II-A
2ejc

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES
"A C A T L A N"

"BASES DE DATOS RELACIONALES: UN MODELO Y UNA HERRAMIENTA PARA LA IMPLEMENTACION DE SISTEMAS DE INFORMACION"

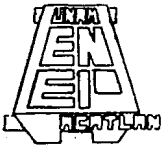
T E S I N A

QUE PARA OBTENER EL TITULO DE

LICENCIADO EN MATEMATICAS APLICADAS Y COMPUTACION

P R E S E N T A :

VELIA ERIKA ESPINOSA SANCHEZ



MEXICO, D. F. 1994

TESIS CON
FALLA DE ORIGEN

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

DERECHOS RESERVADOS © PROHIBIDA SU REPRODUCCIÓN TOTAL O PARCIAL

Todo el material contenido en esta tesis está protegido por la Ley Federal del Derecho de Autor (LFDA) de los Estados Unidos Mexicanos (México).

El uso de imágenes, fragmentos de videos, y demás material que sea objeto de protección de los derechos de autor, será exclusivamente para fines educativos e informativos y deberá citar la fuente donde la obtuvo mencionando el autor o autores. Cualquier uso distinto como el lucro, reproducción, edición o modificación, será perseguido y sancionado por el respectivo titular de los Derechos de Autor.

Este trabajo lo dedico a mis padres y a mis hermanos quienes con su apoyo, confianza y cariño me ayudaron a esforzarme cada día por ser una mejor persona y a obtener las bases necesarias para ser una profesionista.

Muchas Gracias.

También agradezco a mi novio Carlos por ser la persona que siempre me alentó a concluir este trabajo con muchísimo amor y comprensión.

Gracias Mi Amor.

"BASES DE DATOS RELACIONALES: *UN MODELO Y UNA HERRAMIENTA PARA LA IMPLEMENTACION DE SISTEMAS DE INFORMACION*"

CONTENIDO.

INTRODUCCION

El presente trabajo nace de la necesidad de tener un material de apoyo didáctico para la materia de Base de Datos, que actualmente pertenece al plan de estudio de la carrera de Matemáticas Aplicadas y Computación, o alguna otra que investigue y analice este concepto.

Además de conocer el esquema y las tendencias en el procesamiento de datos que utilizan las Bases de Datos Relacionales, ya que actualmente son una herramienta poderosa en el ciclo de vida de múltiples sistemas de información. Su importancia radica en las características de éstas, las cuales nos permiten obtener grandes beneficios, tales como: tener la información centralizada, esto es compartir dicha información entre los múltiples sistemas que así lo requieran, tener menos dependencia entre las aplicaciones (sistemas diseñados ad-hoc a las necesidades de cualquier empresa) y los equipos de cómputo en que son diseñadas, menor esfuerzo de programación, adaptabilidad a los cambios en las estructuras de datos; en fin, un sin número de ventajas que redundan en productividad.

Es entonces que a través de la elaboración de este trabajo resulta muy importante dar a conocer el panorama de Bases de Datos Relacionales, su origen, sus bases, sus características y lo más importante un enfoque práctico para el diseño e implementación de sistemas de información.

ANTECEDENTES

EVOLUCION DE LOS SISTEMAS MANEJADORES DE DATOS

A principios de la década de los sesentas, el punto más importante dentro del estudio de Sistemas Manejadores de Datos fue la introducción por parte de CODASYL (COnference on DAta SYstems Languages) del compilador del lenguaje COBOL, acompañado por la evolución de unidades de almacenamiento en cinta y la aparición subsecuente de los dispositivos de almacenamiento de acceso directo. Al surgir la necesidad de aplicaciones (sistemas diseñados ad-hoc a las necesidades de cualquier empresa) más complejas se observó la necesidad de agregar al compilador de COBOL paquetes (códigos de programación) que facilitaran el ordenamiento y clasificación de datos, asi como la generación de reportes surgiendo también las organizaciones lógicas (estructuras de datos) de alto nivel para los datos y es entonces que las aplicaciones comenzaron a interrelacionarse entre sí para ponerse a disposición de un mayor número de usuarios.

Como productos comerciales surgieron los Sistemas Generalizados para Manejo de Archivos (GFMS), Sistemas Generalizados para la Administración de Bases de Datos (GDBMS) y Sistemas de Bases de Datos/ Telecomunicaciones (DB/DC).

En 1971, el DBTG (Data Base Task Group) de CODASYL presentó un documento acerca de las bases de datos, en el cual quedaron asentadas las bases para el desarrollo de los sistemas DBMS (Data Base Management Systems).

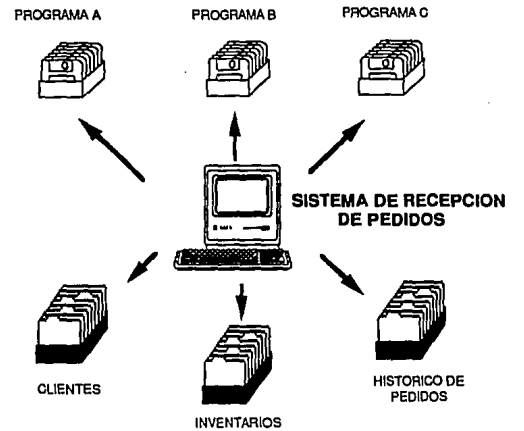
Antes de que aparecieran las primeras computadoras de la tercera generación (la primera fue instalada en 1965) la mayoría de los archivos se organizaban de modo secuencial simple. El programador de aplicaciones diseña la distribución física de los datos y la incorpora a los programas de aplicación. De ahí que los mismos datos dificilmente se comparten entre las distintas aplicaciones.

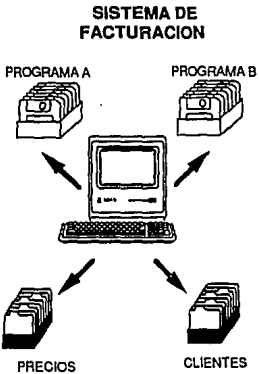
A mediados y finales de la década de los setentas, se reconoció la naturaleza cambiante de los archivos y de los dispositivos de almacenamiento. Los archivos estaban por lo general, diseñados para una aplicación determinada o para un grupo de aplicaciones muy similares. Es posible el acceso secuencial o el acceso directo (al azar) a los registros. El software provee "métodos de acceso", pero no "administración de datos".

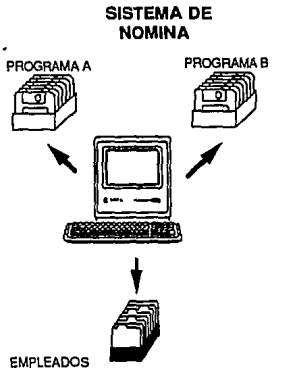
En los años ochentas y principios de los noventas, de los mismos datos físicos se derivan múltiples bases de datos lógicas. Se puede tener acceso a los mismos datos de diferentes maneras, según los requisitos de la aplicación. El software provee los medios para disminuir la redundancia Se utilizan formas de organización de datos muy complejos sin que ello se refleje en los programas de aplicación.

En la etapa actual, el software procura la independencia lógica y física de los datos. Se facilita la administración e integración de las aplicaciones. Los datos pueden evolucionar sin que se incurra en costos de mantenimiento excesivos.

EL ENFOQUE TRADICIONAL







EL ENFOQUE TRADICIONAL

En un departamento tradicional de Informática, la fuerza motriz es el desarrollo y mantenimiento de aplicaciones (sistemas diseñados ad-hoc las necesidades de cualquier empresa). Cada una de estas aplicaciones tiene sus propias estructuras de datos y refleja los requerimientos de alguna gerencia. En otros términos, las aplicaciones son el resultado de los requerimientos individuales de algún área de la organización, que no necesariamente van acordes con las políticas y requerimientos institucionales. Por ello, suele suceder que los cambios en la organización, repercuten fuertemente en las aplicaciones. Adicionalmente, otra característica de este tipo de instalaciones es que emplean a la computadora para almacenar, procesar y producir información, sin preocuparse por controlar y vigilar su integridad.

Indudablemente, de las cosas que cambian con mayor frecuencia son los requerimientos de información. Cuando cada aplicación tiene sus propias estructuras de datos, se generan arquitecturas caóticas y redundantes. Además resulta una labor titánica tratar de obtener información proveniente de estructuras de datos de diversas aplicaciones.

Un ejemplo de esto, es el esquema anterior en el que se ve claramente que dada la independencia de las aplicaciones, no es posible compartir información, lo que provoca tener información redundante e inconsistente.

En conclusión, el enfoque tradicional es una forma de trabajar con los sistemas de información, en la cual no se contemplan futuras modificaciones en los datos que estos sistemas manejan, así como en sus estructuras.

DESVENTAJAS DEL ENFOQUE TRADICIONAL

• Los costos de desarrollo aumentan debido a que no es fácil encontrar equipos de cómputo y software que se adapten a todas las necesidades de la empresa.

Por ejemplo, hace algunos años era muy común encontrarnos con aplicaciones que estuvieran diseñadas y programadas especialmente para un tipo de equipo de cómputo en particular, esto, traía como consecuencia que si una cierta empresa ya contaba con un equipo de cómputo prácticamente tendría que desecharlo para poder seguir operando, lo que redundaba en un gasto considerable cada vez que quisiera cambiar de tecnología.

• La productividad del desarrollador se ve afectada en sentido negativo, ya que los programas son exhaustivos, de manejo complejo y de difícil mantenimiento; por lo que cada cambio en los requerimientos requiere de modificaciones en los programas. También, cada cambio en la estructura de los datos, requiere de cambios en el código de las aplicaciones.

