UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE CIENCIAS

Estudio sobre el
Lenguaje Estructurado de Consulta
SQL, y Bases de Datos Relacionales

T E S I S

Que para optar por el título de

A C T U A R I A

p r e s e n t a

MARGARITA ALCALA CASTAÑEDA

México, D. F. 1991

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

INDICE

INTRODUCCION

La administración de bases de datos es una de las funciones más importantes de los Sistemas de Computación Modernos. Aunque los usuarios tienen distintas necesidades globales, comparten muchas expectativas respecto del uso del sistema de bases de datos.

El presente trabajo pretende coadyuvar en la comprensión de lo que se conoce como base de datos relacional, así como de los usos de los Lenguajes de Consulta Estructurados, específicamente se presenta el SQL.

Por medio de SQL, las consultas de los usuarios a las bases de datos se hacen entendibles, sobre todo para quienes pretendan manejar la información que éstas contienen.

Asimismo, se hace una revisión de los sistemas manejadores de bases de datos relacionales, tales como: Informix, Oracle Profesional, Dbase IV y DB2. Estos utilizan SQL.

Es importante la consideración de las bases de datos relacionales, ya que permiten que los datos se vean como tablas de columnas y renglones, las cuales pueden definirse de tal modo que no redunden datos. Estos arreglos se logran mediante el proceso de normalización.

Con los lenguajes de consulta es posible formular preguntas acerca de alguna base de datos, y con ellos, la computadora puede encontrar el camino para dar la respuesta. Un buen lenguaje de consulta es aquel en el cual se formula la consulta y se obtiene la respuesta, y además, por medio de él se puede agregar renglones, borrarlos, crear tablas, suprimirlas, crear índices, suprimirlos, etc.

En cuanto a SQL, se puede decir que, antes de su existencia no había un lenguaje estándar por medio del cual se pudiera tener acceso directo a los datos. SQL es un lenguaje de consulta estructurada, no-procedural que permite el control, definición y manipulación de datos de una manera fácil, ya que cuenta con 30 comandos aproximadamente, sin que por eso deje de ser un lenguaje rico y poderoso; es dinámico y puede ser adoptado por los usuarios con sistemas de bases de datos relacionales; es usado por sistemas manejadores de bases de datos relacionales tales como: dBASE IV, Informix y Oracle Profesional, los que pueden consultar sus bases, ya sea encajando comandos SQL dentro de sus programas de aplicación, mediante el uso de un lenguaje de cuarta generación o en forma interactiva.

SQL fue desarrollado a fines de los 70's por IBM posteriormente fue adoptado por The American National Standards Institute (ANSI) y la International Standars Organization

(ISO) como el lenguaje estándar para el acceso a los sistemas manejadores de bases de datos relacionales. Varias compañías dedicadas a la elaboración de software para microcomputadoras, adoptaron SQL como el lenguaje de interfaz para poder manipular, accesar y actualizar sus bases de datos, obteniendo gran eficiencia, pues la integración de SQL a sus sistemas dio como resultado reducción de tiempo y costo en el mantenimiento de sus bases de datos, es decir, el manejo, control y actualización se realiza con facilidad a través de SQL, y aunque la rapidez y eficiencia están presentes en la formulación de consultas a las bases de datos con el uso de SQL, la forma precisa de éstas, dependerá principalmente del tamaño del material que se requiera.

Así entonces, se puede esperar que en los 90's aumente considerablemente el número de sistemas manejadores de bases de datos relacionales, que utilicen a SQL como el lenguaje de interfaz entre sistema y usuario, para así lograr mayor facilidad de manipulación, control y definición de datos.

PARTE I

BASES DE DATOS

La humanidad se ha interesado por los datos, al menos durante los pasados 1200 años, y aunque en la actualidad a menudo es asociado el concepto de datos con la computadora, históricamente han existido otros métodos primitivos de manejo de datos, en realidad algunos todavía siguen utilizándose.

Los orígenes primarios del interés de datos pueden seguirse hasta el surgimiento de las ciudades. Los principios de la producción en masa, la especialización de la mano de obra, el empleo de dinero y la posibilidad de adquirir servicios y productos para las necesidades de la vida requieren la conversión de datos en registro.

