TRATAMIENTO DE INFORMACIÓN

A. BASES DE DATOS

I. Modelado y diseño

1. Conceptos generales

- ***Niveles interno, conceptual y externo de las arquitecturas correspondientes***

Una base de datos es en esencia una colección de archivos relacionados entre sí, de la cual los usuarios pueden extraer información sin considerar las fronteras de los archivos. Un objetivo importante de un sistema de base de datos es proporcionar a los usuarios una visión abstracta de los datos, es decir, el sistema esconde ciertos detalles de cómo se almacenan y mantienen los datos. Sin embargo para que el sistema sea manejable, los datos se deben extraer eficientemente. Existen diferentes niveles de abstracción para simplificar la interacción de los usuarios con el sistema; Interno, conceptual y externo, específicamente el de almacenamiento físico, el del usuario y el del programador.

Nivel físico.

Es la representación del nivel más bajo de abstracción, en éste se describe en detalle la forma en como de almacenan los datos en los dispositivos de almacenamiento(por ejemplo, mediante señaladores o índices para el acceso aleatorio a los datos).

Nivel conceptual.

El siguiente nivel más alto de abstracción, describe que datos son almacenados realmente en la base de datos y las relaciones que existen entre los mismos, describe la base de datos completa en términos de su estructura de diseño. El nivel conceptual de abstracción lo usan los administradores de bases de datos, quienes deben decidir qué información se va a guardar en la base de datos.

Consta de las siguientes definiciones:

Definición de los datos: Se describen el tipo de datos y la longitud de campo todos los elementos direccionables en la base. Los elementos por definir incluyen artículos elementales (atributos), totales de datos y registros conceptuales (entidades).

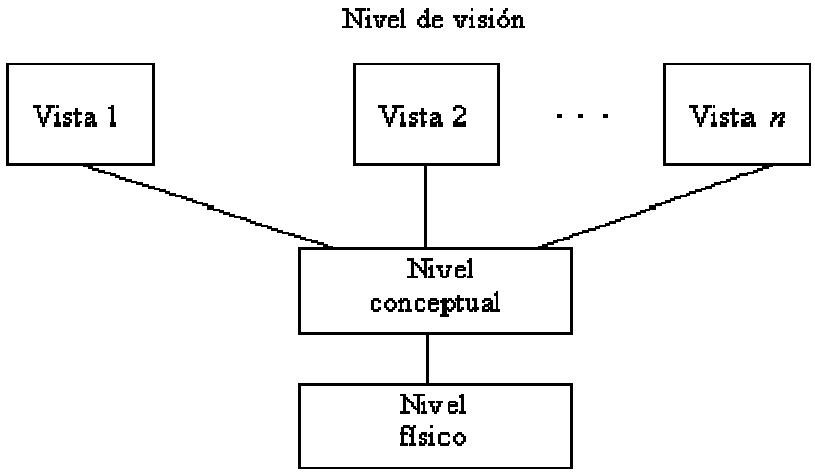
Relaciones entre datos: Se definen las relaciones entre datos para enlazar tipos de registros relacionados para el procesamiento de archivos múltiples.

En el nivel conceptual la base de datos aparece como una colección de registros lógicos, sin descriptores de almacenamiento. En realidad los archivos conceptuales no existen físicamente. La transformación de registros conceptuales a registros físicos para el almacenamiento se lleva a cabo por el sistema y es transparente al usuario.

Nivel de visión.

Nivel más alto de abstracción, es lo que el usuario final puede visualizar del sistema terminado, describe sólo una parte de la base de datos al usuario acreditado para verla. El sistema puede proporcionar muchas visiones para la misma base de datos.

La interrelación entre estos tres niveles de abstracción se ilustra en la siguiente figura.



- ***Administración de la base de datos***

Administrador de Bases de Datos
Denominado por sus siglas como: DBA, Database Administrator.
Es la persona encargada y que tiene el control total sobre el sistema de base de datos, sus funciones principales son:
Definición de esquema.
Es el esquema original de la base de datos se crea escribiendo un conjunto de definiciones que son traducidas por el compilador de DDL a un conjunto de tablas que son almacenadas permanentemente en el diccionario de datos.
Definición de la estructura de almacenamiento del método de acceso.
Estructuras de almacenamiento y de acceso adecuados se crean escribiendo un conjunto de definiciones que son traducidas por e compilador del lenguaje de almacenamiento y definición de datos.
Concesión de autorización para el acceso a los datos.
Permite al administrador de la base de datos regular las partes de las bases de datos que van a ser accedidas por varios usuarios.
Especificación de limitantes de integridad.
Es una serie de restricciones que se encuentran almacenados en una estructura especial del sistema que es consultada por el gestor de base de datos cada vez que se realice una actualización al sistema.
Usuarios de las bases de datos.
Podemos definir a los usuarios como toda persona que tenga todo tipo de contacto con el sistema de base de datos desde que este se diseña, elabora, termina y se usa. Los usuarios que accesan una base de datos pueden clasificarse como:
Programadores de aplicaciones.
Los profesionales en computación que interactúan con el sistema por medio de llamadas en DML (Lenguaje de Manipulación de Datos), las cuales están incorporadas en un programa escrito en un lenguaje de programación (Por ejemplo, COBOL, PL/I, Pascal, C, etc.)
Usuarios sofisticados.
Los usuarios sofisticados interactúan con el sistema sin escribir programas. En cambio escriben sus preguntas en un lenguaje de consultas de base de datos.
Usuarios especializados.
Algunos usuarios sofisticados escriben aplicaciones de base de datos especializadas que no encajan en el marco tradicional de procesamiento de datos.
Usuarios ingenuos.
Los usuarios no sofisticados interactúan con el sistema invocando a uno de los programas de aplicación permanentes que se han escrito anteriormente en el sistema de base de datos, podemos mencionar al usuario ingenuo como el usuario final que utiliza el sistema de base de datos sin saber nada del diseño interno del mismo por ejemplo: un cajero.

2. Organización de archivos

- ***Secuencial indexado***

Archivos de acceso secuencial (con tipo)
Dependiendo de la manera en que se accesen los registros de un archivo, se le clasifica como SECUENCIAL o como DIRECTO. En el caso de los archivos de ACCESO SECUENCIAL, para tener acceso al registro localizado en la posición N, se deben haber accesado los N-1 registros previos, en un orden secuencial. Cuando se tienen pocos registros en un archivo, o que los registros son pequeños, la diferencia entre los tiempos de acceso de forma secuencial y directa puede no ser perceptible para el usuario; sin embargo, la diferencia viene a ser significativa cuando se manejan archivos con grandes cantidades de información. La forma de manejar los archivos de acceso secuencial es más sencilla en la mayoría de los lenguajes de programación, por lo que su estudio se antepone al de los archivos de acceso directo. El manejo secuencial de un archivo es recomendable cuando se deben procesar todos o la mayoría de los registros, como por ejemplo en los casos de una nómina de empleados o en la elaboración de reportes contables.

- ***Árboles B***

Los árboles B y los árboles B+ son casos especiales de árboles de búsqueda. Un árbol de búsqueda es un tipo de árbol que sirve para guiar la búsqueda de un registro, dado el valor de uno de sus campos. Los índices multinivel de la sección anterior pueden considerarse como variaciones de los árboles de búsqueda. Cada bloque o nodo del

índice multinivel puede tener hasta p valores del campo de indexación y p punteros. Los valores del campo de indexación de cada nodo guían al siguiente nodo (que se encuentra en otro nivel), hasta llegar al bloque del fichero de datos que contiene el registro deseado. Al seguir un puntero, se va restringiendo la búsqueda en cada nivel a un subárbol del árbol de búsqueda, y se ignoran todos los nodos que no estén en dicho subárbol. Los árboles de búsqueda difieren un poco de los índices multinivel. Un árbol de búsqueda de orden p es un árbol tal que cada nodo contiene como mucho $p-1$ valores del campo de indexación y p punteros:

$$(P_1, K_1, P_2, K_2, \ldots, P_{q-1}, K_{q-1}, P_q)$$

donde $q \leq p$, cada P_i es un puntero a un nodo hijo y cada K_i es un valor de búsqueda proveniente de algún conjunto ordenado de valores. Se supone que todos los valores de búsqueda son únicos. Un árbol de búsqueda debe cumplir, en todo momento, las siguientes restricciones:

1. Dentro de cada nodo: $K_1 < K_2 < \ldots K_{q-1}$.

2. Para todos los valores X del subárbol al que apunta P_i, se tiene: $K_{i-1} < X < K_i$ para $1 < i < q$, $X < K_i$ para $i = 1$ y $K_{i-1} < X$ para $i = q$.

Al buscar un valor X siempre se sigue el puntero P_i apropiado de acuerdo con las condiciones de la segunda restricción. Para insertar valores de búsqueda en el árbol y eliminarlos, sin violar las restricciones anteriores, se utilizan algoritmos que no garantizan que el árbol de búsqueda esté equilibrado (que todas las hojas estén al mismo nivel). Es importante mantener equilibrados los árboles de búsqueda porque esto garantiza que no habrá nodos en niveles muy profundos que requieran muchos accesos a bloques durante una búsqueda. Además, las eliminaciones de registros pueden hacer que queden nodos casi vacíos, con lo que hay un desperdicio de espacio importante que también provoca un aumento en el número de niveles. El árbol B es un árbol de búsqueda, con algunas restricciones adicionales, que resuelve hasta cierto punto los dos problemas anteriores. Estas restricciones adicionales garantizan que el árbol siempre estará equilibrado y que el espacio desperdiciado por la eliminación, si lo hay, nunca será excesivo. Los algoritmos para insertar y eliminar se hacen más complejos para poder mantener estas restricciones. No obstante, la mayor parte de las inserciones y eliminaciones son procesos simples, se complican sólo en circunstancias especiales: cuando se intenta insertar en un nodo que está lleno o cuando se intenta borrar en un nodo que está ocupado hasta la mitad. Un árbol B de orden p se define del siguiente modo:

1. La estructura de cada nodo interno tiene la forma:

   $$(P_1, (K_1, Pr_1), P_2, (K_2, Pr_2), P_3, (K_3, Pr_3), \ldots P_{q-1}, (K_{q-1}, Pr_{q-1}), P_q)$$

   donde $q \leq p$. Cada P_i es un puntero a un nodo interno del árbol y cada Pr_i es un puntero al registro del fichero de datos que tiene el valor K_i en el campo de búsqueda o de indexación.
2. Dentro de cada nodo se cumple: $K_1 < K_2 < \ldots K_{q-1}$.

3. Para todos los valores X del campo de indexación del subárbol al que apunta P_i, se cumple: $K_{i-1} < X < K_i$ para $1 < i < q$, $X < K_i$ para $i = 1$ y $K_{i-1} < X$, para $i = q$.

4. Cada nodo tiene, como mucho, p punteros a nodos del árbol.

5. Cada nodo, excepto la raíz y las hojas, tiene, al menos, $\lceil p/2 \rceil$ punteros a nodos del árbol. El nodo raíz tiene, como mínimo, dos punteros a nodos del árbol, a menos que sea el único nodo del árbol.

6. Un nodo con q punteros a nodos, $q \leq p$, tiene $q-1$ valores del campo de indexación.

7. Todos los nodos hoja están al mismo nivel. Los nodos hoja tienen la misma estructura que los nodos internos, pero los punteros a nodos del árbol son nulos.

Como se puede observar, en los árboles B todos los valores del campo de indexación aparecen alguna vez en algún nivel del árbol, junto con un puntero al fichero de datos. En un árbol B+ los punteros a datos se almacenan sólo en los nodos hoja del árbol, por lo cual, la estructura de los nodos hoja difiere de la de los nodos internos. Los nodos hoja tienen una entrada por cada valor del campo de indexación, junto con un puntero al registro del fichero de datos. Estos nodos están enlazados para ofrecer un acceso ordenado a los registros a través del campo de indexación. Los nodos hoja de un árbol B+ son similares al primer nivel (nivel base) de un índice. Los nodos internos del árbol B+ corresponden a los demás niveles del índice. Algunos valores del campo de indexación se repiten en los nodos internos del árbol B+ con el fin de guiar la búsqueda. En un árbol B+ de orden p la estructura de un nodo interno es la siguiente:

1. Todo nodo interno es de la forma:

   $$(P_1, K_1, P_2, K_2, P_3, K_3, \ldots P_{q-1}, K_{q-1}, P_q)$$

   donde $q \leq p$. Cada P_i es un puntero a un nodo interno del árbol.
2. Dentro de cada nodo interno se cumple: $K_1 < K_2 < \ldots K_{q-1}$.
3. Para todos los valores X del campo de indexación del subárbol al que apunta P_i, se cumple: $K_{i-1} < X \leq K_i$ para $1 < i < q$, $X \leq K_i$ para $i = 1$ y $K_{i-1} < X$ para $i = q$.
4. Cada nodo interno tiene, como mucho, p punteros a nodos del árbol.
5. Cada nodo interno, excepto la raíz, tiene, al menos, $\lceil p/2 \rceil$ punteros a nodos del árbol. El nodo raíz tiene, como mínimo, dos punteros a nodos del árbol si es un nodo interno.
6. Un nodo interno con q punteros a nodos, $q \leq p$, tiene $q - 1$ valores del campo de indexación.

La estructura de los nodos hoja de un árbol B+ de orden p es la siguiente:

1. Todo nodo hoja es de la forma:

   $$((K_1, Pr_1), (K_2, Pr_2), (K_3, Pr_3), \ldots (K_{q-1}, Pr_{q-1}), P_{siguiente})$$

   donde $q \leq p$. Cada Pr_i es un puntero al registro de datos que tiene el valor K_i en el campo de indexación, y $P_{siguiente}$ es un puntero al siguiente nodo hoja del árbol.
2. Dentro de cada nodo hoja se cumple: $K_1 < K_2 < \ldots K_{q-1}$, $q \leq p$.
3. Cada nodo hoja tiene, al menos, $\lfloor p/2 \rfloor$ valores.
4. Todos los nodos hoja están al mismo nivel.

Como las entradas en los nodos internos de los árboles B+ contienen valores del campo de indexación y punteros a nodos del árbol, pero no contienen punteros a los registros del fichero de datos, es posible ``empaquetar" más entradas en un nodo interno de un árbol B+ que en un nodo similar de un árbol B. Por tanto, si el tamaño de bloque (nodo) es el mismo, el orden p será mayor para el árbol B+ que para el árbol B. Esto puede reducir el número de niveles del árbol B+, mejorándose así el tiempo de acceso. Como las estructuras de los nodos internos y los nodos hoja de los árboles B+ son diferentes, su orden p puede ser diferente. Se ha demostrado por análisis y simulación que después de un gran número de inserciones y eliminaciones aleatorias en un árbol B, los nodos están ocupados en un 69% cuando se estabiliza el número de valores del árbol. Esto también es verdadero en el caso de los árboles B+. Si llega a suceder esto, la división y combinación de nodos ocurrirá con muy poca frecuencia, de modo

que la inserción y la eliminación se volverán muy eficientes. Cuando los árboles se definen sobre un campo no clave, los punteros a datos pasan a ser punteros a bloques de punteros a datos (se añade un nivel de indirección).

- ***Hashig asociativo***

DIRECTO (HASHING) A partir de una clave, y por medio de una función (f) encontrar el RID de ese registro. ¿Cual debe ser la f?:
Algoritmo de Direccionamiento:
f: C ------> 0,...N f(Cj)=n para cada valor no se repite
Se trata de buscar la función en la que se produzca el mínimo de sinónimos y huecos (distribuye los valores entre 0 y N). Las más utilizadas son las de números aleatorios y son:
1.- PLEGAMIENTO se pasa el nº a binario y se suma la primera mitad del número con su segunda mitad y el resultado es la dirección.
2.- CUADRADOS GENERALES elevamos el nº al cuadrado y se cogen las cifras centrales a conveniencia.
3.- METODO DE CONGRUENCIA se sigue la fórmula C=(a*C+b)mod N
4.- DESPLAZAMIENTO se van sumando las cifras, para que quede una cifra más pequeña (en sucesivos pasos).
Para evitar los sinónimos y los huecos se puede utilizar **Páginas de Overflow**, pero es muy lioso.
Entonces aparece el HASHING, se definen una serie de cubos para ir almacenando los valores.
V.Gr. Se sigue la fórmula ((K div 100)div 2)+1
El resultado es el cubo donde hay que almacenarlo. Si el cálculo nos lleva a un cubo cuyos 10 registros están llenos, entonces se va a la página de overflow y se almacena allí. Lo primero que hay que controlar es el tamaño de los cubos, que será lo suficientemente grande para no utilizar mucho la página de overflow y lo suficientemente pequeño para que no queden muchos huecos. Los registros se pueden ordenar dentro del cubo y así las busquedas son más fáciles. La transformación de la clave para obtener la dirección, tiene que ser la más adecuada (pocos sinónimos). Los criterios para elegir el número de páginas de overflow (1/3 del nº de cubos).

- ***Métodos de acceso secundario***

ENCADENADA (POR PUNTEROS) se utiliza para indexar por un campo que no es clave. El campo escogido será un campo con pocos valores (rango) pero que se repite mucho.

V.Gr.

S#	Ciudad	
S1	Vigo	
S2	Ourense	
S3	Lugo	
S4	Coruña	
S5	Ourense	
S6	Vigo	Se enlazan con punteros, con lo que las
S7	Coruña	insercciones son mucho más rápidas y fáciles.

Esta técnica se utiliza porque al trabajar con las tablas de indices, si queremos insertar no es posible. Entonces se utiliza la Tabla Relocalizable:

3. Modelo entidad-relación

- ***Representación del modelo mediante el diagrama entidad-relación***

El modelo E-R se basa en una percepción del mundo real, la cual esta formada por objetos básicos llamados entidades y las relaciones entre estos objetos así como las características de estos objetos llamados atributos.

Entidades y conjunto de entidades

Una entidad es un objeto que existe y se distingue de otros objetos de acuerdo a sus características llamadas atributos . Las entidades pueden ser concretas como una persona o abstractas como una fecha. Un conjunto de entidades es un grupo de entidades del mismo tipo. Por ejemplo el conjunto de entidades CUENTA, podría representar al conjunto de cuentas de un banco X, o ALUMNO representa a un conjunto de entidades de todos los alumnos que existen en una institución. Una entidad se caracteriza y distingue de otra por los atributos, en ocasiones llamadas propiedades, que representan las características de una entidad. Los atributos de una entidad pueden tomar un conjunto de valores permitidos al que se le conoce como dominio del atributo. Así cada entidad se describe por medio de un conjunto de parejas formadas por el atributo y el valor de dato. Habrá una pareja para cada atributo del conjunto de entidades. Ejemplo:
Hacer una descripción en pareja para la entidad alumno con los atributos No_control, Nombre y Especialidad.

Nombre_atributo, Valor
No_control , 96310418
Nombre , Sánchez Osuna Ana
Esp , LI

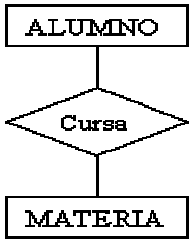
O considerando el ejemplo del Vendedor cuyos aributos son: RFC, Nombre, Salario.

Nombre_atributo, Valor
RFC , COMD741101YHR
Nombre , Daniel Colín Morales
Salario , 3000

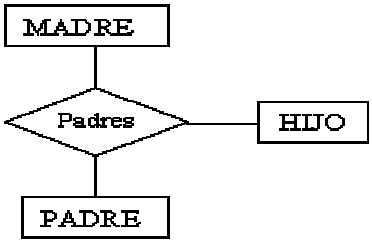
Relaciones y conjunto de relaciones.

Una relación es la asociación que existe entre dos a más entidades. Un conjunto de relaciones es un grupo de relaciones del mismo tipo. La cantidad de entidades en una relación determina el grado de la relación, por ejemplo la relación ALUMNO-MATERIA es de grado 2, ya que intervienen la entidad ALUMNO y la entidad MATERIA, la relación PADRES, puede ser de grado 3, ya que involucra las entidades PADRE, MADRE e HIJO. Aunque el modelo E-R permite relaciones de cualquier grado, la mayoría de las aplicaciones del modelo sólo consideran relaciones del grado 2. Cuando son de tal tipo, se denominan relaciones binarias. La función que tiene una relación se llama papel, generalmente no se especifican los papeles o roles, a menos que se quiera aclarar el significado de una relación.

Diagrama E-R (sin considerar los atributos, sólo las entidades) para los modelos ejemplificados:



Ejemplo de relación de grado 2



Ejemplo de relación de grado 3

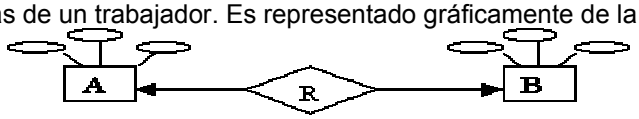
Limitantes de mapeo.

Existen 4 tipos de relaciones que pueden establecerse entre entidades, las cuales establecen con cuantas entidades de tipo B se pueden relacionar una entidad de tipo A:

** Tipos de relaciones:*

Relación uno a uno.

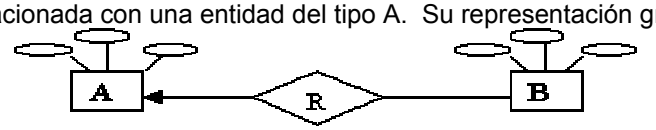
Se presenta cuando existe una relación como su nombre lo indica uno a uno, denominado también relación de matrimonio. Una entidad del tipo A solo se puede relacionar con una entidad del tipo B, y viceversa. Por ejemplo: la relación asignación de automóvil que contiene a las entidades EMPLEADO, AUTO, es una relación 1 a 1, ya que asocia a un empleado con un único automóvil por lo tanto ningún empleado posee más de un automóvil asignado, y ningún vehículo se asigna a más de un trabajador. Es representado gráficamente de la siguiente manera:



A: Representa a una entidad de cualquier tipo diferente a una entidad B.
R: en el diagrama representa a la relación que existe entre las entidades.
El extremo de la flecha que se encuentra punteada indica el uno de la relación, en este caso, una entidad A ligada a una entidad B.

Relación uno a muchos.

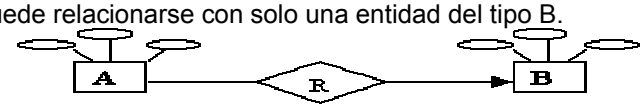
Significa que una entidad del tipo A puede relacionarse con cualquier cantidad de entidades del tipo B, y una entidad del tipo B solo puede estar relacionada con una entidad del tipo A. Su representación gráfica es la siguiente:



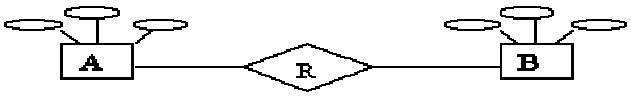
Nótese en este caso que el extremo punteado de la flecha de la relación de A y B, indica una entidad A conectada a muchas entidades B.

Tratamiento de información

Muchos a uno.
Indica que una entidad del tipo B puede relacionarse con cualquier cantidad de entidades del tipo A, mientras que cada entidad del tipo A solo puede relacionarse con solo una entidad del tipo B.



Muchas a muchas.
Establece que cualquier cantidad de entidades del tipo A pueden estar relacionados con cualquier cantidad de entidades del tipo B.



A los tipos de relaciones antes descritos, también se le conoce como cardinalidad. La cardinalidad nos especifica los tipos de relaciones que existen entre las entidades en el modelo E-R y establecer con esto las validaciones necesarias para conseguir que los datos de la instancia (valor único en un momento dado de una base de datos) correspondan con la realidad.
Algunos ejemplos de cardinalidades de la vida común pueden ser:
Uno a uno.
El noviazgo, el RFC de cada persona, El CURP personal, El acta de nacimiento, ya que solo existe un solo documento de este tipo para cada una de las diferentes personas.
Uno a muchos.
Cliente – Cuenta en un banco, Padre-Hijos, Camión-Pasajeros, zoologico- animales, árbol – hojas.
Muchos a muchos.
Arquitecto – proyectos, fiesta – personas, estudiante – materias.
NOTA:
Cabe mencionar que la cardinalidad para cada conjunto de entidades depende del punto de vista que se le de al modelo en estudio, claro esta, sujetándose a la realidad.
** Otra clase de limitantes lo constituye la dependencia de existencia.*
Refiriéndonos a las mismas entidades A y B, decimos que si la entidad A depende de la existencia de la entidad B, entonces A es dependiente de existencia por B, si eliminamos a B tendríamos que eliminar por consecuente la entidad A, en este caso B es la entidad Dominante y A es la entidad subordinada.

Llaves primarias.

Como ya se ha mencionado anteriormente, la distinción de una entidad entre otra se debe a sus atributos, lo cual lo hacen único. Una llave primaria es aquel atributo el cual consideramos clave para la identificación de los demás atributos que describen a la entidad. Por ejemplo, si consideramos la entidad ALUMNO del Instituto Tecnológico de La Paz, podríamos tener los siguientes atributos: Nombre, Semestre, Especialidad, Dirección, Teléfono, Número de control, de todos estos atributos el que podremos designar como llave primaria es el número de control, ya que es diferente para cada alumno y este nos identifica en la institución. Claro que puede haber más de un atributo que pueda identificarse como llave primaria en este caso se selecciona la que consideremos más importante, los demás atributos son denominados llaves secundarias. Una clave o llave primaria es indicada gráficamente en el modelo E-R con una línea debajo del nombre del atributo.