Por ejemplo, ya que precisamente hablamos de que las aplicaciones eran diseñadas especialmente para un cierto equipo, también es cierto que dichas aplicaciones fueran programadas con muchas reglas y restricciones, lo cual hacía bastante tedioso pensar en cambiar operaciones. Nuestros programadores tenía que trabajar por bastante tiempo en esos cambios.

• Encontrar personal que cumpla con los requisitos indispensables para el desarrollo y manejo de las aplicaciones es una tarea difícil, además de que repercute en tiempo y dinero.

RESULTA COSTOSO CONTRATAR Y ENTRENAR AL PERSONAL

• La dependencia entre las aplicaciones y el equipo de cómputo empleado es grande, debido a que los programas manejan características propias del ambiente de cada equipo, por ejemplo; operaciones de I/O, procesos internos y manejo de memoria.

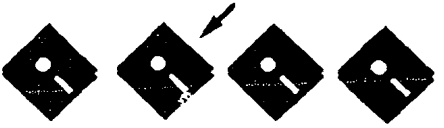
LAS APLICACIONES SE QUEDAN ATRAPADAS EN EL EQUIPO EN EL QUE FUERON DESARROLLADAS. NO SON PORTABLES

• El mantenimiento excesivo de los programas es inevitable y la causa es el cambio continuo en los sistemas de información. Esto hace que la gente dedicada a la programación tenga que estar desarrollando constantemente.

SE REQUIERE MAYOR ESFUERZO DE PROGRAMACION

• La capacidad para compartir datos es muy limitada, ya que existe poco apego a los estándares, y si es necesario modificar algún dato en particular, se tiene que conocer la posición exacta de dónde éste se encuentra.

FRECUENTEMENTE, ES NECESARIO CONOCER LA ORGANIZACION FISICA DE LOS DATOS PARA PODER ACCESARLOS



CAPITULO I. ENFOQUE DE BASE DE DATOS

Dentro de este capítulo presentaremos una serie de conceptos relacionados con Bases de Datos.

I.1 Qué es una Base de Datos?

> "Una Base de Datos es un Conjunto de Datos relacionados entre sí, almacenados, estructurados, no redundante, normalizada y de fácil acceso". (1)

A continuación presentamos algunas definiciones de Base de Datos.

Una Base de Datos es:

- Una colección de archivos relacionados entre sí, de la cual los usuarios pueden extraer información sin considerar las fronteras físicas de los archivos. (1)

- Una colección de datos interrelacionados almacenados en conjunto sin redundancias perjudiciales o innecesarias; su finalidad es la de servir a una o más aplicaciones; los datos son independientes de los programas que los usan ; se emplean métodos bien determinados para incluir datos nuevos y modificar o extraer los datos almacenados. (2)

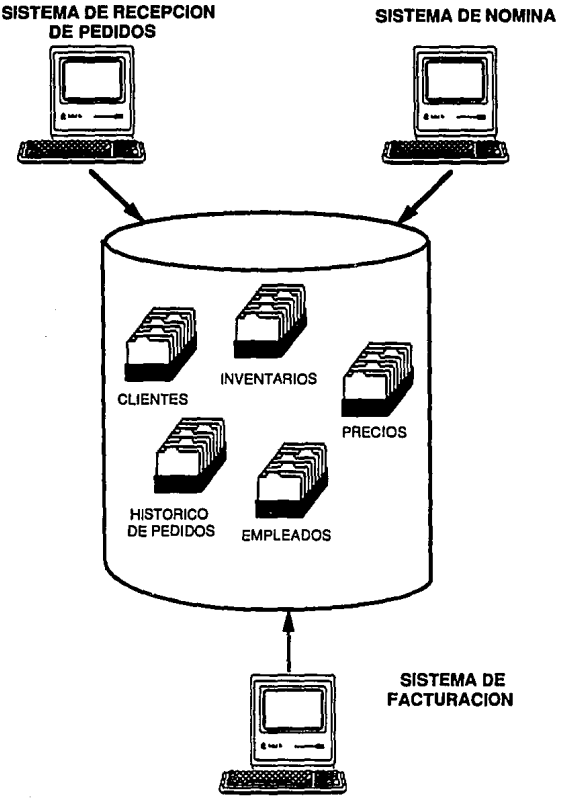
- Es una colección de datos integrada, irredundante y que puede compartirse. (3)

"El enfoque de Base de Datos es la forma de implementar los sistemas de información tomando como la estructura principal una base de datos, la cual organiza, mantiene y controla dicha información". Sus ventajas nos permiten:

- Controlar la redundancia de los datos (que los datos no se repitan)
- Mantener la consistencia (que en distintas salidas de información tengamos los mismos datos)
- Lograr la integración de los datos (que cumplan con las normas establecidas)
- Compartir los datos entre las diferentes aplicaciones
- Cumplir con los estándares
- Tener facilidad en el desarrollo de aplicaciones
- Uniformar los controles de seguridad, privacía e integridad.
- Independencia entre los datos y los programas
- Reducir el mantenimiento a los programas

(1) James Martin, Organización de las Bases de Datos.
(2) E.F. Codd, A Relational Model for Large Shared Data Banks
(3) C.J. Date, An introduction to Database Systems

EL ENFOQUE DE BASE DE DATOS



I.2 Qué es un Sistema Manejador de Base de Datos?

> "Un **DBMS** (**D**ata **B**ase **M**anagement **S**ystem), consta de un Conjunto de Datos Relacionados entre sí y un Conjunto de Programas para tener acceso a esos datos". (1)

Otra definición acerca de un DBMS es la siguiente:

• Un sistema de manejo de base de datos (DBMS) consiste en un conjunto de datos relacionados entre sí y un grupo de programas para tener acceso a esos datos. El conjunto de datos se conoce comunmente como base de datos. Esta contiene información de una organización determinada. El objetivo primordial de un DBMS es crear un ambiente en que sea posible guardar y recuperar información de la base de datos en forma conveniente y eficiente. (5)

(1) James Martin, Organización de las Bases de Datos
(5) Henry F. Korth, Fundamentos de Bases de Datos

- Minimizar la redundancia de los datos significa:

  - No tener datos repetidos.

  - No almacenar datos derivados.

Por ejemplo, si tenemos una aplicación que accesa a dos archivos diferentes y en dichos archivos nos encontramos almacenados los mismos datos, significa que estamos teniendo redundancia en el almacenamiento de esos datos.

- Garantizar la consistencia de los datos es:

  - Obtener la misma información por peticiones similares en un momento dado.

● *GARANTIZAR LA CONSISTENCIA DE LOS DATOS*

Por ejemplo, si bajo diferentes llamadas al mismo archivo obtenemos datos distintos significa que tenemos inconsistencia de información, imaginemos que estamos actualizando un archivo de clientes, en el cual la operación consiste de aumentarle el sueldo a un cierto empleado; un usuario ve información distinta a la de otro usuario, él pensará que no se ha aumentado el sueldo y volverá a realizar la operación, lo cual dará como resultado que a ese empleado se le haya aumentado 2 veces el salario.

- Integridad de los datos se refiere a:

  - Las reglas dictadas por políticas o normas de la empresa y que los datos deben cumplir.

Por ejemplo, si en una empresa existe la política de que para un cierto puesto sólo sean contratadas personas del sexo femenino y solteras, tenemos que llevar a cabo dicha validación cada vez que insertemos los datos de ese nuevo empleado y así no tener ese error y permitir que los datos sean capturados.

- Seguridad de los datos es:

  - La protección de los datos contra accesos, modificaciones o pérdidas, ya sea en forma intencional o no intencional.

● *SEGURIDAD DE LOS DATOS*

- Controlar la concurrencia es:

  - Múltiples usuarios pueden accesar a la misma información al mismo tiempo, sin que con ello se tengan problemas con los datos.

Por ejemplo, tenemos nuestro catálogo de clientes, éste está siendo modificado por un usuario, eso no debe de representar ningún problema para que el resto de los usuarios del sistemas puedan consultar ese catálogo de clientes mientras está siendo actualizado.

• Proteger los datos contra fallas del sistema:

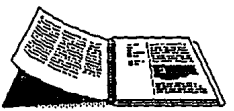
- Es la capacidad de restaurar la integridad y consistencia después de una falla del sistema.

Que sucedería si de repente hay una falta de energía eléctrica y se interrumpen todas las operaciones que se estaban realizando en ese momento, bueno el sistema o la aplicación deberá ser capaz de recuperar la información tal y como se encontraba al último momento antes de la falla, lo cual quiere decir que hasta la última operación que se haya realizado tiene que verse reflejada en los datos.

• El diccionario de datos:

- Es la capacidad que da el manejador de la base de datos de poder tener la descripción de los datos que están almacenados en la base de datos.

● ***DICCIONARIO DE DATOS***



• La interfaz de alto nivel con los programadores es:

- El manejo de un lenguaje de cuarta generación, como lo es SQL.

Generalmente los sistemas manejadores de bases de datos, proveen de un lenguaje que se adapta específicamente al tipo de operaciones que comúnmente se realizan en estos esquemas.

Además un DBMS debe incorporar:

- Independencia de los programas respecto a los cambios en la estructura de los datos.
- Programas de utilería para la administración de la base de datos.
- Mecanismos de seguridad para imponer límites de acceso.
- Facilidades para la afinación (tuning) de la base de datos.
- Capacidad para proceso de transacciones en línea (OLPT).

