales está ahora estandarizando sus nociones de lo que un **"benchmark"** debe ser. Se necesita de una considerable cantidad de trabajo para desarrollar conjuntos de **"benchmarks"** para comparar DBMSs orientados a objetos.

- **Objetos distribuidos.** Adicionalmente a todos los elementos que surgen para soportar bases de datos distribuidas, se presentan otras preguntas cuando las bases de datos orientadas a objetos son colocadas en un medio ambiente de cómputo distribuido. Por ejemplo, ¿cómo van a ser manejados los objetos agregados compuestos de objetos componentes en múltiples nodos? ¿es necesario tener un único esquema de referencia de objetos en la red? ¿cuál es la arquitectura de un medio ambiente fusionado de bases de datos relacional y orientada a objetos?

- **El mercado.** Los retos del mercado abundan también. Una de las preguntas más significativas es "¿Cuál es el valor real de las bases de datos de objetos?" ¿Por qué los usuarios comprarán productos DBMS's orientados a objetos? Muy pocos datos existen

actualmente que soporten las afirmaciones de una mayor productividad a partir de la tecnología de objetos; la mayoría de las experiencias citadas son anecdóticas. La cuantificación de los beneficios de los objetos puede ser uno de los retos críticos a ser satisfechos para el éxito comercial de productos de tecnología de objetos.

Otro reto que se presenta para el mercadeo de tecnología de objetos es posicionar las bases de datos orientadas a objetos con relación a los productos de bases de datos relacionales. ¿Cuando es cada uno aplicable? ¿Cómo deben coexistir las bases de datos de objetos y las relacionales? ¿Cuando empezarán los usuarios de MIS (Management Information Systems) a hacer la transición a bases de datos de objetos?

Más evidente es la pregunta relativa a la diferenciación entre productos emergentes DBMS's de objetos. En el presente es difícil para el usuario promedio establecer como un producto es potencialmente diferente de otro. Es aún más difícil no confundirse por las similitudes en los nombres de los diferentes nombres de los productos y compañías.

Relativas a estos retos del mercado, surgen preguntas con respecto a la estandarización de la terminología y las interfaces para productos con orientación a objetos. Grupos formales e informales están en el presente activamente dedicados a estos aspectos. Un balance razonable debe ser alcanzado cuidadosamente entre la estandarización que posibilita la interoperabilidad y la comunicación, y la estandarización que suprime la innovación temprana.

- **Soporte.** En general, las bases de datos de objetos no es una tecnología que pueda ser lanzada exitósamente sobre la pared a los usuarios. Unos cuantos grupos elitistas serán capaces de evaluar los productos y tal vez elaborar prototipos, pero la gran comunidad de usuarios no sabrá como cosechar los beneficios de

la tecnología de objetos tan sólo leyendo un manual de referencia o conceptos acerca de DBMS's orientados a objetos. Será muy importante que los vendedores sean capaces de proporcionar a sus clientes con el suficiente soporte y servicios relacionados con el proyecto para asegurar el éxito de la aplicación con productos DBMS's de objetos.

Preparándose para la Tecnología de Objetos.

Los usuarios potenciales de bases de datos de objetos pueden hacer varias cosas en este momento para prepararse para la posible disponibilidad de tecnología de objetos.

Conciencia y educación. La disponibilidad de información con respecto a esta área se está incrementando rápidamente. Se ofrecen muchos seminarios públicos sobre tecnología de objetos. La mayoría de estos enfatizan sobre un lenguaje de programación en particular. Muy pocos se enfocan sobre un producto DBMS de objetos en particular o sobre alguna metodología de análisis orientada a objetos. La información tutorial está también disponible en un número creciente de libros sobre la materia. Los capítulos introductorios de varios textos sobre lenguajes de programación orientados a objetos puede ser de mucha utilidad para comprender los conceptos fundamentales de los objetos, aún para un profesional si es que este no quiere atacar los detalles de un lenguaje en particular. Los artículos relativos a tecnología de objetos también están empezando a aparecer en publicaciones periódicas sobre bases de datos y software.