Diagrama Entidad-Relación

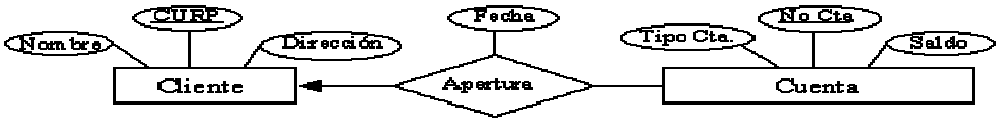
Denominado por sus siglas como: E-R; Este modelo representa a la realidad a través de un esquema gráfico empleando los terminología de entidades, que son objetos que existen y son los elementos principales que se identifican en el problema a resolver con el diagramado y se distinguen de otros por sus características particulares denominadas atributos, el enlace que que rige la unión de las entidades esta representada por la relación del modelo. Recordemos que un rectángulo nos representa a las entidades; una elipse a los atributos de las entidades, y una etiqueta dentro de un rombo nos indica la relación que existe entre las entidades, destacando con líneas las uniones de estas y que la llave primaria de una entidad es aquel atributo que se encuentra subrayado. A continuación mostraremos algunos ejemplos de modelos E-R, considerando las cardinalidades que existen entre ellos:
Relación Uno a Uno.
Problema: Diseñar el modelo E-R, para la relación Registro de automóvil que consiste en obtener la tarjeta de circulación de un automóvil con los siguientes datos:- Automóvil- Modelo, Placas, Color - Tarjeta de circulación - Propietario, No_serie, Tipo.

Tratamiento de información

Indicamos con este ejemplo que existe una relación de pertenencia de uno a uno, ya que existe una tarjeta de circulación registrada por cada automóvil. En este ejemplo, representamos que existe un solo presidente para cada país.

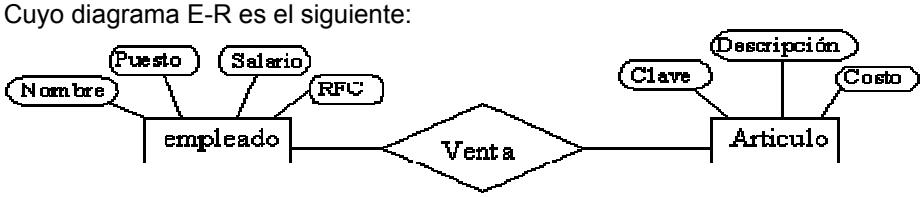
Relación muchos a muchos.

El siguiente ejemplo indica que un cliente puede tener muchas cuentas, pero que una cuenta puede llegar a pertenecer a un solo cliente (Decimos puede, ya que existen cuentas registradas a favor de más de una persona).



Reducción de diagramas E-R a tablas

Un diagrama E-R, puede ser representado también a través de una colección de tablas. Para cada una de las entidades y relaciones existe una tabla única a la que se le asigna como nombre el del conjunto de entidades y de las relaciones respectivamente, cada tabla tiene un número de columnas que son definidas por la cantidad de atributos y las cuales tienen el nombre del atributo. La transformación de nuestro ejemplo Venta en la que intervienen las entidades de Vendedor con los atributos RFC, nombre, puesto, salario y Artículo con los atributos Clave, descripción, costo. Cuyo diagrama E-R es el siguiente:



Entonces las tablas resultantes siguiendo la descripción anterior son:

Tabla Empleado

Nombre	Puesto	Salario	RFC
Teófilo	Vendedor	2000	TEAT701210XYZ
Cesar	Auxiliar ventas	1200	COV741120ABC

Tabla artículo

Clave	Descripción	Costo
A100	Abanico	460
C260	Colcha matrimonial	1200

Tabla Venta

RFC	Clave
TEAT701210XYZ	C260
COV741120ABC	A100

Nótese que en la tabla de relación - Venta -, contiene como atributos a las llaves primarias de las entidades que intervienen en dicha relación, en caso de que exista un atributo en las relaciones, este atributo es anexado como una

fila más de la tabla. Por ejemplo si anexamos el atributo fecha a la relación venta, la tabla que se originaria sería la siguiente:

RFC	Clave	Fecha
TEAT701210XYZ	C260	10/12/96
COV741120ABC	A100	11/12/96

Generalización y especialización

Generalización.

Es el resultado de la unión de 2 o más conjuntos de entidades (de bajo nivel) para producir un conjunto de entidades de más alto nivel. La generalización se usa para hacer resaltar los parecidos entre tipos de entidades de nivel más bajo y ocultar sus diferencias. La generalización consiste en identificar todos aquellos atributos iguales de un conjunto de entidades para formar una entidad(es) global(es) con dichos atributos semejantes, dicha entidad(es) global(es) quedara a un nivel más alto al de las entidades origen.

Ejemplo:

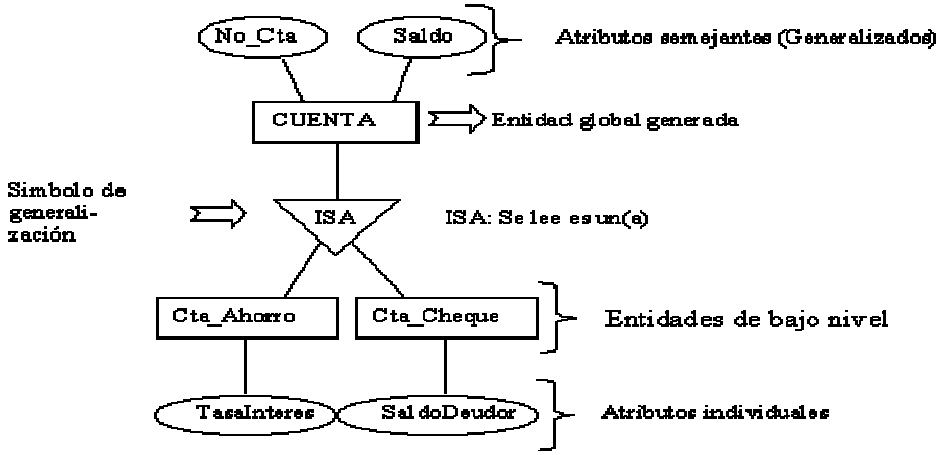
Tomando el ejemplo del libro de fundamentos de base de datos de Henry F. Korth.

Donde:

Se tiene las entidades Cta_Ahorro y Cta_Cheques, ambas tienen los atributos semejantes de No_Cta y Saldo, aunque además de estos dos atributos, Cta_Ahorro tiene el atributo Tasa_Interes y Cta_Cheques el atributo Saldo_Deudor. De todos estos atributos podemos juntar (generalizar) No_Cta y Saldo que son iguales en ambas entidades.

Entonces tenemos:

Podemos leer esta gráfica como: La entidad Cta_Ahorro hereda de la entidad CUENTA los atributos No_Cta y saldo, además del atributo de TasaInteres, de forma semejante Cta_cheque tiene los atributos de No_Cta, Saldo y SaldoDeudor.



Como podemos observar la Generalización trata de eliminar la redundancia (repetición) de atributos, al englobar los atributos semejantes. La entidad(es) de bajo nivel cuentan (heredan) todos los atributos correspondientes.

Especialización:

Es el resultado de tomar un subconjunto de entidades de alto nivel para formar un conjunto de entidades de más bajo nivel.

* En la generalización cada entidad de alto nivel debe ser también una entidad de bajo nivel. La especialización no tiene este limitante.
* se representa por medio de un triángulo denominado con la etiqueta "ISA", se distingue de la generalización por el grosor de las líneas que conectan al triángulo con las entidades.
* La especialización denota la diferencia entre los conjuntos de entidades de alto y bajo nivel.

Agregación.

La agregación surge de la limitación que existe en el modelado de E-R, al no permitir expresar las relaciones entre relaciones de un modelo E-R en el caso de que una relación X se quiera unir con una entidad cualquiera para formar otra relación. La Generalización consiste en agrupar por medio de un rectángulo a la relación (representada por un rombo) junto con las entidades y atributos involucrados en ella, para formar un grupo que es considerado una entidad y ahora sí podemos relacionarla con otra entidad. Para ejemplificar lo anterior consideremos el ejemplo del libro de

fundamentos de Base de Datos de Henry F. Korth. En donde el problema consiste en que existen trabajando muchos empleados que trabajan en diferentes proyectos, pero dependiendo del trabajo que realiza en pueden llegar a utilizar un equipo o maquinaria; en este problema intervienen 3 entidades: Empleado, Proyecto y Maquinaria, el diagrama E-R correspondiente es:

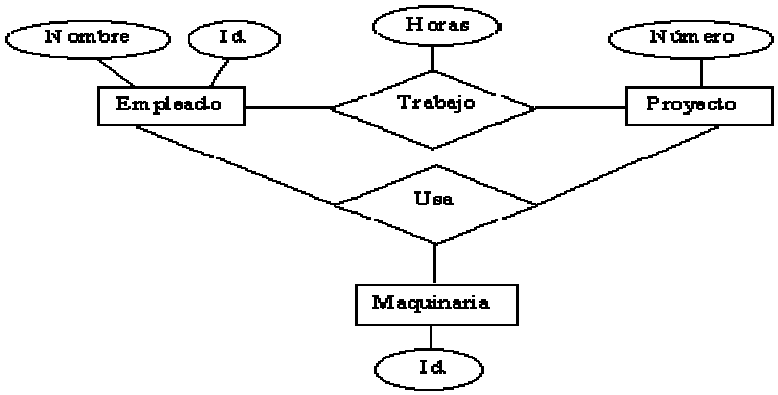


Diagrama E-R con relaciones redundantes

Como el modelo E-R no permite la unión entre dos o más relaciones, la relación trabajo es englobada como si fuera una entidad más de la relación usa, gráficamente queda como:

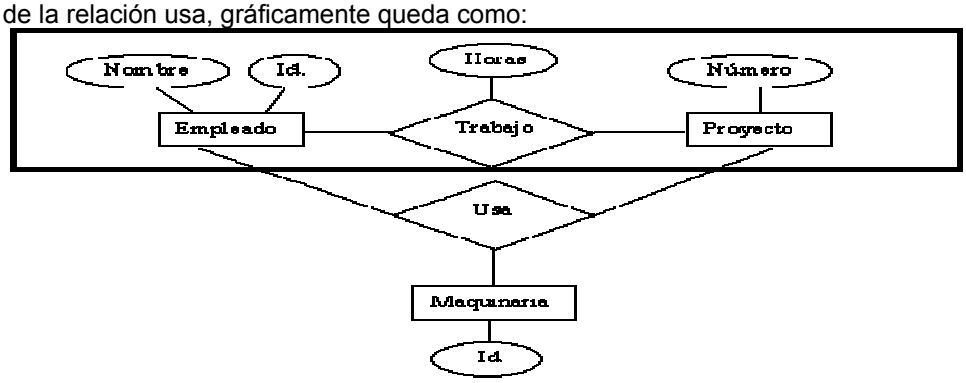


Diagrama E-R con agregación

Ahora podemos decir que la entidad trabajo se relaciona con la entidad maquinaria a través de la relación usar. Para indicarnos que un trabajo usa un determinado equipo o maquinaria según el tipo de trabajo que se trate.

4. El modelo jerárquico

- ***Modelo conceptual***
- ***Estructura de almacenamiento***

Conceptos Básicos.

Una base de datos jerárquica consiste en una colección de registros que se conectan entre sí por medio de enlaces. Los registros son similares a los expuestos en el modelo de red. Cada registro es una colección de campos (atributos), que contienen un solo valor cada uno de ellos. Un enlace es una asociación o unión entre dos registros exclusivamente. Por tanto, este concepto es similar al de enlace para modelos de red. Consideremos la base de datos, nuevamente, que contiene la relación alumno - materia de un sistema escolar. Existen dos tipos de registros en este sistema, alumno y materia. El registro alumno consta de tres campos: NombreA, Control y Esp; El registro Materia esta compuesto de tres campos: Clave, NombreM y Cred. En este tipo de modelos la organización se establece en forma de árbol, donde la raíz es un nodo ficticio. Así tenemos que, una base de datos jerárquica es una colección de árboles de este tipo. El contenido de un registro específico puede repetirse en varios sitios(en el mismo árbol o en varios árboles).

La repetición de los registros tiene dos desventajas principales:

* Puede producirse una inconsistencia de datos
* El desperdicio de espacio.

Diagramas de estructura de árbol

Un diagrama de estructura de árbol es la representación de un esquema de la base de datos jerárquica, de ahí el nombre, ya que un árbol esta desarrollado precisamente en orden descendente formando una estructura jerárquica. Este tipo de diagrama está formado por dos componentes básicos:

Rectángulos: que representan a los de registros.

Líneas: que representan a los enlaces o ligas entre los registros.

Un diagrama de árbol tiene el propósito de especificar la estructura global de la base de datos. Un diagrama de estructura de árbol es similar a un diagrama de estructura de datos en el modelo de red. La principal diferencia es que en el modelo de red los registros se organizan en forma de un grafo arbitrario, mientras que en modelo de estructura de árbol los registros se organizan en forma de un árbol con raíz.

Características de las estructuras de árbol:

El árbol no puede contener ciclos.

Las relaciones que existen en la estructura deben ser de tal forma que solo existan relaciones muchos a uno o uno a uno entre un padre y un hijo.

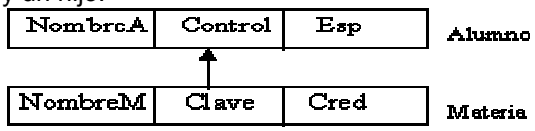
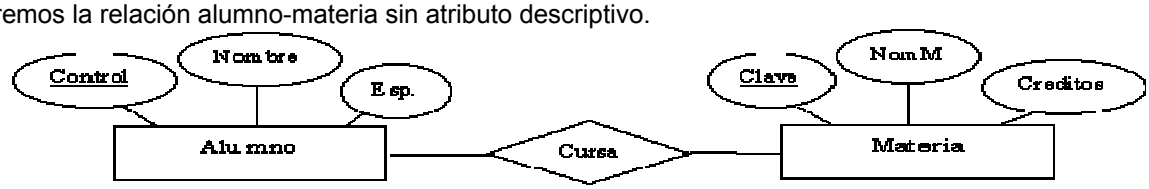


Diagrama de estructura de árbol

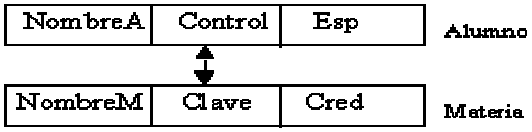
En este diagrama podemos observar que las flechas están apuntando de padres a hijos. Un padre (origen de una rama) puede tener una flecha apuntando a un hijo, pero un hijo siempre puede tener una flecha apuntando a su padre. El esquema de una base de datos se representa como una colección de diagramas de estructura de árbol. Para cada diagrama existe una única instancia de árbol de base de datos. La raíz de este árbol es un nodo ficticio. Los hijos de ese nodo son instancias de los registros de la base de datos. Cada una de las instancias que son hijos pueden tener a su vez, varias instancias de varios registros. Las representaciones según las cardinalidades son:

Consideremos la relación alumno-materia sin atributo descriptivo.

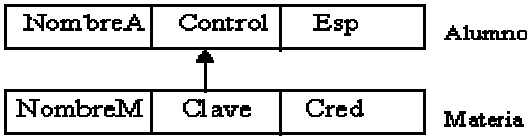


La transformación según las cardinalidades seria:

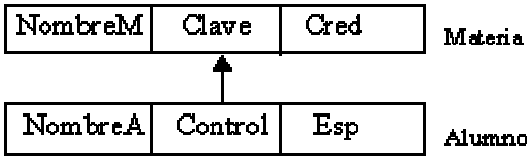
Cuando la relación es uno a uno.



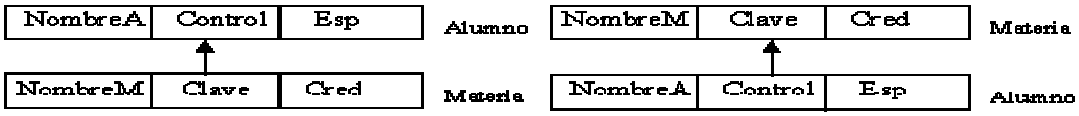
Cuando la relación es uno a muchos.



Cuando la relación es muchos a uno.



Cuando la relación es muchos a muchos.



Cuando la relación tiene atributos descriptivos, la transformación de un diagrama E-R a estructura de árbol se lleva a cabo cubriendo los siguientes pasos:

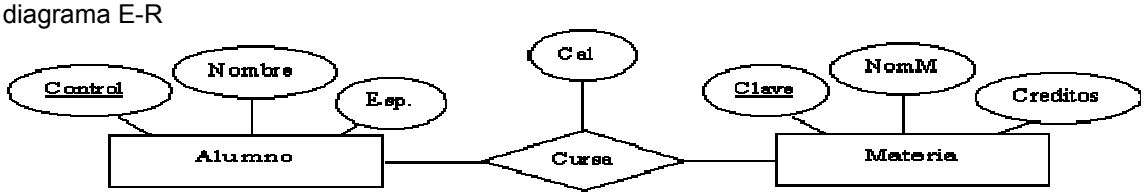
Tratamiento de información

Crear un nuevo tipo de registro.

Crear los enlaces correspondientes.

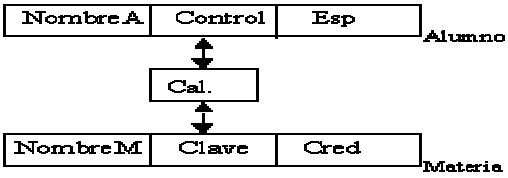
Consideremos que a la relación Alumno-Materia añadimos el atributo Cal a la relación que existe entre ambas, entonces nuestro modelo E-R resulta:

Añadir el diagrama E-R

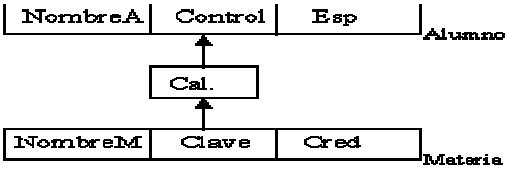


Según las cardinalidades los diagramas de estructura de árbol pueden quedar de la siguiente manera:

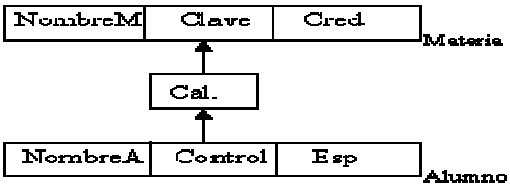
Cuando la relación es uno a uno.



Cuando la relación es uno a muchos.



Cuando la relación es Muchos a uno.



Cuando la relación es Muchos a Muchos.

Si la relación es muchos a muchos entonces la transformación a diagramas de árbol es un poco más compleja debido a que el modelo jerárquico solo se pueden representar las relaciones uno a uno o uno a muchos. Existen varias formas distintas de transformar este tipo de relaciones a estructura de árbol, sin embargo todas las formas constituyen la repetición de algunos registros. La decisión de qué método de transformación debe utilizarse depende de muchos factores, entre los que se incluyen:

El tipo de consultas esperadas en la base de datos.

El grado al que el esquema global de base de datos que se está modelando se ajusta al diagrama E-R dado.

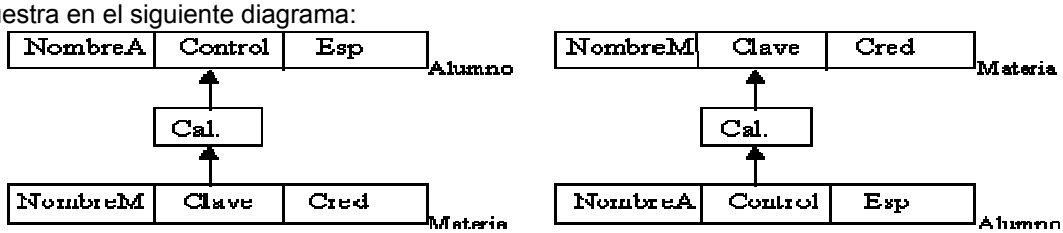
A continuación se describe la forma de transformar un diagrama E-R a estructura de árbol con relaciones muchos a muchos. Suponemos el ejemplo de la relación alumno-materia.

Crear dos diagramas de estructura de árbol distintos T1 yT2, cada uno de los cuales incluye los tipos de registro alumno y materia, en el árbol T1 la raíz es alumno y en T2 la raíz es materia. Crear los siguientes enlaces:

Un enlace muchos a uno del registro cuenta al registro Alumno, en T1

Un enlace muchos a uno del tipo de registro cliente al tipo de registro materia en T2.

Como se muestra en el siguiente diagrama:



Recuperación de datos

Tratamiento de información

Para manipular la información de una base de datos jerárquico, es necesario emplear un lenguaje de manipulación de datos, el lenguaje consta de varias órdenes que están incorporadas en un lenguaje principal, Pascal.

La orden Get

La recuperación de datos se realiza mediante esta orden, se realizan las siguientes acciones:

Localiza un registro en la base de datos y actualiza a un puntero que es el que mantiene la dirección del último registro accesado.

Copia los datos solicitados a un tipo de registro apropiado para la consulta.

La orden Get debe especificar en cual de los árboles de la base de datos se va a buscar.

Para revisar todos los registros de forma consistente, se debe imponer un orden en los registros. El que normalmente se usa es el preorden, el cuál consiste en iniciar la búsqueda por la raíz y continua buscando por los subárboles de izquierda a derecha recursivamente. Así pues, empezamos por la raíz, visitamos el hijo más a la izquierda, y así sucesivamente, hasta que alcancemos un nodo hoja (sin hijos). A continuación movemos hacia atrás al padre de la hoja y visitamos el hijo más a la izquierda no visitado. Continuamos de esta forma hasta que se visite todo el árbol.

Existen dos ordenes Get diferentes para localizar registros en un árbol de base de datos. La orden más simple tiene la forma:

```
get fisrt <Tipo de registro>
where <Condición>
```

La cláusula where es opcional. La <Condición> que se adjunta es un predicado que puede implicar a cualquier tipo de registro que sea un antecesor de <tipo registro> o el <Tipo de registro> mismo. La orden get localiza el primer registro (en preorden) del tipo <Tipo registro> en la base de datos que satisfaga la <condición > de la cláusula where. Si se omite la cláusula where, entonces se localiza el primer registro del tipo <Tipo registro>. Una vez que se encuentra ese registro, se hace que el puntero que tiene la dirección del último registro accesado apunte a ese registro y se copia el contenido del registro en un registro apto para la consulta.

Para ilustrar lo anterior veamos los ejemplos:

Realicemos una consulta que imprima El nombre del alumno llamado Luis A. (consideremos la relación Alumno-Materia que hemos estado manejando.)

```
get first Alumno
where Alumno.NombreA="Luis A.";
print (Alumno.Control)
```

Ahora consideremos que deseamos la consulta de los nombres de las materias en donde el alumno de nombre Luis A. A obtenido una calificación igual a 100 (si es que existe)

```
get first Alumno
where Alumno.NombreA="Luis A." and Cursa.cal= 100;
    if DB-Status=0 then print (Materia.NombreM);
```

La condición involucra a la variable DB-Status, la cual nos indica si se encontró o no el registro. La orden get first, solo nos muestra el primer registro encontrado que satisfaga la orden de consulta, sin embargo puede haber más de ellos, para localizar a los demás registros empleamos la orden Get next.

Cuya estructura es:

```
get next <Tipo de registro>
where <Condición>
```

La cual localiza el siguiente registro en preorden que satisface la condición. Si se omite la cláusula where, entonces se localiza el siguiente registro del tipo <Tipo registro>.

Actualización de datos.

Creación de nuevos registros. La orden utilizada para la inserción de registros es:

```
insert <tipo de registro>
where <Condición>
```

donde: tipo registro contiene los datos de los campos del registro a insertar.

Sí se incluye la cláusula where, el sistema busca en el árbol de la base de datos (en preorden) un registro que satisfaga la condición dada, una vez encontrado, el registro creado se inserta en el árbol como un hijo más a la izquierda. Si se omite la cláusula where, el registro nuevo es insertado en la primera posición (en preorden) en el árbol de la base de datos donde se pueda insertar un registro del mismo tipo que el nuevo.

Ejemplos:

Consideremos que queremos añadir una nueva alumna cuyo nombre es Delia Siordia con número de control 99310168 de la carrera de LI; entonces la inserción del nuevo registro seria de la siguiente manera:

```
Alumno.NombreA:="Delia Siordia";
Alumno.Control:="99310168";
```

```
Alumno.Esp:="ISC";
insert Alumno;
```

Consideremos que deseamos crear la alta de la materia de matemáticas 1 a la alumna con número de control 99310168.

```
Materia.NombreM:="Matemáticas 1";
Materia.Clave:="SCB9334";
Materia.Cred:=8;
insert Materia;
where Alumno.Control="99310168";
```

Modificación de registros existentes.

La instrucción para efectuar cambios a los registros es:

```
Replace
```

Esta instrucción no requiere los datos del registro a modificar como argumento, el registro que se afectará será aquel al que este apuntando el puntero de actualidad, que debe ser el registro que se desea modificar. Ejemplo:

Consideremos que deseamos reemplazar la carrera de la alumna con número de control 99310168.

```
Get hold first Alumno
where Alumno.Control="99310168";
Alumno.Esp:="LI";
replace;
```

Se agrega la palabra hold para que el sistema se entere que se va a modificar un registro.

Eliminación de un registro

Para eliminar un registro se debe apuntar al puntero de actualidad hacia ese registro, después se ejecuta la orden delete, al igual que en la orden replace, se debe poner la orden Hold. Ejemplo:

Consideremos que deseamos borrar al alumno con número de control 99310168.

```
Get hold first Alumno;
where Alumno.Control=99310168;
delete;
```

También se puede borrar un registro(raíz), lo cuál eliminaría todas sus derivaciones (hijos).

Ejemplo:

Consideremos que deseamos eliminar al alumno con número de control 99310168 y todas sus materias, entonces la instrucción quedaría:

```
get hold first Alumno
where Alumno.Control="99310168";
delete;
```

Registros virtuales.

Como se describió anteriormente, en las relaciones muchos a muchos se requería la repetición de datos para conservar la organización de la estructura del árbol de la base de datos. La repetición de la información genera 2 grandes problemas:

La actualización puede generar inconsistencia de los datos.
Se genera un desperdicio considerable de espacio.

Para solventar estos problemas se introdujo el concepto de registro virtual, el cual no contiene datos almacenados, si no un puntero lógico a un registro físico determinado. Cuando se va a repetir un registro en varios árboles de la base de datos, se mantiene una sola copia de ese registro en uno de los árboles y empleamos en los otros registros la utilización de un registro virtual que contiene la dirección del registro físico original.