I.4 Qué es un Diccionario de Datos?

> "Un Diccionario de Datos es una Base de Datos que contiene datos acerca de los datos de la Base de Datos". (1)

Otras definiciones para Diccionario de Datos:

- Es una herramienta para identificar y clasificar los datos almacenados en la base de datos. (5)

- Consiste de archivos, registros y campos que contienen información descriptiva de los datos de la base de datos. Por ejemplo, nos dice cuántas y cuáles son las columnas de la tabla de empleados, además menciona que tipo de datos son válidos para cada columna. (3)

- Es una librería central para definir el significado, uso, características y otros datos relevantes de todas las entidades, sinónimos, referencias cruzadas y relaciones que existen entre ellas. (2)

- No especifica los valores de los datos, sino que define el tipo de valor que debe ir en cada campo. (3)

- Puede consultarse como cualquier conjunto de tablas dentro de la base de datos mediante un lenguaje de consulta sencillo (en un RDBMS). (1)

En resumen, un Diccionario de Datos es una base de datos que contiene datos acerca de los datos de la base de datos.

(1) James Martin, Organización de las Bases de Datos
(2) E.F. Codd, A Relational Model For Large Shared Data Banks
(3) C.J. Date, An Introduction to Database Systems
(5) Henry Korth, Fundamentos de Bases de Datos

I.5 Personajes en el ambiente de Bases de Datos

Dentro del ambiente de bases de datos se han identificado ya, ciertos tipos de roles de personas involucradas en el desarrollo, mantenimiento, diseño y uso de las bases de datos.

Este tipo de personas tiene características y tareas bien específicas, las cuales se dan a continuación:

- Programadores

Son las personas encargadas del desarrollo y programación de las aplicaciones, por lo cual deberá conocer las bases de la tecnología de bases de datos además de que debe poseer conocimientos técnicos y conocer el lenguaje y las herramientas de desarrollo de aplicaciones.

• PROGRAMADORES

- Analistas

Son aquellas personas que interactúan con los usuarios finales, durante las etapas de análisis y diseño de sistemas, sus características son principalmente: conocimiento de teoría de bases de datos, diseño de bases de datos y gran capacidad para interpretar problemas y soluciones posibles en un plano real.

• ANALISTAS

• Soporte Técnico

El personal de soporte técnico es el encargado de asesorar a desarrolladores y administradores sobre el uso correcto y eficiente de la base de datos. Definitivamente, este grupo de trabajo debe tener un panorama amplio de la base de datos y conocimientos profundos del equipo de cómputo.

• SOPORTE TECNICO

• Usuarios Finales

Un usuario final es la persona que requiere hacer uso constante o casual de las aplicaciones desarrolladas en el ambiente de la base de datos. Dentro de este grupo se encuentra el personal directivo o administrativo de su empresa que, mediante alguna aplicación, podrá explotar la información de la base de datos como apoyo en la Toma de Decisiones.

• USUARIOS FINALES

I.6 Quién es un Administrador de la Base de Datos?

LA PERSONA ENCARGADA DE:

El administrador de la base de datos (DBA), es la persona o grupo responsable del control del DBMS. Esto implica:

- Definir el esquema conceptual

En otros términos, decidir qué datos deben estar controlados por el DBMS, es decir, construir el modelo de datos.

- Definir el esquema interno

Crear las estructuras físicas de almacenamiento de forma que aseguren un acceso eficiente a los datos.

- Conocer el negocio

Estar en contacto con los usuarios finales para detectar necesidades de información.

- Definir la estructura de seguridad

Definir propietarios de los datos, otorgar autorización de acceso, establecer procedimientos de auditoría.

- Definir estrategias de respaldo y recuperación

Asegurar la integridad de los datos ante fallas potenciales.

- Monitorear la eficiencia del DBMS

El DBA deberá de asegurar la ejecución eficiente de transacciones y consultas, siguiendo el criterio de lo que "es mejor para la organización".

CAPITULO II. TIPOS DE ENFOQUE DE BASE DE DATOS

En el presente capítulo analizaremos las características de los distintos enfoques: Jerárquico, de Red y Relacional, los cuales nos sirven para modelar e implementar sistemas de información. Además de revisar algunos conceptos ya del modelo relacional. Comenzaremos con el Enfoque Jerárquico:

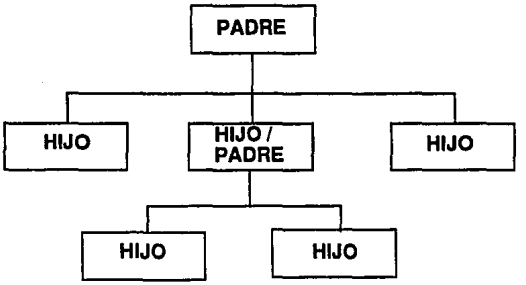
II.1 Enfoque Jerárquico

La estructura lógica en la cual se sustenta la base de datos jerárquica es el árbol. Un árbol se compone de un nodo raíz y varios nodos sucesores, ordenados jerarquicamente. Cada nodo representa una entidad (tipo de registro) y las relaciones entre entidades son las conexiones entre los nodos. El procesamiento es Top-Down, navegacional.

El nodo colocado en la parte superior es llamado padre y los nodos inferiores son los hijos.

En el sistema jerárquico, las conexiones entre archivos no dependen de la información contenida en ellos; se definen al inicio y son fijos.

La característica sobresaliente de este modelo es el manejo de la conexión uno a muchos, entre un padre y varios hijos, en otras palabras, cada hijo sólo tiene un padre.



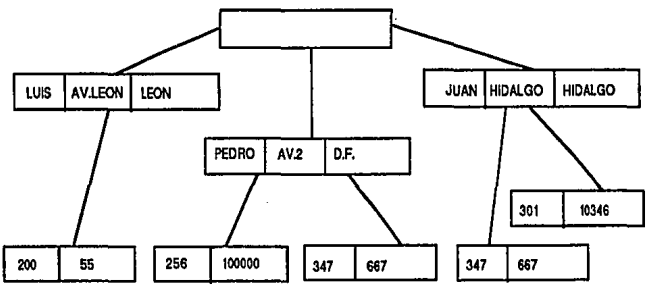
RELACIONES ENTRE REGISTROS (SET) : 1 PADRE, MULTIPLES HIJOS

EJEMPLO DE BASE DE DATOS USANDO EL MODELO JERARQUICO

Tenemos 2 entidades: cuentahabiente y cuenta, relacionadas entre sí con una relación muchos a muchos, es decir, un cuentahabiente puede tener varias cuentas, y una cuenta puede pertenecer a varios cuentahabientes.

El tipo de registro cuentahabiente consiste de 3 campos: nombre, calle y ciudad.

El tipo de registro cuenta tiene 2 campos: número y saldo.



Desventajas en el enfoque de jerárquico:

• No modela sencillamente las relaciones Muchos a Muchos, en lo que más adelante veremos dentro del Modelo Entidad-Relación.

• Anomalías de inserción de registros. Esto es debido, a que hay localizar todos los nodos en los cuales tendría que insertarse información.

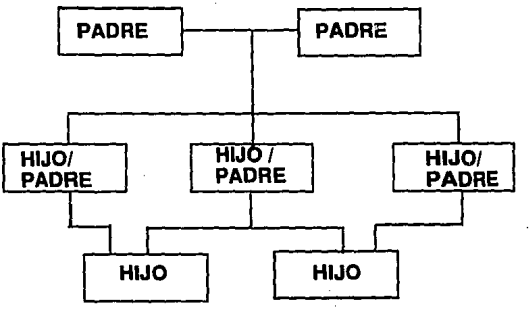
• Anomalías de borrado de registros.Igualmente que en el caso anterior, ya que hay que borra toda la información relacionada para no conservar inconsistencia en los datos

• Anomalías de actualización de registros. Igual que en los 2 casos anteriores.

• Se pueden dar consultas inconsistentes.

• Los datos se representan como registros ligados formando un conjunto de datos intersectados.

• La base de datos de red, a diferencia de las jerárquicas, permite cualquier conexión entre entidades, es decir, se pueden representar relaciones de muchos a muchos. En una red, un hijo puede tener varios padres y varios hijos a la vez.



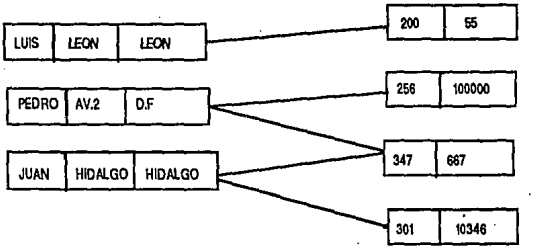
RELACIONES ENTRE REGISTROS (SET) : 1 PADRE - MULTIPLES HIJOS, 1 HIJO - MULTIPLES PADRES

EJEMPLO DE BASE DE DATOS USANDO EL MODELO DE RED

Tenemos 2 entidades: cuentahabiente y cuenta, relacionadas entre sí con una relación muchos a muchos, es decir, un cuentahabiente puede tener varias cuentas, y una cuenta puede pertenecer a varios cuentahabientes.

El tipo de registro cuentahabiente consiste de 3 campos: nombre, calle y ciudad.

El tipo de registro cuenta tiene 2 campos: número y saldo.



Desventajas en el enfoque de red:

- Resulta difícil definir nuevas relaciones.
- Es complicado darle mantenimiento ya que cualquier cambio en la estructura requiere una descarga en los datos.
- Representa mucho desperdicio de recursos.
- Anomalías de inserción.
- Anomalías de borrado.