Evaluación y transmisión de la experiencia. Después de aprender acerca de los conceptos básicos sobre objetos, adquirir experiencia con un producto sobre objetos puede ser benéfico. Esto significa elaborar un prototipo de una aplicación utilizando **Smalltalk,** o

construir un prototipo con una copia de evaluación de un producto DBMS de objetos.

Metodología. Tal vez el paso más grande hacia un exitoso uso de la tecnología de objetos es moverse hacia la adopción de una metodología de análisis y diseño de objetos. Esto tiende a ser más fácil si se enfoca desde datos centralizados en lugar de la filosofía de procesos centralizados. El modelado de datos debe por lo tanto ser incorporado en el trabajo de análisis y diseño. Utilizar diccionarios de datos para inventariar y controlar recursos de datos. Planear para legar los sistemas; identificar aquellas partes que son candidatas para la reimplantación en los años venideros. Investigar la aplicabilidad de los servicios profesionales de análisis y diseño orientados a objetos disponibles para un ambiente dado.

En conclusión; el mayor beneficio de la tecnología de objetos generalmente esperado es el de una mayor productividad sobre el ciclo de vida de los sistemas de información. Esta productividad surge de lo siguiente:

- Los modelos que mejor captan las características de los sistemas del mundo real.

- El software que es más adaptable y flexible para acomodar los cambios en los requerimientos del negocio.

- Los módulos de software que son más fácilmente reutilizables a través de las aplicaciones.

- Modos de interacción que son más naturales para la gente. Las bases de datos orientadas a objetos es tan sólo una de las tecnologías que serán instrumentales para alcanzar los beneficios potenciales de la orientación a objetos. Por lo que toca al usuario, las bases de datos orientadas a objetos serán una parte invisible

del medio ambiente de cómputo. Uno de los beneficios para las organizaciones es que las bases de datos serán más invisibles al desarrollador de la aplicación también.

CONCLUSIONES

El auge actual de las redes de computadoras implica necesariamente mantenerse a la vanguardía respecto al hardware y software que existe en el mercado, así como el manejo de los conceptos que este tema involucra.

Ningún hardware y software es necesariamente el óptimo, dicha resolución dependerá de recursos propios que sean válidos para respaldar el porque de la selección, tal es el caso del propuesto.

Dado que en el manejo de una base de datos se encuentran involucrados muchos usuarios es necesario establecer una función administrativa permanente para coordinar las operaciones relacionadas con ellas y realizar auditorías periódicas para comprobar que realmente se estan cumpliendo las reglas que rigen dicha administración.

Es muy importante tener presente que se puede hacer y que no se puede hacer con los productos, ya que esto implica el nivel del manejo adecuado y productos finales (para el usuario) de mayor calidad.

GLOSARIO

AD HOC: Que va de acuerdo, que conviene a tal objeto.

AMPLIFICADOR: Dispositivo que eleva la potencia de una señal. Utilizado para prevenir la atenuación (deterioro) de las señales transmitidas.

AMPLITUD: Distancia entre dos puntos alto y bajo de forma de onda o una señal.

ANCHO DE BANDA: El rango de frecuencias entre dos límites definidos, expresado en hertz. El ancho de banda determina el porcentaje de información que puede ser transmitida.

ANECDOTA: Narración , historieta, chascarrillo, eco. Relación breve de algún rasgo o suceso particular y curioso.

ANSI: Abreviación de American National Standards Institute, es una institución voluntaria que ayuda a definir estándares, y que también representa a los E.U.A. en la Organización Internacional de Estándares (ISO).

ARCNET: Abreviación de Attached Resource Computer Network, red creada por Data Point. Transmite a 2.5 Mbps. Fue muy utilizada en el mundo debido a su bajo costo, gran confiabilidad y versatilidad del cableado con topología de estrella.

ARPA: Siglas de Advanced Research Projects Agency, agencia dentro del Departamento de Defensa de E.U.A., que da soporte a la red ARPANET.

ARPANET: Es una red de área amplia que utiliza protocolos de paquetes diferidos (tipo X25). La red fué creada por ARPA junto con el Departamento de Defensa de E.U.A. para dar soporte a las comunidades militares. ARPANET se divide en dos partes

interconectadas: Milnet, para uso militar e Internet, para uso comercial y académico.

