Club de Investigación Tecnológica

Bases de Datos

Preparado por: Carlos A. González A.
Diciembre, 1989

Editado y publicado por Rho-Sigma S.A.,
a nombre del Club de Investigación Tecnológica.

San José, Costa Rica

Diciembre, 1989

Resumen Ejecutivo

Los gauchos argentinos poseen al menos diez diferentes términos para referirse a la pampa. Por su parte, los esquimales cuentan con cerca de treinta términos distintos cuando hablan de la nieve. Esto se debe a la importancia que representa para ellos el medio ambiente en que se desenvuelven.

En el campo de la computación, muchas personas que utilizan herramientas para almacenar, manipular y recuperar datos, las llaman, sin serlo, **Sistemas Administradores de Bases de Datos (SABD)**. Además, no tienen una idea clara de lo que representa el concepto de **Bases de Datos**.

El presente informe persigue en parte aclarar las dudas que se puedan tener sobre estos términos.

En el primer capítulo se introduce al lector en las definiciones y conceptos básicos de lo que se ha dado en llamar **Tecnología de Bases de Datos**. Este capítulo se inicia con un recorrido histórico de esta tecnología

En el segundo capítulo se estudiará el modelo de datos relacional, herramienta imprescindible que permite un traslado fiel de la información necesaria en un sistema de información de una empresa, al mundo de los datos almacenados en un computador. Asimismo, se introduce la metodología que se sigue en este modelo para el diseño de las Bases de Datos, es decir, el proceso de **normalización.**

En el tercer capítulo, se introducen dos modelos de datos más, el de redes y el jerárquico. Estos modelos no son actualmente utilizados para desarrollar SABD comerciales. Sin embargo, su importancia radica en la existencia de muchas Bases de Datos de basadas en estos modelos.

En el capítulo IV se hace un estudio sobre los productos comercializados en el campo de los SABD. Se escogió como ejemplo el estándar **SQL.**

En el capítulo V, se darán algunas de las perspectivas sobre la **tercera generación de Bases de Datos**.

Los Sistemas de Bases de Datos de esta generación se caracterizarán, no sólo por la manipulación de datos estructurados (registros), sino que además podrán manipular gráficos, mapas, sonidos, textos, etc.

En este caso, como se verá posteriormente, entran a jugar otras ramas de la computación como son: las redes de computadores, procesamientos en paralelo, gráficos por computador, reconocimiento de la voz, inteligencia artificial, etc.

Finalmente, el informe se concluye con algunas consideraciones a tomar en cuenta y se dan recomendaciones, así como ventajas y deventajas del procesamiento de los datos en el ambiente de Bases de Datos.

También, se incluye un anexo sobre el proceso de normalización, para aquellas personas que deseen profundizar en esta teoría.

Agradecimientos

Se agradece la valiosa colaboración del Dr. Claudio Gutiérrez y del Dr. Roberto Sasso, quienes hicieron innombrables observaciones

que contribuyeron a mejorar en forma sustancial la calidad de la presente investigación. También se agradece la colaboración del Lic. Jorge Walter Bolaños, quién ofreció valiosos comentarios sobre la versión final.

Del autor

Carlos A. González A. es doctor en Informática por la Universidad de París. Actualmente ocupa el cargo de Director del Centro de Investigaciones en Computación y del Programa de Maestría en Computación del Instituto Tecnológico de Costa Rica.

Contenido Página

I Funciones de los Sistemas de Bases de Datos

1. Introducción.c2.

En 1945 Vannebar Bush, ingeniero eléctrico del MIT (Instituto Tecnológico de Massachussets) publica un artículo titulado "**As we may think**" **[Bush45]**. En este artículo se describe la mayoría de los requisitos que debía cumplir lo que posteriormente se daría en llamar Base de Datos.

Sin embargo, no fue sino hasta 1964 que el término **Base de Datos**, apareció por vez primera en una conferencia, en Santa Mónica, California, y que fue organizada por System Development Corporation. El título de la misma fue: "**Development and management of a computer-centered data base**".
A partir de este momento, el desarrollo de lo que hoy se conoce como **Tecnología de Bases de Datos** ha sido bastante acelerado.

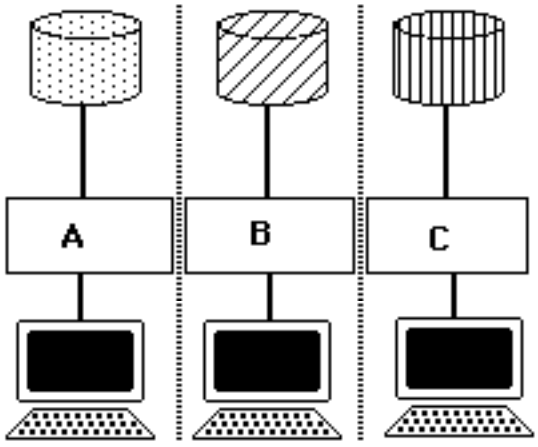
Así por ejemplo, el número de artículos publicados anualmente en revistas de relevancia mundial ha superado la cifra de los 2,000. Esto demuestra que el campo de las Bases de Datos se ha convertido en una parte fundamental de la Computación y por ende de muchas actividades humanas, entre otras, la agricultura, la medicina, la educación, las ingenierías, la productividad industrial y la banca.

El desarrollo de las Bases de Datos, que ha involucrado una serie de métodos y de técnicas para la concepción, la implantación y la manipulación de grandes volúmenes de información estructurada, ha seguido un patrón bastante regular.

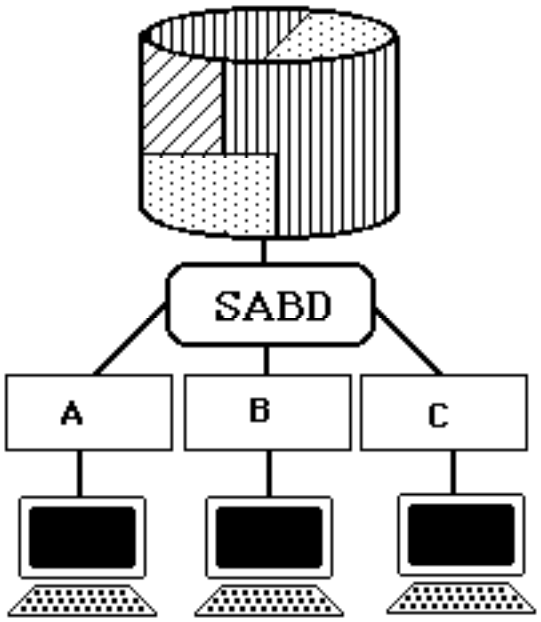
Éstas nacen con el fin de resolver problemas específicos a nivel industrial, y cuando se descubren sus alcances y limitaciones, se formaliza su estudio en centros de investigación industriales, gubernamentales y universitarios.

Históricamente, cada nueva aplicación generaba sus propios archivos y programas. Esto conducía a la redundancia de los datos y la imposibilidad de explotar asociaciones entre ellos.

En la siguiente figura se puede apreciar el enfoque tradicional de los sistemas basados en archivos separados y el enfoque moderno de la integración de la información por medio de Bases de Datos.



Ambiente de archivos separados



Ambiente Bases de Datos

En el primer dibujo se puede observar cómo cada nueva aplicación requiere de sus propios archivos, obteniéndose de esta manera información redundante y posiblemente inconsistente. En el segundo, se aprecia cómo en el ambiente Bases de Datos, varias aplicaciones pueden compartir la misma información.

Sin embargo, cabe mencionar que hubo también otras causas que ayudaron al desarrollo del enfoque Bases de Datos. Entre ellas, se pueden citar las siguientes.

En primer lugar, mientras que la capacidad de almacenamiento en los computadores ha ido en aumento constante, el costo de almacenamiento se ha reducido en forma considerable.

En segundo lugar, la necesidad cada vez mayor, por parte de las organizaciones, de poseer información cada vez más confiable y oportuna.

Finalmente, otro factor es el que se refiere al mantenimiento de aplicaciones. En efecto, la calidad de los sistemas mejoraría considerablemente si se separara la parte conceptual de los archivos y los programas de aplicación.

Se llama **Base de Datos Centralizados** o simplemente **Base de Datos** a un conjunto de datos almacenados en memorias de un computador para satisfacer los requerimientos de información de varios usuarios, eventualmente en forma concurrente.

Entonces, cuando se crea una Base de Datos, es porque se desea poseer mayor centralización e integración, y mejor coordinación y consistencia, así como mayor capacidad de recuperación sobre la información. También, se busca mayor coherencia, eliminar hasta donde sea posible la redundancia de la información almacenada, y reducir los esfuerzos de actualización.

Pero, la centralización de la información en una Base de Datos puede generar una serie de inconvenientes.

En primer lugar, la Base de Datos podría destruirse, debido a fallas del hardware o del software, por acciones de sabotaje, o simplemente por error humano.

En segundo lugar, el uso de la Base de Datos por parte de varios usuarios en forma simultánea, podría presentar problemas ligados a la seguridad de la información, así como a su distribución.

Finalmente, la creación de una Base de Datos podría necesitar gran cantidad de información sobre personas u organismos. Esto podría provocar problemas ligados a la confidencialidad.

El software que garantiza, hasta donde sea posible, que este tipo de problemas no se den y que permita un diálogo Usuario-Base de Datos, se le conoce como **Sistema Administrador de Bases de Datos (SABD).**

En la siguiente sección se estudiarán las funciones principales que deben cumplir un SABD para garantizar una recuperación eficiente de la información residente en una Base de Datos.

2. Funciones de un SABD.c2.

Además de una adecuada interacción Usuario-Base de Datos, el SABD cumple con una serie de funciones, entre las cuales se pueden citar las siguientes.

En primer lugar, el SABD brinda mecanismos que permiten llevar a cabo la **descripción** de los datos que serán almacenados en la Base de Datos.

Existen varios niveles de descripción de estos datos. Por un lado se tiene la **descripción lógica**, la cual traduce la percepción que tiene el usuario de la Base de Datos, y por otro, la **descripción física**, que se refiere a la organización de los datos en las memorias.

También, como se dijo al inicio de la sección, es necesario que el SABD pueda garantizar una interacción de alto rendimiento Usuario-Base de Datos, además, que permita la consulta, la modificación y la actualización de la Base de Datos.

Este diálogo depende del tipo de usuario. Si se trata de un técnico en informática, éste debe poder expresarse por medio de procedimientos algorítmicos. Si se trata de un usuario no informático, como podría ser un cajero de banco con su terminal, éste debe solamente saber las operaciones necesarias que deben efectuarse para actualizar, por ejemplo, las cuentas corrientes.

Es evidente que entre mayor información centralizada se tenga, la posibilidad de pérdida o de uso indebido puede aumentar. Por eso, se debe garantizar la coherencia de la información almacenada con respecto a su significado. Así, la edad de un empleado sólo debe tomar valores entre 18 y 80, por ejemplo, o bien, la cantidad de piezas en una bodega no puede ser negativa. Asímismo, se debe garantizar la consistencia, por ejemplo, un empleado no puede tener una fecha de ingreso mayor que la fecha de nacimiento más veinte años.

Por esta razón el SABD ofrece al usuario la posibilidad de definir ciertas reglas que le permitan mantener la **integridad** de la Base de Datos. A estas reglas se les llama **imposiciones de integridad (integrity constraints)**. Posterior a la definición de estas reglas, corresponde al SABD, verificar su validez, cada vez que se actualice la Base de Datos.

Por ejemplo, supóngase que se ha diseñado una Base de Datos para llevar el control de la matrícula de los estudiantes de un Centro de Educación. Esta Base debería poder asegurar que las notas de los estudiantes, sean números entre 0 y 100, que el monto de las becas sea positivo, etc.

También, se espera que una Base de Datos sea usada por varios usuarios. Sin embargo, es muy posible que algún subconjunto de los datos que se encuentren en esta Base, no deba ser consultado o modificado por cualquiera. Así, el SABD debe garantizar mecanismos de **seguridad** que permitan detectar o verificar los derechos de acceso. Por ejemplo, para un funcionario del Centro que trabaje en la Biblioteca no se justifica que tenga acceso a los datos referentes a los salarios, y que le incumben solamente al Departamento Financiero.

Cuando se da una falla de software o de hardware, el SABD debe garantizar que la Base de Datos se pueda **recuperar** y poder dejarla en un estado satisfactorio. Por supuesto, existen casos en que es difícil recuperar una Base de

Datos después de un incidente como un terremoto, una inundación, etc.

Se han hecho muchos esfuerzos por parte de investigadores, para poder garantizar cada vez mayor integridad y seguridad de la información en una Base de Datos.

A continuación se van a presentar los tres niveles de representación de una Base de Datos, siguiendo el estándar, dado en 1982, por ANSI/SPARC, grupo de estudio de los Sistemas de Administración de Bases de Datos.

3. Niveles de Representación de una Base de Datos

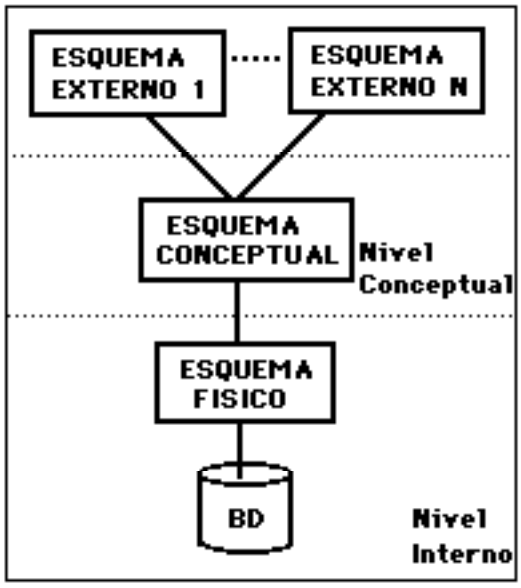
Hoy en día, siguiendo el estándar del grupo ANSI/SPARC, existen tres niveles para la representación de una Base de Datos.

En primer lugar se tiene el **nivel interno** o **físico** con su **esquema interno**. Éste es el de más bajo nivel y en él se describen las estructuras de datos y cómo se almacenan físicamente los datos

En segundo lugar se tiene el **nivel conceptual** con su **esquema conceptual**. Aquí se describen cuáles son los datos que en realidad se encuentran almacenados y las asociaciones entre ellos.

Finalmente, se tiene el **nivel externo** con su **esquema externo**. Este es el nivel de abstracción más alto y describe solamente una parte de la Base de Datos.

En la siguiente figura se pueden apreciar estos tres niveles.



Estos niveles corresponden a diferentes grupos de usuarios. En una representación tal, sólo la Base de Datos tiene una existencia real, los otros niveles corresponden a representaciones descritas en términos abstractos que sirven de referencia a su utilización.

4. Modelaje de una Base de Datos

Con respecto al proceso de modelaje de una Base de Datos, se entiende por **mundo real** de una empresa, la información necesaria para la administración de la misma.

Para llegar a la creación e implementación de una Base de Datos, se requiere previamente pasar del mundo real a un esquema llamado conceptual, como se vió en la figura anterior. Esto es posible si se delimita de ese mundo, la parte de la información relevante para satisfacer los requerimientos de las aplicaciones en cuestión.

Por ejemplo, si se deseara desarrollar un Sistema de Matrícula para alguna institución educativa, la información que sería relevante a este sistema

es la referente a cada estudiante, como por ejemplo: el nombre, número de carné, dirección, edad, cursos aprobados, etc. Sin embargo, sería de muy poca importancia, en este caso, el considerar por ejemplo cuántos tíos tiene cada estudiante o cuál es su pasatiempo favorito.

El mundo real será modelado utilizando cuatro parámetros, a saber:

1. **conjuntos** de objetos, llamados **de entidades**,
2. **propiedades** de esos objetos, llamadas **atributos**,
3. **rango** de las propiedades y
4. **asociaciones** entre conjuntos de entidades.

Un **conjunto de entidades** puede ser un conjunto de objetos con características similares. Por ejemplo, se puede hablar de conjuntos de EMPLEADOS, ESTUDIANTES, COLORES, FECHAS etc.

Un **atributo** es una propiedad de un conjunto de entidades. Por ejemplo: número de cédula, nombre, dirección podrían ser atributos del conjunto de entidades EMPLEADOS.