II.3 Enfoque Relacional

La estructura lógica de una base de datos relacional está basada en la representación de entidades mediante tablas, las cuales constan de columnas(campos) y renglones (registros). Las relaciones entre tablas se llevan a cabo a través de un conjunto de columnas que se tengan en común, logrando una conexión dinámica entre un número ilimitado de ellas mediante el contenido de esas columnas.

La ventaja de los sistemas relacionales es el poder modificar la información sin la preocupación de especificar las combinaciones entre registros.

	COLUMNA 1	COLUMNA 2
RENGLON 1		
RENGLON 2		
RENGLON 3		

SE PUEDEN MODELAR RELACIONES DEL TIPO :
UNO A UNO , UNO A MUCHOS Y MUCHOS A MUCHOS

EJEMPLO DE BASE DE DATOS USANDO EL MODELO RELACIONAL

Tenemos 2 entidades: cuentahabiente y cuenta, relacionadas entre sí con una relación muchos a muchos, es decir, un cuentahabiente puede tener varias cuentas, y una cuenta puede pertenecer a varios cuentahabientes.

La tabla cuentahabiente consiste de 3 columnas: nombre, calle y ciudad.

La tabla cuenta tiene 2 columnas: número y saldo.

CUENTA

NO.CUENTA	SALDO
200	55
256	100 000
347	667
301	10 346

CUENTAHABIENTE

NOMBRE	CALLE	CIUDAD
LUIS	LEON	LEON
PEDRO	AV.2	D.F.
JUAN	HIDALGO	HIDALGO

CUENTA/CUENTAHABIENTE

NO. CUENTA	NOMBRE
200	LUIS
256	PEDRO
347	PEDRO
347	JUAN
301	JUAN

II.4 Sistema Manejador de Base de Datos Relacional

Características:

- Representación de datos a través de tablas.

- Desarrollo de aplicaciones a través de herramientas de alta productividad. (Generadores de Formas, Reportes, Gráficas, etc.)

- Flexibilidad: en el mantenimiento de las estructuras y de los datos, en el tipo de consultas.

- Diccionario de Datos integrado.

- Soporte a todos los operadores relacionales.

Ventajas:

- Simplicidad:

  - Fácil de usar.
  - Fácil obtener respuestas.
  - Fácil insertar y actualizar datos.
  - Fácil cambiar la estructura de los datos.
  - La navegación es responsabilidad del DBMS, NO del programador.

- Poder:

  - Todas las consultas son posibles.

A continuación se nombran distintas marcas de software que han surgido últimamente y que utilizan los distintos modelos de datos comentados anteriormente.

- **SISTEMAS MANEJADORES DE BASE DE DATOS ORIGINALES**

IDMS (RED) CULLINET
IMS (JERARQUICO) IBM
TOTAL (RED) CINCOM

- **SISTEMAS MANEJADORES DE BASE DE DATOS RE-BORNS**

IDMS/R CULLINET
SUPRA CINCOM

- **SISTEMAS MANEJADORES DE BASE DE DATOS DE ORIGEN RELACIONAL**

ORACLE
INGRES
INFORMIX
IDB2
SYBASE

Existen en el mercado sistemas manejadores de base de datos relacionales que originalmente no eran relacionales, como :IDMS de Cullinet que fue originalmente una base de datos en red y ahora se conoce con el nombre de IDMS/R; TOTAL de Cincom originalmente era una base de datos en red y ahora se conoce como SUPRA (relacional).

Sin embargo, existen sistemas manejadores de base de datos que desde su orígen fueron relacionales tal es el caso de: ORACLE, INGRES, DB2, INFORMIX Y SYBASE.

II.6 Reglas de Codd para Bases de Datos Relacionales

En 1985, el Dr. E.F. Codd publicó sus doce reglas (2) para evaluar productos relacionales, las cuales denotan las características principales que debe tener una base de datos relacional, las que son:

1. REGLA DE LA INFORMACION

Toda la información en una base de datos relacional debe ser representada explícitamente, al nivel lógico, en exactamente una manera, por valores en tablas.

2. REGLA DE ACCESO GARANTIZADO

Todos y cada uno de los valores de datos en una base de datos relacional deben ser accesibles logicamente mediante una combinación del nombre de tabla, nombre de columna y valor de la llave primaria.

3. REGLA DE LA INFORMACION FALTANTE

La base de datos relacional deberá ser capaz de manejar información desconocida a través de los llamados valores nulos.

4. DICCIONARIO DE DATOS DINAMICO BASADO EN EL MODELO RELACIONAL

La descripción de la base de datos es representada dinámicamente, al nivel lógico, como datos ordinarios, de tal forma que los usuarios autorizados puedan aplicar el mismo lenguaje relacional para consultarlas.

5. LENGUAJE DE DATOS COMPRENSIBLE

No importa cuantos lenguajes y modos interactivos se soporten, por lo menos un leguaje debe ser soportado, con una sintaxis bien definida, que soporte interactivamente y por programa lo siguiente:

a) Definición de datos
b) Reglas de integridad
c) Manipulación de datos
d) Vistas
e) Control de transacciones
f) Reglas de autorización

(2) E.F. Codd, A Relational Model for Large Shared Data Banks, ACM,Volumen 13, número 6 1970, 377-387 pp.

6. REGLA DE ACTUALIZACION DE VISTAS

Para cada vista el DBMS debe tener una forma de determinar, en el momento de la definicón de la vista, si la vista puede ser utilizada para insertar renglones, borrar renglones, actualizar columnas sobre las tablas en las que está basada, y guardar los resultados de esta decisión en el catálogo del sistema.

7. REGLA DE OPERACIONES DE CONJUNTOS

La capacidad de operar en tablas completas no sólo se aplica a la consulta, sino también a la inserción, modificación y borrado de datos. Las operaciones de conjuntos, independientes de la estructura física de los datos, se logran gracias a un proceso llamado optimización que es único para las bases de datos relacionales.

8. REGLA DE INDEPENDENCIA FISICA DE LOS DATOS

Esto se refiere a una separación, hecha por el DBMS, de los aspectos físicos y lógicos de la base de datos. Las operaciones interactivas y los programas de aplicación no deben ser modificados cuando cambian las estructuras internas de almacenamiento y los métodos de acceso a la base de datos.

9. REGLA DE INDEPENDENCIA LOGICA DE LOS DATOS

Las operaciones interactivas y los programas de aplicación no deben ser modificados cuando se realizan cambios sobre las estructuras de las tablas de la base de datos que no involucren pérdida de información.

10. REGLA DE INDEPENDENCIA DE INTEGRIDAD

Las operaciones interactivas y los programas de aplicación no deben ser modificados cuando se realizan cambios sobre las reglas de integridad definidas y almacenadas en el catálogo del sistema de la base de datos.

11. REGLA DE INDEPENDENCIA DE DISTRIBUCION

Esta regla se refiere a DBMSs distribuídos. El concepto de independencia de distribución es similar a las reglas de independencia física, lógica y de integridad discutidas, pero aplicadas a distribución a través de computadoras.

Esta regla implica que todas las características y reglas requeridas por el modelo deben extenderse a todo el sistema distribuído.

La distribución de datos o de procesamiento en general, y la distribución de DBMS en particular (no son lo mismo!) son tópicos cada vez más populares. Sin embargo, el concepto de un DBMS verdaderamente distribuido no está todavía bien definido, es bastante más complicado, y fuera del alcance de esta discusión.

12. REGLA DE LA NO SUBVERSION

Si un DBMS se maneja con un lenguaje de bajo nivel (procedural), esto no deber de representar el omitir las reglas de integridad y de seguridad que se contemplan con el uso de un lenguaje de alto nivel, y que son almacenadas en el diccionario de la base de datos.

II.7 Qué es la Integridad Referencial?

La integridad referencial es el conjunto de restricciones que se dan a las tablas que forman la base de datos, estas son:

- Llave primaria
- Llave única
- Llave foránea
- Valor nulo
- Valor de default
- Validación de datos

La llave primaria es cuando definimos que para la columna o columnas etiquetadas como tal, no se van a permitir valores nulos, duplicados o cambios sobre todos los renglones almacenados en la(s) columna(s).

La llave única es cuando definimos que para la columna o columnas etiquetadas como tal, no se permiten valores repetidos en los renglones que conforman a la(s) columna(s).

La llave foránea es aquella que definimos sobre una o varias columnas de una tabla, cuya restricción es que siempre deben referenciar a valores existentes en donde esa(s) columna(s) son la llave primaria.

Un valor nulo es una valor desconocido para la(s) columna(s) de una tabla, no es lo mismo que un espacio en blanco o un cero.

Un valor de default es un valor preestablecido para un renglón de una tabla cuando éste es insertado.

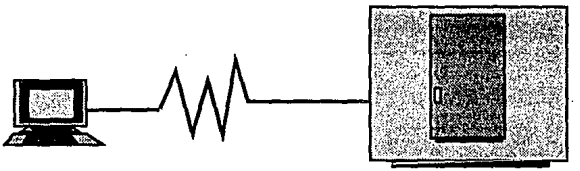
La validación de un dato en un renglón para una cierta columna, puede incluso referenciar a esa misma columna o a ese renglón o a cualquier valor constante. La validación siempre es utilizando operadores de: <,>,=,<>,<= ó >=.

CAPITULO III. BASES DE DATOS DISTRIBUIDAS

Para este capítulo mostraremos un panorama general de las bases de datos distribuídas, las cuales hoy en día son muy utilizadas , ya que nos permiten situarnos en distintos lugares físicamente y no tener ningún problema por manejar la misma información en cualquier momento en que se desee.