Después de la Segunda Guerra Mundial, los avances científicos y las crecientes necesidades comerciales estimularon más los incrementos en los tipos y volúmenes de datos que debían procesarse. En 1953 IBM produjo su primera computadora electrónica: IBM 701, le siguió la 702 y en 1955, la 705. Estas máquinas y otras del mismo tipo llegaron a conocerse como computadores de *primera generación*. A fines de la década de los 50 y principios de los 60 aparecieron computadoras basadas en transistores, fueron las de *segunda generación*. A mediados y fines de la década de los 60 surgió la *tercera generación* de computadoras basadas en circuitos integrados.

Paralelo al crecimiento del equipo para procesar datos se dio el desarrollo de nuevos medios para almacenarlos. Con el tiempo, las fronteras entre las generaciones se volvieron imprecisas. Se habla de que la actual *cuarta generación* está basada en circuitos integrados a muy alta escala, las unidades de diskette son los principales dispositivos de almacenamiento utilizados en las microcomputadoras actuales.

Las facilidades de almacenamiento se han vuelto más veloces, confiables y capaces de empacar datos en forma cada vez más densa.

El software, es decir, los distintos tipos de instrucciones que hacen que las máquinas operen, también ha avanzado en forma continua.

Se necesitaron miles de años para que las personas cultivaran un interés en los tipos más simples de datos. Y sólo se han necesitado los últimos 30 años para que el procesamiento de datos llegue a un punto en que las personas hablen del fin de la *era industrial* y el nacimiento de la *era de la información*. Un aspecto crucial de esta última, que ya ha sido necesario empezar a resolver, es el manejo de enormes volúmenes de datos.

En el caso de los datos de gran volumen, donde cada uno de sus elementos es diferente de otros se tiene la característica de que la mayor parte se encuentra en constante cambio. Y en la actualidad se desea tener acceso (por lo general instantáneo) a los datos con uno u otro fin. Por otro lado, al tener lugar tantos desarrollos en el procesamiento de datos, resulta difícil llevar un control de aquellos datos que ya existen, por esta y otras causas éstos tienden a almacenarse varias veces para distintas aplicaciones.

Por fortuna, existe una metodología para almacenar, tener acceso y en general manejar datos que pueden formar los cimientos para sobrevivir en un mundo que se volverá cada vez más orientado a los datos. Esta clase de metodología o medio ambiente se ha llegado a conocer como BASES DE DATOS y constituye una solución ingeniosa a algunos problemas importantes, como los mencionados, que podrían parecer difíciles si se intentan abarcar de golpe, pero que son más comprensibles si se atacan con precaución, paso a paso y con cuidado, como lo permite un sistema de base de datos, conservando los datos en forma no redundante, y en diferentes secciones interconectados, lo que lleva a muchas más posibilidades de uso que si estuvieran almacenados por separado.

Los sistemas de manejo de bases de datos permiten el almacenamiento no redundante y el acceso a datos complejos.

La computadora se ha vuelto prácticamente una parte de todo proceso de negocios, y esta es una tendencia que seguramente continuará.

A continuación se hace una revisión del modelo relacional, en donde los datos se organizan en tablas. Cada una de estas tablas en realidad es un caso especial de la construcción conocida en matemáticas como Relación, este término es mas preciso que los tradicionales tales como *archivo* o *tabla.*

EL MODELO RELACIONAL

El Dr. E. F. Codd publicó en Computer World (septiembre 14 y 25, 1985), parte del producto de veinte años de trabajo, en investigaciones sobre el concepto de BD relacional para la IBM. Estas publicaciones tuvieron la intención de prevenir el abuso del concepto relacional de Sistemas de Bases de Datos Relacionales, y el concepto de este aquí se explica.

La aproximación relacional a los datos se fundamenta en el hecho de que los archivos que obedecen a ciertas restricciones se pueden considerar relaciones matemáticas y por lo tanto la teoría elemental de las relaciones se puede aplicar a varios problemas prácticos para datos de esos archivos.

Así, una tabla es una relación y los renglones de tales tablas se llaman ocurrencias, del mismo modo las columnas se suelen denominar atributos.

Un concepto, que la teoría relacional pone de relieve, y para el cual parece no haber un término establecido en procesamiento de datos, es el concepto de dominio.