5. El modelo de red

- ***Modelo conceptual***
- ***Estructura de almacenamiento***

Conceptos básicos.

Una base de datos de red como su nombre lo índica, esta formado por una colección de registros, los cuales están conectados entre sí por medio de enlaces. El registro es similar a una entidad como las empleadas en el modelo entidad-relación. Un registro es una colección de campos (atributos), cada uno de los cuales contiene solamente almacenado un solo valor, el enlace es la asociación entre dos registros exclusivamente, así que podemos verla como una relación estrictamente binaria. Una estructura de datos de red, llamada algunas veces estructura plex, abarca más que la estructura de árbol porque un nodo hijo en la estructura de red puede tener más de un padre. En

otras palabras, la restricción de que en un árbol jerárquico cada hijo puede tener un solo padre, se hace menos severa. Así, la estructura de árbol se puede considerar como un caso especial de la estructura de red tal como lo muestra la siguiente figura. Para ilustrar la estructura de los registros en una base de datos de red, consideremos la base de datos alumno-materia, los registros en lenguaje Pascal entonces quedarían como:

```
type alumno= record
                NombreA:string[30];
                Control:string[8];
                Esp: string[3];
              end;
type materia = record
                Clave:string[7];
                NombreM:string[25];
                 Cred=string[2];
               end;
```

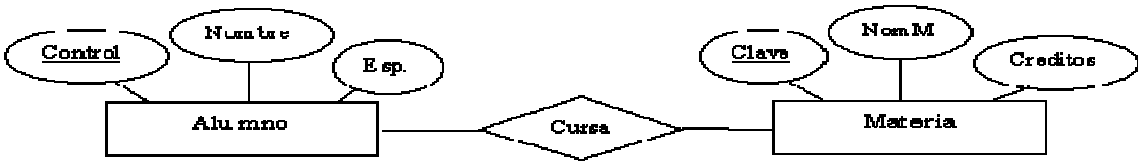
Diagramas de estructura de datos.

Un diagrama de estructura de datos es un esquema que representa el diseño de una base de datos de red. Este modelo se basa en representaciones entre registros por medio de ligas, existen relaciones en las que participan solo dos entidades(binarias) y relaciones en las que participan más de dos entidades (generales) ya sea con o sin atributo descriptivo en la relación. La forma de diagramado consta de dos componentes básicos:

Celdas: representan a los campos del registro.

Líneas: representan a los enlaces entre los registros.

Un diagrama de estructura de datos de red, especifica la estructura lógica global de la base de datos; su representación gráfica se basa en el acomodo de los campos de un registro en un conjunto de celdas que se ligan con otro(s) registro(s), ejemplificaremos esto de la siguiente manera: Consideremos la relación alumno-cursa-materia donde la relación cursa no tiene atributos descriptivos :



Las estructuras de datos según la cardinalidad se representan en los siguientes casos:

Cuando el enlace no tiene atributos descriptivos

Caso 1. Cardinalidad Uno a Uno.

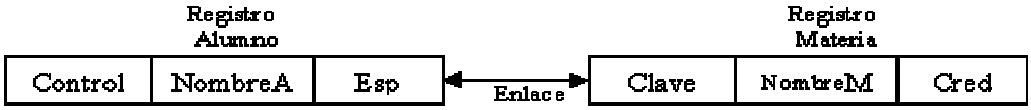


Diagrama de estructura de datos

Caso 2. Cardinalidad Muchos a uno.

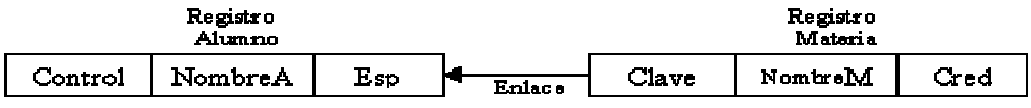


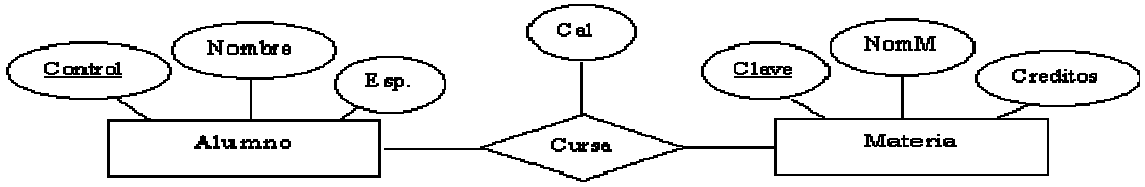
Diagrama de estructura de datos

Caso 3. Cardinalidad Muchos a muchos.

Diagrama de estructura de datos

Cuando el enlace tiene atributos descriptivos.

Consideremos que a la relación cursa le agregamos el atributo Cal (calificación), nuestro modelo E-R quedaría de la siguiente manera:



La forma de convertir a diagramas de estructura de datos consiste en realizar lo siguiente:

1. Realizar la representación de los campos del registro agrupándolos en sus celdas correspondientes.

2. Crear nuevo registro, denominado *Calif,* para este caso, con un solo campo, el de cal (calif).

3. Crear los enlaces indicando la cardinalidad de :

AluCal, del registro Calif al registro *Alumno.*

MatCal, del registro *Calif* al registro *Materia.*

AluCal y MatCal son solo los nombres que emplearemos para identificar el enlace, pueden ser otros y no son empleados para otra cosa.

Los diagramas de estructuras de datos según la cardinalidad se transforman en:

Caso 1. Cardinalidad uno a uno.

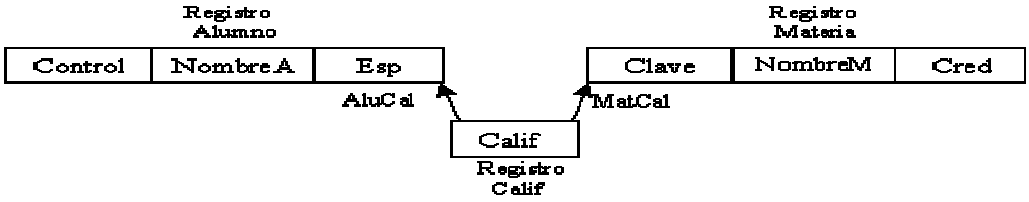


Diagrama de estructura de datos

Caso 2. Cardinalidad Uno a muchos.

Tratamiento de información

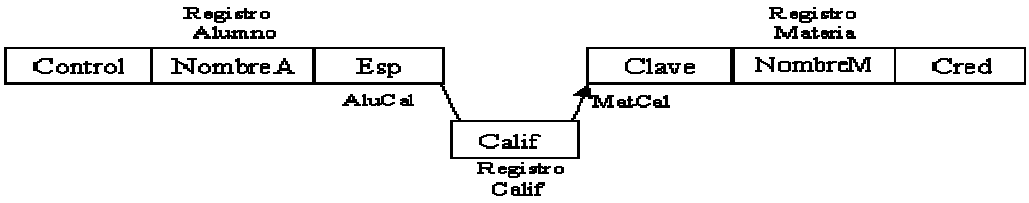


Diagrama de estructura de datos

Caso 3. Cardinalidad Muchos a muchos.

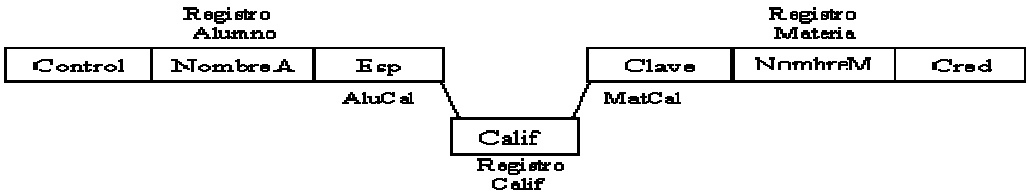
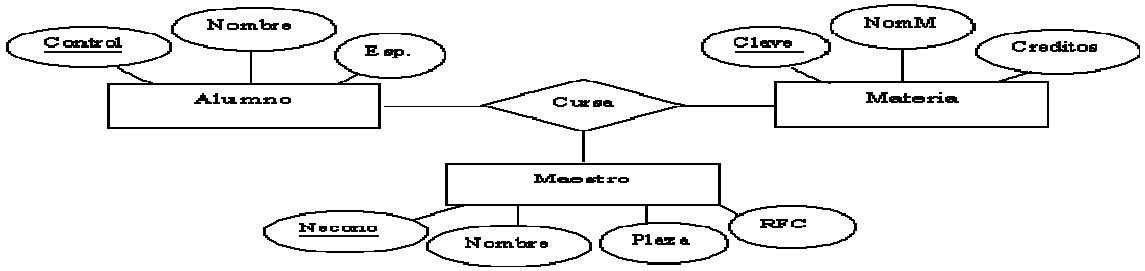


Diagrama de estructura de datos

Diagramas de estructura de datos cuando intervienen más de dos entidades y el enlace no tiene atributos descriptivos. Consideremos que a la relación alumno-cursa-materia le agregamos la entidad maestro, quien es el que imparte dicha materia. Nuestro diagrama E-R quedaría de la siguiente manera:



La transformación a diagramas de estructura de datos se realiza mediante los siguientes pasos:

1.Crear los respectivos registros para cada una de las entidades que intervienen en el modelo.

2.Crear un nuevo tipo de registro que llamaremos Renlace, que puede no tener campos o tener solo uno que contenga un identificador único, el identificador lo proporcionará el sistema y no lo utiliza directamente el programa de aplicación, a este registro se le denomina también como registro ficticio o de enlace o unión. Siguiendo los pasos anteriores nuestra estructura finalmente es: (Considerando una relación con cardinalidad Uno a Uno)

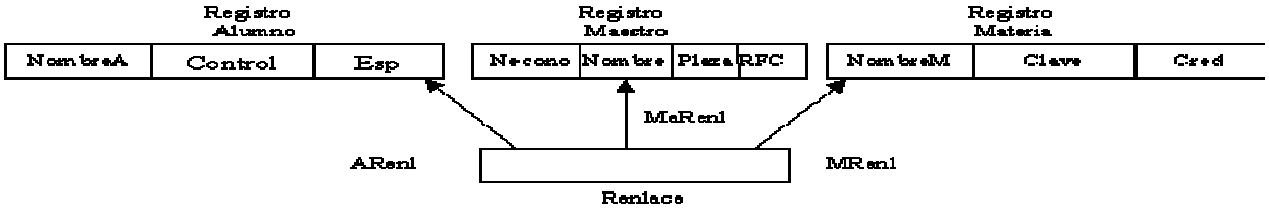


Diagrama de estructura de datos

Ahora si nuestro enlace tuviera atributos descriptivos, se crea el registro con los campos respectivos y se liga indicando el tipo de cardinalidad de que se trate. En este caso tomamos el ejemplo anterior con cardinalidad uno a uno y le agregamos a la relación el atributo calif. (calificación).

Tratamiento de información

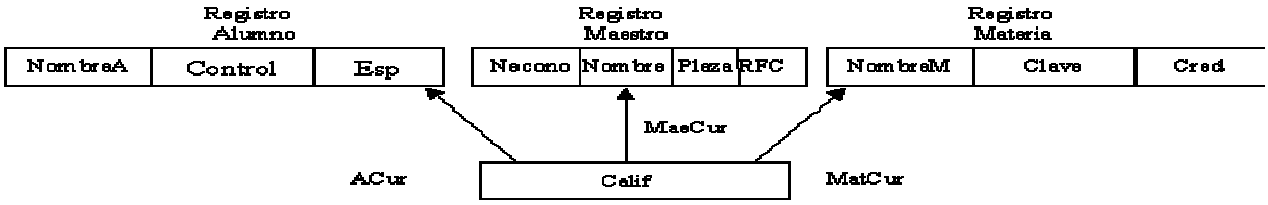
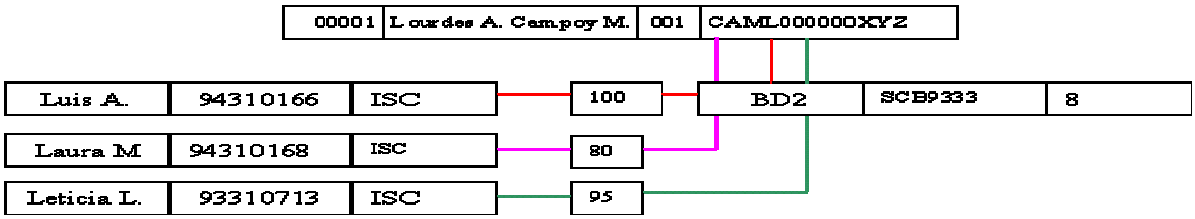


Diagrama de estructura de datos

Considerando el anterior diagrama de estructura de datos, una instancia de este seria: La estructura quedaría:



Este diagrama nos indica que los alumnos Luis A. Laura M. y Leticia L. cursaron la materia Base de datos 2 con La maestra Ing. Lourdes A. Campoy M obteniendo una calificación de 100,80,95 respectivamente. Este modelo fue desarrollado en 1971 por un grupo conocido como CODASYL: Conference on Data System Languages, Data Base Task Group, de ahí el nombre; este grupo es el que desarrolló los estándares para COBOL, el modelo CODASYL ha evolucionado durante los últimos años y existen diversos productos DBMS orientados a transacciones, sin embargo hoy día, estos productos están de salida, ya que este modelo es complejo y no cohesivo; los diseñadores y programadores deben de tener mucho cuidado al elaborar bases de datos y aplicaciones DBTG, además este modelo tiene mucho enfoque de COBOL, gran parte a las deficiencias detectadas en la actualidad se le atribuye a que este modelo fue desarrollado muy pronto antes de que se establecieran correctamente los conceptos esenciales de la tecnología de bases de datos.

6. El modelo relacional

- ***Esquemas de base de datos relacional***

La ventaja del modelo relacional es que los datos se almacenan, al menos conceptualmente, de un modo en que los usuarios entienden con mayor facilidad. Los datos se almacenan como tablas y las relaciones entre las filas y las tablas son visibles en los datos. Este enfoque permite a los usuarios obtener información de la base de datos sin asistencia de sistemas profesionales de administración de información. Las características más importantes de los modelos relacionales son:

1. Es importante saber que las entradas en la tabla tienen un solo valor (son atómicos); no se admiten valores múltiples, por lo tanto la intersección de un renglón con una columna tiene un solo valor, nunca un conjunto de valores.
2. Todas las entradas de cualquier columna son de un solo tipo. Por ejemplo, una columna puede contener nombres de clientes, y en otra puede tener fechas de nacimiento. Cada columna posee un nombre único, el orden de las comunas no es de importancia para la tabla, las columnas de una tabla se conocen como atributos. Cada atributo tiene un dominio, que es una descripción física y lógica de valores permitidos.
3. No existen 2 filas en la tabla que sean idénticas.
4. La información en las bases de datos son representados como datos explícitos, no existen apuntadores o ligas entre las tablas.

En el enfoque relacional es sustancialmente distinto de otros enfoques en términos de sus estructuras lógicas y del modo de las operaciones de entrada/salida. En el enfoque relacional, los datos se organizan en tablas llamadas relaciones, cada una de las cuales se implanta como un archivo. En terminología relacional una fila en una relación representa un registro o una entidad; Cada columna en una relación representa un campo o un atributo. Así, una

relación se compone de una colección de entidades(o registros) cuyos propietarios están descritos por cierto número de atributos predeterminados implantados como campos.

Estructura de las bases de datos relacionales

La arquitectura relacional se puede expresar en términos de tres niveles de abstracción: nivel interno, conceptual y de visión. La arquitectura relacional consta de los siguientes componentes:

Modelo relacional de datos:

En el nivel conceptual, el modelo relacional de datos está representado por una colección de relaciones almacenadas. Cada registro de tipo conceptual en un modelo relacional de datos se implanta como un archivo almacenado distinto.

Submodelo de datos:

Los esquemas externos de un sistema relacional se llaman submodelos relacionales de datos; cada uno consta de uno a más escenarios (vistas) para describir los datos requeridos por una aplicación dada. Un escenario puede incluir datos de una o más tablas de datos. Cada programa de aplicación está provisto de un buffer ("Area de trabajo de usuario") donde el DBMS puede depositar los datos recuperados de la base para su procesamiento, o puede guardar temporalmente sus salidas antes de que el DBMS las escriba en la base de datos.

Esquema de almacenamiento:

En el nivel interno, cada tabla base se implanta como un archivo almacenado. Para las recuperaciones sobre las claves principal o secundaria se pueden establecer uno o más índices para accesar un archivo almacenado.

Sublenguaje de datos:

Es un lenguaje de manejo de datos para el sistema relacional, el álgebra relacional y cálculo relacional, ambos lenguajes son "relacionalmente completos", esto es, cualquier relación que pueda derivarse de una o más tablas de datos, también se puede derivar con u solo comando del sublenguaje. Por tanto, el modo de operación de entrada/Salida en un sistema relacional se puede procesar en la forma: una tabla a la vez en lugar de: un registro a la vez; en otras palabras, se puede recuperar una tabla en vez de un solo registro con la ejecución de un comando del sublenguaje de datos.

Lenguajes de consulta formales.

Los lenguajes de consultas:

Son los lenguajes en el que los usuarios solicitan información de la base de datos. Estos lenguajes son generalmente de más alto nivel que los lenguajes de programación. Los lenguajes de consulta pueden clasificarse como procedimentales y no procedimentales;

En el lenguaje del tipo procedimental el usuario da las instrucciones al sistema para que realice una secuencia de operaciones en la base de datos para calcular el resultado deseado.

En el lenguaje no procedimental, el usuario describe la información deseada sin dar un procedimiento específico para obtener dicha información.

El álgebra relacional es un lenguaje de consulta formal procedimental, el álgebra relacional define operadores que funcionan sobre las tablas (de una manera similar a los operadores +,-,etc. del álgebra común) para llegar al resultado deseado. El álgebra relacional es difícil de utilizar, debido en parte a que es procedimental, esto es, al utilizar el álgebra relacional no sólo debemos saber lo que queremos, también cómo obtenerlo. En el proceso de bases de datos comerciales el álgebra relacional se utiliza de manera poco frecuente. Aunque unos cuantos productos exitosos DBMS sí tienen opciones del álgebra relacional, éstas son poco utilizadas en vista de su complejidad. El álgebra relacional toma dos o más tablas como entrada produce una nueva tabla como resultado de la serie de operaciones. Las operaciones fundamentales en el álgebra relacional son seleccionar, proyectar, producto cartesiano, renombrar, unión y diferencia de conjuntos. Además de las operaciones fundamentales existen otras operaciones como son: intersección de conjuntos, producto natural, división y asignación.

** Operaciones fundamentales **

Las operaciones seleccionar, proyectar y renombrar, son denominadas operaciones unitarias ya que operan sobre una tabla. Las otras operaciones operan sobre pares de relaciones y, por tanto se llaman operaciones binarias.

** La operación seleccionar.*

Esta operación selecciona tuplas (filas) que satisfacen una instrucción(condición) dada de una tabla. Se representa por medio de paréntesis.

(nombre_tabla WHERE condición);

La oración de la instrucción después de la cláusula WHERE puede incluir condiciones de igualdad como =,=,<,>,>=,<=,además que se puede hacer una oración más compleja usando los conectores y (^) y o (v).

** La operación Proyectar.*

Consiste en identificar las columnas (atributos en el modelo E-R) que nos interesa conocer. Se representa por medio de corchetes. Si este se omite indicara que se desea obtener todas las columnas de la tabla en cuestión.

(nombre_tabla WHERE condición) [Nombre_atributo];

** La operación Producto cartesiano.*

Consiste en multiplicar todas las tuplas entre tablas, obteniendo como resultado una tabla que contiene todas las columnas de ambas tablas. Se especifica con la orden TIMES.

Nombre_tabla TIMES Nombre_tabla;

** La operación Join.*

Consiste en obtener el producto (multiplicación) de todas las tuplas de una tabla con las de la otra, para posteriormente evaluar aquellas cuyo campo en común sea igual generando como resultado una nueva tabla que tiene como tuplas (renglones) que cumplen con la condición establecida. Se representa con la orden JOIN. La orden Join es colocada entre las dos tablas a multiplicar después de que la primera especifica la operación de selección y proyección.

(Tabla)[atributo] JOIN (Tabla)[Atributo];

** La operación Divide.*

Toma dos relaciones, una binaria y la otra unaria, construye una relación formada por todos los valores de un atributo de la relación binaria que concuerdan (en el otro atributo) con todos los valores de la relación unaria. Se representa con la orden DIVIDEBY.

NomTablaBin DIVIDEBY NomTablaUna

** La operación Diferencia.*

Construye una relación formada por todas las tuplas (filas) de la primera relación que no aparezcan en la segunda de las dos relaciones especificadas. Se representa con la orden MINUS.

Nom_tablaA MINUS NomTablaB;

** La operación Unión.*

Construye una relación formada por todas las tuplas de la primera relación y todas las tuplas de la segunda relación. El requisito es que ambas relaciones sean del mismo tipo.

Nom_TablaA UNION Nom_tablaB

** La operación intersección.*

Construye una nueva tabla compuesta por todas las tuplas que están en la primera y segunda tabla.

Nom_TablaA INTERSEC Nom_tablaB

Ejemplos:

Para ejemplificar las notaciones anteriores consideremos el ejemplo

ALUMNO - cursa - MATERIA, que tienen los siguientes atributos:

NControl NControl Clave
NombreA Clave NombreM
Especialidad Calif Créditos
Dirección

Representando en tablas a los atributos quedarían de la siguiente forma:

Tabla alumno:

NControl	NombreA	Especialidad	Dirección

Tabla cursa:

NControl	Clave	Calif

Tabla materia:

Clave	NombreM	Créditos

1.- Obtener el nombre de todos los alumnos que están inscritos en la Institución.

(Alumno) [NombreA];

2.- Obtener el nombre de los alumnos que cursan la materia Base de datos 1 cuya clave es SCB9333

(Alumno) JOIN (Cursa where Clave='SCB9333') [NombreA];

3.- Obtener los nombres de los alumnos de la especialidad de Ing. Sistemas que cursan la materia Base de datos 2.

((Alumno)[especialidad,NombreA,NControl]

JOIN (Cursa) where especialidad = 'ISC')[Clave,NombreA])
JOIN (Materia where NombreM='BD2')[NombreA];

En el álgebra relacional no solo debemos saber lo que queremos si no también como obtenerlo, al realizar las consultas debemos especificar el nombre de la tabla a utilizar en caso de que deseemos realizar una operación con un atributo que las otras tablas no tienen debemos "arrastrar" dicho atributo para poder utilizarlo, como lo es en el caso anterior, en donde requerimos el nombre del alumno que solamente lo tiene la tabla alumno, pro también deseamos que se cumpla la condición NombreM=BD2, como no podemos relacionar directamente a ambas tablas empleamos la tabla cursa de donde obtenemos la clave de las materias y mantenemos el nombre del alumno (NombreA) finalmente con la orden JOIN se combinan las tablas por el campo común que tienen que es clave así que obtenemos una tabla con todas las materias que cursan los alumnos de ISC, de donde seleccionamos solo aquella que se llame BD2 con la orden Join obtenemos esta nueva tabla de donde por último proyectamos el atributo NombreA que hemos venido "arrastrando".

Lenguajes de consultas comerciales

Un lenguaje de consulta comercial proporciona una interfaz más amigable al usuario. Un ejemplo de este tipo de lenguaje es el SQL, (Structured Query Languaje, Lenguaje de Consulta Estructurado). Las partes más importantes del SQL son:

DDL: Lenguaje de definición de datos (que nos permite crear las estructuras)

DML: Lenguaje de manipulación de datos (que nos permite tener acceso a las estructuras para suprimir, modificar e insertar)

En este apartado estudiaremos la forma básica para realizar consultas con SQL, en el apartado 3.4: Modificación de la base de datos, estudiaremos lo que concierne a la modificación de las tablas.

La estructura básica de una expresión en SQL contiene 3 partes, Select, From y Where.

La cláusula Select se usa para listar los atributos que se desean en el resultado de una consulta.

From, Lista las relaciones que se van a examinar en la evaluación de la expresión.

Where, es la definición de las condiciones a las que puede estar sujeta una consulta.

La consulta típica de SQL tiene la siguiente forma:

Select A1,A2,A3...An
From r1,r2,r3...rm
Where Condición(es)

Donde:

A1,A2,A3...An: Representan a cada atributo(s) o campos de las tablas de la base de datos relacional.

R1,r2,r3...rm: Representan a la(s) tabla(s) involucradas en la consulta.

Condición: Es el enunciado que rige el resultado de la consulta.

Si se omite la cláusula Where, la condición es considerada como verdadera, la lista de atributos (A1,A2..An) puede sustituirse por un asterisco (*), para seleccionar todos los atributos de todas las tablas que aparecen en la cláusula From.

Funcionamiento del SQL.

El SQL forma el producto cartesiano de las tablas involucradas en la cláusula From, cumpliendo con la condición establecida en la orden Where y después proyecta el resultado con la orden select. Para nuestros ejemplos consideremos una tabla llamada CURSO, que contiene los siguientes campos:

Nombre del campo	**Descripción**
NumC	Número del curso, único para identificar cada curso
NombreC	Nombre del curso, también es único
DescC	Descripción del curso
Creditos	Créditos, número de estos que gana al estudiante al cursarlo
Costo	Costo del curso.
Depto	Departamento académico que ofrece el curso.