ASINCRONO: Forma de transmisión que no requiere que el receptor y el transmisor mantengan en "sincronía" sus relojes. Pero en cambio necesita que el transmisor "inserte" bits antes y después del carácter para que el receptor lo reconozca. Es más barata que la transmisión síncrona, pero menos eficiente.

BANDA BASE: Referencia a señales en su forma original y no cambiada por modulación.

BANDA ANCHA: Se refiere a los medios que pueden soportar un amplio rango de frecuencias electromagnéticas moduladas.

BENCHMARK: Es un método para medir el rendimiento de un sistema en un ambiente controlado, usando una metodología estándar. Los benchmark para medir el rendimiento en manejadores de bases de datos más usados son: el Benchmark de DeWitt y el TP1.

CASE: Siglas de Computer Aided Software Engineering, la utilización de software para ayudar en la definición, elaboración, designación, documentación y algunas otras áreas del desarrollo de programas.

CONECTIVIDAD: La habilidad de enlazar diferentes piezas de hardware (Macintosh, microcomputadoras, minicomputadoras y mainframes) y software en un ambiente de red, donde los recursos (aplicaciones, software, etc.) son compartidos.

DE FACTO: Estándar del mercado, de la industria, resultado del uso común.

FILE SERVER: Dispositivo de red que proporciona acceso a programas y archivos compartidos.

FRAGMENTACIÓN: Es el proceso de partir un datagrama grande en múltiples datagramas menores en tamaño, para transmisión.

FRAME: Unidad de información del nivel 2 del modelo OSI. Usualmente un frame consta de tres partes: un header (o encabezado) que trae información de control, direcciones fuente y destino, etc. Un campo de información y un campo de CRC (verificación de errores).

FRECUENCIA: Número de ciclos por unidad de tiempo. Normalmente medida en Hertz (Hz), que son ciclos por segundo.

GATEWAY: Dispositivo que permite conectar dos redes (locales o geográficas) con diferntes protocolos. Ejecuta operaciones de conversión de protocolos para que se interconecten dos redes o dispositivos incompatibles.

HARDWARE: El equipo usado en una red (tales como: computadoras, impresoras, cable, tarjetas de red, etc.).

HZ: Unidad de frecuencias electromagnéticas igual a 1 ciclo por segundo.

INTERFACE: Límite entre dos programas a través de la cual todas las señales que pasen son cuidadosamente definidas.

IPX: Internetwork Packet Exchange, protocolo puerto a puerto, propio de Novell, que actúa en el nivel 3 del Modelo OSI (nivel de red), entre sus ventajas esta el tener direcciones de tres campos: nodo, red y socket, que le permite tener enlaces entre redes y varios procesos corriendo en diferentes servidores. Está basado en el protocolo de nivel 3 XNS (Xerox Networking Systems).

ISO: Organización Internacional de Estándares.

KHZ: Kilohertz.

LENGUAJE DE ALTO NIVEL: Lenguaje de programación orientado hacia la resolución de problemas, las instrucciones se dan a la

computadora usando letras convenientes, símbolos o texto parecido al inglés, en lugar de usar el código de entrada/salida que entiende la computadora.

LENGUAJE DE CUARTA GENERACION: Lenguaje no procedural, es decir, el usuario indica que datos desea sin decir como obtenerlos.

LENGUAJE DE TERCERA GENERACION: Lenguaje orientado a procedimientos, es decir, el usuario indica cuales datos quiere y como debe obtenerlos.

MAINFRAME: Término empleado para referirse a computadoras grandes que requieren de algún medio ambiente especial (aire acondicionado, etc). Su capacidad de conexión de terminales es de miles.

MINICOMPUTADORA: Computadora de tamaño mediano; las minicomputadoras se encuentran en un punto intermedio entre las microcomputadoras y las mainframes, y su capacidad de conexión de terminales es entre unas cuantas hasta varios cientos.

MODULACION: Mezcla de una señal con la portadora. La modulación es el proceso de entremezclar una señal de voz o una serie de datos con una portadora, para su transmisión a través de la red.

OSI: Siglas de Open System Interconection , estructura lógica y estándar de 7 niveles de protocolos definida por ISO para facilitar la comunicación entre ambientes heterogéneos.