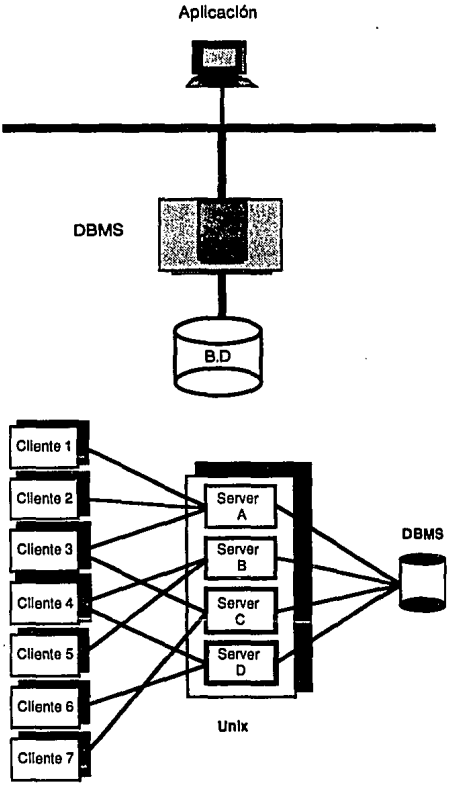
"Las **Bases de Datos Distribuídas** *son una colección de* **Bases de Datos Físicas** *manejadas como una sola* **Base de Datos Lógica".**

III.1 Teleproceso

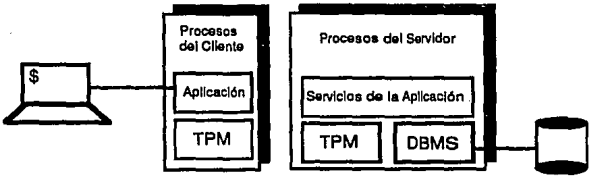


Características:

- La aplicación reside en un nodo remoto.
- El usuario utiliza una terminal con emulación y un modem para conexión.
- Es común utilizar redes públicas. (Ej. TELEPAC).



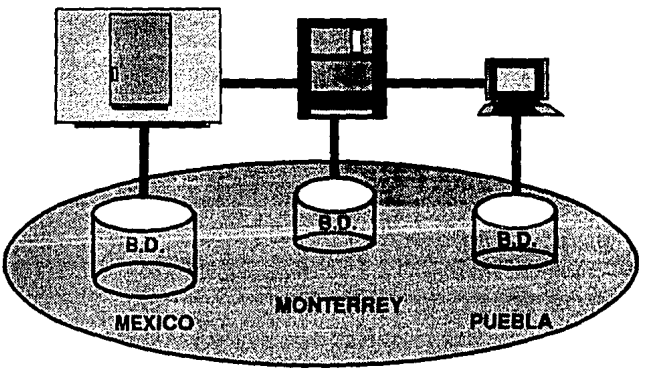
En este tipo de arquitectura se permite que los recursos entre el cliente y el servidor se compartan, hablamos de memoria Ram, de memoria en disco, velocidad de procesamiento de datos, software y muchisimas cosas más.



Características:

- La aplicación reside en un nodo diferente al de la Base de Datos.
- El DBMS es requerido sólo donde la Base de Datos reside.
- La aplicación conoce la localidad de los datos.
- Instrucciones SQL accesan datos en una localidad a la vez.

III.3 Muchas Bases de Datos Físicas = Una Sola Base de Datos Lógica

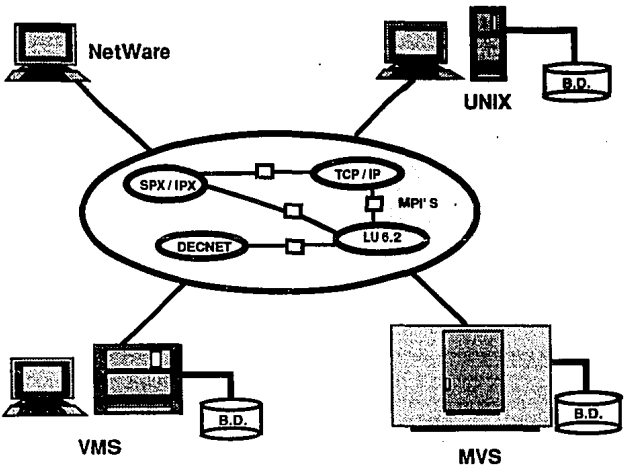


Características:

- Permite ver múltiples Bases de Datos Físicas como una sola Base de Datos Lógica.
- El DBMS se encuentra en cada lugar donde hay una Base de Datos Física.
- Cada DBMS sabe de la localidad de los datos.

Dentro de las organizaciones, es muy frecuente que los usuarios, las fuentes de información y los recursos, en cuanto a equipo se refiere, se encuentren geográficamente distribuídos. Una Base de Datos Distribuída es una red de Bases de Datos Locales almacenadas en múltiples máquinas pero vistas por el usuario como una sola Base de Datos Lógica almacenada en una sola localidad.

III.4 Sistema Manejador de Bases de Datos Distribuídas



Características:

- Colección de DBMSs que trabajan en cooperación para ofrecer una sola base de datos lógica.

- Mecanismos de comunicación y control distribuído son transparentes para el usuario. Se debe manejar multi-protocolo y multi-ruteo.

- DBMSs con estrategias similares para la administración y distribución de datos.

En resumen las Bases de Datos Distribuídas permiten que:

"Cada máquina de la red posea capacidad de *procesamiento autónomo* y pueda efectuar aplicaciones locales. Cada máquina participa también en la ejecución de cuando menos una **aplicación global**, que requiere accesar datos de varias máquinas por medio de un **subsistema de comunicaciones**".

CAPITULO IV. OPERADORES RELACIONALES

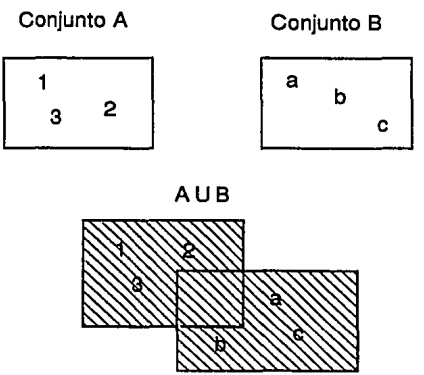
El interés actual en el enfoque relacional se debe en gran medida al trabajo del Dr. E.F. Codd, quien en junio de 1970 publicó el artículo: " A Relational Model of Data for Large Shared Banks". (CACM 13 Num. 6). (2)

A continuación presentaremos un resumen de la teoría matemática involucrada en el modelo relacional del Dr. Codd.

IV.1 Unión

- UNION

El operador de UNION acepta como entrada dos tablas con las mismas columnas en el mismo orden, y produce como resultado todas las columnas y todos los renglones de ambas tablas. Si existe algún renglón con la misma información en ambas tablas, en la tabla que se genera de aplicar el operador de UNION ese renglón sólo aparece una vez.



(2) E.F. Codd, A Relational Model For Large Shared Data Banks, ACM Volumen 13, número 6, 1970, 377- 387 pp.

EJEMPLO:

Tenemos las siguientes tablas,

EMPLEADOS ANTIGUOS

NUMERO DE EMPLEADO	NOMBRE DEL EMPLEADO	SALARIO
1	PEDRO	12,000
2	LUIS	
3	FRANCISCO	36,000

EMPLEADOS NUEVOS

NUMERO DE EMPLEADO	NOMBRE DEL EMPLEADO	SALARIO
3	FRANCISCO	36,000
4	LORENA	24,000
5	GABRIELA	24,000

la UNION dará como resultado,

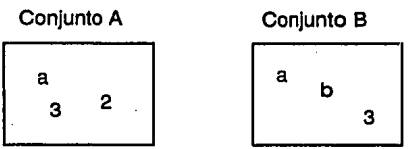
RESULTADO

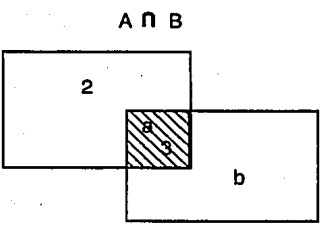
NUMERO DE EMPLEADO	NOMBRE DEL EMPLEADO	SALARIO
1	PEDRO	12,000
2	LUIS	
3	FRANCISCO	36,000
4	LORENA	24,000
5	GABRIELA	24,000

IV.2 Intersección

- INTERSECCION

El operador de INTERSECCION selecciona de ambas tablas los renglones que tengan exactamente la misma información en todas las columnas.





EJEMPLO:

Tenemos las siguientes tablas,

EMPLEADOS ANTIGUOS

NUMERO DE EMPLEADO	NOMBRE DEL EMPLEADO	SALARIO
1	PEDRO	12,000
2	LUIS	
3	FRANCISCO	36,000

EMPLEADOS NUEVOS

NUMERO DE EMPLEADO	NOMBRE DEL EMPLEADO	SALARIO
3	FRANCISCO	36,000
4	LORENA	24,000
5	GABRIELA	24,000

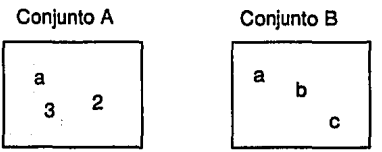
la INTERSECCION dará como resultado,

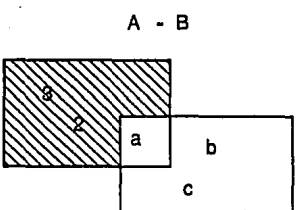
RESULTADO

NUMERO DE EMPLEADO	NOMBRE DEL EMPLEADO	SALARIO
3	FRANCISCO	36,000

- DIFERENCIA

El operador de DIFERENCIA acepta como entrada dos tablas que tengan al menos una columna en común, en donde la tabla resultante tendrá todas las columnas de la primer tabla y los renglones que no aparezcan en la segunda tabla.