Un dominio es un conjunto de valores del cual se sacan los que aparecen en una columna específica, las columnas basadas en este dominio pueden tener el mismo nombre o no. (aunque deben tener un nombre diferente si de otro modo resultara alguna ambigüedad).

Una característica del modelo relacional es que toda la información de la base de datos, tanto *entidades* «datos distinguibles que se tienen y cuyas propiedades o atributos asocian un valor, de un dominio de valores, con cada identidad y los atributos cuyos valores identifican unívocamente cada entidad, la cual se llamará llave» como *asociaciones* «donde una asociación es el elemento que vincula a uno o más conjuntos de entidades de acuerdo con un criterio definido», se representan de una sola manera ordenada en forma de tablas, a diferencia de otros modelos.

Muchos problemas se expresan con mayor naturalidad, no en términos de registros individuales, sino en términos de conjuntos.

Operaciones de Recuperación.

El resultado de cualquier operación de recuperación puede considerarse una tabla en donde un renglón duplicado o redundante se elimina del resultado final, la razón de esto es que en términos matemáticos una tabla o relación es un conjunto (de renglones) y por definición los conjuntos no pueden contener elementos duplicados.

Así se tiene que el resultado de cualquier recuperación es una tabla, por lo que el proceso de recuperación es un proceso de construcción de tablas. Entonces se puede definir un

conjunto de operadores de construcción de tablas, para usarlos en la recuperación.

Estructura del Modelo Relacional.

La base de datos que se construye por medio de relaciones es una **base de datos relacional**, a estas relaciones se les conoce como **tablas.**

Cuando una tabla tiene *n* columnas, se dice que la relación es de **grado***n*.

La manipulación de datos está basada en los conceptos de álgebra relacional y cálculo relacional (cálculo de predicados).

Relación.- Dada una serie de conjuntos D_1 ,D_2 ,...,D_n(no necesariamente distintos). Se dice que R es una relación sobre estos n conjuntos si es un conjunto de n ocurrencias ordenadas d_1 ,d_2 ,...,d_n) tal que $d_1 \in D_1$, $d_2 \in D_2$,...., $d_n \in D_n$. Los conjuntos D_1 ,D_2 ,...,D_n son los dominios de R. El valor de *n* es el grado de R.

Al número de ocurrencias de una relación se le llama **cardinalidad.**

El Producto Cartesiano puede definirse como: Dada una serie de conjuntos: D_1 ,D_2 ,...,D_n (no por fuerza distintos) el Producto Cartesiano de estos n conjuntos, denotado por: $D_1 \times D_2 \times D_3 \times ... \times D_n$, es el conjunto de todas la n ocurrencias ordenadas posibles d_1 ,d_2 ,....,d_n tales que $d_1 \in D_1$, $d_2 \in D_2$,....,$d_n \in D_n$ y se dice que R es una relación sobre los dominios D_1, D_2,...,D_n si es un subconjunto del producto Cartesiano $D_1 \times D_2 \times ... \times D_n$.

Un atributo representa el uso de un dominio dentro de una relación.

En cada intersección de un renglón y una columna de la tabla siempre hay exactamente un valor, nunca un conjunto de valores. Se permite la posibilidad de valores nulos, es decir, valores especiales que representan valores *desconocidos* o *inaplicables.* La relación que cumpla con esta condición se dice que está normalizada.(Sobre normalización se hablará más adelante)

LLAVES

Si dentro de una relación existen uno o varios atributos, cuyos valores son únicos dentro de la relación, y así se pueden usar para identificar las ocurrencias de esa relación, se les denominará *llave primaria.* No toda relación tendrá una llave primaria de un solo atributo, pero cada relación tendrá alguna combinación de atributos que tomados en conjunto, tienen la propiedad de la identificación única.

Dominio Primario.

Un determinado dominio se designa como primario si existe alguna llave primaria de

un solo atributo definida sobre ese dominio.

Se puede encontrar a veces una relación donde hay más de una combinación de atributos que poseen la propiedad de identificación única, es decir, la existencia de más de una llave candidata.

Una *llave candidata* que no es la llave primaria en una relación recibe el nombre de llave alterna.

Un atributo definido, de una Relación resultante es una*llave foránea*, ya que sus valores son valores de la llave primaria de una de las relaciones originales.

Las llaves primarias y foráneas proporcionan un medio para representar asociaciones entre ocurrencias, aunque tales atributos de *asociación* no siempre son llaves.