Por su parte, el **rango** de las propiedades viene a ser el dominio sobre el cual se encuentran definidas. Por ejemplo, para el caso del atributo edad perteneciente al conjunto de entidades EMPLEADOS, se podría definir su rango como el conjunto {00, 01,..., 98, 99}.

Con respecto a las **asociaciones** entre conjuntos de entidades existen cuatro tipos:

1-1 (una a una)
Es el caso en que una entidad de un conjunto está relacionada con una sola entidad de otro conjunto;

1-N (una a muchas)
Cuando una entidad de un conjunto está relacionada con varias entidades de otro conjunto;

N-1 (muchas a una)
Cuando varias entidades de un conjunto se relaciona a una sola entidad de otro conjunto;

M-N (muchas a muchas)
Que es el caso cuando varias entidades de un conjunto se relaciona con varias de otro.

Una vez que se ha hecho un discernimiento sobre la parte del mundo real que interesa al sistema; con ayuda de un modelo de datos, se va a construir el esquema conceptual.

Un **modelo de datos** es una herramienta gráfica que sirve para modelar esa parte del mundo real que interesa para la creación de la Base de Datos, y siempre utilizando los parámetros:

-conjuntos de entidades,
-atributos,
-rango de atributos
-asociaciones entre entidades.

En la actualidad, se cuenta con cerca de 50 modelos de datos diferentes.

El concepto de modelo de datos es fundamental en el ambiente de Bases de Datos. Tan importante es este término que, cuando se habla por ejemplo de un SABD relacional, significa que se encuentra cimentado sobre el modelo de datos relacional.

Sin embargo, es importante recalcar que no todos los modelos de datos poseen SABD respectivos que hayan sido comercializados. El último que ha sido comercializado es el modelo de datos Entidad-Asociación cuyo SABD es el llamado sistema semántico SIM (Semantic Information Manager).

En la siguiente tabla se aprecian algunos SABD comercializados y su modelo de datos

respectivo.

SABD	Modelo de datos
IMS	Jerárquico
DBTG	Redes
SQL	Relacional
SIM	Semántico
ADABAS	Listas invertidas

Con respecto al **modelo de datos jerárquico**, el mismo se basa en la estructura de árbol.
En este caso, los nodos del árbol representan los conjuntos de entidades y los arcos entre los nodos representan las asociaciones entre esos conjuntos de entidades.

Por su misma estructura, el modelo de datos jerárquico sólo permite asociaciones de los tipos 1-1 y 1-N.

Por su parte, **el modelo de datos de redes** se basa, como su nombre lo indica, en redes, en donde los nodos son conjuntos de entidades y los arcos representan las asociaciones entre conjuntos de entidades. Este modelo permite asociaciones de los tipos 1-1, 1-N y N-1. Para el caso de las asociaciones M-N se utiliza la estructura de datos llamada lista circular.

En cuanto al **modelo de datos relacional**, es el de mayor uso en el mundo, no así en Costa Rica y se basa en un conjunto de tablas o relaciones, en donde cada relación representa un conjunto de entidades llamadas **tuplos** o **registros**.

En el modelo relacional, las asociaciones a su vez, serán representadas por medio de relaciones y permiten por lo tanto cualquier tipo de asociación. Estos conceptos serán profundizados en los próximos capítulos.

Una vez que se ha diseñado el esquema conceptual, el SABD suministra, a quién diseña la Base de Datos y que es conocido como el **Administrador de la Bases de datos (ABD)**, lo que se denomina un **Lenguaje de Definición o Descripción de Datos (LDD)**. Esto le permite al **ABD** hacer una descripción de la información.

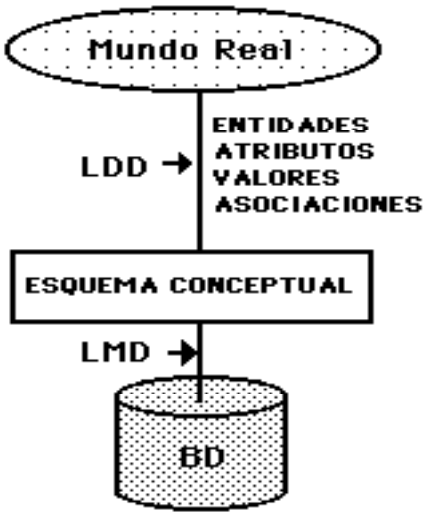
En el caso de un modelo de datos relacional, la descripción significa el nombrar las relaciones, así como los atributos de cada una de ellas y los rangos de los valores eventuales de los atributos.

Por ejemplo, si se tiene la relación (conjunto de entidades o asociación) EMPLEADOS, se describe definiendo sus atributos, como pueden ser CEDULA, NOMBRE, EDAD, etc. También, los rangos de los valores de estos atributos. Por ejemplo, CEDULA se define sobre 10 dígitos, el NOMBRE sobre 40 caracteres, etc.

Una vez que se tenga la descripción de la información, se debe alimentar la Base de Datos, es decir, introducir datos en una primera instancia, así como poder actualizarlos y borrarlos posteriormente. Esto será posible gracias a un **Lenguaje de Manipulación de Datos (LMD)**, suministrado por el SABD.

En los SABD de tipo jerárquico y redes los LDD y los LMD son lenguajes separados. En el caso de los SABD relacionales los dos se integran en uno solo denominado **Lenguaje de Consultas (Query Language)**

La siguiente figura resume el proceso de modelaje de una Base de Datos.

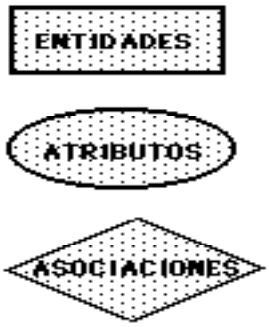


Fases de modelaje de una Base de Datos

5. Ejemplo de Esquema Conceptual

Para este ejemplo, se usará el modelo de datos, debido a P.P. Chen **[Chen76]**, llamado **Entidad-Asociación**.

En este caso las entidades se representan por medio de rectángulos; los atributos, por medio de círculos y las asociaciones por medio de rombos, según se aprecia en la siguiente figura:

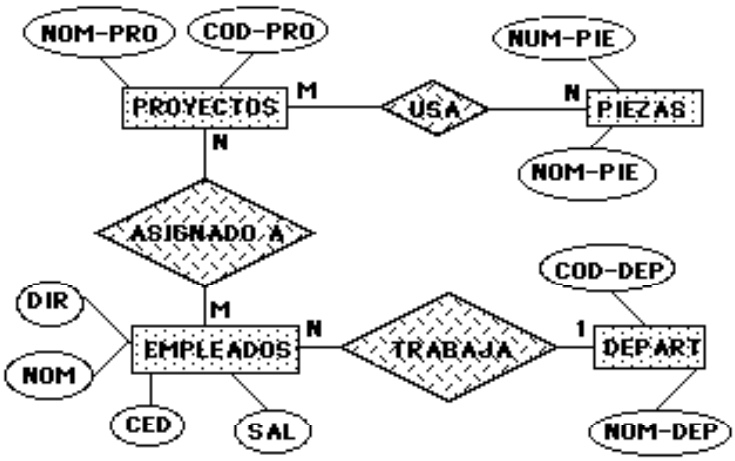


En la figura siguiente, se aprecia el esquema conceptual de una Base de Datos que posee cuatro conjuntos de entidades, a saber: **PROYECTOS**, **PIEZAS**, **EMPLEADOS** y **DEPARTAMENTOS**. Cada uno de estos conjuntos, por su parte, poseen varios atributos. En el caso de **EMPLEADOS**, se tienen los atributos CED (cédula), NOM (nombre), DIR (dirección), SAL (salario).

También se tienen tres asociaciones. En primer lugar, la asociación **USA** entre PROYECTOS y PIEZAS que representa las piezas utilizadas en los proyectos. Esta es una asociación M-N.

En segundo lugar, la asociación **ASIGNADO** entre EMPLEADOS y PROYECTOS que representa los empleados asignados a proyectos. Suponiendo que un empleado puede estar asignado a varios proyectos, se puede considerar una asociación M-N.

Finalmente, **TRABAJA** entre EMPLEADOS y DEPARTAMENTOS, que representa la asociación de los empleados que laboran en un determinado departamento. Esta es una asociación N-1.


NOM-PRO
COD-PRO
NUM-PIE
PROYECTOS
M
USA
N
PIEZAS
N
NOM-PIE
ASIGNADO A
COD-DEP
DIR
M
N
1
EMPLEADOS
TRABAJA
DEPART
NOM
CED
SAL
NOM-DEP

II Modelo de Datos Relacional

1. Introducción

En junio de 1970, aparece en la revista Communications of the ACM, un artículo titulado "**A relational model of data for large shared data banks**". Este artículo, escrito por el matématico inglés Edward F. Codd, revolucionaría en el futuro inmediato, el campo de las Bases de Datos. Codd propone un nuevo modelo de datos, conocido hoy en día como **modelo relacional** o **modelo de Codd**, y una metodología asociada para el diseño de Bases de Datos, llamada **normalización**.

En la actualidad, se cuentan por centenas los SABD basados en el modelo relacional. A mediados de los años setenta se comercializan los dos primeros SABD relacionales.

El primero fue INGRES, desarrollado por M. Stonebraker y su equipo de la Universiodad de Berkeley.

El segundo fue el **Sistema R**, desarrollado en el Centro de Investigación de IBM en San José, California. Este SABD evolucionaría posteriormente al estándar conocido hoy en día como **SQL**.

La propuesta de Codd fue la culminación de investigaciones realizadas a finales de los años sesenta, sobre todo por investigadores franceses y estadounidenses.

Esta nueva forma de modelar el mundo de la información representaba un cambio importante con respecto a los modelos de datos utilizados hasta la fecha, el jerárquico y el de redes.

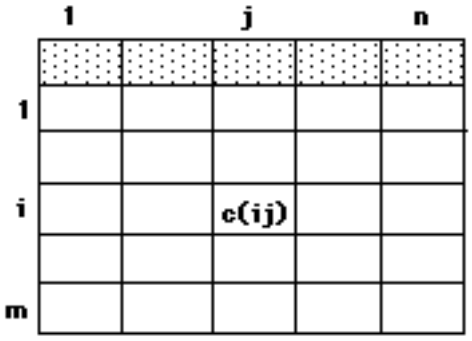
En efecto, en el modelo de datos relacional, la abstracción del mundo de la información, propia de una organización, se representaría por medio de un conjunto de relaciones o tablas.

Una **relación** es un concepto matemático y se define como un subconjunto de un producto cartesiano D_1 **X** D_2 **X** ....**X** D_n, en donde cada D_i es un conjunto finito de elementos indivisibles, llamados **dominios**.

Sin embargo, una relación en el contexto del modelo relacional difiere de la definición matemática. En efecto, una relación en el modelo relacional se puede representar como un arreglo de m+1 filas por n columnas, en donde la primera fila (la fila 0) sirve para nombrar la relación, y estará formada por los llamados atributos de la relación. Las otras filas forman el contenido de la relación y cada una de ellas se llama tuplo.

2. Definiciones Básicas

Un **arreglo (m x n)** es un cuadro, según se aprecia en la siguiente figura:



Un **dominio D** es un conjunto finito de entidades elementales, en donde cada una representa un objeto, un color, un sonido, etc.

Un **atributo** es la identificación que se le da a un conjunto de valores con una característica común. Cada atributo toma sus valores de un

dominio determinado.

Así por ejemplo, Número podría ser un dominio que represente todos los números naturales de 0 a 100. Por su parte, se puede definir el atributo EDAD, cuyo dominio sea Número.

Una **relación R de grado n** es un arreglo m x n completamente utilizado, según se aprecia en la siguiente figura:

R

A1		Aj		An
a11		a1j		a1n
ai1		aij		ain
am1		amj		amn

Cada fila **i (** $x=(a_{i1},...,a_{ij},...,a_{in})$ **)** se llama un **tuplo (o registro)** de la relación, y cada a_{ik} se llama **componente** del tuplo.

Los atributos en una relación se designarán por medio de letras mayúsculas y serán diferentes entre ellos.

En la siguiente figura se puede apreciar una relación llamada **EMPLEADO** que tiene tres atributos: CEDULA, NOMBRE Y CIUDAD.

EMPLEADO

CEDULA	NOMBRE	CIUDAD
2303776	Carlos	Cartago
7204728	Juan	Limón
1227546	María	Alajuela

En el contexto del modelo relacional, a una relación se le pueden intercambiar dos columnas sin que por ello se pierda su significado. Esta es una de las diferencias entre los conceptos: relación matemática y relación en el modelo relacional.

Así por ejemplo, sea la relación **PROFESOR**, según se aprecia en la figura siguiente.

PROFESOR

NOMBRE	INST.	CURSO
Carlos	ITCR	Bases de Datos
Juan	UCR	Programación

Esto podría significar que el tuplo (a,b,c) se encuentra en **PROFESOR** si: **"el profesor de nombre "a" trabaja en la institución "b" e imparte el curso "c""**.

Por otra parte, considérese la relación **PROFESOR1** dada por la figura siguiente:

PROFESOR1

NOMBRE	CURSO	INST
Carlos	Bases de Datos	ITCR
Juan	Programación	UCR

y tal que el tuplo (b,a,c) se encuentra en **PROFESOR1** si: **"en la institución "b" trabaja el profesor de nombre "a" impartiendo el curso "c" ".**

Obsérvese que estas dos relaciones son equivalentes, pues contienen la misma información.

Si **R** representa una relación y sus atributos son $A_1,\ldots,A_n$, se denota esta situación por **$R(A_1,...,A_n)$**.

Una **base de datos relacional** se define como un conjunto de relaciones. Pero, puesto que una Base de Datos evoluciona en el tiempo, las relaciones iniciales y su número pueden variar debido a las actualizaciones que se hagan sobre ella. Así, se tienen las siguientes notaciones:

A una **base de datos inicial** se le denota por B_0, es decir, al conjunto de relaciones definidas en el momento de concebir la Base de Datos. Se denota por B_i el estado de la Base de Datos en la actualización i-ésima. Luego se verá que para pasar de B_i a B_{i+1} se debe mantener la coherencia de los datos, es decir, debe satisfacer un cierto número de imposiciones de integridad propuestos por el ABD.

En los SABD es necesario poseer un lenguaje de manipulación de datos (LMD), para satisfacer las necesidades de consultas de la información por parte de los usuarios de Bases de Datos, según se vió en el capítulo anterior.

Estos lenguajes han recibido mucho interés por parte de los investigadores y se pueden distinguir dos grandes categorías de lenguajes de manipulación de datos relacionales. Por un lado, se tienen **los lenguajes algebraicos**. Éstos se basan en un álgebra de operadores que se aplican sobre las relaciones de base. Y por otro lado, se tienen los **lenguajes predicativos**, basados en la Lógica de Primer Orden.

3. Los Lenguajes Algebraicos

En este tipo de lenguaje, las respuestas a las consultas que los usuarios pueden hacer, se dan bajo forma de relaciones obtenidas por medio de un aplicación finita de operadores unarios y binarios, en donde los operandos son relaciones.

El Sistema **SQL**, así como el **INFORMIX** y el **RBASE 5000**, son ejemplos de SABD relacionales cuyos lenguajes de manipulación de datos fueron construidos sobre la base del lenguaje algebraico.

Antes de continuar con la definición de los diferentes operadores, se va a definir el concepto de término de proyección.

Sea **R** una relación de grado n, sea x un tuplo y X un subconjunto de atributos de **R**. Sea x**[**X**]** el tuplo obtenido a partir del tuplo x, eliminado los componentes no pertenecientes a X. Se llama a x**[**X**]** un **término de proyección.**

Por ejemplo, considérese la relación **EMPLEADO** (#EMP, NOMBRE, SEXO, EDAD, SALARIO, DIRECCION) y X={ #EMP, SEXO, EDAD, SALARIO **}.**
Sea x = (1-404-566, Elena, f, 34, 25000, Alajuela)
Entonces se tiene que x**[**X**]**=(1-404-566,f,34,25000).

A continuación, se definen los operadores relacionales más importantes.