Datos contenidos en la tabla CURSO

NumC	**NombreC**	**DescC**	**Creditos**	**Costo**	**Depto**

Tratamiento de información

A01	Liderazgo	Para público General	10	100.00	Admón.
S01	Introducción a la inteligencia artificial	Para ISC y LI	10	90.00	Sistemas.
C01	Construcción de torres	Para IC y Arquitectura	8	0.00	Ciencias
B01	Situación actual y perspectivas de la alimentación y la nutrición	Para IB	8	80.00	Bioquímica
E01	Historia presente y futuro de la energía solar	IE e II	10	100.00	Electromecánica.
S02	Tecnología OLAP	Para ISC y LI	8	100.00	Sistemas
C02	Tecnología del concreto y de las Estructuras	Para IC	10	100.00	Ciencias
B02	Metabolismo de lípidos en el camarón	Para IB	10	0.00	Bioquímica
E02	Los sistemas eléctricos de potencia	Para IE	10	100.00	Electromecánica
S03	Estructura de datos	Para ISC y LI	8	0.00	Sistemas
A01	Diseño bioclimático	Para Arquitectura	10	0.00	Arquitectura
C03	Matemáticas discretas	General	8	0.00	Ciencias
S04	Circuitos digitales	Para ISC	10	0.00	Sistemas
S05	Arquitectura de Computadoras	Para ISC	10	50.00	Sistemas
I01	Base de Datos Relacionales	Para ISC y LI	10	150.00	Informática

Ejemplos de consultas:
OBTENCIÓN DE UNA TABLA ENTERA
Obtener toda la información disponible sobre un curso donde Costo sea 0.

```
SELECT *
FROM CURSO
WHERE Costo=0.00
```

Resultado de la consulta anterior.

NumC	**NombreC**	**DescC**	**Creditos**	**Costo**	**Depto**
C01	Construcción de torres	Para IC y Arquitectura	8	0.00	Ciencias
B02	Metabolismo de lípidos en el camarón	Para IB	10	0.00	Bioquímica
S03	Estructura de datos	Para ISC y LI	8	0.00	Sistemas
A01	Diseño bioclimático	Para Arquitectura	10	0.00	Arquitectura
C03	Matemáticas discretas	General	8	0.00	Ciencias

Colocamos un * debido a que no nos limitan la información de la tabla, es decir nos piden que mostremos todos los datos atributo de la tabla CURSO. Como la única condición en la sentencia WHERE es que la tarifa del curso sea igual a 0, esta consulta regresa todas las tuplas donde se encuentre que Costo = 0.00. Debido a que Costo es un campo numérico, la condición solo puede comparar con campos del mismo tipo. Para representar valores negativos se antepone a la izquierda el signo (-), en este ejemplo se considera solo el signo (=) para establecer la condición, sin embargo otros operadores que se pueden utilizar son:
Menor que <
Mayor que >
Menor o igual que <=
Mayor o igual que >=

Tratamiento de información

Diferente <>
Además de los operadores booleanos AND, NOT, OR.
Cabe señalar que en la sentencia Where cuando se requiere establecer condiciones con cadenas, estas son delimitadas por apóstrofos (‘’). Las expresiones de cadenas son comparadas carácter por carácter, dos cadenas son iguales solo si coinciden todos los caracteres de las mismas.
Ejemplos de consultas con cadenas:
Obtener toda la información sobre cualquier curso que ofrezca el departamento de Ciencias.

```
SELECT *
FROM CURSO
WHERE Depto = 'Ciencias';
```

Resultado de la consulta.

NumC	NombreC	DescC	Creditos	Costo	Depto
C01	Construcción de torres	Para IC y Arquitectura	8	0.00	Ciencias
C02	Tecnología del concreto y de las Estructuras	Para IC	10	100.00	Ciencias
S04	Circuitos digitales	Para ISC	10	0.00	Sistemas

VISUALIZACIÓN DE COLUMNAS ESPECIFICADAS.
En los ejemplos anteriores obteníamos toda la tabla completa, ahora veremos como mostrar solo algunos atributos específicos de una tabla.
Obtener los valores NumC,NombreC y Depto, en este orden de toda la tabla curso.

```
SELECT NumC, NombreC, Depto
FROM CURSO;
```

Resultado de la consulta:

NumC	NombreC	Depto
A01	Liderazgo	Admón.
S01	Introducción a la inteligencia artificial	Sistemas.
C01	Construcción de torres	Ciencias
B01	Situación actual y perspectivas de la alimentación y la nutrición	Bioquímica
E01	Historia presente y futuro de la energía solar	Electromecánica.
S02	Tecnología OLAP	Sistemas
C02	Tecnología del concreto y de las Estructuras	Ciencias
B02	Metabolismo de lípidos en el camarón	Bioquímica
E02	Los sistemas eléctricos de potencia	Electromecánica
S03	Estructura de datos	Sistemas
A01	Diseño bioclimático	Arquitectura
C03	Matemáticas discretas	Ciencias
S04	Circuitos digitales	Sistemas
S05	Arquitectura de Computadoras	Sistemas
I01	Base de Datos Relacionales	Informática

Tratamiento de información

Observamos que en este caso no se tiene la sentencia Where, no existe condición, por lo tanto, todas las filas de la tabla CURSO se recuperan, pero solo se visualizaran las tres columnas especificadas. Así mismo, empleamos la (,) para separar los campos que deseamos visualizar.

VISUALIZACIÓN DE UN SUBCONJUNTO DE FILAS Y COLUMNAS

Seleccionar los valores NumC, Depto y Costo para todos los cursos que tengan un Costo inferior a $100

```
SELECT NumC, Depto, Costo
FROM CURSO
WHERE Costo < 100.00
```

Como resultado de esta consulta se obtendrán todas aquellas tuplas que tengan un costo en CTARIFA menor que 100, y se visualizaran solo los campos de NumC, Depto,Costo. Podemos observar que este ejemplo cubre el formato general de una consulta SQL.

La palabra clave DISTINCT

DISTINCT, es una palabra reservada que elimina las filas que duplicadas en el resultado de una consulta.

Visualizar todos los departamentos académicos que ofrezcan cursos, rechazando los valores duplicados.

```
SELECT DISTINCT Depto
FROM CURSO;
```

Resultado de la consulta

Depto
Administración
Sistemas
Ciencias
Bioquímica
electromecánica
Arquitectura
Informática

La palabra DISTINCT va estrictamente después de la palabra SELECT. De no haberse utilizado la palabra DISTINCT, el resultado hubiera mostrado todas las tuplas del atributo Depto que se encontraran, es decir, se hubiera visualizado la columna de Depto completamente.

EMPLEO DE LOS CONECTORES BOOLEANOS (AND, OR, NOT)

Para emplear las condiciones múltiples dentro de la sentencia WHERE, utilizamos los conectores lógicos.

El conector AND.

Este conector pide al sistema que seleccione una sola columna únicamente si ambas condiciones se cumplen. Obtener toda la información sobre todos los cursos que ofrece el departamento Sistemas que tengan una tarifa igual a 0.

```
SELECT *
FROM CURSO
WHERE Depto='Sistemas' AND Costo=0.00;
```

El resultado de esta consulta sería todas aquellas tuplas que cumplan exactamente con las dos condiciones establecidas.

El conector OR.

Este conector al igual que el AND permite conectar condiciones múltiples en la sentencia WHERE, a diferencia del conector AND, el OR permite la selección de filas que cumplan con una sola de las condiciones establecidas a través de este conector. Obtener toda la información existente sobre cualquier curso ofrecido por los departamentos Arquitectura o Bioquímica.

```
SELECT *
FROM CURSO
WHERE Depto = 'Arquitectura' OR Depto= 'Bioquímica';
```

El resultado de esta consulta será la de visualizar todas aquellas tuplas donde se cumpla cualquiera de las 2 condiciones, es decir mostrara todas las tuplas que tengan en el atributo Depto=Arquitectura o Bioquímica.

El conector NOT

Este nos permite marcar aquellas tuplas que por alguna razón no deseamos visualizar. Obtener el nombre del curso y del departamento de todos los cursos que no sean ofrecidos por el departamento Sistemas.

```
SELECT NombreC, Depto
FROM CURSO
WHERE NOT (Depto='Sistemas');
```

Tratamiento de información

JERARQUÍA DE OPERADORES BOOLEANOS.
En orden descendente (de mayor a menor prioridad)
NOT
AND
OR
Existen dos formas para realizar consultas: Join de Querys y Subquerys. Cuando en la sentencia From colocamos los nombres de las tablas separados por comas se dice que efectuamos una consulta de la forma Join de Querys, en este caso se requiere anteponer el nombre de la tabla y un punto al nombre del atributo. En el Join de Querys el resultado que se produce con las tablas que intervienen en la consulta es la concatenación de las tablas, en donde los valores de una columna de la primera tabla coinciden con los valores de una segunda tabla, la tabla de resultado tiene una fila por cada valor coincidente que resulte de las dos tablas originales. Para ejemplificar esto, consideremos 2 tablas: Tabla1 y Tabla2, entonces:

C1	**C2**	**C3**		**CA**	**CB**
A	AAA	10		35	R
B	BBB	45		10	S
C	CCC	55		65	T
D	DDD	20		20	U
E	EEE	20		90	V
F	FFF	90		90	W
G	GGG	15		75	X
H	HHH	90		90	Y
				35	Z

Resultado de la operación Join:

C1	**C2**	**C3**	**CA**	**CB**
A	AAA	10	10	S
D	DDD	20	20	U
E	EEE	20	20	U
F	FFF	90	90	V
F	FFF	90	90	W
F	FFF	90	90	Y
H	HHH	90	90	V
H	HHH	90	90	W
H	HHH	90	90	Y

Como podemos observar, la comparación se efectuó por las columnas C3 y CA, que son donde se encontraron valores iguales, el resultado muestra una tupla por cada coincidencia encontrada. Cuando las consultas se anidan se conoce como Subquerys o subconsultas. Este tipo de consulta obtiene resultados parciales reduciendo el espacio requerido para realizar una consulta. Nota: Todas las consultas que se resuelven con subquerys pueden resolverse

con Join de Querys, pero no todas las consultas hechas con Join de Querys pueden resolverse utilizando Subquerys.
Para ejemplificar lo anterior consideremos el ejemplo ALUMNO-cursa-MATERIA, que tienen los siguientes atributos:

NControl	NControl	Clave
NombreA	Clave	NombreM
Especialidad	Calif	Creditos
Dirección		

Representando en tablas a los atributos quedarían de la siguiente forma:

Tabla alumno:

NControl	NombreA	Especialidad	Dirección

Tabla cursa:

NControl	Clave	Calif

Tabla materia:

Clave	NombreM	Creditos

Obtener el nombre de la materia que cursa el alumno con número de control 97310211 con créditos igual a ocho.

```
SELECT NombreA
FROM Materia
WHERE creditos='8' and clave in(SELECT clave
                                FROM cursa
                                WHERE   NControl='97310211';
```

Obtener el número de control del alumno que tenga alguna calificación igual a 100

```
SELECT DISTINC(NControl)
FROM Cursa
WHERE Calif='100';
```

Obtener el nombre de las materias que cursa el alumno Salvador Chávez.

```
SELECT NombreM
FROM Materia
WHERE Clave in (SELECT DISTINC (Clave)
                FROM Cursa
                WHERE NControl in (SELECT NControl)
                                   FROM Alumno
                                   WHERE NombreA='Salvador Chávez'));
```

FUNCIONES AVANZADAS APLICABLES A CONSULTAS

Existen funciones que permiten la agilización de consultas similares a una hoja de cálculo, ya que trabajan en base a renglones y columnas.

COUNT ( ): Cuenta el número de tuplas en la columna establecida
MIN ( ): Localiza el valor mínimo de la columna establecida
MAX ( ): Localiza el valor máximo de la columna establecida.
AVG ( ): Obtiene el promedio de valores de la columna establecida
SUM ( ): Obtiene el valor total que implican los valores obtenidos en la columna establecida.

Ejemplos:

Obtener el número de alumnos que existen en la carrera de Ingeniería en Sistemas Computacionales.

```
SELECT Count (*)
FROM Alumno
WHERE especialidad='ISC';
```

Obtener la máximo calificación que ha obtenido J.M. Cadena.

```
SELECT Max(Calif)
FROM Cursa
WHERE NControl IN (SELECT NControl
                   FROM Alumno
```

```
                                    WHERE NombreA= 'J.M. Cadena ');
```

Obtener el promedio de calificaciones de Salvador Chávez.

```
SELECT Avg (Calif)
FROM Cursa
WHERE NCotrol IN (SELECT NControl
                  FROM Alumno
                  WHERE NombreA='Salvador Chávez');
```

Obtener la suma total de las calificaciones obtenidas por Daniel Colín.

```
SELECT Sum (Calif)
FROM Cursa
WHERE NControl IN (SELECT NControl
                   FROM Alumno
                   WHERE NombreA='Daniel Colín');
```

Hasta aquí hemos visto el manejo sencillo de realizar consultas con SQL, hay que destacar que en la realización de consultas anidadas se tiene que poner cuidando a la prioridad de los operadores, teniendo cuidado también al momento de agrupar los paréntesis que involucran las condiciones con los operadores.

Modificación de la Base de datos

Como se mencionó al inicio de este apartado del SQL, esté cuenta con módulos DDL, para la definición de datos que nos permite crear o modificar la estructura de las tablas. Las instrucciones para realizar estas operaciones son:

CREATE TABLE: Nos permite crear una tabla de datos vacía.
INSERT: Permite almacenar registros en una tabla creada.
UPDATE: Permite modificar datos de registros almacenados en la tabla.
DELETE: Borra un registro entero o grupo de registros de una tabla.
CREATE INDEX: Crea un índice que nos puede auxiliar para las consultas.
DROP TABLE: Permite borrar una tabla.
DROP INDEX: Borra el índice indicado.

Para ejemplificar las instrucciones anteriores consideremos el ejemplo

```
ALUMNO -      cursa          - MATERIA, que tienen los siguientes atributos:
NControl      NControl       Clave
NombreA       Clave          NombreM
Especialidad  Calif          Creditos
Dirección
```

* Estructura de la sentencia CREATE TABLE.

```
CREATE TABLE <Nombre de la tabla>
  (
   Atributo1: tipo de dato longitud ,
   Atributo2: tipo de dato longitud ,
   Atributo3: tipo de dato longitud ,
   :
   :
  Atributon: tipo de dato longitud ,
  PRIMARY KEY (Opcional) ) ;
```

Los campos pueden definirse como NOT NULL de manera opcional excepto en la llave primaria para lo cual es obligatorio. Además al definir la llave primaria se genera automáticamente un índice con respecto al campo llave; para definir la llave la denotamos dentro de los paréntesis de PRIMARY KEY.

Ejemplo:

Crear la tabla alumno con los atributos antes descritos, tomando como llave el numero de control.

```
CREATE TABLE Alumno
        (
        NControl char(8) NOT NULL,
        NombreA char(20),
        Especialidad char(3),
        Dirección char(30),
        PRIMARY KEY (NControl) );
```

Tabla Alumno:

NControl NombreA Especialidad Dirección

Pueden existir más de una llave primaria, esto es si se requiere, se crearán tantos índices como llaves primarias se establezcan. Pueden existir tantos campos Not Null (No nulos) como se requieran; En si estructurar la creación de una tabla es siempre parecida al ejemplo anterior.

* Estructura de la sentencia INSERT

```
INSERT
INTO  Nombre de la tabla a la que se le va a insertar el registro
VALUES  (Conjunto de valores del registro ) ;
```

Ejemplo:

Insertar en la tabla Alumno, antes creada los datos del alumno Daniel colín, con numero de control 95310518 de la especialidad de Ingeniería civil, con domicilio Abasolo Norte #45.

```
INSERT
INTO Alumno
VALUES("95310518","Daniel Colín","IC","Abasolo Norte #45") ;
```

Nótese que la inserción de los datos se realiza conforme la estructura que se implanto en la tabla, es decir en el orden en que se creo dicha tabla. En caso de querer omitir un dato que no sean no nulos solamente se ponen las comillas indicando el vacío de la cadena.

* Estructura de la Sentencia CREATE INDEX

```
CREATE INDEX Nombre que se le asignara al índice.
ON Nombre de la taba a la cual se le creara el índice (Campo(s) por el cual se creara el índice);
```

Ejemplo:

Crear un índice de la tabla Alumno por el campo Especialidad.

```
CREATE INDEX Indice1
ON Alumno(Especialidad);
```

Este índice contendrá a todos los alumnos ordenados por el campo especialidad.

```
CREATE INDEX UNIQUE INDEX Indice2
ON Alumno (Especialidad);
```

En la creación de este índice utilizamos la sentencia UNIQUE, es un indicador para permitir que se cree un índice único por especialidad, esta sentencia siempre se coloca antes de CREATE INDEX. En este ejemplo se creara un índice que contenga un alumno por especialidad existente.

* Estructura de la sentencia UPDATE

```
UPDATE Nombre de la tabla en donde se modificaran los datos.
SET Valores
WHERE (Condición);
```

Ejemplo:

Modificar el número de control del registro de Daniel Colín de la Tabla alumno por el número 96310518.

```
UPDATE Alumno
SET NControl  '96310518'
WHERE NombreA='Daniel Colín';
```

* Estructura de la sentencia DROP TABLE

```
DROP TABLE Nombre de la tabla a borrar ;
```

Ejemplo:

Borrar la tabla Alumno creada anteriormente.

```
DROP TABLE Alumno;
```

* Estructura de la sentencia DROP INDEX

```
DROP INDEX Nombre del índice a borrar;
```

Ejemplo:

Borrar el índice Indice1 creado anteriormente.

```
DROP INDEX Indice1;
```

* Estructura de la sentencia DELETE

```
DELETE
FROM Nombre de la tabla
WHERE Condición;
```

Ejemplos:

- Borrar el registro cuyo número de control es 95310386.

```
DELETE
```

FROM Alumno
WHERE Control='95310386';

- Borrar todos los registros de la tabla alumno.

DELETE
FROM Alumno;

En el primer ejemplo, se borrara todo el registro(todos los datos), del alumno con número de control = 95310386. En el segundo ejemplo se borraran todos los registros de la tabla alumno, pero sin borrar la estructura de la tabla, ya que la orden Delete solo borra registros, la sentencia Drop Table es la que borra toda la estructura de la tabla junto con los registros de la misma.

Vistas.

Una vista se define en SQL usando la orden CRETE VIEW. Para definir una vista debemos dar a la vista un nombre y declarar la consulta que calcula la vista. Una vez que establecemos una vista, podemos ejecutar una sentencia SELECT que referencie a esa vista. El sistema asociará la vista SQL con una tabla base y extraerá y visualizará, entonces, los datos de la tabla base. Esto significa que una vista no contiene datos duplicados de una tabla base. No tiene absolutamente ningún dato, puesto que no es una tabla real, todo el proceso se realiza con los datos almacenados en la tabla base. Es decir se percibe como una tabla virtual. Las ordenes que se utilizan para la manipulación de vistas son:

CREATE VIEW: Crea una tabla virtual.
DROP VIEW: Elimina una vista creada anteriormente.
Estructura de la sentencia CREATE VIEW.

CREATE VIEW Nombre de la vista AS (Expresión de consulta);

Para nuestros ejemplos consideremos de nuevo la tabla llamada CURSO, que contiene los siguientes campos:

Nombre del campo	**Descripción**
NumC	Número del curso, único para identificar cada curso
NombreC	Nombre del curso, también es único
DescC	Descripción del curso
Creditos	Créditos, número de estos que gana al estudiante al cursarlo
Costo	Costo del curso.
Depto	Departamento académico que ofrece el curso.

Que contiene los siguientes datos:

NumC	**NombreC**	**DescC**	**Créditos**	**Costo**	**Depto**
A01	Liderazgo	Para público General	10	100.00	Admón.
S01	Introducción a la inteligencia artificial	Para ISC y LI	10	90.00	Sistemas.
C01	Construcción de torres	Para IC y Arquitectura	8	0.00	Ciencias
B01	Situación actual y perspectivas de la alimentación y la nutrición	Para IB	8	80.00	Bioquímica
E01	Historia presente y futuro de la energía solar	IE e II	10	100.00	Electromecánica.
S02	Tecnología OLAP	Para ISC y LI	8	100.00	Sistemas
C02	Tecnología del concreto y de las Estructuras	Para IC	10	100.00	Ciencias
B02	Metabolismo de lípidos en el camarón	Para IB	10	0.00	Bioquímica
E02	Los sistemas eléctricos de potencia	Para IE	10	100.00	Electromecánica

S03	Estructura de datos	Para ISC y LI	8	0.00	Sistemas
A01	Diseño bioclimático	Para Arquitectura	10	0.00	Arquitectura
C03	Matemáticas discretas	General	8	0.00	Ciencias
S04	Circuitos digitales	Para ISC	10	0.00	Sistemas
S05	Arquitectura de Computadoras	Para ISC	10	50.00	Sistemas
I01	Base de Datos Relacionales	Para ISC y LI	10	150.00	Informática

Ejemplos:

* Crear una vista (tabla virtual), denominada CursosS, que contenga las filas solo correspondientes a cursos ofrecidos por el departamento Sistemas. La vista deberá contener todas las columnas de la tabla CURSO, con la excepción de la columna Depto, la secuencia, de izquierda a derecha de las columnas, deberá ser: NombreC, NumC, Creditos, Costo Y DescC.

```
CREATE VIEW CursosS AS
   SELECT NombreC,NumC,Creditos,Costo,DescC
   FROM CURSO
   WHERE DescC='Sistemas';
```

Observemos que después del nombre de la vista ponemos la sentencia AS, esto para definir la estructura de la vista, la estructura en si de la vista esta formada por la consulta anteriormente vista utilizando la orden SELECT.

* Crear una vista denominada CursosCaros, correspondientes a las filas de la tabla CURSO, en donde la tarifa exceda de $150, las columnas de la vista deberán tener los nombres ClaveCurso, NombreCurso y CostoCaro.

```
CREATE VIEW CursosSCaros(ClaveCurso,NombreCurso,CostoCaro) As
   SELECT NumC,NombreC, Costo
   FROM Curso
   WHERE Costo > 150;
```

Observamos que después del nombre de la vista CursosCaros ponemos los nombres que se nos pidieron tuvieran los campos de la vista(ClaveCurso,...), después se realiza la consulta correspondiente para generar el resultado deseado.

Visualizar las vistas

Creamos una tabla virtual que contiene los datos de las consultas que deseamos, ahora nos falta visualizar estos datos, para ello utilizamos la sentencia SELECT y realizamos la consulta:

```
SELECT * FROM CursosCaros;
```

De esta consulta podemos observar que mostramos todos los campos que la vista contiene, aunque podemos visualizar solo alguno de ellos, también observamos que sustituimos el nombre de la vista por el de la tabla junto a la sentencia FROM, esto es por que una vista es una tabla virtual, pero guarda los datos como cualquier tabla normal.

Eliminar una vista

Como si fuera una tabla normal, las vistas también pueden borrarse, para ello utilizamos la sentencia DROP VIEW.

Estructura de la sentencia DROP VIEW.

DROP VIEW Nombre de la vista a borrar;

Ejemplo: Borrar la vista CursosCaros creada anteriormente.

```
DROP VIEW CursosCaros;
```

7. Diseño relacional

- ***Proceso de normalización***

Peligros en el diseño de bases de datos relacionales.

Uno de los retos en el diseño de la base de datos es el de obtener una estructura estable y lógica tal que:

El sistema de base de datos no sufra de anomalías de almacenamiento.

El modelo lógico pueda modificarse fácilmente para admitir nuevos requerimientos.

Una base de datos implantada sobre un modelo bien diseñado tiene mayor esperanza de vida aun en un ambiente dinámico, que una base de datos con un diseño pobre. En promedio, una base de datos experimenta una reorganización general cada seis años, dependiendo de lo dinámico de los requerimientos de los usuarios. Una base

de datos bien diseñada tendrá un buen desempeño aunque aumente su tamaño, y será lo suficientemente flexible para incorporar nuevos requerimientos o características adicionales. Existen diversos riesgos en el diseño de las bases de datos relacionales que afecten la funcionalidad de la misma, los riesgos generalmente son la redundancia de información y la inconsistencia de datos. La normalización es el proceso de simplificar la relación entre los campos de un registro. Por medio de la normalización un conjunto de datos en un registro se reemplaza por varios registros que son más simples y predecibles y, por lo tanto, más manejables. La normalización se lleva a cabo por cuatro razones:

Estructurar los datos de forma que se puedan representar las relaciones pertinentes entre los datos.
Permitir la recuperación sencilla de los datos en respuesta a las solicitudes de consultas y reportes.
Simplificar el mantenimiento de los datos actualizándolos, insertándolos y borrándolos.
Reducir la necesidad de reestructurar o reorganizar los datos cuando surjan nuevas aplicaciones.

En términos más sencillos la normalización trata de simplificar el diseño de una base de datos, esto a través de la búsqueda de la mejor estructuración que pueda utilizarse con las entidades involucradas en ella.

Pasos de la normalización:

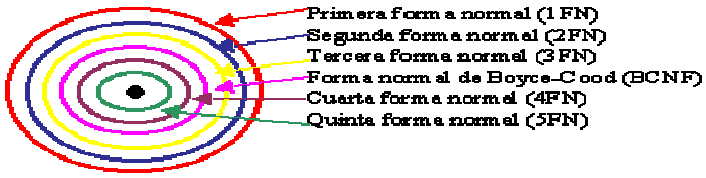
1. Descomponer todos los grupos de datos en registros bidimensionales.
2. Eliminar todas las relaciones en la que los datos no dependan completamente de la llave primaria del registro.
3. Eliminar todas las relaciones que contengan dependencias transitivas.

La teoría de normalización tiene como fundamento el concepto de formas normales; se dice que una relación está en una determinada forma normal si satisface un conjunto de restricciones.

Primera y segunda forma normal.

Formas normales. Son las técnicas para prevenir las anomalías en las tablas. Dependiendo de su estructura, una tabla puede estar en primera forma normal, segunda forma normal o en cualquier otra.

Relación entre las formas normales:



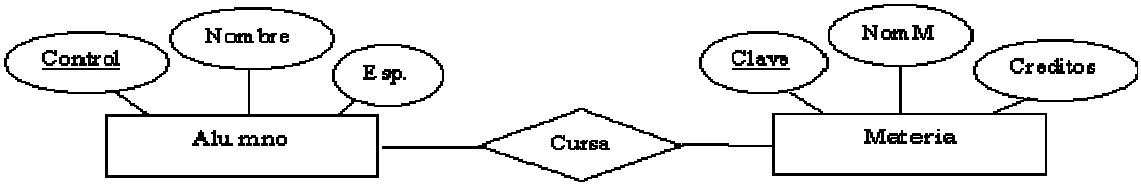
Primera forma normal.