La *extensión* de una relación específica es el conjunto de ocurrencias que aparecen en esa relación en cualquier instante dado. Así pues la extensión cambia a medida que las ocurrencias son creadas, destruidas o actualizadas.

La *comprensión* de una relación específica es independiente del tiempo. Es decir es la parte permanente de la relación. Así pues la comprensión define todas las extensiones admisibles es decir, es la combinación de: una estructura que la define y un conjunto de restricciones de Integridad. La estructura que la define está compuesta por el nombre de la relación y los nombres de los atributos (los cuales van con el nombre de su respectivo dominio). Y las restricciones de Integridad se subdividen en restricciones de: llaves, referencia y otras restricciones.

Las *restricciones de llaves* aparecen por la existencia de llaves candidatas. La comprensión incluye una especificación de los atributos que constituyen la llave primaria y las especificaciones de los atributos que constituyen las llaves alternas, si es que las hay. Luego entonces, hay una *restricción de unicidad* dada por la definición de llave candidata y por la Primera Regla de Integridad existe también la *restricción de la llave primaria* que se refiere a la no existencia de valores nulos.

Las *restricciones de Referencia* aparecen por la existencia de llaves foráneas. La comprensión incluye la especificación de todas las llaves foráneas de la relación, y por la Regla de Integridad Referencial se da la *restricción de referencia*.

Planteado todo lo anterior se puede decir que una Base de Datos Relacional es una Base de datos que el usuario percibe como un conjunto de relaciones normalizadas, que varían con el tiempo, es decir, los operadores a disposición del usuario operan sobre estructuras relacionales. (Esto no significa que los datos estén almacenados físicamente en tablas).

Las relaciones pueden considerarse archivos altamente disciplinados, es decir:

a).- Cada "archivo" o relación, solo contiene un tipo de registro.

b).- Cada ocurrencia en un archivo o registro tiene el mismo número de campos o atributos, es decir, los grupos de repetición no están permitidos.

c).- Cada ocurrencia de registro tiene un identificador único.

d).- Dentro de un archivo o relación, las ocurrencias de registro tienen un orden no especificado.

Reglas de Integridad.

En referencia al Modelo Relacional, se conocen dos reglas generales de integridad, a saber:

- **Regla de Integridad de Identidad.**
- **Regla de Integridad Referencial.**

Son generales en el sentido de que se aplican a todas las bases de datos que se construyen de acuerdo al modelo relacional.

Regla de Integridad de Identidad

Ningún componente de un valor de una llave primaria puede ser nulo, ya que por definición todas las entidades deben ser distinguibles, es decir deben tener alguna identificación única de alguna clase.

En otras palabras, si el valor de alguna llave primaria fuera totalmente nulo significaría que hubo alguna entidad sin identificador único.

La justificación para esta regla es la siguiente:

a). Las relaciones corresponden a entidades en el mundo real.

b). Por definición, las entidades en el mundo real son distinguibles, esto es, tienen alguna identificación de algún tipo.

c). La llave primaria representa la identificación única en el modelo relacional. Entonces el valor de una llave primaria que sea nulo será una contradicción bajo estos términos, ya que entonces sería una entidad sin identificación, i,e., no existiría (de aquí el nombre de "integridad de identidad").

Cabe aclarar que por razones análogas a las anteriores los valores nulos parciales de llave primaria son prohibidos.

Regla de Integridad Referencial

Sea D un dominio primario. Si la Relación R2 incluye una llave foránea F_K haciendo pareja con la llave primaria P_K de alguna relación R1, entonces todos los valores de FK en R2, deben ser:

a). Iguales al valor de P_K en alguna tupla R1 (R1 y R2 no necesariamente distintos) con P_K definida sobre D.

b). Completamente nulos (i.e., cada valor de atributo participante en FK también es nulo). R1 y R2 no necesariamente distintos.

El intento básico de esta regla es simplemente que, si alguna tupla t_2 referencia a alguna tupla t_1, entonces t_1 existe. Esto es evidente, puesto que los valores de una llave foránea dan un valor en la relación referida (sí sus valores son no-nulos).

Hay que notar cuidadosamente que estas dos reglas están construidas en términos del planteamiento de la base de datos. Algún planteamiento de la base de datos que no satisfaga las dos reglas es por definición incorrecta.