3.1. Producto

Sean **R** y **S** dos relaciones. El **producto** de **R** y **S**, denotado por **R*S**, es la relación cuyos atributos son la unión de los atributos de **R** y **S** y los tuplos x son tales que x**[R]** es un tuplo de **R** y x**[S]** uno de **S.**

Ejemplo. Sean **R** y **S** las dos relaciones siguientes:

R

PINTOR	EPOCA
Manet	Realista
Delacroix	Realista
El Greco	Realista

S

PINTOR	CUADRO
Manet	Almuerzo en la hierba
Manet	Retrato de Emilio Zola
El Greco	La resurrección

Entonces el producto de **R** y **S** será la relación siguiente:

R*S

PINTOR	EPOCA	CUADRO
Manet	Realista	Almuerzo en la hierba
Manet	Realista	Retrato de Emilio Zola
El Greco	Realista	La resurrección

Esto podría representar la respuesta a la consulta: " **Dar la lista de los pintores con sus épocas y cuadros respectivos** "
En este caso Delacroix no puede aparecer pues sólo se tiene información de su época y no de los cuadros que pintó.

3.2. Unión

Sean $\mathbf{R}(A_1,A_2,\ldots,A_n)$ y $\mathbf{S}(A_1,A_2,\ldots,A_n)$ dos relaciones con los mismos atributos. Se define la **unión** de **R** y **S**, denotada por $\mathbf{R} \cup \mathbf{S}$, a la relación cuyos tuplos son los pertenecientes a **R** más los de **S**.

Sean **R** y **S** las relaciones siguientes:

R

PINTOR	NACIONALIDAD
Manet	Francesa
Chagall	Francesa

S

PINTOR	NACIONALIDAD
Giotto	Italiana
Vinci	Italiana

Entonces, la unión de R y S es la relación siguiente:

R+S

PINTOR	NACIONALIDAD
Manet	Francesa
Chagall	Francesa
Giotto	Italiana
Vinci	Italiana

3.3. Diferencia

Sean $\mathbf{R}(A_1,A_2,\ldots,A_n)$ y $\mathbf{S}(A_1,A_2,\ldots,A_n)$ dos relaciones con los mismos atributos. Se define la **diferencia** de **R** y **S**, denotada por **R - S**, a la relación cuyos tuplos son los pertenecientes a **R** y que no se encuentran en **S**.

Esta vendría a ser la operación inversa a la unión. En el caso del ejemplo anterior se tiene que $(\mathbf{R} \cup \mathbf{S}) - \mathbf{S} = \mathbf{R}$

3.4. Producto Cartesiano

Se define el producto cartesiano de dos relaciones **R** y **S**, denotado por $\mathbf{R} \times \mathbf{S}$, a la relación cuyos tuplos son las posibles concatenaciones entre los tuplos de **R** y los de **S**.

Ejemplo. Sean **R** y **S** las dos relaciones siguientes siguientes:

R

PINTOR	NACIONALIDAD
Manet	Francesa
Chagall	Francesa
Giotto	Italiana

S

PINTOR	EPOCA
Manet	Realista
El greco	Barroca

Entonces, el producto cartesiano entre **R** y **S** es la relación siguiente:

RxS

PINTOR	NACIONALIDAD	PINTOR	EPOCA
Manet	Francesa	Manet	Realista
Manet	Francesa	El greco	Barroca
Chagall	Francesa	Manet	Realista
Chagall	Francesa	El greco	Barroca
Giotto	Italiana	Manet	Realista
Giotto	Italiana	El greco	Barroca

Recuérdese que en una relación los atributos son diferentes. En el caso del producto cartesiano, si aparecen los mismos atributos en las dos relaciones, éstos deben aparecer en la relación producto cartesiano, sin embargo, el SABD los considera diferentes.

3.5. Proyección

Sea **R**(A_1,...,A_n) una relación y X un subconjunto de atributos de **R**. Se denota la **proyección de R** a X por **R**[X] y se define como la relación cuyos tuplos son todos los términos de proyección x[X], en donde x es cualquier tuplo de **R.**

Ejemplo. Sea **R** la siguiente relación:

R

PINTOR	NACIONALIDAD	CUADRO
Manet	Francesa	La olimpia
Yinci	Italiana	La gioconda
Amighet	Costarricense	La celestina
Manet	Francesa	Retrato

Ahora se desea, a partir de **R**, obtener una lista de los pintores y sus nacionalidades. Esto se puede lograr de la siguiente forma. Sea X={PINTOR, NACIONALIDAD}. Entonces, se tiene que **R**[X] es la relación siguiente:

PINTOR	NACIONALIDAD
Manet	Francesa
Yinci	Italiana
Amighet	Costarricense
Manet	Francesa

3.6. Selección

Para definir la relación selección es necesario que se defina previamente el concepto de expresión de selección.

Se dice que **F** es una **fórmula atómica** si
F = ATRIBUTO θ ATRIBUTO o
F = ATRIBUTO θ 'a', donde a es el valor de una entidad y $\theta \in \{\leq, \geq, =, \neq, >, <\}$

Una **expresión de selección** se define de la siguiente manera:

1.Si **F** es una fórmula atómica entonces **F** es una expresión de selección.

2.Si **E 1** y **E2** son expresiones de selección, entonces

E_1 **AND** E_2,
E_1 **OR** E_2 , y
NOT E_1

también lo son, en donde **AND**, **OR** y **NOT** son las conectivas lógicas.

Ejemplos de expresiones de selección son los siguientes:

1. E1: { (Pintor='Manet' **0R** Pintor='Renoir')
AND (Epoca='Impresionista') }

2. E2: {(AÑO-OBRA **>** 1600) **AND**
NOT (NACIONALIDAD= 'Italiana')}

Sea **R** una relación, **E** una expresión de selección. Se define la **selección R%E** como la relación cuyos atributos son los de **R** y sus tuplos los que satisfacen la condición **E.**

Ejemplo. Sea **R** la siguiente relación:

R

PINTOR	NACIONALIDAD	CUADRO
Manet	Francesa	La olimpia
Yinci	Italiana	La gioconda
Amighet	Costarricense	La celestina
Manet	Francesa	Retrato

Sea la expresión de selección **E** :
{ PINTOR = 'Manet' **OR**
NACIONALIDAD = 'Costarricense'}

Entonces, **R%E** es la relación siguiente:

PINTOR	NACIONALIDAD	CUADRO
Manet	Francesa	La olimpia
Amighet	Costarricense	La celestina
Manet	Francesa	Retrato

3.7. Producto-θ

Sean **R**(A_1,...,A_n) y **S**(B_1,...,B_n) dos relaciones, con **dom**(A_l) **= dom**(B_1) y sean x e y dos tuplos de **R** y **S** respectivamente. Se define la relación **producto-selección** sobre A_1, y se denota por **R**{A_1 θ B_l}**S** al conjunto de tuplos (x,y), del producto cartesiano **R × S**, que satisfacen (x,y)[A_1] θ (x,y)[B_l], con θ en {≤,≥, = , ≠, >,<}

Ejemplo. Sean **R** y **S** las dos relaciones siguientes:

R

PINTOR	AÑO-NAC
Rembrandt	1606
Rubens	1577
Raphael	1483
Velazquez	1599
Picasso	1881
Ingres	1780
Degas	1834

S

AÑO	ESCUELA
1850	Impresionista
1600	Flamenca

Entonces, la relación **R**{AÑO-NAC**>**AÑO}**S** es la siguiente.

R{AÑO-NAC > AÑO}S

PINTOR	AÑO-NAC	AÑO	ESCUELA
Picasso	1881	1850	Impresionista
Rembrandt	1606	1600	Flamenca
Picasso	1881	1600	Flamenca
Ingres	1780	1600	Flamenca
Degas	1834	1600	Flamenca

Esta relación podría representar la respuesta a la consulta: **Dar la información sobre todos los pintores cuyo nacimiento es posterior al inicio del movimiento en que se desenvolvieron.**

A continuación, se resume en una tabla los operadores anteriormente expuestos

Nombre	**Simbología**
Producto	**R * S**
Unión	**R ∪ S**
Producto Cartesiano	**R × S**
Proyección	**R[X]**

Selección	**R % E**
Diferencia	**R - S**
Producto-Selección	**R**{A_i θ B_i}**S**

3.8. Expresiones Relacionales

Como se mencionó anteriormente, la repuesta a una consulta en el caso de los lenguajes algebraicos es una relación obtenida por medio de manipulaciones entre operadores y relaciones de la Base de Datos.

Por ejemplo, considérese una Base de Datos con las siguientes relaciones:
R(PINTOR, EPOCA, CUADRO),
S(PINTOR,NACIONALIDAD,AÑO-NACIMIENTO),

Ahora, sea la siguiente consulta a la Base de Datos:
Dar la lista de los pintores Impresionistas cuyo año de nacimiento es posterior a 1850.

La respuesta a dicha consulta se puede obtener efectuando primero las dos expresiones de selección siguientes:

E = (EPOCA='Impresionista')
F = (AÑO-NACIMIENTO >1850)

Luego, se calculan las dos selecciones **R%E**, **S%F** y posteriormente el producto de éstas, seguida de una proyección sobre **PINTOR**.

En la mayoría de los lenguajes algebraicos que se encuentran en los sistemas relacionales actuales, la totalidad de los operadores no están implementados. Algunos de ellos no son primitivos, es decir, se pueden derivar de otros operadores que se llamarán de base. Los **operadores de base** son: la unión, la diferencia, el producto cartesiano, la proyección y la selección (**∪, × , -, [ ], %**).

Se dice que un lenguaje es **relacionalmente completo** si puede generar los operadores de base.

Los SABD relacionales poseen también los llamados **optimizadores de consultas**, es decir, conjuntos de programas que buscan, a partir de la expresión como la anterior, construir una equivalente (dando el mismo resultado), de tal manera que su procesamiento y accesos a disco sea el menor posible.

En efecto, supóngase que se tienen dos relaciones **R** y **S** de 20,000 y 40,000 tuplos respectivamente. Si en una expresión se tiene el producto cartesiano **R × S** involucrado, se generarían 800 millones de registros. De ahí la necesidad de tener optimizadores que permitan disminuir estos números tan grandes.

Se ha hecho un estudio exhaustivo del lenguaje algebraico con el fin de mejor entender el funcionamiento del lenguaje de consultas SQL, que será estudiado en el capítulo 4 de este informe. El SQL es un lenguaje de consultas construido sobre el lenguaje algebraico.

A continuación se introducirá, sin entrar en detalles, los lenguajes predicativos.

4. Lenguajes Predicativos

Los lenguajes predicativos están basados en la Lógica de Primer Orden (o Cálculo de Predicados) .

Las consultas en este caso son expresiones del tipo **{x/P(x)}**, que significa: Dar todos los valores x que satisfacen el predicado P(x). La respuesta a dicha consulta serán los objetos "a" tal que P(a) sea verdadero, es decir al conjunto **{a/P(a)=v}**.

En **[Codd71]** se define el primer lenguaje de

consultas de tipo predicativo. Se le llamó "sublenguaje de datos **ALPHA**". En realidad, ALPHA nunca fue implementado. Sin embargo, mucha de su filosofía fue tomada en cuenta para realizar **QUEL**, el lenguaje de consultas del SABD INGRES.

Codd también mostró que toda consulta expresada en forma predicativa puede convertirse en una expresión algebraica equivalente y recíprocamente. Así, los lenguajes predicativos y algebraicos poseen el mismo poder de respuesta.

Entre los SABD que se han comercializado y cuyos lenguajes de consultas se encuentran basados en los lenguajes predicativos, se puede citar a **QUEL**, el lenguaje de consultas del SABD INGRES, el **Paradox** y el **QBE (Q**uery **B**y **E**xample). Todos estos SABD se encuentran disponibles para un ambiente de microcomputadores.

A continuación se introduce la metodología llamada **normalización**. Ésta es necesaria aplicarla si se desea un buen diseño en las Bases de Datos relacionales, como se verá en el ejemplo de la siguiente sección.

5. Normalización

Cuando se desea traducir el mundo real de la empresa a un esquema conceptual por medio del modelo relacional, esto se logra definiendo un conjunto de relaciones. Sin embargo, surge la siguiente pregunta: ¿ **Cuál debe ser el conjunto de relaciones que traduzcan fielmente el esquema conceptual, evitando los problemas de coherencia** ? Cuando se habla de evitar problemas de coherencia significa eliminar anomalías en la Base de Datos que pueden darse a la hora de introducir, borrar o actualizar datos. A partir de las reglas semánticas que traducen las imposiciones de la organización modelada, el ABD debe definir las "**dependencias**" que serán incorporadas en la defínción del esquema conceptual.

Las dependencias son propiedades del esquema relacional de la Base de Datos y no de una relación en particular, es decir son invariantes que deben ser satisfechas por las relaciones a través de la vida de la Base de Datos.

La teoría de la normalización se basa en el hecho de que ciertas relaciones poseen mejores propiedades a la hora de la actualización, que otras equivalentes (es decir, con la misma información). Además, brinda un cuadro riguroso para la definición del esquema relacional y se basa en un proceso de descomposición reversible que garantiza que la relación inicial puede recuperarse usando "*" (producto) entre las relaciones resultantes y por lo tanto la información inicial no se pierde.

Antes de dar un ejemplo que aclare lo expuesto anteriormente, y mostrando la importancia y la necesidad de la normalización, se va a introducir los términos de dependencia funcional y llave de una relación.

Sean X,Y dos subconjuntos de atributos de la relación **R**(X,Y,Z). Se dice que **R verifica la dependencia funcional X->Y** si para cada valor de X se tiene asociado un sólo valor de Y.

Además, si X->X∪Y ∪ Z (denotado por X->X,Y,Z) se verifica y no existe un subconjunto de X que cumpla esta propiedad, X se dice **llave** de la relación y se denota por **<u>X</u>**

Como ejemplo, considere la relación **EMPLEADO**
(#EMP, NOM-EMP, DIRECION, SALARIO).

En este caso #EMP se puede considerar una llave, pues a cada número de empleado se tiene asociado sólo un nombre, una dirección y un salario. Es decir, se satisface la dependencia funcional
#EMP->NOM-EMP,DIRECION,SALARIO.

En el siguiente ejemplo se verán algunas de las anomalías de actualización que se pueden presentar en un modelo relacional si no se aplica el proceso de normalización.

Ejemplo. Sea **R** la relación siguiente

R

P	PIEZA	PROV	CIUDAD	CANT
P1	tornillo	Ana	Cartago	100
P1	tuerca	Ana	Cartago	100
P2	empaque	Juan	Limón	500
P3	arandela	María	Cartago	200
P3	tornillo	Juan	Limón	300
P2	tuerca	Ana	Cartago	700

P (Nombre del proyecto)
PIEZA (Pieza utilizada)
PROV (Nombre del proveedor)
CIUDAD (Ciudad del proveedor)
CANT (Cantidad de piezas suplidas)

y considérense las dependencias funcionales P,PIEZA->PROV y PROV->CIUDAD que significan respectivamente: "**Para cada proyecto y pieza sólo se tiene un proveedor**" y "**cada proveedor se localiza en una sola ciudad**".

En este caso se tiene que P,PIEZA,CANT es una llave para **R,** pues determinan en forma única al resto de los atributos de **R**. Además, la relación **R** se encuentra mal diseñada pues se presentan varias anomalías de almacenamiento:

Anomalías de actualización:
Si se desea actualizar la dirección de Ana, se debe hacer en todos los tuplos que contienen el valor Ana, en caso contrario la dependencia funcional PROV->CIUDAD no se satisface.
Este problema surge debido a la redundancia de la información presente en la relación.

Anomalías de inserción:
Se puede introducir un proveedor sólo en el caso en que suministre una pieza a un proyecto.
Una solución a este problema podría ser el introducir valores nulos en los atributos PROV,CANT. Si embargo, esto llevaría a generar otro tipo de problema. En efecto, al ser P,PIEZA,CANT llave, sería difícil buscar tuplos de **R**, en donde la llave posee valores nulos.

Anomalías de borrado:
Si se suprime el tuplo con el proyecto P3, se pierde la información relacionada con María y en particular la dirección de María.