Definición formal: Una relación R se encuentra en 1FN si y solo sí por cada renglón columna contiene valores atómicos.

Abreviada como 1FN, se considera que una relación se encuentra en la primera forma normal cuando cumple lo siguiente:

- Las celdas de las tablas poseen valores simples y no se permiten grupos ni arreglos repetidos como valores, es decir, contienen un solo valor por cada celda.
- Todos los ingresos en cualquier columna(atributo) deben ser del mismo tipo.
- Cada columna debe tener un nombre único, el orden de las columnas en la tabla no es importante.
- Dos filas o renglones de una misma tabla no deben ser idénticas, aunque el orden de las filas no es importante.

Por lo general la mayoría de las relaciones cumplen con estas características, así que podemos decir que la mayoría de las relaciones se encuentran en la primera forma normal. Para ejemplificar como se representan gráficamente las relaciones en primera forma normal consideremos la relación alumno cursa materia cuyo diagrama E-R es el siguiente:

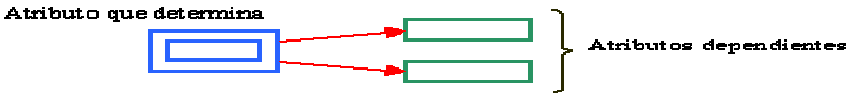


Como esta relación maneja valores atómicos, es decir un solo valor por cada uno de los campos que conforman a los atributos de las entidades, ya se encuentra en primera forma normal, gráficamente así representamos a las relaciones en 1FN.

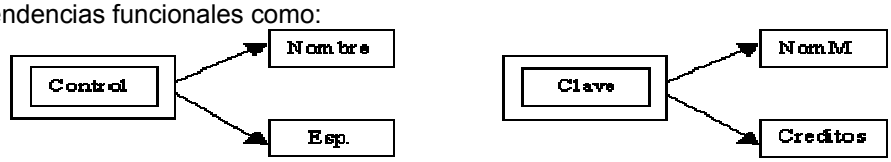
Segunda forma normal.

Tratamiento de información

Para definir formalmente la segunda forma normal requerimos saber que es una dependencia funcional: Consiste en edificar que atributos dependen de otro(s) atributo(s).



Definición formal: Una relación R está en 2FN si y solo si está en 1FN y los atributos no primos dependen funcionalmente de la llave primaria.
Una relación se encuentra en segunda forma normal, cuando cumple con las reglas de la primera forma normal y todos sus atributos que no son claves (llaves) dependen por completo de la clave . De acuerdo con está definición, cada tabla que tiene un atributo único como clave, esta en segunda forma normal. La segunda forma normal se representa por dependencias funcionales como:



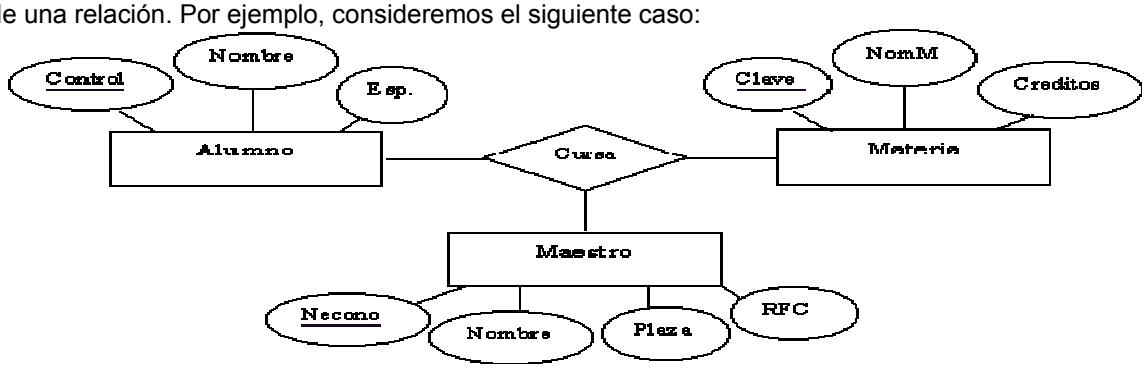
Nótese que las llaves primarias están representadas con doble cuadro, las flechas nos indican que de estos atributos se puede referenciar a los otros atributos que dependen funcionalmente de la llave primaria.

Tercera forma normal y la forma normal de Boyce Codd.

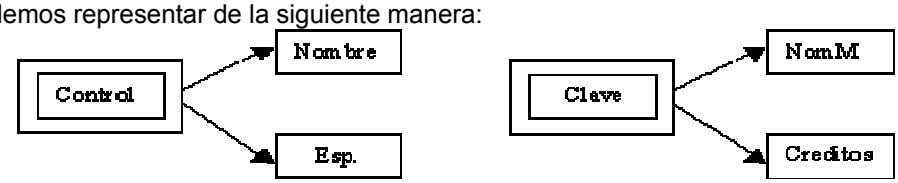
Para definir formalmente la 3FN necesitamos definir dependencia transitiva: En una afinidad (tabla bidimensional) que tiene por lo menos 3 atributos (A,B,C) en donde A determina a B, B determina a C pero no determina a A.

Tercera forma normal.

Definición formal: Una relación R está en 3FN si y solo si esta en 2FN y todos sus atributos no primos dependen no transitivamente de la llave primaria.
Consiste en eliminar la dependencia transitiva que queda en una segunda forma normal, en pocas palabras una relación esta en tercera forma normal si está en segunda forma normal y no existen dependencias transitivas entre los atributos, nos referimos a dependencias transitivas cuando existe más de una forma de llegar a referencias a un atributo de una relación. Por ejemplo, consideremos el siguiente caso:

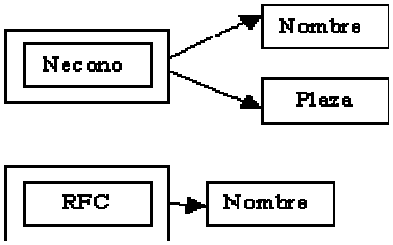


Tenemos la relación alumno-cursa-materia manejada anteriormente, pero ahora consideramos al elemento maestro, gráficamente lo podemos representar de la siguiente manera:



Podemos darnos cuenta que se encuentra graficado en segunda forma normal, es decir que todos los atributos llave están indicados en doble cuadro indicando los atributos que dependen de dichas llaves, sin embargo en la llave Necono tiene como dependientes a 3 atributos en el cual el nombre puede ser referenciado por dos atributos: Necono y RFC (Existe dependencia transitiva), podemos solucionar esto aplicando la tercera forma normal que consiste en eliminar estas dependencias separando los atributos, entonces tenemos:

Tratamiento de información



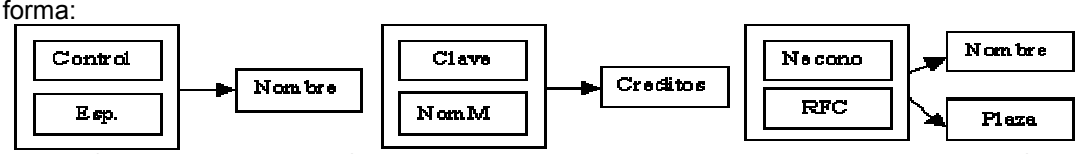
Forma normal de Boyce Codd.

Determinante: Uno o más atributos que, de manera funcional, determinan otro atributo o atributos. En la dependencia funcional (A,B)-->C, (A,B) son los determinantes.

Definición formal:

Una relación R esta en FNBC si y solo si cada determinante es una llave candidato.

Denominada por sus siglas en ingles como BCNF; Una tabla se considera en esta forma si y sólo sí cada determinante o atributo es una llave candidato. Continuando con el ejemplo anterior, si consideramos que en la entidad alumno sus atributos control y nombre nos puede hacer referencia al atributos esp., entonces decimos que dichos atributos pueden ser llaves candidato. Gráficamente podemos representar la forma normal de Boyce Codd de la siguiente forma:



Obsérvese que a diferencia de la tercera forma normal, agrupamos todas las llaves candidato para formar una global (representadas en el recuadro) las cuales hacen referencia a los atributo que no son llaves candidato.

Cuarta y quinta forma normal

Cuarta forma normal.

Definición formal: Un esquema de relaciones R está en 4FN con respecto a un conjunto D de dependencias funcionales y de valores múltiples sí, para todas las dependencias de valores múltiples en D de la forma X->->Y, donde X<=R y Y<=R, se cumple por lo menos una de estas condiciones:

* X->->Y es una dependencia de valores múltiples trivial.
* X es una superllave del esquema R.

Para entender mejor aún esto consideremos una afinidad (tabla) llamada estudiante que contiene los siguientes atributos: Clave, Especialidad, Curso tal y como se demuestra en la siguiente figura:

Clave	Especialidad	Curso
S01	Sistemas	Natación
S01	Bioquímica	Danza
S01	Sistemas	Natación
B01	Bioquímica	Guitarra
C03	Civil	Natación

Suponemos que los estudiantes pueden inscribirse en varias especialidades y en diversos cursos. El estudiante con clave S01 tiene su especialidad en sistemas y Bioquímica y toma los cursos de Natación y danza, el estudiante B01 tiene la especialidad en Bioquímica y toma el curso de Guitarra, el estudiante con clave C03 tiene la especialidad de Civil y toma el curso de natación. En esta tabla o relación no existe dependencia funcional porque los estudiantes pueden tener distintas especialidades, un valor único de clave puede poseer muchos valores de especialidades al igual que de valores de cursos. Por lo tanto existe dependencia de valores múltiples. Este tipo de dependencias produce redundancia de datos, como se puede apreciar en la tabla anterior, en donde la clave S01 tiene tres registros para mantener la serie de datos en forma independiente lo cual ocasiona que al realizarse una actualización se requiera de demasiadas operaciones para tal fin. Existe una dependencia de valores múltiples cuando una afinidad tiene por lo menos tres atributos, dos de los cuales poseen valores múltiples y sus valores dependen solo del tercer atributo, en otras palabras en la afinidad R (A,B,C) existe una dependencia de valores múltiples si A determina valores múltiples de B, A determina valores múltiples de C, y B y C son independientes entre sí. En la tabla anterior Clave determina valores múltiples de especialidad y clave determina valores múltiples de curso, pero especialidad y curso son independientes entre sí. Las dependencias de valores múltiples se definen de la siguiente manera: Clave ->->Especialidad y Clave->->Curso; Esto se lee "Clave multidetrmina a Especialidad, y clave multidetermina a Curso"

Para eliminar la redundancia de los datos, se deben eliminar las dependencias de valores múltiples. Esto se logra construyendo dos tablas, donde cada una almacena datos para solamente uno de los atributos de valores múltiples. Para nuestro ejemplo, las tablas correspondientes son:

Tabla Eespecialidad

Clave	Especialidad
S01	Sistemas
B01	Bioquímica
C03	Civil

Tabla ECurso

Clave	Curso
S01	Natación
S01	Danza
B01	Guitarra
C03	Natación

Quinta forma normal.

Definición formal: Un esquema de relaciones R está en 5FN con respecto a un conjunto D de dependencias funcionales, de valores múltiples y de producto, si para todas las dependencias de productos en D se cumple por lo menos una de estas condiciones:

* (R1, R2, R3, ... Rn) es una dependencia de producto trivial.
* Toda Ri es una superllave de R.

La quinta forma normal se refiere a dependencias que son extrañas. Tiene que ver con tablas que pueden dividirse en subtablas, pero que no pueden reconstruirse.

8. Bases de datos orientadas a objetos

- ***Conceptos de bases de datos orientadas a objetos (clases, objetos, métodos, mensajes, persistencia y herencia)***

Las aplicaciones de las bases de datos en áreas como el diseño asistido por computadora, la ingeniería de software y el procesamiento de documentos no se ajustan al conjunto de suposiciones que se hacen para aplicaciones del estilo de procesamiento de datos. El modelo de datos orientado a objetos se ha propuesto para tratar algunos de estos nuevos tipos de aplicaciones. El modelo de bases de datos orientado a objetos es una adaptación a los sistemas de bases de datos. Se basa en el concepto de encapsulamiento de datos y código que opera sobre estos en un objeto. Los objetos estructurados se agrupan en clases. El conjunto de clases esta estructurado en sub y superclases basado en una extensión del concepto ISA del modelo Entidad - Relación. Puesto que el valor de un dato en un objeto también es un objeto, es posible representar el contenido del objeto dando como resultado un objeto compuesto. El propósito de los sistemas de bases de datos es la gestión de grandes cantidades de información. Las primeras bases de datos surgieron del desarrollo de los sistemas de gestión de archivos. Estos sistemas primero evolucionaron en bases de datos de red o en bases de datos jerárquicas y, más tarde, en bases de datos relacionales.

Estructura de objetos.

El modelo orientado a objetos se basa en encapsular código y datos en una única unidad, llamada objeto. El interfaz entre un objeto y el resto del sistema se define mediante un conjunto de mensajes. Un objeto tiene asociado:

Un conjunto de variables que contienen los datos del objeto. El valor de cada variable es un objeto.

Un conjunto de mensajes a los que el objeto responde.

Un método, que es un trozo de código para implementar cada mensaje. Un método devuelve un valor como respuesta al mensaje.

El término mensaje en un contexto orientado a objetos, no implica el uso de un mensaje físico en una red de computadoras, si no que se refiere al paso de solicitudes entre objetos sin tener en cuenta detalles específicos de

implementación. La capacidad de modificar la definición de un objeto sin afectar al resto del sistema está considerada como una de las mayores ventajas del modelo de programación orientado a objetos.

Jerarquía de clases.

En una base de datos existen objetos que responden a los mismos mensajes, utilizan los mismos métodos y tienen variables del mismo nombre y tipo. Sería inútil definir cada uno de estos objetos por separado por lo tanto se agrupan los objetos similares para que formen una clase, a cada uno de estos objetos se le llama instancia de su clase. Todos los objetos de su clase comparten una definición común, aunque difieran en los valores asignados a las variables. Así que básicamente las bases de datos orientados a objetos tienen la finalidad de agrupar aquellos elementos que sean semejantes en las entidades para formar un clase, dejando por separado aquellas que no lo son en otra clase. Por ejemplo: Retomemos la relación alumno-cursa-materia agregándole la entidad maestro; donde los atributos considerados para cada uno son alumno: Nombre, Dirección, Teléfono, Especialidad, Semestre, Grupo; Maestro: Nombre, Dirección, Teléfono, Número económico, Plaza, RFC; Materia: Nombre, Créditos, Clave. Los atributos de nombre, dirección y teléfono se repiten en la entidad alumno y maestro, así que podemos agrupar estos elementos para formar la clase Persona con dichos campos. Quedando por separado en alumno: Especialidad, semestre, Grupo. Y en maestro: Número económico, Plaza y RFC; la materia no entra en la agrupación (Clase persona) ya que la clase específica los datos de solo personas, así que queda como clase materia.

Herencia.

Las clases en un sistema orientado a objetos se representan en forma jerárquica como en el diagrama anterior, así que las propiedades o características del elemento persona las contendrán (heredaran) los elementos alumno y maestro. Decimos que tanto la entidad Alumno y maestro son subclases de la clase persona este concepto es similar al utilizado en la de especialización (la relación ISA) del modelo E-R. Se pueden crear muchas agrupaciones (clases) para simplificar un modelo así que una jerarquía (en forma gráfica) puede quedar muy extensa, en estos casos tenemos que tener bien delimitados los elementos que intervienen en una clase y aquellos objetos que las heredan.

Consultas orientadas a objetos:

Los lenguajes de programación orientados a objetos requieren que toda la interacción con objetos se realiza mediante el envío de mensajes. Consideremos el ejemplo de alumno-cursa-materia deseamos realizar la consulta de los alumnos que cursan la materia de Base de Datos 1, para realizar esta consulta se tendría que enviar un mensaje a cada instancia alumno. Así un lenguaje de consultas para un sistema de bases de datos orientado a objetos debe incluir tanto el modelo de pasar el mensaje de objeto a objeto como el modelo de pasar el mensaje de conjunto en conjunto.

Complejidad de Modificación.

En base de datos orientados a objetos pueden existir los siguientes cambios:

Adición de una nueva clase: Para realizar este proceso, la nueva clase debe colocarse en la jerarquía de clase o subclase cuidando las variables o métodos de herencia correspondientes.

Eliminación de una clase: Se requiere la realización de varias operaciones, se debe de cuidar los elementos que se han heredado de esa clase a otras y reestructurar la jerarquía.

En sí la estructuración de modelos orientados a objetos simplifica una estructura evitando elementos o variables repetidas en diversas entidades, sin embargo el precio de esto es dedicarle un minucioso cuidado a las relaciones entre las clases cuando en modelo es complejo, la dificultad del manejo de objetos radica en la complejidad de las modificaciones y eliminaciones de clases, ya que de tener variables que heredan otros objetos se tiene que realizar una reestructuración que involucra una serie de pasos complejos.

9. Arquitecturas cliente-servidor

- ***Protocolos de acceso nativos y ODBC***

Arquitectura cliente/servidor, arquitectura hardware y software adecuada para el proceso distribuido, en el que la comunicación se establece de uno a varios. Un proceso es un programa en ejecución. Proceso cliente es el que solicita un servicio. Proceso servidor es el capaz de proporcionar un servicio. Un proceso cliente se puede comunicar con varios procesos servidores y un servidor se puede comunicar con varios clientes. Los procesos pueden ejecutarse en la misma máquina o en distintas máquinas comunicadas a través de una red. Por lo general, la parte de la aplicación correspondiente al cliente se optimiza para la interacción con el usuario, ejecutándose en su propia máquina, mientras que la parte correspondiente al servidor proporciona la funcionalidad multiusuario centralizada y se ejecuta en una máquina remota. Una aplicación cliente/servidor típica es un servidor de base de datos al que varios usuarios realizan consultas simultáneamente. El proceso cliente realiza una consulta, el proceso servidor le envía las tablas resultantes de la consulta y el proceso cliente las interpreta y muestra el resultado en pantalla.

10. Bases de datos distribuidas

• Una Base de Datos Distribuida es, una base de datos construida sobre una red computacional y no por el contrario en una máquina aislada. La información que constituye la base de datos esta almacenada en diferentes sitios en la red,y las aplicaciones que se ejecutan accesan datos en distintos sitios.
• Una Base de Datos Distribuida entonces es una colección de datos que pertenecen lógicamente a un sólo sistema, pero se encuentra fisicamente esparcido en varios "sitios" de la red. •Un sistema de base de datos distribuidas se compone de un conjunto de sitios, conectados entre sí mediante algún tipo de red de comunicaciones, en el cual :
1. cada sitio es un sistema de base de datos en sí mismo, pero
2. los sitios han convenido en trabajar juntos (si es necesario) con el fin de que un usuario de cualquier sitio pueda obtener acceso a los datos de cualquier punto de la red tal como si todos los datos estuvieran almacenados en el sitio propio del usuario.
En consecuencia, la llamada "base de datos distribuida" es en realidad una especie de objeto virtual, cuyas partes componentes se almacenan físicamente en varias bases de datos "reales" distintas ubicadas en diferentes sitios. De hecho, es la unión lógica de esas bases de datos. En otras palabras, cada sitio tiene sus propias bases de datos "reales" locales, sus propios usuarios locales, sus propios DBMS y programas para la administración de transacciones (incluyendo programas de bloqueo, bitácoras, recuperació, etc), y su propio administrador local de comunicación de datos (administrador DC). En particular un usuario dado puede realizar operaciones sobre los datos en su propio sitio local exactamente como si ese sitio no participara en absoluto en el sistema distribuido (al menos, ése es uno de los objetivos). Así pues, el sistema de bases de datos distribuidas puede considerarse como una especie de sociedad entre los DBMS individuales locales de todos los sitios. Un nuevo componente de software en cada sitio (en el aspecto lógico, una extensión del DBMS local) realiza las funciones de sociedad necesarias; y es la combinación de este nuevo componente y el DBMS ya existente lo que constituye el llamado "sistema de administración de bases de datos distribuidas" (DDBMS, distributed database management system).
Conceptos básicos.
• El sistema de administración de Base de Datos Distribuida (DDBMS), esta formado por las transacciones y los administradores de base de datos distribuidos de todas las computadoras. Tal DDBMS en un esquema genérico implica un conjunto de programas que operan en diversas computadoras. Estos programas pueden ser subsistemas de un producto único DDBMS, concesionado por un sólo fabricante, o también pudiera resultar de una colección de programas de fuentes dispares : algunos considerados por fabricantes y algunos otros escritos en casa.
• Un administrador de base de datos (DTM) es un programa qeu recibe solicitudes de procesamiento de los programas de consulta o de transacciones y a su vez las traduce en acciones para los administradores de la base de datos . Una función importante del DTM es coordinar y controlar dichas acciones. • Cada sitio tiene sus propias bases de datos "reales" locales, sus propios usuarios locales, sus propios DBMS y programas para administración de transacciones y su propio administrador local de comunicación de datos.
La diferencia principal entre los sistemas de bases de datos centralizados y los distribuidos es que en los primeros, los datos residen en una sola localidad, mientras que, en lo últimos, se encuentran en varias localidades. Cada localidad puede procesar transaccion locales , es decir, aquellas que sólo acceden a datos que residen en esa localidad. Además, una localidad puede participar en la ejecución de transacciones globales , es decir, aquellas que acceden a datos de varias localidades, ésta requiere comunicación entre las localidades. • Una transacción local es la que accede a cuentas en la localidad individual donde se inicio. En cambio, una transacción global accede a cuentas de una localidad distinta a la localidad donde se inicio o a cuentas de varias localidades diferentes.
¡ IMPORTANTE !
Se asume que cada localidad ejecuta el mismo software de control de base de datos distribuida. Si no se cumple esta premisa, será muy difícil manejar transacciones globales.
¿Por qué son deseables las bases de datos distribuidas?
La respuesta básica a esta pregunta es que por lo regular las empresas ya están distribuidas, por lo menos desde el punto de vista lógico (en divisiones, departamentos, proyectos, etc) y muy probablemente en el sentido físico también (en plantas, talleres, laboratorios, y demás), de lo cual se desprende que en general la información ya está también distribuida, porque cada unidad de organización dentreo de la empresa mantendrá por fuerza los datos pertinentes a su propio funcionamiento. Así pues, un sistema distribuido permite que la estructura de la base de datos refleje la estructura de la empresa : los datos locales se pueden mantener en forma local, donde por lógica deben estar, pero al mismo tiempo es posible obtener acceso a datos remotos en caso necesario. Un principio fundamental de los sistemas de bases de datos distribuidos
El principio fundamental de las bases de datos distribuidas :
• Desde el punto de vista del usuario, un sistema distribuido deberá ser idéntico a un sistema no distribuido.

En otras palabras, los usuarios de un sistema distribuido deberán comportarse exactamente como si el sistema no estuviera distribuido. Todos los problemas de los sistemas distribuidos son (o deberían ser) internos o a nivel de realización, no externos o a nivel del usuario. Llamaremos al principio fundamental recién identificado la "regla cero" de los sistemas distribuidos. La regla cero conduce a varios objetivos o reglas secundarios - doce en realidad- siguientes :

1. Autonomía local.
2. No dependencia de un sitio central.
3. Operación continua.
4. Independencia con respecto a la localización.
5. Independencia con respecto a la fragmentación.
6. Independencia de réplica.
7. Procesamiento distribuido de consultas.
8. Manejo distribuido de transacciones.
9. Independencia con respecto al equipo.
10. Independencia con respecto al sistema operativo.
11. Independencia con respecto a la red.
12. Independencia con respecto al DBMS.

Estas doce reglas no son todas independientes entre sí, ni son por fuerza exhaustivas, ni tienen todas la misma importancia (diferentes usuarios darán diferentes grados de importancia a diferentes reglas en diferentes ambientes). Sin embargo, sí son útiles como fundamento para entender la tecnología distribuida y como marco de referencia para caracterizar la funcionalidad de sistemas distribuidos específicos. Un último punto introductorio: es importante distinguir los sistemas distribuidos de bases de datos verdaderos, generalizados, de los sistemas que tan solo ofrecen algún tipo de acceso remoto a los datos (llamados a veces sistemas de procesamiento distribuido o sistemas de red). En un " sistema de acceso remoto a los datos ", el usuario podría ser capaz de trabjar con datos de un sitio remoto, o aun con datos de varios sitios remotos al mismo tiempo, pero " se notan las costuras" ; el usuario definitivamente está consciente (en mayor o menor grado) de que los datos son remotos, y debe comportarse de manera acorde. En cambio, en un sistema distribuido verdadero, las costuras son invisibles. Las doce reglas.

1. Autonomía Local. Los sitios de un sistema distribuido deben ser autónomos . La autonomía local significa que todas las operaciones en un sitio dado se controlan en ese sitio; ningún sitio X deberá depender de algún otro sitio Y para su buen funcionamiento (pues de otra manera el sitio X podría ser incapaz de trabajar, aunque no tenga en sí problema alguno, si cae el sitio Y, situación a todas luces indeseable). La autonomía local implica también un propietario y una administración locales de los datos, con responsabilidad local : todos los datos pertenecen " en realidad" a una base de datos local, aunque sean accesibles desde algún sitio remoto. Por tanto, las cuestiones de seguridad, integridad y representación en almacenamiento de los datos locales permanecen bajo el control de la instalación local.