12 Reglas de Codd

El Dr. E. F. Codd ideó el fundamento para la definición del Modelo Relacional *, y señala que un sistema puede ser llamado Relacional, si cumple por lo menos con 6 de las siguientes reglas que él mismo plantea:

1.- Todos los datos se almacenan en tablas.

2.- Todos los datos son accesibles a los usuarios por una combinación de tablas, usando los valores de llaves primarias y columnas.

3.- Los valores nulos son permitidos.

4.- La descripción de una base de datos puede ser vista como datos normales que pueden ser manipulados usando todas las herramientas disponibles en datos normales.

5.- El sistema de base de datos, soporta hasta el lenguaje más pequeño que pueda ser expresado en cadena de caracteres y provisto de un soporte que comprenda lo siguiente:
a). Definición de datos.
b). Vista de la definición.
c). Manipulación de datos (interactiva y por medio de programas).
d). Restricciones de integridad.
e). Autorización.
f). Parámetros de transacción (comienzo, acción y recorrido).

6.- Todas las vistas, donde una vista es una subrelación derivada de una relación, que son modificadas teóricamente, son modificadas por el sistema en su correspondiente relación.

7.- La manipulación de la base de datos es lógicamente atómica, i, e, no hay división.

8.- Los cambios en la presentación de almacenamiento o métodos de acceso para cualquier Base de Datos no deteriorarán el programa de aplicación.

9.- Los cambios en las tablas o relaciones que guardan información lógicamente no deteriorarán el programa de aplicación.

10.- Las restricciones de integridad forman parte de la base de datos.

11.- El administrador de base de datos del sistema, tiene independencia en la distribución de los datos.

12.- Si un sistema relacional tiene un lenguaje de bajo-nivel (por registro), no será capaz de burlar las restricciones de integridad.

* Publicado en Computer World, 1985

ALGEBRA RELACIONAL

El álgebra relacional se define como un conjunto de operaciones sobre las relaciones, en donde cada operación toma una o más relaciones como sus operandos y produce otra relación como resultado.

Las relaciones se representan en forma de tablas, en donde cada renglón de la tabla representa una n-ocurrencia de la relación: el número de correspondencias de éstas se llama *cardinalidad* de una relación.

Las relaciones a las que se aplican las operaciones de teoría de conjuntos, deben estar contenidas en la relación universo, la cual se conforma con todas las combinaciones posibles de los valores de los atributos (dominios).

Dentro del álgebra relacional hay dos grupos de operadores, a saber: 1.) Operador de la Teoría de Conjuntos y 2.) Operadores relacionales especiales.

1. Operadores de la teoría de conjuntos:unión, intersección, diferencia y producto cartesiano, los cuales son modificados para operar con atributos de las relaciones y no con conjuntos arbitrarios.

   Para todos estos operadores excepto producto cartesiano, las dos relaciones operando deben ser compatibles con la unión, es decir, deben tener el mismo grado.

Unión.- La unión de R1 y R2, se forma con la combinación de todas la n-ocurrencias que pertenecen a cada uno de las relaciones, y se cuenta solamente una vez las n-ocurrencias que estén tanto en R1 como en R2, y se representa por el símbolo $\cup$.

Entonces tenemos que: Sí $R \in R1 \cup R2 \Rightarrow R \in R1$ ó $R \in R2$, es decir una n-ocurrencia pertenece a la unión de dos relaciones, sí está en cualquiera de las dos o en ambas, antes de efectuar la operación unión.

Intersección.- La intersección de dos relaciones R1 y R2 que estén contenidas en la relación universo, se forma con las n-ocurrencias comunes de ambas relaciones, es decir: $R \in R1 \cap R2 \Rightarrow R \in R1$ y R2. Así entonces R esta en la intersección de R1 y R2, cuando la n-ocurrencia R se encuentre en ambas relaciones R1 y R2.

Diferencia.- La diferencia de las relaciones R1 y R2 (R1 - R2) se forma por todas las n-ocurrencias de R1 quitando las n-ocurrencias de R1 que se encuentren en R2, es decir: $R \in R1 - R2 \Rightarrow R \in R1$ y $R \in R2$.