EJEMPLO:

Tenemos las siguientes tablas,

DEPARTAMENTO

COD. DEPART.	NOMBRE DEL DEPARTAMENTO
VE	VENTAS
NO	NOMINA
IN	INVESTIGACION
ME	MERCADOTECNIA
RE	RESULTADOS

EMPLEADO

NUMERO DE EMPLEADO	NOMBRE DEL EMPLEADO	SALARIO	COD. DEPART.
1	PEDRO	12,000	VE
2	LUIS		NO
3	FRANCISCO	36,000	
4	LORENA	24,000	
5	GABRIELA	24,000	NO

la DIFERENCIA dará como resultado,

RESULTADO

COD. DEPART.	NOMBRE DEL DEPARTAMENTO
IN	INVESTIGACION
ME	MERCADOTECNIA
RE	RESULTADOS

- PROYECCION

El operador de PROYECCION tiene como entrada una tabla, y produce como resultado sólo aquellas columnas especificadas por el usuario.

El orden en el cual aparecen las columnas, es el que se indica cuando se hace la proyección.

El número de columnas que se pueden proyectar es como máximo el mismo número de columnas de la tabla y como mínimo una sola columna.

EJEMPLO:

Tenemos la siguiente tabla,

EMPLEADO

NUMERO DE EMPLEADO	NOMBRE DEL EMPLEADO	SALARIO	COD. DEPART.
1	PEDRO	12,000	VE
2	LUIS		NO
3	FRANCISCO	36,000	
4	LORENA	24,000	
5	GABRIELA	24,000	NO

La PROYECCION de las columnas Nombre y Número de empleado, muestran el siguiente resultado,

RESULTADO

NOMBRE DEL EMPLEADO	NUMERO DE EMPLEADO
PEDRO	1
LUIS	2
FRANCISCO	3
LORENA	4
GABRIELA	5

IV.5 Selección

- SELECCION

El operador de SELECCION acepta una sola tabla como entrada, y produce como resultado las mismas columnas que contiene la tabla de entrada y los renglones que sean especificados por el usuario.

Las condiciones de selección de renglones pueden ser de varios grados de complejidad y pueden incluir a los operadores booleanos AND, OR y NOT (se pueden utilizar paréntesis para indicar precedencia de operación).

Las comparaciones pueden realizarse con valores literales, valores contenidos en las columnas, o expresiones matemáticas que involucren valores literales de las columnas.

EJEMPLO:

Tenemos la siguiente tabla,

EMPLEADO

NUMERO DE EMPLEADO	NOMBRE DEL EMPLEADO	SALARIO	COD. DEPART.
1	PEDRO	12,000	VE
2	LUIS		NO
3	FRANCISCO	36,000	
4	LORENA	24,000	
5	GABRIELA	24,000	NO

La SELECCION de todos los empleados que trabajan en el departamento de nomina muestra el siguiente resultado,

RESULTADO

NUMERO DE EMPLEADO	NOMBRE DEL EMPLEADO	SALARIO	COD. DEPART.
2	LUIS		NO
5	GABRIELA	24,000	NO

IV.6 Join

- JOIN

El operador de JOIN acepta como entrada dos o más tablas, teniendo cada una al menos una columna en común con las otras tablas, y produce como resultado a todas las columnas de las tablas de entrada, y los renglones se concatenan con aquellos renglones cuyos valores en las tablas de entrada cumplen la condición que indica el usuario para hacer el join.

Los operadores relacionales para indicar las condiciones de join pueden ser >, <, =, !=. Las columnas en común sólo se muestran una vez.

EJEMPLO:

Tenemos las siguientes tablas,

DEPARTAMENTO

COD. DEPART.	NOMBRE DEL DEPARTAMENTO
VE	VENTAS
NO	NOMINA
IN	INVESTIGACION
ME	MERCADOTECNIA
RE	RESULTADOS

EMPLEADO

NUMERO DE EMPLEADO	NOMBRE DEL EMPLEADO	SALARIO	COD. DEPART.
1	PEDRO	12,000	VE
2	LUIS		NO
3	FRANCISCO	36,000	
4	LORENA	24,000	
5	GABRIELA	24,000	NO

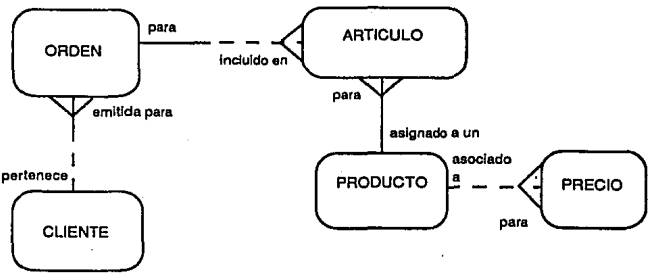
el JOIN con la columna en común del Código de Departamento muestra el siguiente resultado,

RESULTADO

COD. DEPART.	NOMBRE DEL DEPARTAMENTO	NUMERO DE EMPLEADO	NOMBRE DEL EMPLEADO	SALARIO
VE	VENTAS	1	PEDRO	12,000
NO	NOMINA	2	LUIS	
NO	NOMINA	5	GABRIELA	24,000

CAPITULO V. MODELO ENTIDAD-RELACION

En este capítulo aprenderemos una técnica para modelar bases de datos relacionales, la cual involucra diferentes conceptos como: Entidad, Relación y Atributo, en sus distintas variantes.

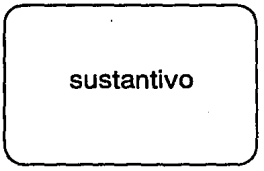


El Modelo Entidad-Relación es una técnica para definir las necesidades de información de cualquier empresa. Esta técnica involucra conceptos que se identifican con varios objetos de importancia para la empresa, a los cuales se les denomina ENTIDADES, a las características de dichos objetos se les denomina ATRIBUTOS y a cómo se relacionan estos objetos entre sí se le denomina RELACIONES.

Todos estos conceptos se modelan a través de cierto tipo de esquemas gráficos, los cuales muestran a los usuarios una manera más sencilla y práctica de visualizar sus necesidades de información.

ENTIDAD

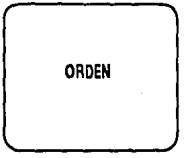
Una entidad es una persona, cosa o lugar, que cae dentro del alcance del sistema, acerca de la cual el sistema debe mantener, correlacionar y desplegar información.



La ENTIDAD se representa por medio de una caja con las esquinas redondeadas y dentro de ésta se escribe el nombre de la entidad, el cual debe estar en sigular.

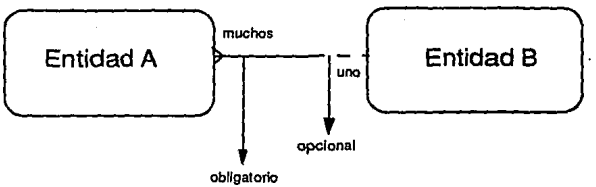
Cada ENTIDAD debe tener un nombre único dentro del sistema, lo que indica que no pueden existir dos entidades dentro del mismo sistema con el mismo nombre.

EJEMPLO:



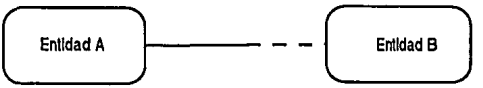
RELACION

Una relación requiere de una o más entidades, la cual debe caer dentro del alcance del sistema, acerca de la cual el sistema debe mantener, correlacionar y desplegar información.

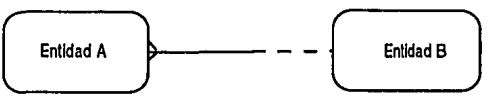


Las relaciones se representan en varias modalidades:

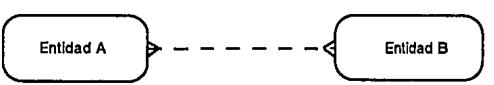
• **Relaciones Uno a Uno** con las combinaciones de obligatorio a opcional, opcional a opcional y obligatorio a obligatorio.



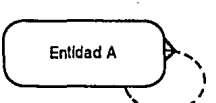
• **Relaciones Uno a Muchos** con las combinaciones de obligatorio a opcional, opcional a opcional, obligatorio a obligatorio y opcional a obligatorio.



• **Relaciones Muchos a Muchos** con las combinaciones de opcional a opcional, obligatorio a opcional y obligatorio a obligatorio.



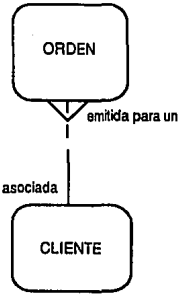
• Relaciones Recursivas con las combinaciones de muchos a uno opcional a opcional y de uno a uno opcional a opcional.



La manera formal en que se deben leer las relaciones es:

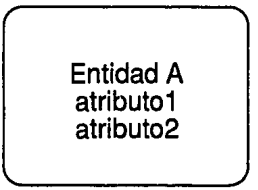
" Cada elemento de la Entidad A debe o puede relacionarse con uno y sólo un o (muchos) elemento (s) de la Entidad B".