La forma para eliminar estos problemas es por medio de la normalización. Así, la relación **R** sería reemplazada por las relaciones siguientes:

R1

P	PIEZA	PROV	CANT
P1	tornillo	Ana	100
P1	tuerca	Ana	100
P2	empaque	Juan	500
P3	arandela	María	200
P3	tornillo	Juan	300
P2	tuerca	Ana	700

R2

PROV	CIUDAD
Ana	Cartago
Juan	Limón
María	Cartago

En este caso si se desea modificar la dirección de Ana, sólo se hará en un tuplo de **R2**. Además, se puede introducir un nuevo proveedor en **R2** sin necesidad de que éste se encuentre suministrando alguna pieza a un proyecto. Finalmente, se puede suprimir el proyecto P3, sin perder la información relacionada con María. Por otra parte, se puede recuperar **R** haciendo el producto de **R1** con **R2**, es decir **R = R1 * R2.**

En el Anexo del presente documento se profundiza este tema y puede servir como guía para aquellas personas interesadas en utilizar dicha teoría en el diseño de sus Bases de Datos.

III Otros Modelos de Datos

1. Introducción

Los SABD denominados de la primera generación se caracterizan por basarse en los modelos de datos de redes o jerárquicos. En estos casos, la división en tres niveles para la representación de Bases de Datos, como se estudió en el capítulo 1, no existe. Por lo tanto, no se puede hablar de independencia física y lógica de los datos reales. En efecto, es posible que se desee pasar de un tipo de organización de datos a otra, por ejemplo de Hash a ISAM y esto con el fin de mejorar el acceso a los datos. En el caso de estos SABD, los esquemas y los programas de aplicación no se pueden mantener, se deben reprogramar.

Si el diseño de una Base de Datos voluminosa no va a variar en el tiempo, un SABD de este tipo puede cumplir satisfactoriamente su cometido, pero si por el contrario, la Base de Datos es cambiante, no son recomendables estos sistemas.

El SABD jerárquico de mayor relevancia es sin lugar a dudas el **IMS** (Information Management System), creado a mediados de los años sesenta por IBM, mientras que el de redes es el **DBTG** (Database Task Group) de **CODASYL** y fue desarrollado a finales de los años sesenta.

En Costa Rica se le conoce con el término **DL/1**, pero en realidad este es sólo el lenguaje de manipulación y de descripción de los datos del sistema IMS.

En este capítulo se van a introducir las características principales de estos dos modelos de datos.

2. Modelo de Datos de Redes

El modelo de redes se basa en la construcción de **conjuntos (sets)**. El término conjunto en el modelo de redes difiere de su significado en el sentido matemático. En efecto, en matemática un conjunto se compone de objetos, sin ningún tipo de restricción. Mientras que en la construcción de conjuntos en el modelo de redes se usan objetos provenientes de dos tipos de objetos: los **objetos propietarios** y los **objetos miembros**. Además, cada conjunto consiste de un solo objeto propietario y cualquier número finito de objetos miembros.

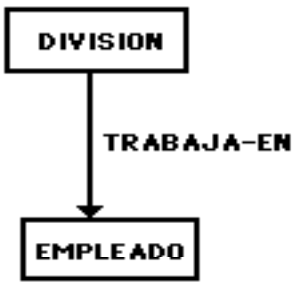
Los conjuntos, a su vez, pueden agruparse en tipos de conjuntos. Todos los conjuntos del mismo tipo consisten de un objeto perteneciente al mismo tipo de objetos propietarios y varios objetos provenientes del mismo tipo de objetos miembros.

En la siguiente figura se dan dos conjuntos del tipo de conjunto denominado **TRABAJA-EN**. En este caso, los objetos del tipo EMPLEADO se encuentran agrupados según la DIVISION para la cual trabajan.

Aquí se tienen dos objetos del tipo DIVISON: DIVISION1 y DIVISION2 y siete objetos del tipo EMPLEADO. Cada objeto DIVISION es un propietario de conjunto del tipo TRABAJA-EN. Los miembros de tal conjunto son EMPLEADOS que trabajan en la DIVISION.

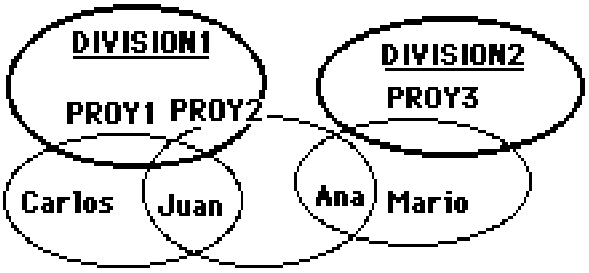
Por ejemplo, Carlos, Felipe, María y Juan trabajan en la DIVISION1, mientras que Mario, Carmen y Luisa trabajan en la DIVISION2.

El modelo de datos de red se representa por medio de un diagrama de estructura de datos, introducido por C.W. Bachman, a finales de los años sesenta. En este diagrama, los objetos de tipo propietario y tipo miembro se representan por medio de rectángulos. Por su parte, el tipo de conjunto **propietario-miembro** se representa por medio de una flecha que va desde el tipo propietario hacia el tipo miembro, según se aprecia en la siguiente figura:



En este caso DIVISION es el tipo de objetos propietarios, y EMPLEADO es el tipo de objetos miembros y TRABAJA-EN el tipo de conjunto. Cada conjunto de TRABAJA-EN consiste de un objeto del tipo DIVISION y un número cualquiera de objetos del tipo EMPLEADO.

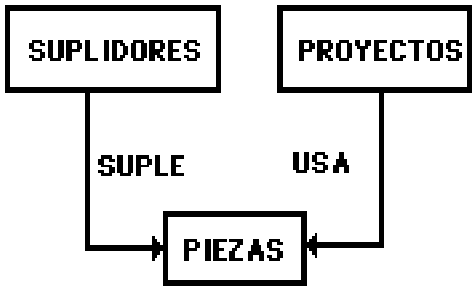
En el modelo de redes no existe ninguna restricción sobre el número de tipos de objetos y tipos de conjuntos que se pueden usar. Tampoco, sobre la utilización de los mismos tipos de objetos propietarios o miembros en diferentes tipos de conjuntos. Esto significa que es posible, tener una jerarquía de tipos de objetos, según se muestra en la siguiente figura:

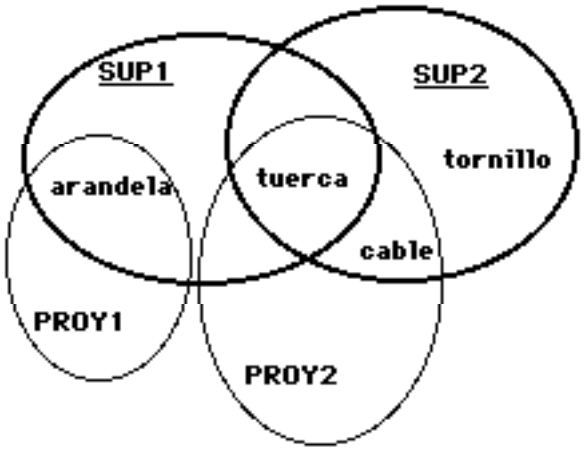


En este caso se tienen tres tipos de objetos: DIVISION, PROYECTO y EMPLEADO. Cada DIVISION administra varios PROYECTOS. A su vez, cada PROYECTO cuenta con varios EMPLEADOS. Además, se tienen dos tipos de conjuntos: ADMINISTRA y TRABAJA-EN. Cada conjunto del tipo ADMINISTRA posee como propietario a un objeto del tipo DIVISION y como miembros, objetos del tipo PROYECTO. En la figura anterior DIVISION1 administra PROY1 y PROY2. Por su parte, cada conjunto del tipo TRABAJA-EN tiene como proietarios a objetos del tipo PROYECTO y como miembros a objetos del tipo EMPLEADO.

Note que en esta figura algunas personas trabajan en más de un proyecto. Por ejemplo, Juan trabaja en PROY1 y PROY2. Por lo tanto, aparecen en más de un conjunto del tipo TRABAJA-EN.

Otra estructura de conjunto común en el modelo de redes es el de un objeto que aparece en varios conjuntos de diferente tipo. En este caso el objeto tiene más de un propietario y cada uno de los propietarios puede estar en más de un tipo de conjunto. Por ejemplo sea la siguiente figura.



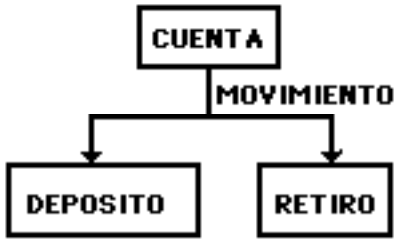


En este caso hay dos tipos de conjuntos: SUPLE y USA. Los conjuntos de ambos tipos tienen como miembros a objetos del tipo PIEZAS. El objeto tuerca, por ejemplo, es miembro de los conjuntos del tipo USA y SUPLE que tienen como propietarios a PROY2 y SUP2 respectivamente.

El conjunto definido en el ambiente del modelo de redes puede ampliarse para permitir traducir situaciones más complejas. Las ampliaciones más naturales son las siguientes:

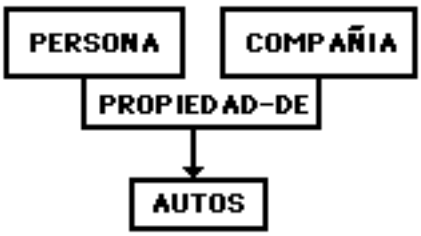
Permitir que los conjuntos tengan el mismo tipo de objetos propietarios y diferentes tipos de objetos como miembros.

En la siguiente figura los conjuntos del tipo MOVIMIENTO tienen como propietarios a los objetos del tipo CUENTA y como miembros a objetos de los tipos DEPOSITO o RETIRO.



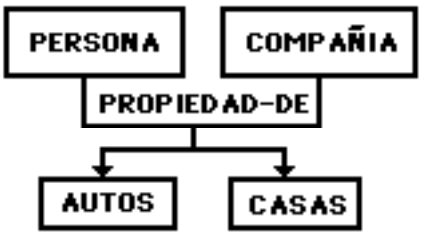
Permitir, el recíproco del caso anterior, es decir que un conjunto de un tipo dado pueda tener como propietarios a objetos de diferentes tipos.

En la siguiente figura el propietario de un conjunto del tipo PROPIEDAD-DE puede ser un objeto del tipo COMPAÑIA o del tipo PERSONA. Es decir, un auto puede pertenecer a una COMPAÑIA o a una PERSONA.



Permitir una combinación de las dos ampliaciones anteriores.

En la siguiente figura el propietario de cada conjunto del tipo PROPIEDAD-DE, puede ser un objeto del tipo PERSONA o un objeto del tipo COMPAÑIA, y los miembros pueden ser objetos de los tipos AUTOS y CASAS.



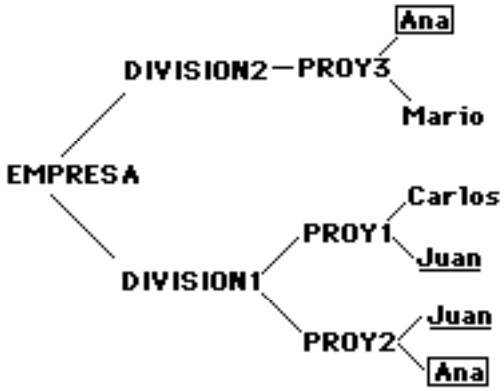
3. Modelo de Datos Jerárquico

A semejanza del modelo de datos de redes, el modelo jerárquico trabaja sobre tipos de objetos y las asociaciones entre ellos. El modelo jerárquico sin embargo, difiere del de redes en el sentido de que restringe a que cada tipo de objeto tenga a lo sumo un tipo de objeto **padre** (propietario). Un tipo de objeto padre puede sin embargo tener más de un tipo de objetos **hijo** (miembro).

En el modelo jerárquico, los **registros** (objetos) se ven como nodos de un árbol. Los objetos se organizan en una asociación tipo **padre-hijo**. En esta asociación un objeto padre puede tener varios objetos hijos.

Así, en la siguiente figura los registros DIVISION son padres de los registros PROYECTOS siempre y cuando los proyectos sean administrados dentro de las DIVISIONES respectivas.

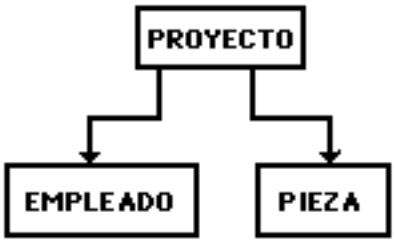
La siguiente figura ilustra una consecuencia importante de este hecho.



Los objetos que pertenecen a más de un padre deben aparecer más de una vez en la Base de Datos. Así, Juan trabaja en PROY1 y PROY2, y Ana en PROY2 y PROY3. En el caso del modelo de redes, sólo debe aparecer una vez. Por esta razón **el modelo jerárquico puede causar problemas cuando se desean modelar asociaciones del tipo M-N ya que es a veces necesario duplicación de los datos** .

La base de datos jerárquica puede representarse por medio de un diagrama similar al de redes. Es decir, los tipos de objeto se representan por rectángulos y las asociaciones padre-hijo por medio de flechas.

La siguiente figura es una estructura jerárquica en que el padre tiene varios tipos de objetos hijo.



Los hijos de PROYECTO son los EMPLEADOS que trabajan en el PROYECTO así como las PIEZAS usadas en el mismo. Los objetos hijos se encuentran al mismo nivel.

En muchas implementaciones el orden de los hijos es significativa ya que los registros de la

Base de Datos se recuperan según el orden de estos objetos.

4. Conclusiones

Para concluir el presente capítulo es importante destacar lo siguiente. A pesar de que los SABD más numerosos actualmente son relacionales, existen grandes Bases de Datos cimentadas sobre SABD jerárquicos y de redes. Por ejemplo se podría hablar de la Base de Datos SIGNE que sirve para la administración de los asuntos escolares de la provincia de Quebec, en Canadá. Esta es una Base de Datos de redes con cerca de 40 millones de registros.

También, se podría mencionar el Sistema MUMPS, desarrollado en el MIT. Este SABD jerárquico es utilizado para la administración hospitalaria de muchos hospitales en los Estados Unidos.

Sin embargo, los SABD jeráquicos y de redes, son bastante complejos para el ABD. Basta revisar la cantidad de manuales que acompañan al IMS.

Por ejemplo, en los sistemas jerárquicos, uno de los principales problemas reside en la rigidez de los mismos. Se debe tener cuidado en organizar la Base de Datos según un esquema correspondiente a los accesos más frecuentes. En este modelo sólo se puede llegar a los elementos terminales vía los elementos raíz.

Un problema aun más grave se presenta cuando se hacen consultas del tipo: " **Dar la lista de todos los padres que poseen raíces de un tipo dado**". Entonces se requerirá de un recorrido total de la Base de Datos y si estos recorridos son numerosos, el uso de la misma será más costosa que un sistema de archivos clásico.

En cuanto a los SABD de redes su rigidez se mantiene; además, son más complicados en su uso que los jerárquicos.

Actualmente se pueden encontrar SABD jerárquicos y de redes disfrazados de relacionales. Esto se logra por medio de una interfaz que simula un lenguaje de consultas relacional. Así, para el usuario no informático es como si se tratara de un SABD relacional. Sin embargo, aquí se acaba toda similitud, pues para el ABD, la complejidad de implementar una Base de Datos se mantiene.

Fuera de los SABD basados en los modelos de datos vistos hasta ahora, existen otros basados en otros modelos. Es el caso del SABD semántico **SIM** (Semantic Information Manager) de UNISYS. Este SABD, el único en su tipo, simplifica la tarea de modelaje, pues trabaja a un nivel superior que los SABD anteriores. En efecto, además de las posibilidades de recuperación y manipulación de la Base de Datos, acepta un mayor significado a la información almacenada, ya que va a permitir definiciones más complejas entre conjuntos de entidades.

La semántica se refiere al estudio del significado. El SIM, como SABD semántico tiene la posibilidad de capturar más significado de los datos por medio de los objetos de la Base de Datos que emulan los objetos del mundo real. Además de permitir las asociaciones 1-1, 1-N, M-N, brinda los significados para predefinir asociaciones entre varios tipos de datos.

También, como otros sistemas, el sistema SIM permite al usuario definir rangos y valores y ser verificados cada vez que se consulta la Base de Datos.

La definición, el mantenimiento, la recuperación y la actualización de la Base de Datos, se hace por medio de una familia de programas llamada **InfoExec** (Information Executive).

A todas luces y siendo el primer SABD basado en el modelo de datos Entidad-Asociación (ver el ejemplo del primer capítulo), su poder es notorio y se le augura una gran aceptación por parte de los diferentes usuarios de Bases de Datos.