Producto Cartesiano.- El producto cartesiano de dos relaciones P y Q representado por P x Q, es el conjunto de todas las n-ocurrencias r, tales que r es la concatenación de la n-ocurrencia $p \in P$ y la n-ocurrencia $q \in Q$.

Ahora bien, la concatenación de una n-ocurrencia $p = (p_1, p_2, ..., p_m)$ y una n-ocurrencia $q = (q_{m+1}, q_{m+2}, ..., q_{m+n})$ en ese orden es la n-ocurrencia:
$r = (p_1, p_2, p_3, ..., p_m, q_{m+1}, q_{m+2}, ..., q_{m+n})$.

2.- Operadores relacionales especiales: selección, proyección y reunión.

Selección.- Examina todas las n-ocurrencias de una relación R y saca solamente aquellas en que su valor de atributo coincida con el valor del atributo que fue proporcionado por el usuario, entonces: SELECT R (A = x) = Rx, nos representa todas las n-ocurrencias para las cuales el atributo A=x de la relación R, y la relación resultante se denota Rx.

En otras palabras, el operador SELECT construye una nueva relación al tomar un *subconjunto horizontal* de una relación existente es decir, todos los renglones de una tabla existente que satisfaga cierta condición.

Ejemplo:
Teniendo los registros "cédula" y "custodio", que es un control de bienes arqueológicos muebles, obtenemos la selección.

Cédula donde Cultura = Otomí, que resulta

Cédula	Pieza	Tipo	Cultura
48	PJ 15	vasija	Otomí
23	PJ 2	cajete	Otomí

Proyección.- Actúa sobre una relación dada a fin de reducir el número de atributos que ésta contiene eliminando los duplicados y creando una nueva relación (subrelación) de grado menor, es decir el operador PROYECT forma un subconjunto vertical al extraer determinadas columnas y suprimir renglones duplicados.

Ejemplo:
Del registro "Cédula" queremos la proyección sobre pieza, cultura y tipo.

Cédula [Pieza, Cultura, Tipo]

Pieza	Cultura	Tipo
1	Golfo	cajete
2	Altiplano Central	comal
3	Mexica	collar
4	Golfo	cajete

la proyección sobre Cultura será:

Cultura
Golfo
Altiplano Central
Mexica

Reunión.- Agrupa simultáneamente dos o más relaciones, formando así una nueva relación. Dicho de esta manera: Teniendo dos tablas, si cada una tiene una columna definida sobre algún dominio común se puede reunir sobre esas dos columnas y el resultado será una nueva tabla, más ancha, donde cada renglón se forma concatenando dos renglones, uno de cada tabla original. Si un renglón en una de las tablas originales no tiene una contraparte en la otra, no participa en el resultado. La condición de reunir puede fundamentarse en: igualdad, mayor que, no igual, etc. La más utilizada es la de igualdad y se conoce como equireunión, que contiene por fuerza dos columnas iguales, una de las cuales puede ser eliminada con la operación proyección. Se dice que hay una reunión natural cuando se tiene una equireunión en donde ya fue eliminada una de las columnas duplicadas. La reunión natural es importante en el contexto de las normalizaciones.

Ejemplo:
Se pretende la Reunión de las Relaciones "Cédula" y "Custodio", combinando proyección se obtiene la equirreunión, es decir se eliminan columnas duplicadas.

((Cédula REUNION Custodio) donde tipo = 'vaso de obsidiana')

Cédula	Pieza	Cultura	Tipo	Custodio
48	PJ 28	Tolteca	vaso obsidiana	C.R.Noreste
27	PJ 15	Tolteca	vaso obsidiana	C.R.Noreste
16	PJ 13	Occidente	vaso obsidiana	C.R.Occidente
15	PJ 1	Teotihuacana	vaso obsidiana	C.R.Occidente
15	PJ 2	Teotihuacana	vaso obsidiana	Museo Templo Mayor

CALCULO RELACIONAL

Este sistema de cálculo, ideado por E.F. Codd*, para bases de datos relacionales, fue creado con la finalidad de que el usuario únicamente defina el resultado que requiera, y deje que el sistema decida las operaciones que necesite para la obtención de la respuesta a partir de la Base de Datos**. La definición de los operadores que aquí se utilizan están basados en la noción de:

Variable libre .- donde la variable es análoga al de una variable global en un lenguaje de programación, es decir, es una variable definida fuera del procedimiento actual.