EJEMPLO:



ATRIBUTO

Un atributo es una característica o cualidad de una entidad o relación, que cae dentro del alcance del sistema, acerca del cual el sistema debe mantener, correlacionar y desplegar información.



Para representar uno o varios atributos, se escribe el nombre del atributo dentro de la entidad.

Indica que el atributo es llave primaria (PK).
• Indica que el atributo es obligatorio.

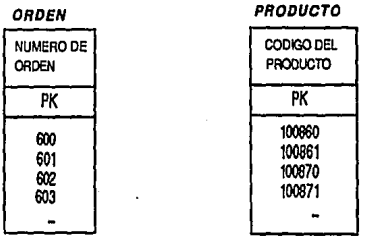
CAPITULO VI. MODELO RELACIONAL

A continuación mostraremos otra técnica para modelar bases de datos relacionales, dicha técnica es muy utilizada y además puede funcionar muy bien no importa si hablamos de un sistema de información grande o de uno pequeño.

VI.1 Modelado de Entidades

La ENTIDAD dentro del modelo relacional se representa por medio de una tabla, donde el nombre de la tabla corresponde al nombre de la entidad. Además, cada tabla debe contener una columna que identifique de forma única a cada renglón de ésta. Esta columna recibe el nombre de llave primaria (PK), la cual no puede contener valores nulos, ni duplicados.

EJEMPLO:



VI.2 Modelado de Relaciones

Las RELACIONES dentro del modelo relacional se representan de la siguiente forma:

• Relación Uno a Uno entre dos entidades se modela dibujando la llave primaria (PK) de una de las tablas como llave foránea (FK) en la otra tabla.
Se sugiere que se dibuje la llave primaria (PK) de la tabla con más renglones como llave foránea (FK) en la otra tabla.

• Una llave foránea es una o más columnas que son llave primaria en otra tabla, una llave foránea permite nulos y valores duplicados.

EJEMPLO:

PRODUCTO

NUMERO DE PRODUCTO
PK
100860
100861
100870
100871
--

PRECIO

CLAVE DE PRECIO	NUMERO DE PRODUCTO
PK	FK, ND
1	100870
2	100876
3	100101
4	100860
--	--

Nota: ND significa que sobre la columna no se permiten duplicados.

• Relación Uno a Muchos o Muchos a Uno entre dos entidades se modela dibujando la llave primaria (PK) de la tabla que tiene la correspondencia de uno como llave foránea (FK) en la otra tabla.

EJEMPLO:

CLIENTE

NUMERO DE CLIENTE
PK
100
101
102
103
--

ORDEN

NUMERO DE ORDEN	NUMERO DE CLIENTE
PK	FK
600	101
601	101
602	102
603	
604	103
--	--

• Relación Muchos a Muchos entre dos entidades se modela dibujando una tercer tabla, la cual se compone de una llave primaria (PK) compuesta de dos columnas, las cuales son llaves primarias y foráneas a la vez.

EJEMPLO:

ORDEN

NUMERO DE ORDEN
PK
600 601 602 603 ...

PRODUCTO

CODIGO DEL PRODUCTO
PK
100860 100861 100870 100872 ...

ORDEN / PRODUCTO

NUMERO DE ORDEN	CODIGO DEL PRODUCTO
PK	
FK	FK
600 603 600 602 ...	100860 100860 100872 101860 ...

VI.3 Modelado de Atributos

Los ATRIBUTOS dentro del modelo relacional se representan por medio de columnas dentro de una tabla.

EJEMPLO:

PRODUCTO

CODIGO DEL PRODUCTO	DESCRIPCION
PK	
100860	Tenis
100861	Raquetas I
100870	Raquetas II
100871	Pans
--	--

CAPITULO VII. INTRODUCCION A SQL

En este capítulo presentaremos los comandos básicos de SQL (Structured Query Language).

VII.1 Comandos de Recuperación de Datos

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

La cláusula FROM es una lista de las tablas que se van a examinar durante la ejecución de la expresión.

La claúsula WHERE corresponde al predicado de selección del álgebra relacional. Se compone de un predicado que incluye atributos de las relaciones que aparecen en la claúsula FROM.

EJEMPLOS:

Tenemos las siguientes tablas,

ORDEN

# ORDEN	FECHA EMI.	FECHA ENV.	TOTAL	# CLIENTE	TIPO COMM.
600	21-MAY-92	27-MAY-92	2400	100	A
601	22-MAY-92	26-MAY-92	500	103	C
602	2-JUN-92	11-JUN-92	1100	100	B
603	4-JUN-92	6-JUN-92	5560	103	A

ARTICULO

# ORDEN	# PRODUCTO	CANTIDAD	TOTAL	PRECIO ACTUAL	# ARTICULO
600	100860	2	550	275	1
600	100872	1	300	300	2
601	100890	1	640	320	1
601	102345	3	330	110	2
602	103500	4	400	100	3
602	100860	2	250	125	1
603	101700	1	100	100	1

Ejemplos:

- Obtener las órdenes de la tabla de Orden

```
SELECT   No_orden
FROM     Orden
```

- Obtener las fechas de orden y las fechas de envío de las órdenes

```
SELECT   Fecha_emision, Fecha_envio
FROM     Orden
```

- Obtener toda la información de las órdenes del cliente 100

```
SELECT   *
FROM     Orden
WHERE    No_cliente = 100
```

- Obtener el total de las órdenes cuyo plan de comisión es A

```
SELECT   Total
FROM     Orden
WHERE    Tipo_comm = 'A'
```

- Obtener los números de producto y la cantidad de cada producto en cada orden

```
SELECT   No_articulo, Cantidad
FROM     Orden, Articulo
WHERE    Orden.No_orden = Articulo.No_orden
```

- Obtener el precio actual de los artículos cuya cantidad es mayor que 1

```
SELECT   Precio_actual
FROM     Articulo
WHERE    Cantidad > 1
```

- Obtener todos los números de orden en las que el total por artículo es mayor que 100

```
SELECT   No_orden
FROM     Articulo
WHERE    Total > 100
```

VII.2 Comandos de Manipulación de Datos

Los comandos de manipulación de datos (DML) son:

- Altas : **INSERT**
- Bajas : **DELETE**
- Cambios : **UPDATE**

La claúsula de INSERT trabaja junto con la claúsula de INTO y la claúsula de VALUES y sirve para poder datos a una tabla.

La claúsula de DELETE trabaja junto con la claúsula de FROM y sirve para poder borrar renglones de una tabla, los borrados son de todos los valores almacenados en las columnas para un determinado renglón, lo cual indica que no existen los borrados parciales. Es muy importante la utilización de la claúsula de WHERE, ya que si se omite provoca que todos los renglones de la tabla sean borrados.

La claúsula de UPDATE trabaja junto con la claúsula de SET y la claúsula de WHERE y sirve para actualizar la información almacenada en la tabla.

EJEMPLOS:

- Insertar en la tabla de orden una orden nueva

```
INSERT INTO  Orden
VALUES ( 604, '17-JUN-93','20-JUN-93', 2000, 101, 'A')
```

- Insertar en la tabla de artículo un artículo con datos No_orden, No_producto y el total

```
INSERT INTO Articulo (No_orden, No_producto, Total)
VALUES (604, 100860, 2000)
```

- Borrar de la tabla de Orden a la orden 603

```
DELETE FROM  Orden
WHERE No_orden = 603
```

- Borrar de la tabla de artículo a la información correspondiente a las órdenes cuyo total sea menor de 300

```
DELETE FROM  Artículo
WHERE Total < 300
```

- Actualizar los datos de la orden 605, ya que el total debe ser 2500

```
UPDATE   Orden
SET      Total = 2500
WHERE    No_orden = 605
```

- Actualizar todas las órdenes cuyo No_producto haya sido 100860 y poner el nuevo número que es 100760

```
UPDATE   Articulo
SET      No_producto = 100760
WHERE    No_producto = 100860
```

VII.3 Comandos de Definición de Datos

Los comandos de definición de datos (DDL) son:

- Creación de Objetos : **CREATE**
- Modificación de Objetos : **ALTER**
- Borrado de Objetos : **DROP**

La claúsula de CREATE trabaja junto con la claúsula de TABLE, con la claúsula de VIEW, con la claúsula de INDEX, y con algunas otras; y sirven para crear objetos en la base de datos.

La claúsula de ALTER trabaja junto con la claúsula de TABLE, con la claúsula de VIEW, con la claúsula de INDEX, y con algunas otras; y sirven para alterar la definición de dichos objetos en la base de datos.

La claúsula de DROP trabaja junto con la claúsula de TABLE, con la claúsula de VIEW, con la claúsula de INDEX, y con algunas otras; y sirven para borrar los objetos de la base de datos.

EJEMPLOS:

- Crear una tabla que sirva para manejar una agenda

```
CREATE TABLE   Agenda
( Nombre            Char(30),
  Direccion         Char(60),
  Telefono_casa     Number(11),
  Telefono_oficina  Char(20),
  Fecha_nacimi      Date,
  Tipo_de_relacion  Char(20))
```

- Alterar la definición de la tabla Agenda y agregar una columna que sirva para almacenar comentarios

```
ALTER TABLE   Agenda
ADD ( Comentarios   Char(40))
```

- Borrar la tabla Agenda

```
DROP TABLE   Agenda
```

- Crear una vista de la tabla Agenda que muestre únicamente el nombre y el telefono

```
CREATE VIEW   Directorio  AS
SELECT  Nombre, Telefono
FROM    Agenda
```

- Crear un índice a la tabla de Agenda para la columna de Nombre

```
CREATE INDEX  uno
ON   Agenda (Nombre)
```

VII.4 Comandos de Seguridad de los Datos

Los comandos de seguridad de los datos son:

- Otorgamiento de Privilegios : **GRANT**
- Eliminación de Privilegios : **REVOKE**

La claúsula GRANT dependiendo de los privilegios que se quieran otorgar puede trabajar a nivel de sistema, de tablas o de vistas.