IV Lenguaje de Consultas SQL

1. Introducción

En este capítulo se va desarrollar un ejemplo de la creación y mantenimiento de una Base de Datos, usando para ello el estándar **SQL**.

El lenguaje de consultas SQL (**S**tructured **Q**uery **L**anguage) fue diseñado, a mediados de los años setenta por D.D. Chamberlin en el laboratorio de investigación de la IBM, en San José, California. Inicialmente se llamaba SQUARE y servía como lenguaje de consultas del SABD **R**. Posteriormente evoluciona a otros sistemas como SEQUEL, DB2 y SQL/DS.

Para mejor entender las características del SQL, se va a construir una Base de Datos relacional para la administración del transporte de mercadería a través del territorio nacional, a solicitud de una Compañía de Transportistas. Esta Compañía posee para tal fin camiones de diferente tipo. Por ejemplo, se tienen camiones cisterna para el transporte de líquidos: leche, aceite, gasolina, etc. También, se tienen camiones ganaderos, planos, vagonetas, etc.

Para efectuar un envío de mercaría de una ciudad a otra, el cliente hace la solicitud de envío caracterizada por: el tipo de mercadería y el peso de ella. A partir de esta información y en función de los camiones disponibles, se le asigna al cliente un camión en una fecha determinada.

Para modelar esta situación se podría usar el siguiente esquema relacional:

CAMION(#PLACA,#EMP,MODELO, TIPO,CAPACIDAD)
CLIENTE(#CLIENTE,NOM-CLI, CIUDAD)
TRANSPORTE(#CLIENTE,CIUDAD-O, CIUDAD-D,#PLACA, FECHA)
PEDIDO(#CLIENTE,TIPO-MERCADERIA, PESO-MERCADERIA,MONTO)
EMPLEADO(#EMP, NOMBRE, DIRECCION, PUESTO, #DEP)
DEPARTAMENTO(#DEP, NOM-DEPARTAMENTO)

2. Descripción de los Datos

Como se vió en el primer capítulo, antes de que se puedan manipular las relaciones (actualizar, introducir y borrar tuplos) éstas se deben definir.

A modo de ejemplo se va a definir la relación **TRANSPORTE**, usando el SQL :

```
CREATE TABLE TRANSPORTE
  (#CLIENTE   SMALLINT NOT NULL,
   CIUDAD-O        CHAR(20),
   CIUDAD-D        VARCHAR(20),
   #PLACA      INTEGER,
   FECHA       INTEGER)
```

En este ejemplo, SMALLINT indica que el atributo #CLIENTE corresponde a un número entero pequeño, sin decimales. NOT NULL significa que el valor no puede ser omitido, es decir, no se pueden considerar valores nulos. CHAR(20) significa que la información referente a la ciudad de origen corresponde a un número máximo de 20 caracteres. Por su parte, VARCHAR(20) significa que el atributo ciudad de destino puede tener un número de caracteres variable, sin embargo, los 20 caracteres se respetan en toda impresión de informes.

3.Inserción de la Información

Una vez hecha la descripción de las relaciones, las mismas se encuentran vacías. Para introducir

los tuplos en las respectivas relaciones se puede utilizar el mandato **INSERT**, brindado por el SQL.

Por ejemplo, si se desea introducir un tuplo en la relación **CAMION** se haría de la siguiente forma: **INSERT INTO** CAMION(C2450,234,Toyota, cisterna,10)

Por otra parte, si se desea cambiar en la relación **TRANSPORTE** las ciudades de destino del cliente #34, de San Carlos por Guápiles, se podría hacer usando el mandato **UPDATE** de la siguiente forma:

```
UPDATE TRANSPORTE
SET  CIUDAD-DESTINO= 'Guápiles'
WHERE #CLIENTE = 34;
```

Finalmente, para suprimir tuplos de una relación se puede hacer, usando el mandato **DELETE.**

4. Consultas a la Base de Datos

En SQL se tiene un bloque básico de respuesta a las consultas. Este es conocido como el **SELECT-FROM-WHERE** básico. Su estructura es la siguiente:

```
SELECT X1, X2,..., Xk
FROM R
WHERE E
```

Este bloque dará como resultado la relación **(R%E)[X1,X2,...,Xk]**.

A continuación se va a responder a una serie de consultas con el fin de mostrar las características del SQL.

1. Dar los números de placa de los camiones de tipo ganadero cuya capacidad sea superior a las 10 toneladas.

La respuesta a dicha consulta queda de la siguiente forma.

```
SELECT  #PLACA
FROM    CAMION
WHERE   TIPO='ganadero'
AND     CAPACIDAD >10;
```

Sea la relación **R(**A_1, A_2,...,A_n**)**. Si se deseara imprimir toda la relación, se podría escribir, en vez de

```
SELECT  A1, A2,...,An
FROM    R
WHERE   E
```

lo siguiente:

```
SELECT  *
FROM    R
WHERE   E
```

2. Dar los nombres de los clientes que se les transportará mercadería desde San Carlos el 15 de setiembre de 1989.

Esta consulta ya requiere de dos relaciones **TRANSPORTE** y **CLIENTE**
Para responder a esto el SQL permite anidar los bloques de calificación, según se aprecia en la estructura siguiente:

```
SELECT      NOM-CLI
FROM        CLIENTE
WHERE       #CLIENTE  IN
   (SELECT  #CLIENTE
        FROM    TRANSPORTE
        WHERE   CIUDAD-0=  'San-Carlos'
         AND FECHA=150989);
```

También, podría utilizarse la siguiente expresión equivalente:

```
SELECT NOM-CLI
FROM   CLIENTE C, TRANSPORTE T
```

```
WHERE C.#CLIENTE = T.#CLIENTE
    AND T.CIUDAD-0= 'San-Carlos'
    AND T.FECHA=150989;
```

La expresión CLIENTE C significa que C es una variable tuplo que va a recorrer la relación CLIENTE. Y por otra parte, el término C.#CLIENTE es equivalente al término de proyección C[#CLIENTE] visto en los lenguajes algebraicos.

3. Dar los nombres de los empleados que le han transportado mercadería a la Fábrica de Cemento antes del 10 de diciembre de1989.

Esta consulta tiene cierto grado de complejidad pues involucra a cuatro relaciones. En SQL podría verse de la siguiente forma:

```
SELECT NOMBRE
FROM   CLIENTE C, TRANSPORTE T
          CAMION D, EMPLEADO E
WHERE C.#CLIENTE = T.#CLIENTE
   AND  D.#EMP = E.#EMP
   AND  D.#PLACA = T.#PLACA
    AND C.NOM-CLI='Fábrica de Cemento'
    AND T.FECHA≤101289;
```

Con estas consultas se puede apreciar el poder de SQL. Sin embargo, algunas consultas son más complejas que éstas y requieren de herramientas propias de un lenguaje de alto nivel. A continuación se verá cómo se resuelve este problema.

5. SQL Inmerso

Ahora, si se solicitara responder a la siguiente consulta: **¿Cuál es el peso total de la mercadería transportada por todos los camiones cisterna?**; con ayuda de los lenguajes relacionales que se han visto, no se podría hacer.

Estos lenguajes sólo permiten seleccionar la información que se requiera, siempre y cuando no se ejecute ningún cálculo sobre ella. Frente a esta situación, el uso de estos lenguajes necesita el construir, en una primera fase, una función de cálculo.
La primera fase podría efectuarse elaborando una relación con los componentes #PLACA, CAPACIDAD, en donde TIPO = 'cisterna', a partir de la relación **CAMION.**

R = (CAMION %(TIPO='cisterna'))
[#PLACA,CAPACIDAD**].**

Luego, se deberían sumar los valores de la columna CAPACIDAD. Esto puede hacerse, ya sea usando un **lenguaje anfitrión**, en donde los tuplos se manipularían uno después del otro, o bien dotando a los lenguajes relacionales de funciones de cálculo.

Entre las funciones de cálculo del SQL se tienen

SUM (suma de valores),
COUNT (contador de valores),
AVG (promedio),
MAX (máximo de un conjunto),
MIN (mínimo de un conjunto).

Con respecto al SQL, éste acepta a los lenguajes de alto nivel COBOL, PL/1, FORTRAN, como anfitriones.

El siguiente es un fragmento de un programa PL/1 con SQL inmerso.

```
1   $DCL    #PLA CHAR(10);
2   $DCL     NOM-C CHAR(30);
3   $DCL     #EMPLEA  FIXED BIN(4);
    $DCL ....;
    $DCL .....;
20  IF #EMPLEADO > 1000 THEN
....
45  $SELECT NOM-CLI, #EMP
46    INTO $NOM-C, $EMPLEA
47  FROM CLIENTE, EMPLEADO
48  WHERE  #EMP =$EMPLEA;
.....
98  PUT SKIP LIST (NOM-C,EMPLEA);
```

En este caso se debe comentar lo siguiente.

Primero, las instrucciones del SQL llevan de prefijo un signo $, con el fin de ser reconocidas por el precompilador llamado RDS. Segundo, una proposición ejecutable de SQL puede aparecer en cualquier lugar del programa. Tercero, las proposiciones de SQL pueden incluir referencias a variables de PL/1, siempre prefijando con el símbolo $.

Así, el poder de un lenguaje de consultas inmerso dentro de un lenguaje de alto nivel, es evidentemente de mayor nivel y poder.

6. Vistas

Con mucha frecuencia, para facilitar las operaciones, los usuarios prefieren trabajar con una sola relación y no con varias. El SQL permite, crear **vistas**, es decir una relación derivada, como expresión algebraica, de relaciones de base o de otras vistas. En el caso del SQL, así como en otros sistemas, las vistas son virtuales, es decir son las expresiones las que residen en la memoria y no las relaciones. Cuando se solicitan las vistas, el sistema en forma automática genera la relación.

En el siguiente ejemplo se va crear una vista llamada **CAMION1** con tres atributos: PLACA, AÑO, NOMBRE-EMP a partir de dos relaciones de base **EMPLEADO** y **CAMION**.

```
CREATE VIEW CAMION1
     (PLACA,AÑO,NOMBRE-EMP)
AS SELECT      #PLACA,
               MODELO, NOMBRE
     FROM EMPLEADO E, CAMION C
     WHERE E.#EMP =C.#EMP
     AND C.#CLIENTE =35;
```

Obsérvese que los atributos #PLACA, MODELO y NOMBRE toman en la vista, los nombres: PLACA, AÑO y NOMBRE-EMP, respectivamente. Sin embargo, pueden ser los mismos atributos.

7.Seguridad e Integridad de los Datos

Un aspecto muy importante en el control y administración de la información es la referente a la seguridad e integridad de los datos. De ahí que algunos programas de aplicación requieran niveles de privilegio de acceso. El SQL posee mandatos que permiten brindar privilegios sobre las relaciones, así como eliminarlas, mediante los mandatos **GRANT** y **REVOKE**.

A modo de ejemplo supóngase que se desea autorizar a cualquier persona el privilegio de lectura sobre la relación **TRANSPORTE**. Esto quedaría así:

```
GRANT    SELECT
ON TRANSPORTE
WHERE    PUBLIC
```

Si por su parte, el usuario Salas le desea eliminar el privilegio de escritura que previamente le había dado al usuario Torres sobre la relación CAMION, lo haría de la siguiente forma:

SALAS:
REVOKE INSERT ON CAMION
FROM TORRES.

Otros aspectos pueden tomarse en cuenta en la seguridad. Entre ellos, la **identificación del usuario**. Es decir, el sistema debe solicitar a cualquier usuario que afirma ser el usuario X, que pruebe la verdad de esa afirmación. A modo de ejemplo, sea lo siguiente:

El usuario digita
GONZALEZ
El sistema responde
HOLA GONZALEZ, INTRODUZCA
CONTRASEÑA

Se digita contraseña errónea. El sistema responde.
CONTRASEÑA INCORRECTA. PROBAR DE NUEVO
Se digita contraseña correcta. El sistema responde
CONTRASEÑA ACEPTADA, DIGITE MANDATO.

Algo así ocurre cuando se utilizan los cajeros automáticos.

Otras formas de mantener la seguridad de los datos es por medio de la **criptografía**, que consiste en codificar en clave parte de la información de la Base de Datos. Esto significa que si un usuario desea accesar a la misma, sin tener el derecho, sólo verá un conjunto de 0 y 1 sin ningún sentido para él.

Por otra parte, qué ocurriría cuando se producen problemas, por ejemplo fallas del Sistema Operativo o alguna falla de harware?. Se encuentra el usuario a merced de estas fallas o por el contrario se encuentra protegido?

El SQL posee varios mecanismos de recuperación, manteniendo varias copias de los tuplos y relaciones. Este proceso es dinámico y permanente. Esto garantiza la recuperación de la Base de Datos luego de una falla; sin embargo, esto sacrifica almacenamiento secundario.

Con respecto a las imposiciones de integridad, el SQL ofrece varios mandatos. A modo de ejemplo, supóngase que se desea asegurar que el tipo de un carro sólo puede tener cuatro valores:
ASSERT II1 ON CAMION: TIPO **IN**
('ganadero', 'cisterna', 'plano', 'vagoneta').

El SQL fue escogido para introducir un poco al lector en lo que significa la creación y mantenimiento de una Base de Datos, sin entrar en mucho detalle. Se escogió el SQL pues, tanto a nivel de Mainframes como Micros se ha convertido en un estándar ANSI/SPARC. Es un lenguaje mucho más poderoso y de más fácil manejo que los SABD jerárquicos y redes como el IMS, DBTG, ADABAS o IDMS.

V Tendencias actuales de los Sistemas de Bases de Datos

1.Introducción

En la actualidad, los SABD son numerosos y ofrecen posibilidades poderosas y variadas. Sin embargo, se encuentran limitados por tres razones fundamentales.

En primer lugar, se puede decir que los SABD se han concebido para administrar datos de gestión, es decir estructurados de forma regular, sobre todo bajo forma de relaciones o archivos. Pero, muchos datos no pueden ser administrados por uno de estos SABD. Es el caso de los textos, las imágenes, los resultados de medida y documentos. Se debe entonces mejorar los SABD actuales para poder manejar este tipo de datos, que se han dado en llamar **datos multi-medios**.

En segundo lugar, los SABD actuales sólo permiten recuperar datos almacenados en la Base de Datos. No son capaces de "razonar" con el fin de inferir, a partir de datos y leyes generales conocidas, nuevos datos. Se debe entonces mejorar los SABD con el fin de permitir el almacenamiento de mayores cantidades de reglas (es decir reglas generales) y poder usar a la vez las reglas y los datos para inferir nuevos datos, y nuevas reglas. También, los SABD actuales que se dicen cada vez más "**user friendly**" (amigables) están lejos de presentar interfaces sencillas, accesibles a los no informáticos.

Finalmente, el modelo relacional permite una representación muy simple de las estructuras de datos pero posee una semántica muy pobre. Esta insuficiencia puede neutralizarse con la adición de conceptos derivados de la inteligencia artificial y del desarrollo de software.

Los SABD comercializados actualmente sólo manipulan datos con una cierta estructura llamados registros y que a su vez constituyen los archivos y las Bases de Datos. Es por eso que investigadores se están interesando actualmente en la concepción de sistemas que puedan manipular datos multi-medios.

2. Sistemas de Bases de Datos Expertos

Hoy en día la Inteligencia Artificial (IA), las Bases de Datos y los Lenguajes de Programación se encuentran bastante relacionados en muchas investigaciones que se están llevando a cabo en Computación.

Para expresar la riqueza semántica de la abstracción de una empresa (toda organización real como un banco, universidad, fábrica,...), los modelos conceptuales no son suficientemente poderosos. No toman en cuenta uniformemente las estructuras de datos (esquemas), la manipulación (transacciones) y el significado. Por esta razón, los investigadores han querido introducir la IA para producir herramientas aplicables al modelaje conceptual de una Base de Datos y para poder así definir un cuadro de integración formal de la especificación y de la manipulación de una Base de Datos.

Las herramientas de la IA pueden aplicarse a todo dominio de representación de conocimientos. Entre éstas herramientas que actualmente se utilizan se pueden citar:

- las redes semánticas
- los sistemas de producción
- los esquemas
- la lógica de primer orden

- los conjuntos borrosos

Las Bases de Datos y las Bases de Conocimiento, representan dos métodos para la aprehensión del mundo real.