2. No dependencia de un sitio central. La autonomía local implica que todos los sitios deben tratarse igual; no debe haber dependencia de un sitio central "maestro" para obtener un servicio central, como por ejemplo un procesamiento centralizado de las consultas o una administración centralizada de las transacciones, de modo que todo el sistema dependa de ese sitio central. Este segundo objetivo es por tanto un corolario del primero (si se logra el primero, se logrará pro fuerza el segundo) . Pero la "no dependencia de un sitio central" es deseable por sí misma, aun si no se logra la autonomía local completa. Por ello vale la pena expresarlo como un objetivo separado. La dependencia de un sitio central sería indeseable al menos por las siguientes razones : en primer lugar, eses sitio central podrí ser un cuello debotella; en segundo lugar, el sistema sería vulnerable ; si el sitio central sufriera un desperfecto, todo el sistema dejaría de funcionar.

3. Operación continua. En un sistema distribuido, lo mismo que en uno no distribuido, idealmente nunca debería haber necesidad de apagar a propósito el sistema Es decir, el sistema nunca debería necesitar apagarse para que se pueda realizar alguna función, como añadirse un nuevo sitio o instalar una versión mejorada del DBMS en un sitio ya existente.

4. Independencia con respecto a la localización. La idea básica de la independencia con respecto a la localización (también conocida como transparencia de localización) es simple : no debe ser necesario que los usuarios sepan dónde están almacenados físicamente los datos, sino que más bien deben poder comportarse - al menos desde un punto de vista lógico - como si todos los datos estuvieran almacenados en su propio sitio local. La independencia con respecto a la localización es deseable porque simplifica los programas de los usuarios y sus actividades en la terminal. En particular, hace posible la migración de datos de un sitio a otro sin anular la validez de ninguno de esos programas o actividades. Esta posibilidad de migración es deseable pues permite modificar la distribución de los datos dentro de la red en respuesta a cambios en los requerimientos de desempeño.

5. Independencia con respecto a la fragmentación. Un sistema maneja fragmentación de los datos si es posible dividir una relación en partes o "fragmentos" para propósitos de almacenamiento físico. La fragmentación es deseable por razones de desempeño: los datos pueden almacenarse en la localidad donde se utilizan con mayor frecuencia, de manera que la mayor parte de las operaciones sean sólo locales y se reduzca al tráfico en la red. Por ejemplo, la relación empleados EMP podría fragmentarse de manera que los registros de los empleados de Nueva York se almacenen en el sitio de Nueva York, en tanto que los registros de los empleados de Londres se almacenan en el sitio de Londres. Existen en esencia dos clases de fragmentación, horizontal y vertical, correspondientes a las operaciones relacionales de restricción y proyección; respectivamente. En términos más generales, un fragmento puede ser cualquier subrelación arbitraria que pueda derivarse de la relación original mediante operaciones de restricción y proyección (excepto que, en el caso de la proyección es obvio que las proyecciones deben conservar la clave primaria de la relación original). La reconstrucción de la relación origianl a partir de los fragmentos se hace mediante operciones de reunión y unión apropiadas (reunión en el caso de fragmentación vertical,y la unión en casos de fragmentación horizontal). Ahora llegamos a un punto principal : un sistema que maneja la fragmentación de los datos deberá ofrecer también una independencia con respecto a la fragmentación (llamada también transparencia de fragmentación). La independencia con respecto a la fragmentación (al igual que la independencia con respecto a la independencia con respecto a la localización) es deseable porque simplifica los programas de los usuarios y sus actividades en la terminal.
6. Independencia de réplica. Un sistema maneja réplica de datos si una relación dada (ó en términos más generales, un fragmento dado en una relación) se puede representar en el nivel físicio mediante varias copias réplicas, en muchos sitios distintos. La réplica es deseable al menos por dos razones : en primer lugar, puede producir un mejor desempeño (las aplicaciones pueden operar sobre copias locales en vez de tener que comunicarse con sitios remotos) ; en segundo lugar, también puede significar una mejor disponibilidad (un objeto estará disponible para su procesamiento en tanto esté disponible por lo menos una copia, al menos para propósitos de recuperación). La desventaja principal de las réplicas es desde luego que cuando se pone al día un cierto objeto copiado, deben ponerse al día todas las réplicas de ese objeto : el problema de la propagación de actualizaciones. La réplica como la fragmentación, debe ser "transparente para el usuario". En otras palabras , un sistema que maneja la réplica de los datos deberá ofrecer también una independencia de réplica (conocida también como transparencia de réplica); es decir, los usuarios deberán poder comportarse como si sólo existiera una copia de los datos. La independencia de réplica es buena porque simplifica los programas de los usuarios y sus actividades en la terminal. En particular, permite la creación y eliminación dinámicas de las réplicas en cualquier momento en respuesta a cambios en los requerimientos, sin anular la validez de esos programas o actividades de los usuarios.
7. Procesamiento distribuido de consultas. En este aspecto debemos mencionar dos puntos amplios. Primero consideremos la consulta "obtener los proveedores de partes rojas en Londres". Supongamos que el usuario está en la instalación de Nueva York y los datos están en el sitio de Londres. Supongamos también que son n/ n registros de Londres a Nueva York. Si, por otro lado, el sistema no es relacional, sino de un registro a la vez, la consulta implicará en esencia 2n mensajes : n de Nueva York a Londres solicitando el siguiente registro, y n de Londres a Nueva York para devolver eses siguiente registro. Así, el ejemplo ilustra el punto de que un sistema relacional tendrá con toda probabilidad un mejor desempeño que uno no relacional (para cualquier consulta que solicite varios registros), quizá en varios órdenes de magnitud. En segundo lugar, la optimización es todavía más importante en un sistema distribuido que en uno centralizado. Lo esencial es que, en una consulta como la anterior, donde están implicados varios sitios, habrá muchas maneras de trasladar los datos en al red para satisfacer la solicitud, y es crucial encontrar una estrategia suficiente. Por ejemplo, una solicitud de unión de una relación Rx almacenada en el sitio X y una relación Ry almacenada en el sitio Y podría llevarse a cabo trasladando Rx a Y o trasladando Ry a X, o trasladando las dos a un tercer sitio Z .
8. Manejo distribuido de transacciones. El manejo de transacciones tiene dos aspectos principales, el control de recuperación y el control de concurrencia, cada uno de los cuales requiere un tratamiento más amplio en el ambiente distribuido. Para explicar eses tratamiento más amplio es preciso introducir primero un término nuevo, "agente". En un sistema distribuido, una sola transacción puede implicar la ejecución de código en varios sitios (en particular puede implicar a actualizaciones en varios sitios). Por tanto, se dice que cada transacción está compuesta de varios agentes, donde un agente es el proceso ejecutado en nombre de una transacción dada en determinado sitio. Y el sistema necesita saber cuándo dos agentes son parte de la misma transacción; por ejemplo, es obvio que no puede permitirse un bloqueo mutuo entre dos agentes que sean parte de la misma transacción. La cuestión especifica del control de recuperación; : para asegurar, pues que una transacción dada sea atómica (todo o nada) en el ambiente distribuido, el sistema debe segurarse de que todos los agentes correspondientes a esa transacción se comprometan al unísono o bien que retrocedan al unísono. Este efecto puede lograrse mediante el protocolo de compromiso en dos

fases. En cuanto al control de concurrencia, esta función en un ambiente distribuido estará basada con toda seguridad en el bloqueo, como sucede en los sistemas no distribuidos.
9. Independencia con respecto al equipo. En realidad, no hay mucho que decir acerca de este tema, el título lo dice todo. Las instalaciones de cómputo en el mundo real por lo regular incluyen varias máquinas diferentes -máquinas IBM, DEC, HP, UNISYS, PC etc- y existe una verdadera necesidad de poder integrar los datos en todos esos sistemas y presentar al usuario "una sola imagen del sistema". Por tanto conviene ejecutar el mismo DBMS en diferentes equipos, y ademáS lograr que esos diferentes equipos participen como socios iguales en un sistema distribuido.
10. Independencia con respecto al sistema operativo. Este objetivo es un corolario del anterior. Es obvia la conveniencia no sólo de poder ejecutar el mismo DBMS en diferentes equipos, sino también poder ejecutarlo en diferentes sistemas operativos y lograr que una versión MVS y una UNIX y una PC/DOS participen todas en el mismo sistema distribuido.
11. Independencia con respecto a la red. Si el sistema ha de poder manejar múltiples sitios diferentes, con equipo distinto y diferentes sistemas operativos, resulta obvia la conveniencia de pdoer manejar también varias redes de comunicación distintas.
12. Independencia con respecto al DBMS. Bajo este título consideramos las implicaciones de relajar la suposición de homogeneidad estrictia. Puede alegarse que esa suposición es quizá demasiado rígida. En realidad, no se requiere sino que los DBMS en los diferentes sitios manejen todos la misma interfaz ; no necesitan ser por fuerza copias del mismo sistema.
Ventajas
Existen cuatro ventajas del procesamiento de bases de datos distribuidas. La primera, puede dar como resultado un mejor rendimiento que el que se obtiene por un procesamiento centralizado. Los datos pueden colocarse cerca del punto de su utilización, de forma que el tiempo de comunicación sea más corto. Varias computadoras operando en forma simultánea pueden entregar más volumen de procesamiento que una sola computadora. Segundo, los datos duplicados aumentan su confiabilidad. Cuando falla una computadora, se pueden obtener los datos extraídos de otras computadoras. Los usuarios no dependen de la disponibilidad de una sola fuente para sus datos .Una tercera ventaja , es que los sistemas distribuidos pueden variar su tamaño de un modo más sencillo. Se pueden agregar computadoras adicionales a la red conforme aumentan el número de usuarios y su carga de procesamiento. A menudo es más fácil y más barato agregar una nueva computadora más pequeña que actualizar una computadora única y centralizada. Después, si la carga de trabajo se reduce, el tamaño de la red también puede reducirse.
Por último , los sistemas distribuidos se puede adecuar de una manera más sencilla a las estructuras de la organización de los usuarios.
Nota:
Los siguientes puntos están basados en una referencia bibliográfica distinta, y me pareció importante hablar sobre ello, espero y aclare el tema en cuestión. Utilización compartida de los datos y distribución del control. Si varias localidades diferentes están conectadas entre si, entonces un usuario de una localidad puede acceder a datos disponibles en otra localidad. La ventaja principal de compartir datos por medio de la distribución es que cada localidad pueda controlar hasta cierto punto los datos almacenados localmente.
Fiabilidad y disponibilidad: Si se produce un fallo en una localidad en un sistema distribuido, es posible que las demás localidades puedan seguir trabajando. En particular si los datos se repiten en varias localidades, una transacción o aplicación que requiere un dato especifico puede encontrarlo en más de una localidad. Así el fallo, de una localidad no implica necesariamente la desactivación del sistema.
Agilización del procesamiento de consultas: Si una consulta comprende datos de varias localidades, puede ser posible dividir la consulta en varias subconsultas que se ejecuten en paralelo en distintas localidades. En los casos en que hay repetición de los datos, el sistema puede pasar la consulta a las localidades mas ligeras de carga.
Ejemplo de sistemas: Para efectos de referencia posterior, mencionaremos brevemente algunas de las realizaciones de sistemas distribuidos más conocidas. En primer término, los prototipos. Entre los sistemas investigados, tres de los más conocidos son:
a) SDD-1 creado en la división de investigación de Computer Corporation of America (CCA) a finales de la década de los 1970 y principios de la siguiente.
b) R* (pronunciado "R estrella"), versión distribuida del prototipo System R elaborada en IBM Research a principios de la década de 1980 y ;
c) INGRES distribuido, versión distribuida del prototipo INGRES, creada también a principios de la década de 1980 en la University of California en Berkeley.
Pasando a productos comerciales, algunos de los más conocidos son :
a) INGRES/STAR de Relational Technology, Inc.

b) SQL*STAR, de Oracle Corp. y
c) DB2 versión 2 Edición 2, de IBM.
Desventajas
Procesamiento de consultas
Administración de catálogo
Propagación de Actualizaciones
Control de recuperación
Control de Concurrencia
Las primeras dos desventajas de las bases de datos distribuidas son las mismas que las dos primeras ventajas. Primero, el rendimiento puede ser peor para el procesamiento distribuido que para el procesamiento centralizado. Depende de la naturaleza de la carga de trabajo, la red, el DDBMS y las estrategias utilizadas de concurrencia y de falla, así como las ventajas del acceso local a los datos y de los procesadores múltiples, ya que éstos pueden ser abrumados por las tareas de coordinación y de control requeridas. Tal situación es probable cuando la carga de trabajo necesita un gran número de actualizaciones concurrentes sobre datos duplicados, y que deben estar muy distribuidos. Segundo, el procesamiento de base de datos distribuida puede resultar menos confiable que el procesamiento centralizado. De nuevo, depende de la confiabilidad de las computadoras de procesamiento, de la red, del DDBMS, de las transacciones y de las tasas de error en la carga de trabajo. Un sistema distribuido puede estar menos disponible que uno centralizado. Estas dos desventajas indican que un procesamiento distribuido no es ninguna panacea. A pesar de que tiene la promesa de un mejor rendimiento y de una mayor confiabilidad, tal promesa no está garantizada. Una tercera desventaja es su mayor complejidad, a menudo se traduce en altos gastos de construcción y mantenimiento. Ya que existen más componentes de hardware, hay más cantidad de cosas por aprender y más interfaces susceptibles de fallar. El control de concurrencia y recuperación de fallas puede convertirse en algo complicado y difícil de implementar, puede empujar a una mayor carga sobre programadores y personal de operaciones y quizá se requiera de personal más experimentado y más costoso. El procesamiento de bases de datos distribuido es difícil de controlar. Una computadora centralizada reside en un entorno controlado, con personal de operaciones que supervisa muy de cerca, y las actividades de procesamiento pueden ser vigiladas, aunque a veces con dificultad. En un sistema distribuido, las computadoras de proceso, residen muchas veces en las áreas de trabajo de los usuarios. En ocasiones el acceso físico no está controlado, y los procedimientos operativos son demasiado suaves y efectuados por personas que tienen escasa apreciación o comprensión sobre su importancia. En sistemas centralizados, en caso de un desastre o catástrofe, la recuperación puede ser más difícil de sincronizar.
Nota: De las desventajas que se mencionan a continuación, están relacionadas con las doce reglas mencionadas en el capítulo anterior.
Procesamiento de Consultas
El problema más grande es que las redes de comunicación (las de larga distancia en especial) son lentas. El objetivo es reducir al mínimo el tráfico en la red y esto implica que el proceso mismo de optimización de consultas debe ser distribuido, además del proceso de ejecución de las consultas. Es decir un proceso representativo consistirá en un paso de optimización global, seguido de pasos de optimización local en cada unos de los sitios afectados.
Administración de Catálogo
En un sistema distribuido, el catálogo del sistema incluirá no solo la información usual acerca de las relaciones, índices, usuarios, sino también toda la información de control necesaria para que el sistema pueda ofrecer la independencia deseada con respecto a la localización, la fragmentación y la replicación.
1. Centralizado (" no depender de un sitio central")
2. Replicas completas (" falta de autonomía, toda la actualización debe ser propagada a cada sitio ")
3. Dividido (muy costoso)
4. Combinación de 1 y 3 (" no depender de un sitio central ")
Propagación de Actualizaciones
El problema básico con la réplica de datos, es la necesidad de propagar cualquier modificación de un objeto lógico dado a todas las copias almacenadas de ese objeto. Un problema que surge es que algún sitio donde se mantiene una copia del objeto puede NO estar disponible, y fracasaría; la modificación si cualquiera de las copias no esta disponible. Para tratar este problema se habla de " una copia primaria " y funciona así :
• Una de las copias del objeto se designa como copia primaria, y las otras serán secundarias.
• Las copias primarias de los distintos objetos están en sitios diferentes.
• Las operaciones de actualización se consideran completas después de que se ha modificado la copia primaria.
El sitio donde se encuentra esa copia se encarga entonces de propagar la actualización a las copias secundarias.
Recuperación

Basado en el protocolo de compromiso de dos fases. El compromiso de dos fases es obligatorio en cualquier ambiente en el cual una sola transacción puede interactuar con varios manejadores de recursos autónomos, pero tiene especial importancia en un sistema distribuido porque los manejadores de recursos en cuestión (o sea los DBMS locales) operan en sitios distintos y por tanto son muy autónomos. En particular, son vulnerables a fallas independientes. Surgen los siguientes puntos :
1. El objetivo de "no dependencia de un sitio central" dicta que la función de coordinador no debe asignarse a un sitio específico de la red, sino que deben realizarla diferentes sitios para diferentes transacciones. Por lo regular se encarga de ella el sitio en el cual se inicia la transacción en cuestión.
2. El proceso de compromiso en dos fases requiere una comunicación entre el coordinador y todos los sitios participantes, lo cual implica más mensajes y mayor costo extra.
3. Si el sitio Y actúa como participante en un proceso de compromiso en dos fases coordinado por el sitio X, el sitio Y deberá hacer lo ordenado pro el sitio X (compromiso o retroceso, según se aplique), lo cual implica otra pérdida de autonomía local.
4. En condiciones ideales, nos gustaría que el proceso de compromiso en dos fases funcionara aun en caso de presentarse fallas de sitios o de la red en cualquier punto. Idealmente, el proceso debería ser capaz de soportar cualquier tipo concebible de falla. Por desgracia es fácil ver que este problema es en esencia imposible de resolver; es decir, no existe un protocolo finito que garantice el compromiso al unísono de una transacción exitosa por parte de todos los agentes, o el retroceso al unísono de una transacción no exitosa en caso de fallas arbitrarias.
Concurrencia
Este concepto tiene que ver con la definición de un agente. El manejo de transacciones tiene dos aspectos principales, el control de recuperación y el control de concurrencia. En un sistema distribuido, una sola transacción puede implicar la ejecución de código en varios sitios (puede implicar actualizaciones en varios sitios), entonces se dice que una transacción esta compuesta por varios agentes, donde un agente es el proceso ejecutado en nombre de una transacción dada en determinado sitio. Y el sistema necesita saber cuando dos agentes son parte de la misma transacción.

- ***Fragmentación***

Existen tres tipos de fragmentación:
- Horizontal
- Vertical
- Mixta

Las reglas para validar la fragmentación son:
- Completitud: la descomposición de una relación R en un conjunto de fragmentos R_1, R_2, R_3,....R_N es completo si y solo si cada elemento de información en R se encuentra en algún R_i.
- Reconstrucción: si la relación R se descompone en un conjunto de fragmentos R_1, R_2, R_3,....R_N, entonces debe existir algún operador relacional ∇ tal que: $R = \nabla_{1 <= i <= n} R_i$
- No-Join: si la relación R se descompone en un conjunto de fragmentos R_1, R_2, R_3,....R_N y un elemento de dato d_i está en R_j, entonces d_i no debe estar en otro fragmento R_k ($k \neq j$)

- ***Replicación***

Localización de fragmentos
No replicada: cada fragmento está en un solo site
Replicada: que existe una copia del fragmento en algún site. Hay dos tipos:
Parcialmente: sólo para algunos fragmentos existe una copia.
Totalmente: de todos los fragmentos existe al menos una copia.

II. Lenguajes de bases de datos

1. Tipos de lenguajes de consulta

Un lenguaje de consulta es un lenguaje en el que el usuario solicita información de la base de datos, Los lenguajes de consulta se pueden clasificar en procedurales y no procedurales. Los lenguajes procedurales son aquellos en los cuales el usuario instruye al sistema para que lleve a cabo una serie de operaciones en la base de datos con el fin de calcular el resultado deseado. En los lenguajes no procedurales, en cambio, el usuario describe la información deseada sin dar un procedimiento concreto para obtener esta información. Los lenguajes puros para la consulta de datos son 3. El álgebra relacional, el cual es procedural y los cálculos relacionales tanto de tuplas como de dominios,

los cuales son lenguajes no procedurales. Estos 3 lenguajes son rígidos y formales, por lo tanto la mayor parte de los sistemas comerciales de bases de datos relacionales ofrecen lenguajes de consulta mixtos, que además de ser ricos en sintaxis ofrecen adicionalmente sublenguajes de definición de datos, administración de seguridad y otras características. Se estudiarán brevemente dos de los tres lenguajes puros de consultas de base de datos, con el fin de presentar una base teórica para introducir SQL. Se excluirá el cálculo relacional de dominios dado que es una generalización del cálculo relacional de tuplas y queda fuera del alcance de este trabajo.

- ***SQL ANSI 92: conceptos básicos, definición de daros, consultas y actualización, manejo de vistas, SQL embebido***

SQL como lenguaje de consultas relacionales.
Introducción. Los lenguajes formales presentados en las secciones 2.2 y 2.3 proporcionan una notación concisa para la representación de consultas. Sin embargo, los sistemas de base de datos necesitan un lenguaje de consultas más cómodo para el usuario. Aunque SQL se considere un lenguaje de consultas, contiene muchas otras capacidades que incluyen características para definir estructuras de datos, modificación de datos y la especificación de restricciones de integridad. SQL se ha establecido como el lenguaje estándar de base de datos relacionales. Hay numerosas versiones de SQL. La versión original se desarrollo en el laboratorio de investigación de San Jose, California (San Jose Research Center) de IBM, este lenguaje originalmente denominado Sequel, se implementó como parte del proyecto System R, a principios de 1970 [McJones97]. Desde entonces ha evolucionado a lo que ahora se conoce como SQL (Structured Query Language, o lenguaje estructurado de consultas). En 1986, ANSI (American National Standards Institute, Instituto Nacional Americano de Normalización) e ISO (International Standards Organization, Organización Internacional de Normalización) Publicaron una norma de SQL denominada SQL-86. En 1987 IBM publicó su propia norma de SQL denominada SAA-SQL(System Application Architecture Database Interfaz, Interfaz de base de datos para arquitecturas de aplicación de sistemas). En 1989 se publicó una norma extendida para SQL (SQL-89) y actualmente los SGBD son compatibles al menos con esta norma. La norma actual de SQL de ANSI/ISO es la SQL-92. Se debe tener en cuenta que algunas implementaciones de SQL pueden ser compatibles sólo con SQL-89, no siéndolo con SQL-92. SQL proporciona dos tipos de lenguajes diferentes: uno para especificar el esquema relacional y el otro para expresar las consultas y actualizaciones de la base de datos.

- ***Lenguajes de programación 3GL y 4GL***

Lenguajes de tercera generación. 3GL
Third Generation Language. Los lenguajes de tercera generación son aquellos lenguajes de programación utilizados por los especialistas para construir aplicaciones que incluyen el procedimiento. Es decir, el programador especifica en su programa qué tiene que hacer el ordenador y cómo debe hacerlo. Se trata de un paso más allá del lenguaje máquina. Son lenguajes de tercera generación Cobol, C, Pascal o Fortran.
Lenguajes de cuarta generación. 4GL
No existe consenso sobre lo que es un lenguaje de cuarta generación (4GL). Lo que en un lenguaje de tercera generación (3GL) como COBOL requiere cientos de líneas de código, tan solo necesita diez o veinte líneas en un 4GL. Comparado con un 3GL, que es procedural, un 4GL es un lenguaje no procedural: el usuario define qué se debe hacer, no cómo debe hacerse. Los 4GL se apoyan en unas herramientas de mucho más alto nivel denominadas herramientas de cuarta generación. El usuario no debe definir los pasos a seguir en un programa para realizar una determinada tarea, tan sólo debe definir una serie de parámetros que estas herramientas utilizarán para generar un programa de aplicación. Se dice que los 4GL pueden mejorar la productividad de los programadores en un factor de 10, aunque se limita el tipo de problemas que pueden resolver. Los 4GL abarcan:

Lenguajes de presentación, como lenguajes de consultas y generadores de informes.
Lenguajes especializados, como hojas de cálculo y lenguajes de bases de datos.
Generadores de aplicaciones que definen, insertan, actualizan y obtienen datos de la base de datos.
Lenguajes de muy alto nivel que se utilizan para generar el código de la aplicación.
Los lenguajes SQL y QBE son ejemplos de 4GL. Hay otros tipos de 4GL:

Un generador de formularios es una herramienta interactiva que permite crear rápidamente formularios de pantalla para introducir o visualizar datos. Los generadores de formularios permiten que el usuario defina el aspecto de la pantalla, qué información se debe visualizar y en qué lugar de la pantalla debe visualizarse. Algunos generadores de formularios permiten la creación de atributos derivados utilizando operadores aritméticos y también permiten especificar controles para la validación de los datos de entrada. Un generador de informes es una herramienta para crear informes a partir de los datos almacenados en la base de datos. Se parece a un lenguaje de consultas en que permite al usuario hacer preguntas sobre la base de datos y obtener información de ella para un informe. Sin embargo, en el generador de informes se tiene un mayor control sobre el aspecto de la salida. Se puede dejar que el

generador determine automáticamente el aspecto de la salida o se puede diseñar ésta para que tenga el aspecto que desee el usuario final. Un generador de gráficos es una herramienta para obtener datos de la base de datos y visualizarlos en un gráfico mostrando tendencias y relaciones entre datos. Normalmente se pueden diseñar distintos tipos de gráficos: barras, líneas, etc. Un generador de aplicaciones es una herramienta para crear programas que hagan de interfase entre el usuario y la base de datos. El uso de un generador de aplicaciones puede reducir el tiempo que se necesita para diseñar un programa de aplicación. Los generadores de aplicaciones constan de procedimientos que realizan las funciones fundamentales que se utilizan en la mayoría de los programas. Estos procedimientos están escritos en un lenguaje de programación de alto nivel y forman una librería de funciones entre las que escoger. El usuario especifica qué debe hacer el programa y el generador de aplicaciones es quien determina cómo realizar la tarea.

III. Seguridad en las bases de datos

1. Seguridad en bases de datos

Introducción.

Seguridad: Es la protección de los datos contra una alteración, destrucción o revelación no autorizada de los mismos.
Integridad: Consiste en la exactitud o validez de los datos.
La seguridad implica la protección de los datos contra usuarios no autorizados y el controlar el que los usuarios se ajustan a realizar el trabajo que les está permitido.
La integridad comprueba que los usuarios autorizados hagan las cosas correctamente.
Similitudes de integridad y seguridad: En ambos casos el sistema debe conocer las restricciones que no se pueden violar. Las restricciones las va a establecer el administrador de la base de datos, además el sistema de gestión de bases de datos va a ser el encargado de que la interacción con los usuarios cumple las restricciones.