OS/2: Es un sistema operativo para microcomputadoras que permite tomar todas las ventajas del procesador 80286, además permite ejecutar más de 1 aplicación al mismo tiempo.

PARSIMONIOSO: Economía, moderación, escasez.

PC: Personal Computer, computadora de tamaño pequeño (en comparación con mainframes y minicomputadoras) o de escritorio, estos computadores son de uso generalizado.

PERFORMANCE: Factor que determina la productividad total de un sistema. Así pues el desempeño se determina por una combinación de los siguientes factores: disponibilidad, tiempo de respuesta y through-put.

PLATAFORMA: Referencia el sistema operativo que funciona en una máquina computadora tal como: OS/2, DOS, UNIX, etc.

PROTOCOLO: Son los lenguajes de la red. Son un conjunto de reglas por medio de las cuales se establece, mantiene y controla la comunicación.

PUENTE: Dispositivo que permite enviar datos de una red a otra.

RELACION: Asociación entre conjuntos de entidades (tablas en una RDBMS).

REMOTO: Físicamente distante.

REPETIDOR: En transmisión analógica, equipo que recibe una serie de señales, las amplifica y reenvía a otros dispositivos. En transmisión digital, equipo que recibe una serie de señales, las reconstruye y entonces amplifica y reenvía.

RUTEADOR: Los ruteadores pueden enviar paquetes de datos sobre diferentes trayectoria en una red.

SEMANTICA: Relativo a la significación, ciencia que trata de los cambios de significación de las palabras.

SERVER: Servidor de base de datos, también llamado back-end en la arquitectura cliente-servidor. Consiste en proporcionar acceso a archivos de bases de datos compartidas en una red.

SINCRONO: Forma de transmisión en la que ambos extremos deben tener un mismo pulso de reloj, y con base a éste, ambos extremos conocen en que momento pueden transmitir. Aunque en la transmisión síncrona no se necesitan bits de inicio y final por cada carácter, el hardware requerido para sincronizar los pulsos de reloj, la hace más cara que asíncrona.

SINTAXIS: Parte de la gramática que enseña a coordinar y unir las palabras para formar oraciones.

SISTEMA ABIERTO: Capacidad del dispositivo o computadora de poder comunicarse con cualquier otro dispositivo para el intercambio de información.

SNA: Siglas de Systems Network Architecture, la arquitectura de protocolos para redes creada por IBM.

SOFTWARE: Instrucciones de computador que ejecutan funciones comunes para todos los usuarios, como también aplicaciones específicas para necesidades de usuarios particulares.

SPX: Secuenced Packet Exchange, debido a que IPX sólo envía los paquetes, SPX revisa que todos los paquetes esten completos y que lleguen en orden.

TARJETA: Dispositivo que va instalado dentro de una microcomputadora y según su especificación cada tarjeta determina la forma de concexión de la red.

TELENET: Red privada, disponible comercialmente, que provee servicios de paquetes conmutados y circuitos conmutados a sus abonados en Norte America, Europa y Asia.

TOPOLOGIA: El arreglo físico de estaciones de la red en relación de una con otra.

TUPLA: Referencia a un renglón de una tabla (relación) definida en una base de datos con modelo relacional.

TYMNET: Red de paquetes conmutados instalada en E.U.A., Europa y otras partes del mundo British Telecom se encarga de la administración de esta red.

UNIX: Un sistema operativo multiusuario, desarrollado por los laboratorios Bell.

BIBLIOGRAFIA

FRANK J. DERFLER, JR.
Guide to Connectivity.
Editorial: Ziff-Davis.
Año de edición 1991.

ALABAU MUÑOS ANTONIO
Teleinformática y Redes de Computadoras.
Editorial: Publicaciones Macombo
2a. edición México.
1987.

DATE C. J.
Introducción a los Sistemas de Bases de Datos.
Editorial: SITESA.
3a. Edición México.
1989.

HENRY F. KORTH & ABRAHAM SILBERSCHATZ.
Fundamentos de Bases de Datos.
Editorial: Mc Grawhill.
1a. Edición México.
1987.

MILLER MARK A.
LAN Troubleshooting Handbook.
Editorial: M&T Books.
1a. Edición Estados Unidos.
1989.