Con las Bases de Datos, por lo general, se tendrán grandes volúmenes de datos con la misma estructura. En cambio, con las Bases de Conocimiento, el volumen de datos generalmente es reducido pero prácticamente cada ocurrencia de cada entidad tiene su propio formato descriptivo de forma que el factor de repetición es casi inexistente.

Así, sería importante poder hacer converger esta dos ideas de manera tal que se puedan tener Bases de Datos que sean voluminosas en forma virtual, es decir, que realmente la información no resida en disco, pero sí reglas que puedan generar, a partir de datos que se podrían llamar iniciales, la información requerida

El mecanismo de inferencia es primordial en todo sistema de representación de conocimientos para poder deducir nuevos hechos independientemente de una intervención externa.

Esta diferencia ha llevado a la investigación en dos vías:

En las Bases de Datos, se ha hecho énfasis en los mecanismos sofisticados de estructuración de la información (esquema conceptual) y sobre los mecanismos eficientes de almacenamiento y de acceso a los datos en disco y no sobre los descriptores.

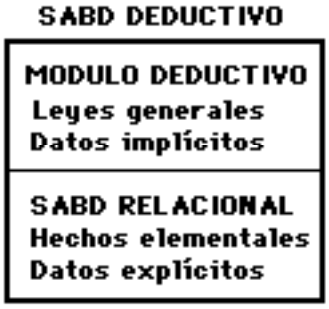
En el caso de las Bases de Conocimiento, el acento se ha puesto sobre los descriptores, sobre técnicas sofisticadas que permitan representar muchos conocimientos a partir de un número pequeño de objetos, así como sobre la semántica de los datos y los mecanismos de inferencia.

3. Bases de Datos Deductivas

Con el fin de mejorar la capacidad de manipulación y generación de nuevos datos, actualmente se trabaja en la idea de incorporar, a un SABD relacional convencional, un módulo deductivo escrito en un lenguaje de programación lógica como PROLOG.

El módulo deductivo implantado sobre un SABD relacional estaría encargado de la administración de las leyes generales mientras que el SABD se encargaría de los hechos elementales.

La siguiente figura muestra la arquitectura de una Base de Datos deductiva



Se puede decir que un **SABD Deductivo** es un sistema que permite derivar nueva información a partir de la que fue explícitamente introducida por el usuario en la Base de Datos.

Si los lenguajes utilizados en los dos módulos son los mismos (lenguaje predicativo para el SABD relacional y lenguaje de la Lógica de Primer Orden para el módulo deductivo), el SABD aparecerá como un módulo del interpretador de la lógica, especializado en la manipulación de hechos elementales.

Ahora se va a estudiar un poco las Bases de Datos Deductivas y los Sistemas de Integridad

3.1 Imposición de Integridad y Regla de Deducción

Una ley general puede percibirse: Ya sea como una imposición de integridad que pueda implicar el rechazo de un estado de la Base de Datos si la imposición no se verifica (en este caso no podemos hablar de una Base de Datos Deductiva). O bien, como una regla de deducción, permitiendo derivar informaciones implícitas y produciendo un estado coherente de la Base de Datos.

El problema de la utilización de una ley general como imposición de integridad es un problema abierto.

Regla de derivación y regla de generación

La noción de regla de deducción puede afinarse en

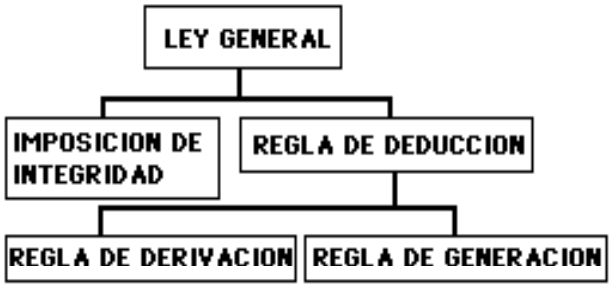
- regla de derivación y
- regla de generación

Una **regla de derivación** se asocia a la operación de consulta a la Base de Datos, permitiendo la derivación de información implícita. Esto corresponde a la ejecución de una regla de deducción en "encadenamiento hacia atrás".

El establecimiento de reglas de derivación puede hacerse en forma interpretada siguiendo la ligazón estrecha entre el proceso deductivo y el proceso de búsqueda de datos elementales.

Una **regla de generación** se asocia a la operación de actualización de la Base de Datos permitiendo el hacer explícita información deducible. Esto corresponde a la ejecución de una regla de deducción en "encadenamiento hacia adelante"

Los diferentes tipos de reglas de una ley general se resumen en el siguiente cuadro:



El lector puede ampliar estos conceptos con el documento de Sistemas Expertos, del Dr. Claudio Gutiérrez **[Gutiérrez88]**, específicamente la parte que trata de los tipos de encadenamiento.

4. Bases de Datos Multi-medios

Se trata de ampliar los SABD actuales con el fin de poder manejar objetos complejos como textos, gráficos, imágenes, almacenados bajo forma de células, series estadísticas resultantes de medidas, voz codificada, etc. El conjunto de estos objetos complejos, adicionado a las relaciones de los sistemas clásicos necesarios en la administración, debe conducir a poder almacenar y generar documentos generales compuestos de un conjunto ordenado de objetos complejos y relaciones. Tales documentos se usan mucho en automatización de oficinas.

Con respecto a la investigación en este campo, dos escuelas tienden a desarrollarse para concebir SABD multi-medios.

Un primer enfoque, representado particularmente por los constructores de INGRES y que actualmente está en estudio en el proyecto SABRE, tiende a usar un SABD relacional y a aportar mejoras, sobre todo, ampliando los tipos de datos administrados. Por

ejemplo, se espera partir de un sistema relacional y permitirle aceptar dominios mucho más ricos, definidos por el usuario bajo forma de tipos abstractos a partir de tipos de base del sistema o bien previamente definidos. Se podrá por ejemplo definir tipos que sean mensajes, cartas, segmentos, paralelogramas, círculos, etc.

Una segunda escuela, tal vez representada por los proyectos de IBM, trata de construir un nuevo SABD basado en un modelo de datos innovador, más adaptado a los documentos complejos. Así, el proyecto de IBM Heidelberg utiliza un modelo completamente recursivo, en donde cada objeto está compuesto de un conjunto de objetos a dos dimensiones. En consecuencia, una relación puede verse como un conjunto de atributos en donde algunos son a su vez relaciones (rompiendo la definición de primera forma normal). **[Gardarin85]**

5. Bases de Datos Distribuidos

Sintetizando lo expuesto en los capítulos anteriores, se puede decir que un SABD centralizado se compone de:

1. Un conjunto de transacciones.
2. Un **Módulo de Transacción** que controla la concurrencia de las transacciones.
3. Un **Módulo de Base de Datos** que administra la Base de Datos.
4. Una Base de Datos.

En los últimos años, gracias al desarrollo de las redes de computadores, ha ido tomando fuerza la idea de distribuir, en sitios, geográficamente alejados, los datos que componen una Base de Datos convencional. Esto también se ajusta más a la idea de descentralización que tienen muchas organizaciones.

Por otra parte, si una empresa requiere adquirir nuevo equipo, para dar abasto a sus necesidades de información, sólo se requiere conectar el nuevo equipo con el ya existente, sin necesidad de modificar las aplicaciones que se han desarrollado.

Se puede decir que una **Base de Datos Distribuidos** es una Base cuya información se encuentra localizada en sitios geográficamente distantes.

Por su parte, el software que permite que varios usuarios puedan interactuar con este tipo de Base de Datos se denomina **Sistema Administrador de Bases de Datos Distribuidos (SABDD)**

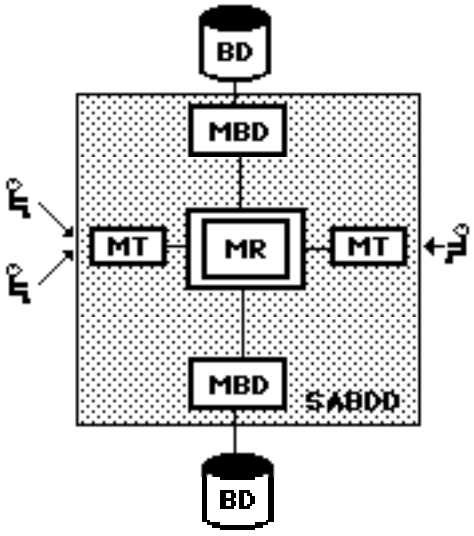
Con respecto a su arquitectura, el SABDD consiste de un conjunto de nodos y conexiones entre ellos.

En este caso, cada nodo representa un Módulo de Transacción o un Módulo de Base de Datos.

Por otra parte, cada Módulo de Transacción está asociado a un conjunto de transacciones, cuya concurrencia controla. Las conexiones entre nodos representan un módulo funcional: el **Módulo de Red**. Este implementa conexiones entre Módulos de Transacción y de Base de Datos

En una Base de Datos convencional, los valores y los nombres de los objetos son únicos. En cambio, en una Base de Datos Distribuidos, el valor de un objeto X (por ejemplo una relación) puede almacenarse en más de un sitio.

El siguiente diagrama esquematiza un SBDD



Con respecto al Módulo de Transación (MT en el dibujo), éste controla la concurrencia de transacciones. Cada transacción en un SABDD está controlada por un sólo MT. Cada MT traduce sus transacciones en programas en paralelo y éstos se ejecutan con ayuda de varios Módulos de Base de Datos (MBD en el dibujo). El control de esta ejecución distribuida la realiza el MT quién explota al máximo el paralelismo. El MT realiza la sincronización de la ejecución de sus transacciones con las ejecuciones de las otras transacciones del sistema.

Con respecto al Módulo de Base de Datos, éste en realidad es un SABD convencional. Responde a los mensajes de los MT, efectuando las operaciones requeridas en la Base de Datos de su localidad.

El Módulo de Red (MR en el dibujo anterior), por su parte, implementa conexiones entre MT y MBD. Sin embargo, no se permiten conexiones entre MT, ni entre MBD.

Actualmente se tienen varios **prototipos** de SABDD. Entre ellos, se pueden citar los sistemas INGRES DISTRIBUIDO, PRECI, SDD-1 y Sistema R*.

Debido a que en un ambiente distribuido se dan problemas de costo y beneficio en la transferencia de datos entre diferentes sitios y tomando en cuenta que desde cualquier localidad se pueden ejecutar transacciones, el sistema que deba soportar este ambiente es bastante sofisticado. Los problemas de concurrencia y de procesamiento, por ejemplo, son difíciles y caros de resolver.

Comercialmente se cuenta con varios SABDD distribuidos disponibles, pero de alta complejidad y muy caros.

Para muchas empresas es más viable contar con una Base de Datos centralizada y terminales geográficamente distantes, conectadas al sistema central.

6. Máquinas de Bases de Datos

Hasta principios de los años sesenta, la administración de las comunicaciones, la ejecución de los programas de aplicación y la administración de los datos en un sistema computacional, se encontraban dentro de un mismo componente, según se aprecia en la siguiente figura:

Ya para inicios de los años setenta aparecieron los **frontales** (**front-end machine**), que son componentes separados del sistema central y utilizados para la administración de las comunicaciones. Veáse la siguiente figura.

Como se ha visto en los capítulos anteriores, un SABD tradicional está constituido básicamente de un conjunto de programas de sistema y un conjunto de programas de aplicación.

Este enfoque no es completamente satisfactorio. En efecto, algunas Bases de Datos alcanzan volúmenes de varios giga-bytes (GB). En este caso, se tienen procesamientos que necesitan la transferencia masiva de datos entre los discos y la memoria central. Esto implica una saturación de la memoria central con datos no utilizables, y se satura el computador con ejecuciones de los datos involucrados.

A inicios de los años ochenta, y para evitar la saturación de memorias y procesadores, aparece la idea de descentralizar el procesamiento de datos. Esto se lograría trasladando una parte importante del SABD a un computador especializado, periférico del computador central, llamado **dorsal (back-end)**. Así, se buscaría desplazar los algoritmos de búsqueda y de actualización, lo más cerca posible de las memorias secundarias, según se aprecia en la figura siguiente.

El dorsal encargado de esto puede ser un computador tradicional con software específico; hablaríamos en este caso de **máquinas bases de datos (MAQBD)**.

Con respecto a las ventajas del enfoque MAQBD, se pueden citar las siguientes:

a. Sólo los datos necesarios se transfieren al computador central.

b. El procesador central no tiene que efectuar las búsquedas y las actualizaciones.
c. Ejecución mas rápida de las consultas que se hagan al SABD, debido a :
1.Especialización del dorsal en el procesamiento de las Bases de Datos.
2.Posibilidad de desarrollar equipos de tecnología especilizada que sean muy rápidos.
3.Mayor uso de las posibilidades del paralelismo, a nivel del procesamiento de datos yacceso a disco.

Esto sin embargo podría conducir a comunicaciones complejas y caras entre los dorsales y los computadores centrales. Por ejemplo, si tenemos sistemas que necesitan una comunicación para cada grupo de datos transferidos (por ejemplo tuplos), este costo de comunicación puede ser muy alto.

Aún así, las primeras experiencias realizadas con dorsales han dado en un factor de 5 en la reducción del tiempo de respuesta.

En cuanto a la arquitectura de las MAQBD, ésta puede ser: **autónoma** o **dorsal** de un computador anfitrión.

Se han definido hasta el momento cuatro tipos de MAQBD, no necesariamente disjuntos:

1. Máquinas con procesadores asociativos (enfoque desechado)
2. Máquinas con memorias asociativas (enfoque actual)
3. Máquinas paralelas (enfoque actual)
4. Procesadores VLSI (enfoque por desarrollarse)

En la actualidad se tienen varias MAQBD comercializadas. Entre ellas:

- DORSAL 32 de Copernique.
- CAFS de ICL.
- IDM de Britton-Lee.
- IDBP de Intel.

También se tienen prototipos en varias universidades y centros de unvestigación. Entre otros,

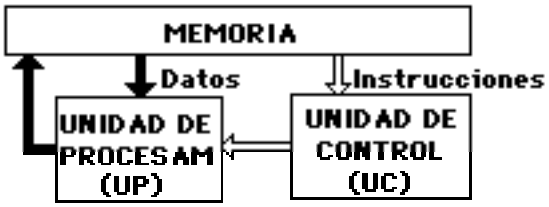
- SABRE, Universidad de París VI, Francia.
- VERSO, INRIA, Francia.
- RDBM, Universidad de Braunschweig, Alemania.
- DBMAC, CNR, Italia.
- DSDM,GDS, Japón.
- DIRECT, Universidad de Madisson, Wisconssin.
- DBC, Universidad de Columbus, Ohio.
- CASSM, Universidad de Gainseville, Florida.

Las MAQBD son, en menor o mayor grado, computadores a varios procesadores con hardware especializado, que administran los datos para uno o varios computadores centrales, llamados anfitriones.

Mientras que la mayoría de las máquinas comercializadas ejecutan las consultas a la Base de Datos una tras otra, en forma secuencial, es posible construir máquinas que ejecuten varias consultas en paralelo y que procesen varios flujos de datos en paralelo.

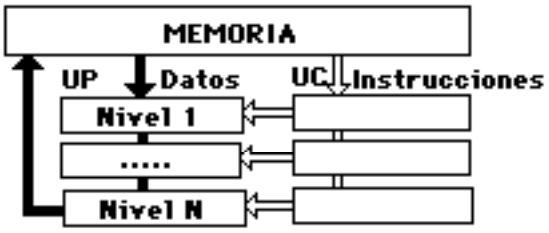
En arquitectura de computadores se puede hablar de cuatro tipos de computadores, según la clasificación de Michael Flynn **[Sansonnet88].**

En primer lugar se tienen las máquinas **SISD (Single Instruction Single Data stream)**. Éstas son conocidas como **máquinas de von Neuman**, o **máquinas secuenciales** y ejecutan una instrucción a la vez, según se aprecia en la siguiente figura.

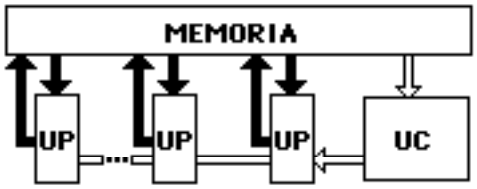


La mayoría de los computadores instalados en el país, incluyendo los micros, son del tipo SISD.