- ***Seguridad***

Seguridad: Consideraciones generales.

Aspectos a tratar por la seguridad:

1.- Legales, sociales y éticos.
2.- Controles físicos.
3.- Cuestiones de política interna.
4.- Problemas de operación: Para indicar a un sistema lo que un usuario puede hacer se realiza mediante contraseñas.
5.- Controles del equipo: La CPU tiene que tener claves de protección de las áreas de almacenamiento.
6.- La seguridad del S.O.
7.- Materias de relevancia específica para el sistema mismo de la base de datos.
8.- Las decisiones de lo que los usuarios pueden hacer o ver lo que van a tomar el administrador de la base de datos y la dirección de la empresa: El sistema de gestión de la base de datos debe velar para que se cumpla esto y para ello ha de contener las siguientes características:
   - Los resultados de las decisiones deben de darse a conocer al sistema y el sistema debe ser capaz de recordarlas.
   - Debe existir alguna forma de verificar una solicitud de acceso, contra las restricciones de seguridad aplicable. Para saber que restricciones se le aplican a una solicitud el sistema debe ser capaz de reconocer esa solicitud, de ahí que no solo baste con tener el identificador del usuario, sino que sea necesario tener también una contraseña.

- ***Integridad***

Reglas de integridad

Una vez definida la estructura de datos del modelo relacional, pasamos a estudiar las reglas de integridad que los datos almacenados en dicha estructura deben cumplir para garantizar que son correctos. Al definir cada atributo sobre un dominio se impone una restricción sobre el conjunto de valores permitidos para cada atributo. A este tipo de restricciones se les denomina restricciones de dominios. Hay además dos reglas de integridad muy importantes que son restricciones que se deben cumplir en todas las bases de datos relacionales y en todos sus estados o instancias (las reglas se deben cumplir todo el tiempo). Estas reglas son la regla de integridad de entidades y la regla de integridad referencial. Antes de definirlas, es preciso conocer el concepto de nulo.

Nulos

Cuando en una tupla un atributo es desconocido, se dice que es nulo. Un nulo no representa el valor cero ni la cadena vacía, éstos son valores que tienen significado. El nulo implica ausencia de información, bien porque al insertar la tupla se desconocía el valor del atributo, o bien porque para dicha tupla el atributo no tiene sentido. Ya que los nulos no son valores, deben tratarse de modo diferente, lo que causa problemas de implementación. De hecho, no todos los SGBD relacionales soportan los nulos.

Regla de integridad de entidades

La primera regla de integridad se aplica a las claves primarias de las relaciones base: ninguno de los atributos que componen la clave primaria puede ser nulo. Por definición, una clave primaria es un identificador irreducible que se utiliza para identificar de modo único las tuplas. Que es irreducible significa que ningún subconjunto de la clave primaria sirve para identificar las tuplas de modo único. Si se permite que parte de la clave primaria sea nula, se está diciendo que no todos sus atributos son necesarios para distinguir las tuplas, con lo que se contradice la irreducibilidad. Nótese que esta regla sólo se aplica a las relaciones base y a las claves primarias, no a las claves alternativas.

Regla de integridad referencial

La segunda regla de integridad se aplica a las claves ajenas: si en una relación hay alguna clave ajena, sus valores deben coincidir con valores de la clave primaria a la que hace referencia, o bien, deben ser completamente nulos. La regla de integridad referencial se enmarca en términos de estados de la base de datos: indica lo que es un estado ilegal, pero no dice cómo puede evitarse. La cuestión es ¿qué hacer si estando en un estado legal, llega una petición para realizar una operación que conduce a un estado ilegal? Existen dos opciones: rechazar la operación, o bien aceptar la operación y realizar operaciones adicionales compensatorias que conduzcan a un estado legal. Por lo tanto, para cada clave ajena de la base de datos habrá que contestar a tres preguntas:

Regla de los nulos: ¿Tiene sentido que la clave ajena acepte nulos?

Regla de borrado: ¿Qué ocurre si se intenta borrar la tupla referenciada por la clave ajena?

Restringir: no se permite borrar la tupla referenciada.

Propagar: se borra la tupla referenciada y se propaga el borrado a las tuplas que la referencian mediante la clave ajena.

Anular: se borra la tupla referenciada y las tuplas que la referenciaban ponen a nulo la clave ajena (sólo si acepta nulos).

Regla de modificación: ¿Qué ocurre si se intenta modificar el valor de la clave primaria de la tupla referenciada por la clave ajena?

Restringir: no se permite modificar el valor de la clave primaria de la tupla referenciada.

Propagar: se modifica el valor de la clave primaria de la tupla referenciada y se propaga la modificación a las tuplas que la referencian mediante la clave ajena.

Anular: se modifica la tupla referenciada y las tuplas que la referenciaban ponen a nulo la clave ajena (sólo si acepta nulos).

Reglas de negocio

Además de las dos reglas de integridad anteriores, los usuarios o los administradores de la base de datos pueden imponer ciertas restricciones específicas sobre los datos, denominadas reglas de negocio. Por ejemplo, si en una oficina de la empresa inmobiliaria sólo puede haber hasta veinte empleados, el SGBD debe dar la posibilidad al usuario de definir una regla al respecto y debe hacerla respetar. En este caso, no debería permitir dar de alta un empleado en una oficina que ya tiene los veinte permitidos. Hoy en día aún existen SGBD relacionales que no permiten definir este tipo de restricciones ni las hacen respetar.

Integridad: Consideraciones generales.

La integridad la hay que programar a mano por medio del código del programa. La forma de programar una integridad es de forma declarativa:

Ejemplo: Forall SX (SX > 0).

Otra forma de programar la integridad es por medio de técnicas de compensación.

Consideraciones finales.

Seguridad: Consiste en proteger la base de datos contra usuarios no autorizados.

Integridad: Consiste en proteger la base de datos de los usuarios autorizados.

Tanto seguridad o integridad implican las siguientes características:

1.- Definición de restricciones.

2.- Especificar las medidas a tomar si se violan esas restricciones.

3.- Supervisión por parte del sistema de las operaciones de los usuarios para detectar tales violaciones..

B. RECUPERACIÓN DE INFORMACIÓN

I. Archivos para grandes volúmenes de datos

1. Métodos de acceso y almacenamiento

- ***Estructuras de datos auxiliares para organización de información: reservorios, diccionarios de datos, directorios, estructuras de tipo hipertexto***

Diccionario de datos.
Base de datos acerca de la terminología que se utilizará en un sistema de información. Para comprender mejor el significado de un diccionario de datos, puede considerarse su contenido como "datos acerca de los datos"; es decir, descripciones de todos los demás objetos (archivos, programas, informes, sinónimos...) existentes en el sistema. Un diccionario de datos almacena la totalidad de los diversos esquemas y especificaciones de archivos, así como sus ubicaciones. Si es completo incluye también información acerca de qué programas utilizan qué datos, y qué usuarios están interesados en unos u otros informes. Por lo general, el diccionario de datos está integrado en el sistema de información que describe. Contiene la información referente a la estructura de la base de datos.
Directorio (informática)
Organización jerárquica de nombres de archivos almacenados en un disco. El directorio superior se denomina directorio raíz; los directorios existentes dentro de otro directorio se denominan subdirectorios. Según la forma en que el sistema operativo soporte los directorios, los nombres de los archivos allí contenidos pueden verse y ordenarse de distintos modos, como por ejemplo alfabéticamente, por fecha o por tamaño, o en forma de iconos en una interfaz gráfica de usuario. Lo que el usuario ve como directorio está soportado en el sistema operativo en forma de tablas de datos, guardadas en el disco, que contienen las características asociadas con cada archivo, así como la ubicación de éste dentro del disco.
Hipertexto
Método de presentación de información en el que el texto, las imágenes, los sonidos y las acciones están unidos mediante una red compleja y no secuencial de asociaciones que permite al usuario examinar los distintos temas, independientemente del orden de presentación de los mismos. Normalmente es el autor el que establece los enlaces de un documento hipertexto en función de la intención del mismo. Por ejemplo, viajando a través de los enlaces de Encarta, la palabra hierro dentro de un artículo puede llevar al usuario a un sistema periódico de elementos o a un artículo referido a la edad del hierro. El término hipertexto fue creado por Ted Nelson en 1965, con el fin de describir los documentos que se presentan en un ordenador o computadora, o sea, expresando la estructura no lineal de las ideas, al contrario de la estructura lineal de los libros, las películas y el habla. El término hipermedia es prácticamente un sinónimo, pero recalca los componentes no textuales del hipertexto, como animaciones, sonido y vídeo.

- ***Selección de métodos de acceso y almacenamiento en función del volumen de los datos***
- ***Manejo de índices***

POR TABLAS ÍNDICE Mediante Índices densos. Cada uno de ellos está compuesto de dos campos:
* valor semántico
* puntero a donde apunta el valor semántico

V.Gr.	S1	S1 ... Londres	Londres
	S2	S2 ... París	Londres
	S3	S3 ... Londres	París
	S4	S4 ... Madrid	París
	S5	S5 ... París	Madrid
	S6	S6 ... Roma	Madrid
	S7	S7 ... Madrid	Roma

Una de las ventajas es que al hacer un barrido secuencial, nos devuelve la BdD ordenada. Además si solo nos interesa una parte de los registros (acceso directo) no tenemos que recorrer toda la BdD, ya que están agrupados. Los inconvenientes son el poseer tantas tablas indices como campos tenga nuestra BdD, lo que nos lleva a tener muchas tablas (mucho espacio); y si hacemos modificaciones, hay que transmitirlas a todas las tablas indices asociadas a los campos modificados. Una Tabla indice es una lista invertida (a partir de un campo encontramos un registro). Cuando una BdD tiene tantas tablas indices como campos, se dice que está totalmente invertida.

TABLAS INDICE MULTIPLES En el caso de que poseamos muchas tablas indice, podemos indexarlas de forma compuesta unas con otras. Es muy importante el orden por el que se vaya componiendo. Se las conoce como *Tablas de indices múltiples*. Si nuestra tabla es muy grande, entonces necesitamos utilizar los Indices no densos que se componen de dos campos:

* Apunta a la página donde está el registro
* El mayor valor semántico de esa página

V.Gr.

S#	*RID*
S1	011
S2	012
S3	013
S4	021
S5	022
S6	031
S7	032

TABLA	
S3	01
S5	02
S7	03

Todos los Registros tienen una llave primaria, que es un campo que nos permite diferenciar unívocamente al registro

C. SISTEMAS DE INFORMACIÓN

I. Teoría de sistemas

1. Conceptos generales

- ***Conceptos de teoría general de sistemas***

Introducción:
Un sistema de información es un conjunto de elementos que interactúan entre sí con el fin de apoyar las actividades de una empresa o negocio. El equipo computacional: el hardware necesario para que el sistema de información pueda operar. El recurso humano que interactúa con el Sistema de Información, el cual está formado por las personas que utilizan el sistema. Un sistema de información realiza cuatro actividades básicas: entrada, almacenamiento, procesamiento y salida de información.
Entrada de Información: Es el proceso mediante el cual el Sistema de Información toma los datos que requiere para procesar la información. Las entradas pueden ser manuales o automáticas. Las manuales son aquellas que se proporcionan en forma directa por el usuario, mientras que las automáticas son datos o información que provienen o son tomados de otros sistemas o módulos. Esto último se denomina interfases automáticas. Las unidades típicas de entrada de datos a las computadoras son las terminales, las cintas magnéticas, las unidades de diskette, los códigos de barras, los escáners, la voz, los monitores sensibles al tacto, el teclado y el mouse, entre otras.
Almacenamiento de información: El almacenamiento es una de las actividades o capacidades más importantes que tiene una computadora, ya que a través de esta propiedad el sistema puede recordar la información guardada en la sección o proceso anterior. Esta información suele ser almacenada en estructuras de información denominadas archivos. La unidad típica de almacenamiento son los discos magnéticos o discos duros, los discos flexibles o diskettes y los discos compactos (CD-ROM).
Procesamiento de Información: Es la capacidad del Sistema de Información para efectuar cálculos de acuerdo con una secuencia de operaciones preestablecida. Estos cálculos pueden efectuarse con datos introducidos recientemente en el sistema o bien con datos que están almacenados. Esta característica de los sistemas permite la transformación de datos fuente en información que puede ser utilizada para la toma de decisiones, lo que hace posible, entre otras cosas, que un tomador de decisiones genere una proyección financiera a partir de los datos que contiene un estado de resultados o un balance general de un año base.
Salida de Información: La salida es la capacidad de un Sistema de Información para sacar la información procesada o bien datos de entrada al exterior. Las unidades típicas de salida son las impresoras, terminales, diskettes, cintas magnéticas, la voz, los graficadores y los plotters, entre otros. Es importante aclarar que la salida de un Sistema de Información puede constituir la entrada a otro Sistema de Información o módulo. En este caso, también existe una interfase automática de salida. Por ejemplo, el Sistema de Control de Clientes tiene una interfase automática de salida con el Sistema de Contabilidad, ya que genera las pólizas contables de los movimientos procesales de los clientes. A continuación se muestran las diferentes actividades que puede realizar un Sistema de Información de Control de Clientes:
Actividades que realiza un Sistema de Información:
Entradas:

- Datos generales del cliente: nombre, dirección, tipo de cliente, etc.
- Políticas de créditos: límite de crédito, plazo de pago, etc.
- Facturas (interfase automático).
- Pagos, depuraciones, etc.

Proceso:

- Cálculo de antigüedad de saldos.
- Cálculo de intereses moratorios.
- Cálculo del saldo de un cliente.

Almacenamiento:

- Movimientos del mes (pagos, depuraciones).
- Catálogo de clientes.
- Facturas.

Salidas:

- Reporte de pagos.
- Estados de cuenta.
- Pólizas contables (interfase automática)
- Consultas de saldos en pantalla de una terminal.

Las diferentes actividades que realiza un Sistema de Información se pueden observar en el diseño conceptual ilustrado en la en la figura 1.2.

2. Sistemas de información

- ***Operativos***

¿Qué es un Sistema Operativo?

Desde su creación, las computadoras digitales han utilizado un sistema de codificación de instrucciones en sistema de numeración binaria, es decir con los 0S. Esto se debe a que los circuitos integrados funcionan con este principio, es decir, hay corriente o no hay corriente. En el origen de la historia de las computadoras (hace unos cuarenta años), los sistemas operativos no existían y la introducción de un programa para ser ejecutado se convertía en un increíble esfuerzo que solo podía ser llevado a cabo por muy pocos expertos. Esto hacia que las computadoras fueran muy complicadas de usar y que se requiriera tener altos conocimientos técnicos para operarlas. Era tan complejo su manejo, que en algunos casos el resultado llegaba a ser desastroso. Además, el tiempo requerido para introducir un programa en aquellas grandes máquinas de lento proceso superaba por mucho el de ejecución y resultaba poco provechosa la utilización de computadoras para resolución de problemas prácticos. Se buscaron medios más elaborados para manipular la computadora, pero que a su vez simplificaran la labor del operador o el usuario. Es entonces cuando surge la idea de crear un medio para que el usuario pueda operar la computadora con un entorno, lenguaje y operación bien definido para hacer un verdadero uso y explotación de esta. Surgen los sistemas operativos. Un sistema operativo es el encargado de brindar al usuario una forma amigable y sencilla de operar, interpretar, codificar y emitir las ordenes al procesador central para que este realice las tareas necesarias y especificas para completar una orden. El sistema operativo, es el instrumento indispensable para hacer de la computadora un objeto útil. Bajo este nombre se agrupan todos aquellos programas que permiten a los usuarios la utilización de este enredo de cables y circuitos, que de otra manera serian difíciles de controlar. Un sistema operativo se define como un conjunto de procedimientos manuales y automáticos, que permiten a un grupo de usuarios compartir una instalación de computadora eficazmente.

Interfaz de Línea de Comandos.

La forma de interfaz entre el sistema operativo y el usuario en la que este escribe los comandos utilizando un lenguaje de comandos especial. Los sistemas con interfaces de líneas de comandos se consideran más difíciles de aprender y utilizar que los de las interfaces gráficas. Sin embargo, los sistemas basados en comandos son por lo general programables, lo que les otorga una flexibilidad que no tienen los sistemas basados en gráficos carentes de una interfaz de programación.

Interfaz Gráfica del Usuario.

Es el tipo de visualización que permite al usuario elegir comandos, iniciar programas y ver listas de archivos y otras opciones utilizando las representaciones visuales (iconos) y las listas de elementos del menú. Las selecciones pueden activarse bien a través del teclado o con el Mouse. Para los autores de aplicaciones, las interfaces gráficas de usuario ofrecen un entorno que se encarga de la comunicación con el ordenador o computadora. Esto hace que el programador pueda concentrarse en la funcionalidad, ya que no esta sujeto a los detalles de la visualización ni a la

entrada a través del Mouse o el teclado. También permite a los programadores crear programas que realicen de la misma forma las tareas más frecuentes, como guardar un archivo, porque la interfaz proporciona mecanismos estándar de control como ventanas y cuadros de diálogo. Otra ventaja es que las aplicaciones escritas para una interfaz gráfica de usuario son independientes de los dispositivos: a medida que la interfaz cambia para permitir el uso de nuevos dispositivos de entrada y salida, como un monitor de pantalla grande o un dispositivo óptico de almacenamiento, las aplicaciones pueden utilizarlos sin necesidad de cambios.

Funciones de los Sistemas Operativos.

- ✓ Interpreta los comandos que permiten al usuario comunicarse con el ordenador.
- ✓ Coordina y manipula el hardware de la computadora, como la memoria, las impresoras, las unidades de disco, el teclado o el Mouse.
- ✓ Organiza los archivos en diversos dispositivos de almacenamiento, como discos flexibles, discos duros, discos compactos o cintas magnéticas.
- ✓ Gestiona los errores de hardware y la pérdida de datos.
- ✓ Servir de base para la creación del software logrando que equipos de marcas distintas funcionen de manera análoga, salvando las diferencias existentes entre ambos.
- ✓ Configura el entorno para el uso del software y los periféricos; dependiendo del tipo de máquina que se emplea, debe establecerse en forma lógica la disposición y características del equipo. Como por ejemplo, una microcomputadora tiene físicamente dos unidades de disco, puede simular el uso de otras unidades de disco, que pueden ser virtuales utilizando parte de la memoria principal para tal fin. En caso de estar conectado a una red, el sistema operativo se convierte en la plataforma de trabajo de los usuarios y es este quien controla los elementos o recursos que comparten. De igual forma, provee de protección a la información que almacena.

Categoría de los Sistemas Operativos.

Sistema Operativo Multitareas.

Es el modo de funcionamiento disponible en algunos sistemas operativos, mediante el cual una computadora procesa varias tareas al mismo tiempo. Existen varios tipos de multitareas. La conmutación de contextos (context Switching) es un tipo muy simple de multitarea en el que dos o más aplicaciones se cargan al mismo tiempo, pero en el que solo se esta procesando la aplicación que se encuentra en primer plano (la que ve el usuario). Para activar otra tarea que se encuentre en segundo plano, el usuario debe traer al primer plano la ventana o pantalla que contenga esa aplicación. En la multitarea cooperativa, la que se utiliza en el sistema operativo Macintosh, las tareas en segundo plano reciben tiempo de procesado durante los tiempos muertos de la tarea que se encuentra en primer plano (por ejemplo, cuando esta aplicación esta esperando información del usuario), y siempre que esta aplicación lo permita. En los sistemas multitarea de tiempo compartido, como OS/2, cada tarea recibe la atención del microprocesador durante una fracción de segundo. Para mantener el sistema en orden, cada tarea recibe un nivel de prioridad o se procesa en orden secuencial. Dado que el sentido temporal del usuario es mucho más lento que la velocidad de procesamiento del ordenador, las operaciones de multitarea en tiempo compartido parecen ser simultáneas.

Sistema Operativo Monotareas.

Los sistemas operativos monotareas son más primitivos y es todo lo contrario al visto anteriormente, es decir, solo pueden manejar un proceso en cada momento o que solo puede ejecutar las tareas de una en una. Por ejemplo cuando la computadora esta imprimiendo un documento, no puede iniciar otro proceso ni responder a nuevas instrucciones hasta que se termine la impresión.

Sistema Operativo Monousuario.

Los sistemas monousuarios son aquellos que nada más puede atender a un solo usuario, gracias a las limitaciones creadas por el hardware, los programas o el tipo de aplicación que se este ejecutando.

Estos tipos de sistemas son muy simples, porque todos los dispositivos de entrada, salida y control dependen de la tarea que se esta utilizando, esto quiere decir, que las instrucciones que se dan, son procesadas de inmediato; ya que existe un solo usuario. Y están orientados principalmente por los microcomputadores.

Sistema Operativo Multiusuario.

Es todo lo contrario a monousuario; y en esta categoría se encuentran todos los sistemas que cumplen simultáneamente las necesidades de dos o más usuarios, que comparten mismos recursos. Este tipo de sistemas se emplean especialmente en redes. En otras palabras consiste en el fraccionamiento del tiempo (timesharing).

Secuencia por Lotes.

La secuencia por lotes o procesamiento por lotes en microcomputadoras, es la ejecución de una lista de comandos del sistema operativo uno tras otro sin intervención del usuario. En los ordenadores más grandes el proceso de recogida de programas y de conjuntos de datos de los usuarios, la ejecución de uno o unos pocos cada vez y la entrega de los recursos a los usuarios. Procesamiento por lotes también puede referirse al proceso de almacenar

transacciones durante un cierto lapso antes de su envío a un archivo maestro, por lo general una operación separada que se efectúa durante la noche. Los sistemas operativos por lotes (batch), en los que los programas eran tratados por grupos (lote) en ves de individualmente. La función de estos sistemas operativos consistía en cargar en memoria un programa de la cinta y ejecutarlo. Al final este, se realizaba el salto a una dirección de memoria desde donde reasumía el control del sistema operativo que cargaba el siguiente programa y lo ejecutaba. De esta manera el tiempo entre un trabajo y el otro disminuía considerablemente.

Tiempo Real.

Un sistema operativo en tiempo real procesa las instrucciones recibidas al instante, y una vez que han sido procesadas muestra el resultado. Este tipo tiene relación con los sistemas operativos monousuarios, ya que existe un solo operador y no necesita compartir el procesador entre varias solicitudes.

Su característica principal es dar respuestas rápidas; por ejemplo en un caso de peligro se necesitarían respuestas inmediatas para evitar una catástrofe.

Tiempo Compartido.

El tiempo compartido en ordenadores o computadoras consiste en el uso de un sistema por más de una persona al mismo tiempo. El tiempo compartido ejecuta programas separados de forma concurrente, intercambiando porciones de tiempo asignadas a cada programa (usuario). En este aspecto, es similar a la capacidad de multitareas que es común en la mayoría de los microordenadores o las microcomputadoras. Sin embargo el tiempo compartido se asocia generalmente con el acceso de varios usuarios a computadoras más grandes y a organizaciones de servicios, mientras que la multitarea relacionada con las microcomputadoras implica la realización de múltiples tareas por un solo usuario.

- ***Estratégicos***

Sus principales características son:

- Su función primordial no es apoyar la automatización de procesos operativos ni proporcionar información para apoyar la toma de decisiones.
- Suelen desarrollarse in house, es decir, dentro de la organización, por lo tanto no pueden adaptarse fácilmente a paquetes disponibles en el mercado.
- Típicamente su forma de desarrollo es a base de incrementos y a través de su evolución dentro de la organización. Se inicia con un proceso o función en particular y a partir de ahí se van agregando nuevas funciones o procesos.
- Su función es lograr ventajas que los competidores no posean, tales como ventajas en costos y servicios diferenciados con clientes y proveedores. En este contexto, los Sistema Estratégicos son creadores de barreras de entrada al negocio. Por ejemplo, el uso de cajeros automáticos en los bancos en un Sistema Estratégico, ya que brinda ventaja sobre un banco que no posee tal servicio. Si un banco nuevo decide abrir sus puerta al público, tendrá que dar este servicio para tener un nivel similar al de sus competidores.
- Apoyan el proceso de innovación de productos y proceso dentro de la empresa debido a que buscan ventajas respecto a los competidores y una forma de hacerlo en innovando o creando productos y procesos.

Un ejemplo de estos Sistemas de Información dentro de la empresa puede ser un sistema MRP (Manufacturing Resoure Planning) enfocado a reducir sustancialmente el desperdicio en el proceso productivo, o bien, un Centro de Información que proporcione todo tipo de información; como situación de créditos, embarques, tiempos de entrega, etc. En este contexto los ejemplos anteriores constituyen un Sistema de Información Estratégico si y sólo sí, apoyan o dan forma a la estructura competitiva de la empresa. Por último, es importante aclarar que algunos autores consideran un cuarto tipo de sistemas de información denominado Sistemas Personales de Información, el cual está enfocado a incrementar la productividad de sus usuarios.

- ***Sistemas para trabajo en grupo (GDSS)***

Son sistemas como Lotus Notes que permite la colaboración de grupos de trabajo e incluye herramientas como Chat, lista de correos, colaboración, recuperación de archivos, mesa de ayudas, etc.

- ***Sistemas de flujo (Workflow)***

CSCWP: Computer Supported Collaborative Work Processing

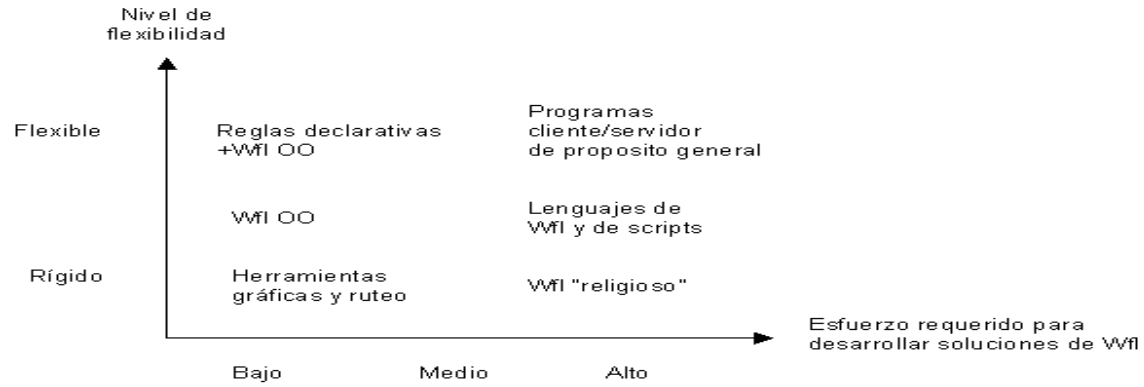
"Automatización de los procesos que se usan diariamente en una empresa. Una aplicación de Workflow automatiza la secuencia de acciones, actividades y tareas usadas para ejecutar los procesos, incluyendo monitoreo en cada instante del proceso, así como las herramientas para administrar el proceso mismo".