La claúsula de REVOKE sirve para eliminar cualquier tipo de privilegio.

EJEMPLOS:

- Dar privilegios a el usuario Scott con password Tiger de poder conectarse a la base de datos

```
GRANT CONNECT TO Scott
IDENTIFIED BY Tiger
```

- Dar privilegios de poder conectarse y crear objetos en la base de datos al usuario Uno con password X

```
GRANT CONNECT, RESOURCE TO Uno
IDENTIFIED BY X
```

- Dar privilegios de administrador de base de datos (todos lo privilegios) al usuario Dos con password Y

```
GRANT CONNECT, RESOURCE, DBA TO Dos
IDENTIFIED BY Y
```

- Dar privilegios de seleccionar y actualizar la información de la tabla Agenda al usuario Scott

```
GRANT SELECT, UPDATE
ON Agenda
TO Scott
```

- Dar todos los privilegios sobre la vista Directorio al usuario Uno

```
GRANT ALL
ON Directorio
TO Uno
```

- Eliminar el privilegio de conexión a la base de datos del usuario Scott

```
REVOKE CONNECT
FROM Scott
```

- Eliminar el privilegio de actualizar información de la tabla de Agenda al usuario Scott

```
REVOKE UPDATE
ON Agenda
FROM Scott
```

- Eliminar los privilegios al usuario Uno sobre la vista del Directorio

```
REVOKE ALL
ON Directorio
FROM Uno
```

CAPITULO VIII. TENDENCIAS FUTURAS EN EL PROCESAMIENTO DE DATOS

VIII.1 Productos en el Mercado

La década de los 70s fue la era de la computación homogénea, una época cuando un mismo proveedor ofrecía el 90% de la solución, el 90% del tiempo. Los 80s cambiaron la naturaleza de la computación con la introducción de nuevas tecnologías: la computadora personal. Esto culminó en los 90s con la tendencia "*downsizing*" de mainframes a estos nuevos ambientes de cómputo. De esta forma, la computación se volvió heterogénea teniendo diferentes marcas y tamaños de equipo de cómputo en una misma organización.

Hoy en día, las corporaciones están invirtiendo en una gran variedad de PCs y estaciones de trabajo (clientes), redes, y servidores para comenzar la migración de sus sistemas de información anteriores a los nuevos sistemas abiertos cliente/servidor.

La era del "*downsizing*" es una realidad en la mayoría de las corporaciones y ciertamente ha comenzado la era de la computación heterogénea y distribuída. La migración de sistemas propietarios a sistemas abiertos se ha acelerado a medida que las nuevas tecnologías emergen.

Computación Cliente / Servidor

La computación cliente/servidor, que utiliza redes de computadoras para ofrecer mejor rendimiento y para enfatizar la integridad de la información, es una nueva arquitectura para el manejo de datos basado en sistemas abiertos. Los clientes se comunican con el servidor solicitando y recibiendo datos a través de un API (Application Programming Interface) soportado por el servidor. La característica principal de esta arquitectura es la división del trabajo. El cliente se concentra en la interacción con el usuario desplegando formas, produciendo reportes y soportando interfaces gráficas. El servidor se concentra en acceso compartido a los datos por múltiples clientes forzando la integridad de los datos ejecutando eficientemente ls requisiciones de los clientes.

En tanto que la computación cliente/servidor se vuelve la característica dominante de mediados de los 90s, una gran variedad de productos en el mercado permiten a las corporaciones sacar el mayor provecho a su inversión en este nuevo ambiente. Nuevas y poderosas herramientas de desarrollo de aplicaciones como Visual Basic, Excel, HyperCard, Lotus 1-2-3 y muchas más proveen un ambiente de desarrollo de alto nivel que incrementa notablemente la productividad con respecto a los lenguajes tradicionales de 3a. generación y hasta de 4a. generación.

VIII.2 Manejo de Multimedia

Desarrollo de Aplicaciones con Nuevas Herramientas y Nuevos Datos

Para entender completamente los aspectos de desarrollo de aplicaciones a los que se enfrentan los desarrolladores en estos días, es importante dar un paso atrás y revisar la forma como era hecho el desarrollo de aplicaciones en el pasado. Tradicionalmente, los desarrolladores de aplicaciones construían aplicaciones utilizando lenguajes de 3a. generación como COBOL o C, o herramientas de 4GL como Focus. Ellos ligaban la aplicación a la fuente de datos, ya sea una base de datos o un archivo plano, utilizando un precompilador (como el Pro*C de Oracle) o una interfaz a nivel llamada (como el OCI).

Mientras que la mayoría de los desarrollos continúan en este esquema tradicional de 3GL y 4GL, los desarrolladores están comenzando a utilizar una nueva generación de herramientas. Estas proveen desarrollo de aplicaciones rápido ya que permiten al desarrollador trabajar en un nivel de abstracción arriba del sistema operativo. Esencialmente, esta herramienta sirve como el complemento al ambiente de desarrollo.

Estas nuevas herramientas proveen un incremento dramático en la productividad especialmente en el ambiente gráfico. Sin éstas, el desarrollador se vería forzado a dominar los detalles de la construcción y el manejo de ventanas, desarrollo de interfaces amigables para los usuarios o manejo de memoria. Productos como Visual Basic de Microsoft, HyperCard de Apple, Oracle Card de Oracle o Pen Apps de Slates han sido reconocidos como productos de alta productividad en el mundo de los GUIs en un tiempo de pocos recursos de este tipo.

VIII.3 Programación Orientada a Objetos

Hoy en día el término Orientado a Objetos ha rebasado su ámbito de orígen en el área de lenguajes de programación y es usado de manera extensiva como sinónimo de lo novedoso, lo bueno y lo interesante. El mercado está inundado de estos productos, todos ellos con fuertes campañas publicitarias que son anunciados como Orientado a Objetos.

La programación Orientada a Objetos abre una nueva perspectiva al desarrollo de sistemas. Es sin duda el paradigma de los 90s, su ámbito de aplicación es abierto y abarca en la actualidad áreas tales como bases de datos, programación distribuída, redes, sistemas abiertos y los sistemas heterogéneos basados en objetos. Los lenguajes de programación y en general los ambientes de desarrollo están siendo extendidos hacia la Programación Orientada a Objetos.

El futuro de las bases de datos está orientada a objetos, un ejemplo de ello es la versión Oracle 8, que saldrá al mercado en diciembre de 1994, la cual utilizará una versión de Structured Query Language (SQL) que se denomina "SQL++", se romperá con ello la limitación actual de las bases de datos relacionales bidimensionales al crear un producto capaz de almacenar datos como objetos en vez de tablas.

Con Oracle 8, se usará SQL++ para añadir un nivel esquemático arriba de la base de datos para que los documentos puedan ser almacenados y examinados mientras aparecen, lo cual los hará más familiares para los usuarios finales. Las bases de datos relacionales solamente pueden contemplar abstracciones de datos. Los usuarios finales conocen las facturas, órdenes de compra y reportes de ventas, pero no conocen las tablas bidimensionales.

Con este nivel esquemático, los usuarios podrán hojear, accesar e imprimir documentos tal y como aparecen en la vida real. También porque Oracle 8 añadirá interfaces de programación de aplicaciones orientadas a objetos (API) a Oracle 8 las cuales serán abstraídas a un nivel más alto que los APIs actuales, las aplicaciones serán más fáciles de escribir en las futuras bases de datos.

CONCLUSIONES

El resultado obtenido de la realización del presente trabajo es tener un material de apoyo didáctico a los estudiantes de materias relacionadas con Bases de Datos, además de ser un compendio de los puntos más relevantes dentro de este interesante tema.

Cabe mencionar que el crecimiento acelerado de las necesidades de obtener más información dentro de las empresas ha obligado a que se realicen e implementen más y nuevas técnicas y procedimientos que satisfagan dichos requerimientos, razón por la cual siempre tendremos temas de estudio relacionados con lo expuesto en este trabajo.

Una solución práctica y muy precisa a esta problemática es la implementación de sistemas que usen bases de datos relacionales, así como la integración de sistemas que incluyan voz, datos e imágenes en la misma aplicación, cuyo desempeño sea completamente eficaz y confiable.

ESTA TESIS NO DEBE
SALIR DE LA BIBLIOTECA

BIBLIOGRAFIA.

1. MARTIN, James
   Organización de las Bases de Datos
   1a. ed.
   Ed. Prentice Hall, México, 1975
   544 pp.

2. CODD, E.F.
   A Relational Model For Large Shared Data Banks
   Communications of the ACM, Volumen 13, Número 6, 1970
   377-387 pp.

3. DATE, C.J.
   An Introduction to Databse Systems
   Volumen II, Addison-Wesley, Massachussets, 1983
   756 pp.

4. BARKER, Richard
   CASE*Method Entity Relationship Modelling
   1a. ed.
   United Kingdom, Addison-Wesley, 1990
   402 pp.

5. KORTH, Henry , SIBERSHATZ, Abraham
   Fundamentos de Bases de Datos
   1a. ed.
   Ed. Mc Graw Hill, México, 1987
   525 pp.