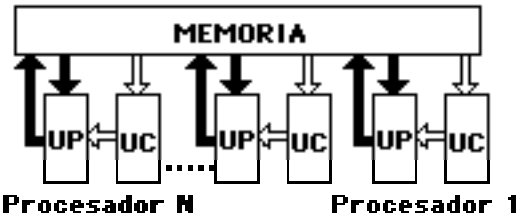
Las máquinas MISD (Multiple Instruction Single Data stream), por su parte, ejecutan varias instrucciones de búsqueda en forma simultánea sobre un único flujo de datos. Ésta máquinas son también llamadas del tipo **pipeline**. En este caso el procesador se compone de varios niveles, encargado cada uno de una parte de la instrucción proveniente de la memoria, según se aprecia en la siguiente figura.



En cuanto a las máquinas **SIMD (Single Instruction Multiple Data stream)**, éstas ejecutan una misma instrucción sobre varios flujos de datos posibles. Ver la siguiente figura.

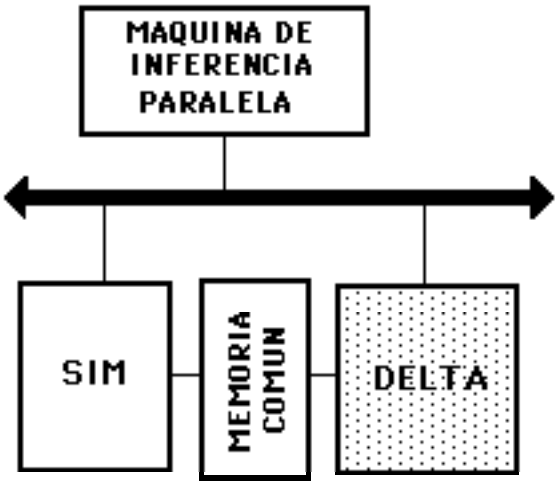


Finalmente, se tiene la cuarta categoría denomina **máquinas MIMD (Multiple Instruction Multiple Data stream)** Las MIMD ejecutan varias instrucciones sobre varios flujos de datos posibles, según se aprecia en la siguiente figura.



Entre las MAQBD, se tienen de los cuatro tipos. Esto debido a que no existe un estándar para el diseño de una MAQBD. En efecto, se tienen MAQBD, que son máquinas convencionales con software específico, como el caso de SABRE, o por el contrario, MAQBD en todo sentido de la palabra, más orientado a nivel de hardware, como es el caso del DORSAL 32.

La MAQBD del tipo MIMD más avanzada que se conoce es la denominada **DELTA**. Construida por el Centro de Investigación ICOT en el Japón, es la unidad de base del computador de quinta generación.
La primera propuesta de una arquitectura de la quinta generación se puede apreciar en la siguiente figura.



Esta arquitectura se compone de:

1. Una máquina de inferencia (SIM), secuencial por el momento.
2. Una **MAQBD** relacional (DELTA) que puede interconectarse vía una red local o vía una memoria común.

En esta arquitectura la MAQBD, quién ofrece por interfaz externa el álgebra relacional, aparece como la unidad central del computador de quinta generación.

DELTA puede administrar 65,535 relaciones de hasta 128 Mb y 255 atributos, y ejecutar hasta 254 transacciones en forma concurrente.

Según las primeras experiencias, en los próximos años se espera un aumento considerable en el uso de los dorsales.

Actualmente, muchos productores de computadores están incorporando procesadores especiales dentro de los sistemas que ofrecen. Estos procesadores, por lo general de varios Mega bytes de memoria, son llamados p**rocesadores de Bases de Datos**. Éstos cumplen básicamente con las mismas funciones de una máquina de Bases de Datos. Entre estos sitemas computacionales se pueden citar 3090 de IBM, la serie A de Unisys, ARIX 875, etc.

VI Consideraciones Finales

1.Síntesis

A través de los diferentes capítulos de este documento, se ha hecho un recorrido sobre lo que hoy en día se ha dado en llamar Tecnología de Bases de Datos.

El estudio ha abarcado tanto el nivel del diseño, creación e implementación de una Base de Datos, así como del software que las hace funcionar, es decir el SABD. También, se estudiaron las nuevas tendencias en este campo.

Sintetizando, se dirá que la Tecnología de las Bases de Datos permite administrar los datos de una organización, en forma integral. Además, elimina la rigidez impuesta por los archivos separados y permite un acceso a los datos en forma más natural.

A continuación, se resumen algunas de las ventajas y desventajas de la Tecnología de Bases de Datos.

1.1. Ventajas

Mayor información, a partir de los mismos datos.

Para mejor entender esta afirmación, supóngase que se tienen tres archivos separados denominados ESTUDIANTES, GRUPOS y DEPARTAMENTOS. En este caso sólo se puede derivar información a partir de los datos de un departamento, un grupo o un estudiante. No es posible obtener información que sea combinación de un departamento y un grupo, un grupo y un estudiante, etc. Esto, sin embargo, sí es posible en una Base de Datos. Esta es quizás la mayor ventaja de las Bases de Datos

Eliminación de la redundancia de los datos

En el caso del ejemplo anterior es posible que se tenga información sobre un estudiante, como puede ser su nombre, su fecha de nacimiento, tanto en el archivo ESTUDIANTES como en el archivo GRUPOS.
En una Base de Datos este no es el caso.

Independencia de los datos versus programas de aplicación

En efecto, es una característica importante del enfoque Bases de Datos el permitir reorganizar físicamente los datos, por razones de eficiencia, sin necesidad de alterar los programas de aplicación.

Mejor administración de los datos

Cuando se tiene un conjunto de datos centralizados es más fácil su mantenimiento. Así, el encargado de esta función en la Base de Datos puede especificar estándares sobre los datos y asegurar que todos los datos de la Base cumplen dichos estándares.

Representación de asociaciones entre tuplos

Esta es una característica importante, pues estando los datos centralizados y según las necesidades de la organización, éstos pueden relacionarse entre sí y obtenerse entonces nueva información.

1.2. Deventajas

Entre las desventajas de la Tecnología de Bases de Datos se pueden citar las siguientes.

Alto costo

Lo que se ha denominado SABD en este documento, se refiere a un software especializado y para un ambiente multiusuario. Así, la instalación de un SABD conlleva una serie de gastos respetables, como son la compra y el mantenimiento del SABD mismo, así como el requerimineto de hardware con mayor capacidad de almacenamiento y de procesamiento. Además, se va a dar un incremento en los costos de operación.

Existen, sin embargo, SABD para microcomputadores, como es el caso del DBASE IV, de un costo relativamente bajo, pero sin muchas de las características propias de un SABD para equipos grandes como es el caso del SQL, el ORACLE y el INGRES.

Complejo

Si bien es cierto una Base de Datos es de fácil manejo para un usuario terminal (no informático), para un ABD su diseño y mantenimiento son más complejos, cuanto más voluminosa sea ésta.

Recuperación más difícil

Cuando se tienen Bases de Datos muy grandes, el problema de la recuperación, luego de una falla, es un asunto bastante delicado. Algunos sistemas, por ejemplo, duplican completamente la Base de Datos, usando lo que se da en llamar discos espejo. Pero esto es bastante costoso.

A pesar de que el SABD posee mecanismos que permiten dejar la Base de Datos en un estado coherente, éste puede ser, dependiendo de los puntos de control, un estado que dejó por fuera una serie de transacciones, y pueden significar grandes pérdidas para la organización.

Vulnerabilidad mayor en cuanto a fallas

Finalmente, se puede decir que la integración, y por ende la centralización, aumenta la vulnerabilidad. Por ejemplo, la falla en un componente puede dejar fuera de servicio el sistema completo. Esto es realmente crítico cuando se tiene una organización que depende, para su funcionamiento, de la Base de Datos.

2. ¿ COMO MEJORAR EL DISEÑO ?

En cuanto mayor sea el volumen de datos que constituye una Base de Datos, mayor importancia adquiere el diseño y por ende el aspecto de los Sistemas de Información.

En la llamada primera generación de Bases de Datos (redes y jerárquicos), el Administrador de la Base de Datos debía tomar en cuenta, a la hora del diseño de una Base de Datos, los métodos de organización de los datos y los caminos de acceso a los mismos.

Actualmente esta labor ha descendido a nivel del SABD y por lo tanto no forma parte importante del diseño. Sin embargo, un imperativo para lograr un buen diseño en Bases de Datos es el referente al análisis de los requerimientos. Es decir, la construcción de un modelo conceptual que especifique los requerimientos, el desarrollo y optimización de un esquema de caminos de acceso lógico. Para este fin, existe una cantidad innumerable de metodologías.

A modo de ejemplo, se presentan los objetivos de una, debida a N. Roussopoulos y R. Yeh de la Universidad de Maryland **[Roussopoulos84].**

Esta metodología se divide en cuatro etapas:

1. Análisis del medio ambiente y los requerimientos.
2. Análisis del sistema y las especificaciones.
3. Modelaje conceptual de los datos.
4. Derivación de los caminos de acceso lógicos

del esquema.

El objetivo de la **primera fase** es el entendimiento del estado actual de la empresa y la recolección de toda la información necesaria para los pasos subsiguientes. La información de los requerimientos puede ser usada en la fase de modelaje conceptual para así generar el esquema conceptual. Los requerimientos del procesamiento de datos pueden ser utilizados en la fase de derivación de caminos de acceso en la cual la optimización lógica y los programas de aplicación son diseñados.

El objetivo de la **segunda fase** es identificar las diferentes aplicaciones que se le podría suministrar a la Bases de Datos. La entrada a esta fase es el análisis del medio ambiente con lo cual se obtiene un entendimiento claro del sistema. La salida es un grupo de tareas para ser ejecutadas durante la operación de la Bases de Datos.

El objetivo de la **tercera fase** es el traslado de los conocimientos recolectados en las fases 1 y 2 en una representación formal, es decir, un modelo conceptual. Aquí, se interesa a las características de las entidades aplicadas y de las asociaciones entre ellas y no en la representación física del computador.

En la **cuarta fase** se tienen varios objetivos:

a. Traducir el esquema conceptual de la organización en un modelo estructurado de datos.

b. Generar los caminos lógicos de acceso de las tareas.

c. Integrar todos estos caminos dentro de un esquema que modela estas relaciones y su valor de uso.

ch. Optimizar el esquema de caminos de acceso.

3. CONCLUSION

Después de la década de los sesenta, los SABD se han desarrollado en forma importante. En cada una de las generaciones se integran nuevas funciones haciendo cada vez su utilización, más sencilla.

Supóngase que se pueden evaluar los alcances de la informática en nuestro país por medio de un termómetro imaginario numerado de 0 a 100. Si el termómetro marca 0, significa que la informática no ha tenido ninguna incidencia en el país; si por el contrario marca 100, significa que se llegó al grado de saturación y que la informática como tal ha cumplido plenamente con lo que se esperaba de ella.

Para lograr que el termómetro marque 100, o se aproxime considerablemente, se requiere que entren a jugar una serie de variables. Sin embargo, una variable fundamental para que el termómetro marque un número cercano a 100, es la de poseer información confiable y poder recuperarla y transmitirla en forma eficiente.

Para lograr esto adecuadamente se requiere del buen diseño e implementación de Bases de Datos. Como bien se sabe, este término ha sido manipulado en forma un poco superficial pues se ha llegado incluso a confundir una cinta magnética con una Base de Datos.

Muchas de las actividades humanas requieren de información exacta y recuperación eficiente y rápida.

Por ejemplo, si se pudieran tener datos confiables sobre rotación de cultivos, manejo de suelos, dietas de ganado, épocas de siembra, condiciones ecológicas, etc., la agricultura se vería realmente favorecida.

Se podría también hablar de los beneficios que sobre la medicina, la educación, y las ingenierías, tendría una utilización adecuada de las Bases de Datos. Por otra parte, algunas

personas, detrás de diseños inadecuados en los sistemas que desarrollan, solicitan equipo cada vez más costoso y software cada vez más caro, como los SABD. Es cierto que algunas aplicaciones van a requerir necesariamente equipos muy poderosos, pero también es cierto que en muchos casos los equipos se encuentran mal utilizados.

Es importante resaltar que si una empresa posee un sistema de archivos para administrar su información y que funciona relativamente bien, el traslado del mismo a un ambiente de Bases de Datos debe hacerse con cautela, pues el tener un software más poderoso para la administración de la información no implica necesariamente que se tendrá un sistema de mayor rendimiento, sin no se hacen las previsiones del caso. Esto se puede lograr contando con personas conocedoras del campo, así como buscar la formación adecuada del recurso humano de la empresa.

Este documento no pretende ser exhaustivo en lo que a la Tecnología de las Bases de Datos se refiere. Solamente un compendio de ideas, consideradas las más importantes en este campo.

Se concluye el presente informe con un pensamiento de E. Green:

"**El valor del conocimiento no radica en su acumulación, sino en su empleo** "

Anexo: Proceso de Normalización

1. Primera Forma Normal (1FN)

En todo lo que sigue, considérese la siguiente Base de Datos:

CHOFER(CEDULA,
NOM-EMP,
DIR-EMP)
CAMION(#PLACA,
MARCA,
CAPACIDAD,
UBICACION)
ENVIO(#ENVIO,
CEDULA,
CIUDAD-ORIGEN,
CIUDAD-DESTINO)

Una relación se encuentra en **primera forma normal (1FN)**, si ninguno de sus atributos es a su vez una relación, es decir si todo atributo es atómico (no se puede descomponer).

Ejemplo. Las siguientes relaciones no se encuentran en **1FN**

R

#ENVIO	CAMION
50	(NISSAN,5)
52	(ISUZU,7)
63	(ISUZU,10)
78	(YOLYO,10)

S

MARCA	CAPACIDAD
NISSAN	(5,7,9)
YOLYO	(10)
ISUZU	(7,10)

Toda relación que no se encuentra en **1FN** puede convertirse en una, reemplazando el atributo compuesto por los elementales correspondientes, o bien, insertando los tuplos tantas veces como valores para un atributo se tenga. Así, en el caso del ejemplo anterior se tendría:

R

#ENVIO	CAMION	CAPACIDAD
50	NISSAN	5
52	ISUZU	7
63	ISUZU	10
78	YOLYO	10

S

MARCA	CAPACIDAD
NISSAN	5
NISSAN	7
NISSAN	9
YOLYO	10
ISUZU	7
ISUZU	10

También S podría representarse por medio de la siguiente relación:

S

MARCA	CAP1	CAP2	CAP3
NISSAN	5	7	9
YOLYO	10	@	@
ISUZU	7	10	@

@ valor nulo

2. Segunda y Terceras Formas Normales

(2FN/3FN/BCNF)

A continuación se va enunciar un resultado, llamado **teorema de descomposición**.

Sea **R** una relación que verifica la dependencia funcional X->Y. Entonces **R**(X,Y,Z) = **R**[X,Y] * **R**[X,Z]. **Ejemplo.** Sea **R** la siguiente relación: **CAMION**(#PLACA,MARCA,CAPACIDAD, UBICACION) y la dependencia funcional**:** MARCA->CAPACIDAD

[Recuérdese que aquí #PLACA representa la llave de **CAMION]**

CAMION

*PLACA	MARCA	CAPACIDAD	UBICACION
C13455	NISSAN	7	Alajuela
C2123	NISSAN	7	Heredia
C4321	NISSAN	7	Heredia
C22334	YOLYO	10	Alajuela
C23249	ISUZU	9	Cartago
C79898	ISUZU	9	Heredia

Como la relación **CAMION** verifica la dependencia funcional MARCA->CAPACIDAD; por el teorema de descomposición ésta se puede descomponer en las dos relaciones siguientes:

CAMION1

MARCA	CAPACIDAD
NISSAN	7
YOLYO	10
ISUZU	9

CAMION2

*PLACA	MARCA	UBICACION
C13455	NISSAN	Alajuela
C2123	NISSAN	Heredia
C4321	NISSAN	Heredia
C22334	YOLYO	Alajuela
C23249	ISUZU	Cartago
C79898	ISUZU	Heredia

En este caso no existen anomalías de alamcenamiento, es decir, de inserción, borrado y modificación. En efecto, no se tiene anomalías de inserción pues se puede introducir el camión FORD con 10 toneladas de capacidad en **CAMION1** sin ningún problema.