Tratamiento de información

"Flujo de trabajo es la secuencia de acciones y pasos usados en procesos de negocios. Flujo de trabajo automatizado aplica tecnología al proceso, aunque no necesariamente en cada acción. En esta definición es necesario que más de una persona esté involucrada".
Flujo de trabajo en imágenes de documentos
Workflow tiene sus raíces en imágenes de documentos:

- Scanning
- OCR
- Indexación (archivos estructurados, contenido)
- Almacenamiento
- Registros, archivos, versiones
- Consulta
- Procesamiento del flujo de trabajo
- Taxonomías de sistemas de flujo de trabajo

¿Cuánta programación se requiere?
Nivel de programación requerido por usuarios y corporación que adoptan esta tecnología
¿Es un compilador de C un sistema de workflow?
Otro extremo: bloques predefinidos donde el usuario construye su aplicación



Ejemplo
Winograd y Flores, de Action Technologies, proponen un modelo basado en teorías de lenguaje y comunicación llamado "Coordinador". Ven a una organización como una "red de compromisos". Los gerentes realizan conversaciones donde crean, se preocupan e inician nuevos compromisos. Compromiso para una acción, característica que precede a una acción. Su modelo de Workflow está basado en este modelo:

- Abrir
- Negociar
- Rendimiento o acción
- Satisfacción o aceptación
- Implica cambiar la organización

Workflow basado en mensajes o servidores
También se conoce como basado en "e-mail" o en "bases de datos".
Dos enfoques para Workflow basado en mensajes:
Aplicaciones sobre e-mail mejorado: norificaciones, recibos, formularios, reenvio basado en reglas, llenado automático, firma digital, etc. Suficientes para soportar Workflow liviano, Workflow administrativo y Workflow ad-hoc.
Aplicaciones apoyadas por e-mail pero con extensiones de Workflow: paquetes de formularios, ruteo secuencial de formularios, monitoreo del progreso en la ruta de un formulario, etc.
Workflow basado en bases de datos:
Implementado sobre un producto comercial, que generalmente tienen manejo de documentos.
Algunos están generando sus propios productos Workflow.
Típico en arquitecturas cliente/servidor.
Poder a los usuarios
"Downsizing" en empresas pretende dar más responsabilidad en desarrollo a los usuarios.
Herramientas generan aplicaciones Workflow desarrolladas por trabajadores especialistas ("knowledge workers").
Workflow pensado como asistentes del trabajador: creatividad, manejo de excepciones, procesamiento de tareas.
Tipos de tecnología Workflow

Tratamiento de información

Workflow Transaccional o de Producción
Existen políticas claras y complejas impuestas por la empresa, desarrolladas durante el tiempo.
Bajo nivel de cambios en los procedimientos, con transacciones recurrentes.
Mucho trabajo realizado por personal no calificado: auditorías frecuentes.
Ejemplos: procesamiento de préstamos, pólizas de seguros, procesamiento de reclamos.
Workflow Ad-Hoc
Apoyan procesos que pueden cambiar y no tan intensivos en transacciones.
Ejemplos: definición de un nuevo producto, marketing de un producto existente, contratar una nueva persona. Existen fechas límites y responsables, pero éstos pueden cambiar.
Workflow Administrativo
Usualmente basado en sistemas de e-mail, maneja tareas administrativas rutinarias.
E-mail con extensiones tales como:
Creación de formularios simples.
Ruteo de formularios.
Iteración para completación del formulario.
Fechas críticas, notificaciones, alarmas, etc.
Ayuda al procesamiento de trabajo
Al diseñar un sistema de Workflow, considerar objetos involucrados tales como dispositivos, personal, agenda de tareas, etc.
Diseñar diagramas de flujo de datos para procesos de negocios, procedimientos cooperativos, proyectos departamentales y corporativos.
Para generar y evaluar alternativas:
Dibuje un modelo de información del dominio del problema antes de especificar y diseñar diagramas de flujo de datos y de control. Este modelo debe describir la estructura de los objetos involucrados en los procedimientos, procesos y proyectos (documentos, formularios, carpetas, cardex, etc.)
Diseñe y construya un modelo de los departamentos, junto con sus roles, grupos y trabajadores especializados. Aquí se debe identificar a los participantes del Workflow.
Análisis de los flujos de trabajo y datos de los objetos, utilizando herramientas de diseño para identificar relaciones dinámicas de los objetos.
Evaluación de estrategias de implementación, especialmente aquellas que involucran productos Workflow.
Controlando el flujo de trabajo
Los productos de Workflow muestran cuatro elementos comunes:
Objetivo: Se refiere al flujo de trabajo para completar un objetivo. El objetivo puede ir desde un entregable simple hasta múltiples entregables.
Participantes o nodos participantes: Tipo de participantes tales como trabajadores, procesos, dispositivos, etc. Para que el trabajo fluya, deben existir enviadores y receptores. La forma de definir la semántica de los participantes depende de la herramienta. Los participantes asumen roles.
Flujo o conexión de participantes: Conexión o flujo de información, documentos, instrucciones entre participantes. Varios tipos de control de flujo:
Por ejemplo, ruteo de documentos. Este puede manejar decisiones dependiendo del estátus, disponibilidad, colas, tráfico, etc.
Documentos y formularios procesados: Tipos de datos a intercambiar entre los participantes.
Más que flujo
En general Workflow se especifica en términos de trabajo y flujo de información entre participantes.
El comité de estandarización "Workflow Management Coalition" (WMC) incluye:
Actividad: Una pieza de trabajo hecha por una aplicación, un usuario, un paso, un script, un nodo.
Definición de proceso: Cuando las actividades están unidas por una red, el resultado es la definición de un proceso que incorpora la semántica del flujo de trabajo, incluyendo los punteros que conectan estas actividades. WMC define estas actividades dentro de una "red de actividades".
Instancia de proceso: Son las activaciones e instancias de un proceso. En un sistema Workflow, las instancias de procesos tienen nombre, se activan y son monitoreadas.
Características y conceptos
Mostraremos componentes de sistemas Workflow:
Definición gráfica de Workflow: diagramas estáticos de flujo de control del Workflow.

Definición de procesos y activación: El proceso captura las reglas y pasos para un procedimiento particular. Luego nodos, conexiones, participantes, periféricos, acciones, reglas, etc. en la definición de una plantilla de Workflow deben asociarse con los objetos que participan en el Workflow.
Monitoreo, estátus y estadísticas. Una de las más importantes características de un Workflow es la habilidad de monitorear el estátus y estadísticas. Información crítica debiese ser automáticamente generada y administrada por el motor Workflow en progreso.
Colas de trabajo: el trabajo se almacena en orden FIFO para su procesamiento.
Cajas: objetos donde almacenar o agrupar tipos de información, tales como folders.
Grupos y roles. Los dispositivos cumplen roles bien definidos y específicos: faxes, impresoras, scanners, etc. En el caso de personas es más complejo: los privilegios de acceso a la información dependen de su rol en la empresa.
Reasignación: permite cambiar o reasignar responsabilidades de un trabajador a otro.
Reglas y condiciones: permiten definir mecanismos de enrutamiento de información. Por ejemplo:
If "Aplicación de crédito" en Nodo "Contratación de Gerente para Aprobación" tiene más de 3 días AND pedido de contratación es URGENTE THEN envíe memo al Gerente de Personal;
Notificaciones: puede ser considerado un caso especial de reglas. Por ejemplo:
SEND RECEIPTS to <trabajador siendo notificado> IF Workflow ha llegado a un cierto estado;
Suspensión y Randezvous: a veces se requiere juntar resultados de varios procesos antes de continuar con otro. Por ejemplo, procesos de aprobación.
Iteración: caso especial de suspensión.

- ***Sistemas para la toma de decisiones***

Las principales características de estos son:

- Suelen introducirse después de haber implantado los Sistemas Transaccionales más relevantes de la empresa, ya que estos últimos constituyen su plataforma de información.
- La información que generan sirve de apoyo a los mandos intermedios y a la alta administración en el proceso de toma de decisiones.
- Suelen ser intensivos en cálculos y escasos en entradas y salidas de información. Así, por ejemplo, un modelo de planeación financiera requiere poca información de entrada, genera poca información como resultado, pero puede realizar muchos cálculos durante su proceso.
- No suelen ahorrar mano de obra. Debido a ello, la justificación económica para el desarrollo de estos sistemas es difícil, ya que no se conocen los ingresos del proyecto de inversión.
- Suelen ser Sistemas de Información interactivos y amigables, con altos estándares de diseño gráfico y visual, ya que están dirigidos al usuario final.
- Apoyan la toma de decisiones que, por su misma naturaleza son repetitivos y de decisiones no estructuradas que no suelen repetirse. Por ejemplo, un Sistema de Compra de Materiales que indique cuándo debe hacerse un pedido al proveedor o un Sistema de Simulación de Negocios que apoye la decisión de introducir un nuevo producto al mercado.
- Estos sistemas pueden ser desarrollados directamente por el usuario final sin la participación operativa de los analistas y programadores del área de informática.

Este tipo de sistemas puede incluir la programación de la producción, compra de materiales, flujo de fondos, proyecciones financieras, modelos de simulación de negocios, modelos de inventarios, etc.

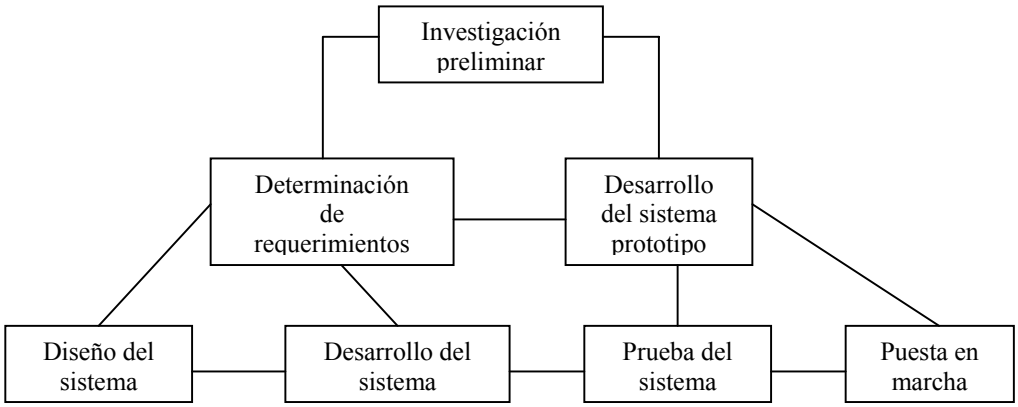
II. Análisis y diseño de sistemas de información

1. Ciclo de vida de sistemas

El desarrollo de sistemas es un proceso que consiste en dos etapas principales de análisis y diseño de sistemas; comienza cuando la gerencia, o en algunas ocasiones el personal de desarrollo de sistemas, se da cuenta de cierto sistema del negocio necesita mejorarse. El ciclo de vida del desarrollo de sistemas es el conjunto de actividades de los analistas, diseñadores y usuarios, que necesitan llevarse a cabo para desarrollar y poner en marcha un sistema de información. Se debe tener presente que en la mayoría de las situaciones del negocio, las actividades están íntimamente relacionadas y son inseparables. El ciclo de vida del desarrollo de sistemas consiste en las siguientes actividades:

1. Investigación preliminar
2. Determinación de requerimientos
3. Desarrollo de sistema prototipo
4. Diseño de sistema

5. Desarrollo de software
6. Prueba de los sistemas
7. Puesta en marcha



- ***Análisis preliminar***

Investigaciones preliminares

¿Cuantas veces se está en situaciones en donde se pregunta si no existe una mejor manera de hacer algo? Por ejemplo, abrir una tienda departamental adicional que creará una necesidad para nuevos procedimientos de facturación, cuando un alto porcentaje de clientes utiliza la cuenta de crédito de esta compañía de esta compañía y compra en todas las tiendas. Duplicar el número de clientas para agrandar las instalaciones y la introducción de muchos nuevos productos, puede traer nuevos requerimientos de pago e cuentas. Un cambio en las áreas de los gerentes departamentales puede guiarlos hacia nuevas formas para registrar las ventas, con implicaciones para el sistema de entrada de pedidos basado en computadora. Una compañía en crecimiento, puede contemplara los sistemas de información computarizados como una forma para hacer posible el crecimiento continuo, sin tener dificultades en el proceso de los pedidos de los clientes. Se puede inicias una petición por muchas razones, pero la clave es que alguien, ya sea gerente, un empleado o un especialista de sistemas, inicie un requerimiento para recibir ayuda de un sistema de información. Cuando ese requerimiento se realiza, la primera actividad de sistemas, es decir, la investigación preliminar, se inicia. Esta actividad tiene tres partes: clasificación de requerimiento, estudio de la factibilidad y aprobación del requerimiento. El resultado será aprobar el requerimiento para la atención posterior o rechazarlo como no factible para un desarrollo futuro.

Clarificación del requerimiento

En las empresas muchos requerimientos de los empleados y usuarios no están establecidos claramente; por lo tanto, antes de que pueda considerarse la investigación del sistema, le proyecto requerido debe examinarse para determinar para determinar precisamente lo que desea la empresa. Una simple llamada telefónica puede ser suficiente si la persona que requiere el servicio tiene una idea clara, pero no sabe cómo establecerla. Por otro lado, la persona que hace el requerimiento puede estar simplemente pidiendo ayuda sin saber qué es lo que está mal o por qué existe un problema. La clarificación del problema es este caso, antes de poder llagar a otro paso, el requerimiento de proyecto debe estar claramente establecido.

Estudio de Factibilidad

Un resultado importante de la investigación preliminar es la determinación de que el sistema requerido es factible. Existen tres aspectos en el estudio de factibilidad de la investigación preliminar:

1. Factibilidad técnica. ¿Puede realizarse el trabajo para el proyecto con el equipo actual, tecnología de software y el personal disponible? Si se requiere nueva tecnología, ¿qué probabilidades hay de que pueda desarrollarse?
2. factibilidad económica. ¿Existen suficientes beneficios en la creación del sistema para hacer que los costos sean aceptables? O, en forma inversa, ¿son tan altos los costos como para que el proyecto no deba llevarse a cabo?
3. Factibilidad operativa. ¿Se utilizará el sistema si se desarrolla y pone en marcha? Habrá resistencia de los usuarios, que los posibles beneficios reducirán del sistema.

Tratamiento de información

El estudio de factibilidad se lleva a cabo con un pequeño grupo de gente, familiarizada con las técnicas de los sistemas de información, que entienden la parte de la empresa que será afectada por el proyecto y tienen los conocimientos suficientes del proceso de análisis y diseño de sistemas.

- ***Análisis detallado***

Aprobación del requerimiento

No todos los proyectos requeridos son deseables o factibles. Sin embargo, aquellos que son tanto factibles como deseables deben anotarse para tomarlos en cuenta. En algunos casos, el desarrollo puede comenzar inmediatamente, pero en la mayor parte, los miembros del departamento de sistemas están ocupados en otros proyectos que se encuentran en marcha. Cuando esto sucede, la gerencia decide que los proyectos son más importantes y entonces los programas. Después de que se aprueba la requisición de un proyecto, se estima su costo, la prioridad, el tiempo de terminación y los requerimientos del personal que se utilizan, para determinar qué lista existente los proyectos se incluirá. Posteriormente, cuando se terminan algunos proyectos anteriores, puede iniciarse el desarrollo de la aplicación propuesta. En este momento, comienza la recabación de datos y la determinación de los requerimientos.

Determinación de requerimientos

El punto clave de análisis de sistemas se consigue al adquirir un conocimiento detallado de todas las facetas importantes dentro del área de negocios que se investiga. (Por esta razón, a menudo esta actividad se conoce como *investigación detallada*.) Los analistas, al trabajar con los empleados y gerentes, deben estudiar el proceso que actualmente se efectúa para contestar estas preguntas clave:

1. ¿Qué se está haciendo?
2. ¿Cómo se está haciendo?
3. ¿Qué tan frecuentemente ocurre?
4. ¿Qué tan grande es la cantidad de transacciones o decisiones?
5. ¿Qué tan bien se lleva acabo la tarea?
6. ¿Existe algún problema?
7. ¿Si el problema existe, qué tan serio es?
8. ¿Si el problema existe, cuál es la causa principal?

Para contestar estas preguntas, los analistas de sistemas hablarán con diferentes personas para recabar los detalles en relación con el proceso, así como sus opiniones sobre las causas por las cuales suceden las cosas de esa manera y algunas ideas en relación a modificarlas. Se utilizan cuestionarios para recopilar esta información, aplicándolos a grandes que no pueden entrevistarse en forma individual. Las investigaciones detalladas también requieren el estudio de manuales y reportes, la observación real de las actividades de las actividades de trabajo y algunas veces la recabación de formas y documentos para entender completamente el proceso. Conforme se recopilan los elementos, los analistas estudian los requerimientos de datos para identificar las características que tendrá el nuevo sistema, incluyendo la información que el sistema debe producir y las características operativas, como son controles de procesamiento, tiempos de respuesta y métodos de entrada y salida.

- ***Diseño preliminar***

Desarrollo del sistema prototipo

La **preparación de prototipos** es el proceso de crear, desarrollar y refinar un modelo funcional del sistema final. Se puede crear un **modelo prototipo preliminar** durante la etapa de definición del problema. Un miembro del equipo de reconocimiento -suponga que se trata de un especialista en el procesamiento de datos- puede construir un modelo de este tipo que muestre la composición de las pantallas y los formatos de los informes. Durante una sesión de requerimientos, otros miembros del equipo y usuarios del futuro sistema examinan esta muestra en la forma con el constructor del modelo entiende en principio el problema y los resultado que debe producir el sistema. En este momento puede iniciarse un proceso de refinación si los usuarios señalan omisiones y equivocas. Durante este proceso de refinación, cuyo objetivo es definir la necesidad que existe, uno o más miembros del equipo pueden utilizar una computadora personal y un **paquete de programas de prototipos** a fin de crear una serie de pantallas en la computadora personal. Estas pantallas no son las salidas que producen los programas ya terminados, pero pueden parecerse mucho a esos resultados. Es posible exhibir en el monitor de la computadora, como una secuencia de diapositivas, menús de captura de datos, la interfaz con el usuario debe servir para buscar, consultar y manipular datos y el formato de los informes de salida. Por ejemplo, se pueden simular los resultados de una serie de selecciones hechas en menús para que los usuarios tengan una idea más clara de la forma como el constructor o los constructores del sistema están interpretando el problema. Si los usuarios no están convencidos de lo que se exhibe define con precisión sus necesidades, pueden modificar fácilmente las plantillas prototipo hasta que estén

satisfechos. La creación de un modelo preliminar de prototipo en este punto produce varios beneficios: los usuarios pueden ver que se está avanzado, se les motiva para que participen activamente en la definición del problema, se mejora la comunicación 4entre todas las partes interesadas y se aclaran los equívocos en una etapa temprana del estudio de sistemas, antes de que se conviertan en costosos errores. Como se acaba e ver, puede ser necesario un proceso repetitivo (o *interactivo*) para terminar el paso de definición del problema. No existe un procedimiento definido que se deba seguir antes de que se pueda iniciarse el análisis detallado del sistema. Un alto ejecutivo puede creer que existen diferencias de información. Puede preparar una declaración general de los objetivos y nombrar a un gerente para que realice un reconocimiento. Pueden realizarse varias sesiones de requerimientos para traducir los deseos generales a objetivos más específicos. Asimismo, pueden crearse y refinarse modelos preliminares de prototipo; se puede ampliar o reducir el alcance del estudio y es posible también que cambien los objetivos conforme se reúnan los datos. Una vez que parezca haberse logrado la aprobación en cuanto a la definición del problema, el equipo de reconocimiento deberá poner la definición detallada *por escrito* y enviarla a todas las personas interesadas, las cuáles deberán aprobarla también por escrito. Si persisten diferencias, deberán resolverse en sesiones adicionales de requerimientos. Hay quienes se impacientan con los "retrasos" en el desarrollo del sistema causados por estas sesiones adicionales. Sin embargo, las personas más prudentes saben que los retrasos verdaderamente largos y costosos se presentan cuando los usuarios descubren, ya muy avanzados el proceso del desarrollo, que el sistema diseñado no es satisfactorio por haberse pasado por alto algunos requerimientos.

- ***Diseño detallado***

Diseño del sistema

El diseño de un sistema de información produce los elementos que establecen cómo el sistema cumplirá los requerimientos indicados durante el análisis de sistemas. A menudo los especialistas de sistemas se refieren a esta etapa como en *diseño lógico*, encontraste con desarrollo del software de programas, que se conoce como *diseño físico*. Los analistas de sistemas comienzan por identificar los informes y otras salidas que el sistema producirá. A continuación los datos específicos con éstos se señalan, incluyendo su localización exacta sobre el papel, la pantalla de despliegue u otro medio. Usualmente, los diseñadores dibujan la forma o la visualización como la esperan cuando el sistema esta terminado. El diseño del sistema también describe los datos calculados o almacenados que se introducirán. Los grupos de datos individuales y los procedimientos de calculo se describen con detalle. Los diseñadores seleccionan las estructuras de los archivos y los dispositivos de almacenamiento, como son discos magnéticos, cintas magnéticas o incluso archivos en papel. Los procedimientos que ellos escriben muestran cómo se van a procesar los datos y a producir la salida. Los documentos que contienen las especificaciones de diseño utilizan muchas formas para representar los diseños, diagramas, tablas y símbolos especiales, algunos de los cuales el lector puede haber utilizado ya y otros que pudieran ser totalmente nuevos. La información del diseño detallado se pasa al grupo de programación para que pueda comenzar el desarrollo del software. Los diseñadores son responsables de proporcionar a los programadores las especificaciones completas y escritas con claridad, que establezcan lo que debe hacer el software. Conforme comienza la programación, los diseñadores están pendientes para contestar preguntas, esclarecer ideas confusas y manejar los problemas que confronten los programadores cuando utilicen las especificaciones de diseño.

- ***Construcción***

Desarrollo del Software

Los desarrollares del software pueden instalar o modificar; por ejemplo, software comercial que se haya comprado, o pueden escribir programas nuevos diseñados a la medida. La decisión de qué se va a hacer depende del costo de cada una de las opciones, el tiempo disponible para describir el software y la disponibilidad de programadores. En forma usual, en las grandes empresas los programadores de computadoras (o la combinación de analistas-programadores) son parte del grupo profesional permanente. Las compañías más pequeñas en donde los programadores permanentes no se han contratado, pueden obtener servicios externos de programación con base en un contrato. Los programadores también son responsables de documentar el programa e incluir los comentarios que expliquen tanto cómo y por qué se utilizo cierto procedimiento conforma se codifico de cierta forma. La documentación es esencial para probar el programa y darle mantenimiento una vez que la aplicación se ha puesto en marcha.

- ***Pruebas***

Durante la prueba, el sistema se utiliza en forma experimental para asegurar que el software no falle; es decir, Que corra de acuerdo a sus especificaciones y a la manera que los usuarios esperan que lo haga. Se examinan datos especiales de prueba en la entrada del procesamiento y los resultados para localizar algunos problemas inesperados.

Puede permitirse también a un grupo limitado de usuarios que utilice el sistema, de manera que los analistas puedan captar si tratan de utilizarlo en forma no planeadas. Es preferible detectar cualquier anomalía antes de que la empresa ponga en marcha el sistema y dependa de él. En muchas compañías la prueba se lleva a cabo por personas diferentes a aquellos que los escriben en forma original; es decir si se utilizan personas que no conocen como se diseñaron ciertas partes de los programas, se asegura una mayor y más completa prueba, además de ser imparcial, lo que da a un software más confiable.

- ***Instalación***

Cuando el personal de sistemas verifica y pone en uso el nuevo equipo, entrena al personal usuario; instala la nueva aplicación y constituye los archivos de datos que se necesiten, entonces el sistema está puesto en marcha. De acuerdo con el tamaño de la empresa que empleará la aplicación y el riesgo asociado con su uso, los desarrolladores del sistema pueden escoger una prueba piloto para la operación del sistema solamente en un área de la compañía; por ejemplo, en un departamento o sólo con una o dos personas. A veces correrán en forma paralela tanto el sistema anterior como el nuevo para comparar los resultados de ambos; en otras situaciones, los desarrolladores pararán por completo el sistema anterior un día y al siguiente empezarán a utilizar el nuevo. Como se puede apreciar, cada estrategia para la puesta en marcha tiene sus méritos, que dependen de la situación del negocio considerado. Sin importar la estrategia para la puesta en marcha que se haya utilizado, los desarrolladores tendrán que asegurarse que el uso inicial del sistema esté libre de problemas. Una vez instalada, con frecuencia la aplicación se utiliza por muchos años; sin embargo, tanto la empresa como los usuarios cambiarán, y el medio ambienta será diferente también a través del tiempo. Por lo tanto, la aplicación indudable mente necesitará mantenimiento; es decir, se harán cambios y modificaciones al software, y a los archivos o procedimientos para cubrir los requerimientos nuevos de los usuarios. Los sistemas de la empresa y el medio ambiente de los negocios están en continuo cambio. Los sistemas de información deben mantenerse de la misma forma; es este sentido, la propuesta en marcha es un proceso continuo.