De igual manera, las anomalías de supresión no se dan, pues si se suprime el camión C22334 en **CAMION2**, no perdemos la información en **CAMION1** de que VOLVO tiene una capacidad de10 toneladas.

Y tampoco se dan las anomalías de modificación, pues si la capacidad del ISUZU pasa de 9 a 12 toneladas, esta modificación involucra un solo tuplo en **CAMION1**.

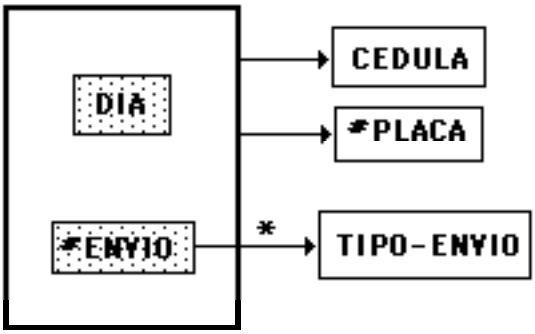
La descomposición permite aislar conceptos diferentes que se encontraban inicialmente en una sola relación. Una de las consecuencias de esto, es el poder enriquecer con nuevos atributos a las relaciones resultantes.

Así por ejemplo, en **CAMION1** se podría adicionar el atributo CONSUMO.

Sin embargo, la descomposción introduce un nuevo problema llamado "emigración del atributo". Este problema, resulta de la duplicación del atributo X en dos relaciones, mientras que antes de la descomposición sólo se tenía en la relación inicial. En realidad se ha reemplazado una dependencia funcional al interior de una relación por una dependencia funcional entre relaciones. En el caso del

ejemplo anterior, MARCA se duplicó y se debe satisfacer lo siguiente: "cada vez que exista un valor de MARCA de camión en **CAMION2**, este valor debe existir en **CAMION1**."

No se puede, por ejemplo, introducir un nueva marca de camión (TOYOTA) en **CAMION2** si esta marca no existe previamente en **CAMION1**.



2.1. Segunda Forma Normal (2FN)

Una relación se dice que está en **segunda forma normal (2FN)** si los atributos no llave son plenamente dependientes de la llave.

[A se dice **plenamente dependiente** de X si no existe X' subconjunto de X tal que X' **->** A se verifique]

Ejemplo. (Descomposición de una relación **1FN** en una **2FN**)

Sea **ENVIO1**(#ENVIO, DIA, CEDULA, #PLACA, TIPO-ENVIO) y la dependencia funcional
#ENVIO**->**TIPO-ENVIO (*)

Es evidente que también se verifica la dependencia
#ENVIO,DIA**->**TIPO-ENVIO, pues #ENVIO,DIA es llave.

Se puede visualizar ésta dependencias de la siguiente forma:

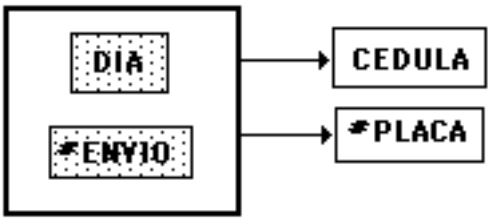
Aquí, el rectángulo más negro incluye la llave de la relación y las flechas representan las dependencias funcionales.

ENVIO1 no está en **2FN** pues TIPO-ENVIO no depende plenamente de la llave #ENVIO, DIA.

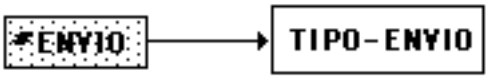
Las anomalías de almacenamiento se dan, debido a la existencia de esta dependencia. En efecto, con respecto a las anomalías de inserción, no se puede insertar una ocurrencia de #ENVIO, TIPO-ENVIO a no ser que exista el valor correspondiente de #ENVIO, DIA que es la llave.

Existe además anomalías de supresión, pues si se suprime un valor de la llave, hay riesgo de perder un valor único de la ocurrencia de #ENVIO, TIPO-ENVIO. Finalmente, se dan anomalías de modificación, pues en la relación **ENVIO1** generalmente se tendrá redundancia de ciertas ocurrencias de #ENVIO, TIPO-ENVIO

Para obtener una relación en **2FN** se aplica el teorema de descomposición aislando en una de las dos relaciones resultantes, la dependencia funcional interna(*), según se aprecia en la siguiente figura:



ENVIO11(*ENVIO,DIA,CEDULA,*PLACA)



ENVIO12(*ENVIO,TIPO-ENVIO)

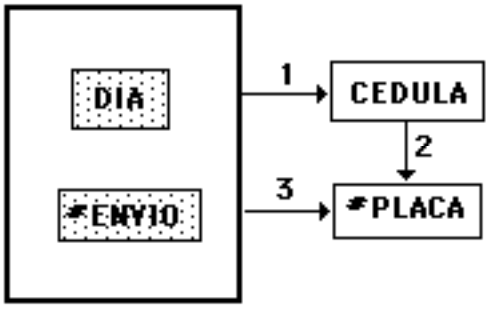
2.2.Tercera Forma Normal (3FN/BCFN)

Una relación **R** se dice en **tercera forma normal (3FN)** si
1. R está en 2FN
2. Ningún atributo no llave es transitivamente dependiente de una llave de **R.**

[Sea **R**(X,Y,Z,U) una relación. Se dice que Z es t**ransitivamente dependiente** de X si se dan las siguientes condiciones:
1. se verifica X->Y,
2. no se verifica Y->X,
3. se verifica Y->Z **]**

Una relación se dice que se encuentra en **forma normal Boyce-Codd (BCFN)** si cada vez que X**->**A, A $\notin$ X, se verifica en **R**, entonces X contiene una llave de **R**.

Ejemplo. Sea la relación **ENVIO11** (#ENVIO,DIA,CEDULA,#PLACA) y supóngase la imposición en el universo real: "**cada chofer conduce siempre el mismo camión**". Esto se traduce por medio de la dependencia funcional CEDULA->#PLACA(2)**.** El conjunto de dependencias funcionales de **ENVIO11** se puede visualizar en el siguiente diagrama:



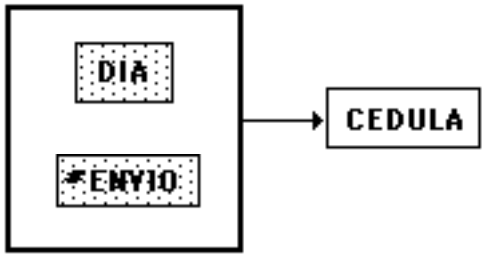
(2): CEDULA->#PLACA se dice **transitiva**, pues la combinación de (1) y (2) da (3) usando la propiedad de transitividad.

En este caso, aún persiten las anomalías:

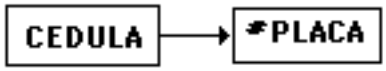
En efecto, no se puede insertar un valor de CEDULA, #PLACA sin que exista un valor de #ENVIO, DIA. Además, si se suprime un valor de #ENVIO, DIA, se puede perder un valor único de CEDULA, #PLACA.

También, si se desea modificar un valor de CEDULA,#PLACA, se tiene un alto costo de actualización (debido a la redundancia lógica) o incoherencia (si se hace la modificación en un sólo lugar).

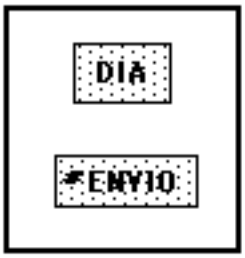
El resultado de descomposición va a permitir aislar la dependencia funcional transitiva y obtener relaciones en **3FN** sin este tipo de problemas, según se aprecia en el siguiente diagrama:



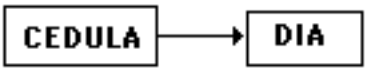
ENVIO111(*ENVIO,DIA,CEDULA)



ENVIO112(CEDULA,*PLACA)



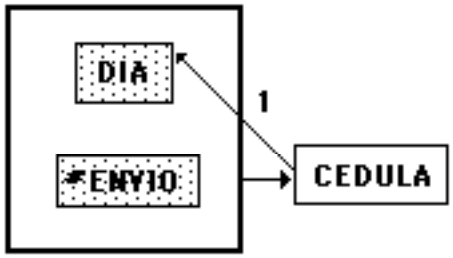
ENVIO1112(*ENVIO,CEDULA)



ENVIO1111(CEDULA,DIA)

Ahora se va a dar un ejemplo de una relación en **3FN** que no está en **BCFN.** Sea la relación **ENVIO111**(#ENVIO,DIA,CEDULA) y la imposición siguiente: "**cada chofer trabaja un solo día a la semana**". Esto se traduce por por medio de CEDULA**->**DIA (1)

Las dependencias funcionales de **ENVIO111** pueden visualizarse de la siguiente forma:



ENVIO111 se encuentra en **3FN** pues no existe dependencia transitiva entre atributos no llave. Sin embargo no está en **BCFN** pues CEDULA no es llave. Además, presenta esta relación una redundancia lógica debido a la dependencia funcional (1).

Si se aplica el resultado de descomposición a la relación **ENVIO111** para aislar la dependencia funcional (1) se obtienen las dos relaciones en **BCFN**

3. Cuarta Forma Normal (4FN)

Sea **R**(X,Y,Z) una relación. Se dice que la **dependencia multivaluda** X**->->**Y se verifica en **R** si y solamente si cada vez que (x_1,y_1,z_1), (x_1,y_2,z_2) son tuplos en **R**, entonces (x_1,y_2,z_1), (x_1,y_1,z_2) también están en **R**.

El resultado de descomposición para el caso de las dependencias multivaluas es el siguiente.

Sean X,Y subconjuntos de atributos de **R**. Entonces X**->->**Y se verifica en **R** si y solamente si **R**(X,Y,Z) = **R**[X,Y] * **R**[X,Z]

Se dice que **R** se encuentra en **cuarta forma normal (4FN)** si está en **BCNF** y si cada vez que una dependencia multivaluada X**->->**Y no trivial se verifica en **R** entonces la dependencia funcional X**->**A se verifica, para todo A en **R**.

[X **->->** Y es trivial si X,Y = **R** o si Y = Ø **]**

Ejemplo. Sea **ENVIO1**(CEDULA,#PLACA, CIUDAD-ORIGEN,CIUDAD-DESTINO) y la dependencia multivaluada CEDULA**->->** #PLACA.

ENVIO1

CEDULA	*PLACA	C-ORIG	C-DEST
2303776	C13455	Heredia	Cartago
2303776	C2123	Cartago	Limón
1123009	C4321	Heredia	Puntar.
3098789	C22334	Puntar.	Heredia
1123009	C13455	Heredia	Alajue.

ENVIO1 no está en **4FN** pues CEDULA no es llave.

El resultado de descomposición aplicado a **ENVIO1** permite aislar las dependencia multivaluada y reducir la redundancia lógica.

Así, se obtiene:

ENVIO11

CEDULA	C-ORIG	C-DEST
2303776	Heredia	Cartago
2303776	Cartago	Limón
1123009	Heredia	Puntar.
3098789	Puntar.	Heredia
1123009	Heredia	Alajue.

ENVIO12

CEDULA	*PLACA
2303776	C13455
2303776	C2123
1123009	C4321
3098789	C22334
1123009	C13455

La inserción del tuplo (2303776,C255,Heredia, Alajuela) se traduce por la inserción de dos tuplps (Heredia, Alajuela) en **ENVIO11** y (2303776,C255) en **ENVIO12**.

4. Quinta Forma Normal (5FN)

Finalmente, se estudiará la forma normal de mayor nivel. Esta fue introducida por Carlo Zaniolo, a finales de la década pasada.

Una relación está en **quinta forma normal (5FN)**, llamada también **forma normal de proyección-producto (PJ/FN)** si y solamente si cada dependencia producto de **R** está inducida por las llaves de **R**.

¿Qué quiere decir esto?

Considérese un ejemplo. Sea la relación **R**(A,B,C,D). Supongáse que A y B son llaves. Entonces la relación **R** satisface varias dependencias producto .

Si por ejemplo **R** satisface la dependencia producto ***[**A,B,C**],[**A,D**]** (es decir que **R**= **R[**A,B,C**]*R[**A,D**]**), entonces esta dependencia producto está inducida por el hecho de que A es una llave.

De igual manera, si **R** satisface la dependencia ***[**A,B**],[**A,C**],[**B,D**]**, esta dependencia producto está inducida por el hecho de que A,B son llaves.

Se puede notar que **5FN**, **4FN**, y **FNBC** corresponden a la definición única " todo determinante de una dependencia es una llave" considerando respectivamente la dependencia producto, la dependencia multivaluada y la dependencia funcional. **5FN** es una generalización de **4FN** que es una generalización de **FNBC**.

Fagin da en 1979 **[Fagin79]** un algoritmo que permite saber si una dependencia producto está inducida por las llaves de una inducción. Muestra además que toda relación en **5FN** no puede descomponerse sin que haya pérdida.

Se puede determinar si una relación **R** está en **5FN** conociendo las llaves y las **DP** de **R**.

Bibliografía

[Akoka85] Akoka J., **Les systèmes de Gestion de Bases de Données**, Eyrolles, París, 1985.

[Alagic86] Alagic, **Relational Database Technology**, Springer Verlag, New York, 1986.

[Autores86], **Compendio de 16 artículos sobre Bases de Datos,** I.T.C.R. Cartago, 1986.

[Bush45] Bush V., **As We May Think**, Atlantic Monthly, vol 176, Nº1, enero, 1945, pp. 101-108.

[Ceri80] Ceri S., Pelagatti G., **Distribuited Databases Principles and Systems**, McGraw-Hill Company, Admsterdam, 1980.

[Codd70] Codd E.F., **A relational model of data for large shared data banks**, Communications ACM, vol 13, Nº6, pp. 377-387.

[Codd71] Codd E.F., **Relational completeness of data base sublanguages**, Data Base Systems, Prentice Hall, pp. 65-98.

[Chen80] Chen P., **Entity-Relationship Approach to Systems Analysis and Design**, North-Holand Company, Amsterdam, 1980

[Date83a] Date C.J., **An introduction to Database Systems**. (**Fourth Editon**), Addison-Wesley, Massachussets, 1983.

[Delobel82] Delobel C., Adiba M.: **Bases de Données et Systemes Relationnels**, Dounod, París, 1982.

[Fagin79] Fagin R., **Normal Forms and relational data base operators**, Proc. ACM SIGMOD, 1979.

[Gardarin84] Gardarin G.(ED), **New Applications of Data Bases**, Academic Press, 1984.

[Gardarin85] Gardarin G., Valduriez P., **Bases de Données Relationnelles**, Eyrolles, París, 1985.

[Gallaire81] Gallaire H. (ED), **Advances in Database Theory**, Plenum Press, 1981.

[González86] González C., Jiménez E., **Modelo Relacional N-ario**, ITCR Cartago 1986.

[Gutiérrez88] Gutiérrez C., **Sistemas Expertos**, Club de Investigación Tecnológica, Costa Rica, 1988.

[Hisiao83] Hisiao D.K. (ED), **Advanced Database Machine Architecture**, Prentice-Hall, 1983.

[Hawryszkiewycz84] Hawryszkiewycz I.T., **Database Analysis and Design**, Science Research Associates, Chicago, 1984

[Korth86] Korth H., Silberchatz A.,**Database Systems Concepts**, McGraw-Hill, 1986.

[Miranda86] Miranda y Busta, **Les Bases de Données Relationnelles**, Eyrolles, París, 1986.

[Sansonnet88]Sansonnet J.P., **L´architecture des nouveaux ordinateurs,** La Recherche, Nº 204, 1988, pp 1298-1307.

[Ullman82] Ullman J.D., **Principles of Data Base Systems (Second Editon)**, Pitman Publishing Limited London, 1982.

[Vetter81] Vetter M., Maddison R.N, **Database Design Methodology**, Prentice-Hall International,1981.

[Wiederhold83] Wiederhold G., **Database Design (Second Edition)**, McGraw-Hill Company, New York, 1983.

[Wiederhold84] Wiederhold G., **Databases**, IEEE Computer, octubre 1984, pp. 211-223

[Yao85] Yao S.B. (ED).: **Principles of Database Design**, **Vol. I,** Prentice-Hall, 1985.