Variable acotada .- donde la variable acotada es semejante a una variable local que se define en el procedimiento actual.

Variable de tupla.- es una variable que "varía sobre" alguna relación, es decir, si la variable de la n-ocurrencia de T varía sobre la relación R, entonces en cualquier instante dado T representa alguna n-ocurrencia individual de R.

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

- Variables de ocurrencias T,U,V... Cada variable de n-ocurrencia se restringe a variar sobre alguna relación con nombre. Si la variable T representa a la ocurrencia t (en algún instante dado) entonces la expresión T.A representa a la componente A de t (en ese instante), donde A es un atributo de la relación sobre la cual varía T.

- Condiciones de la forma $x*y$ donde * es cualquiera de los símbolos: $=, <, \leq, >, \geq$, y al menos una de las dos $(x$ e $y)$ es una expresión de la forma T.A y la otra es una expresión semejante o una constante.

- Fórmulas Bien Formadas. Estas se construyen a partir de condiciones, operadores booleanos (AND, OR, NOT) y cuantificadores ($\forall$, $\exists$) de acuerdo con las siguientes reglas:

1.- Toda condición es una Fórmula Bien Formada (FBF).

2.- Si f es una FBF entonces también lo son (f) y NOT(f).

3.- Si f y g son FBFs, también lo son (f AND g) y (f OR g).

* E.F.Codd, "Relational Completeness of Data Base Sublanguages". En: *Data Base Systems*. Courant Computer Science Symposia Series, Vol. 6, Englewood Cliffs, New Jersey: Prentice Dall, 1972.

** Cabe aclarar que el objetivo de esta tesis no es el de profundizar en el cálculo relacional, por lo que sólo se hará mención a sus elementos fundamentales.

4.- Si *f* es una FBF en la cual T aparece como variable libre entonces $\exists$ T(*f*) y $\supset$ T(*f*) son FBFs.

5.- Ninguna otra es una Fórmula Bien Formada (FBF).

Variables libres y acotadas.- Cada ocurrencia de una variable de n-ocurrencia dentro de una FBF es libre o acotada, en donde la "ocurrencia" es una aparición del nombre de la variable dentro de la hilera de símbolos que constituye la FBF que se considera. Una variable de tupla ocurre dentro de una FBF en el contexto de una expresión de la forma: T.A, o como la variable que sigue a uno de los símbolos de cuantificación $\supset$, $\exists$., así tenemos que:

1.- Dentro de una condición, todas las ocurrencias de las variables de n-ocurrencia son libres.

2.- Las ocurrencias de las variables de n-ocurrencia en las FBFs (*f*), NOT (*f*) son libres o acotadas según sean libres o acotadas en f. Las ocurrencias de las variables de ocurrencias en las FBFs (*f* AND *g*), (*f* OR *g*) son libres o acotadas según sean libres o acotadas en *f* o en *g* (cualquiera de las dos en donde aparezcan).

3.- Las ocurrencias de T que sean libres en *f* son acotadas en las FBFs $\exists$ T(*f*), $\supset$ T(*f*). Otras ocurrencias de variables de n-ocurrencias en *f* son libres o acotadas en estas FBFs según sean libres o acotadas en *f*.

Cálculo Relacional orientado a los dominios.- Este difiere del cálculo de ocurrencias en que toman valores sobre dominios en vez de relaciones, sus expresiones se construyen a partir de los siguientes elementos:

- Variables de dominio D,E,F,... cada variable se limita a variar sobre algún dominio especificado, cuyos valores son elementos de los dominios y no dominios.

- Condiciones, que pueden adoptar dos formas:

a). Comparaciones simples de la forma $x*y$, donde x e y son variables de dominio (o constantes).

b). Condiciones de pertenencia, de la forma R (término, término,...) donde R es una relación y cada "termino" es un par A:V donde A es un atributo de R y V es una variable de dominio o una constante.

- *Fórmulas Bien Formadas*.- las mismas de variable de n-ocurrencia (pero con la definición modificada de "condición").

Las reglas referentes a variables libres y acotadas dadas, para las variables de n-ocurrencia, se aplican también al cálculo de dominio.

Entonces vemos que mientras el álgebra relacional es un conjunto de *operaciones* sobre las relaciones, el cálculo relacional es un *cálculo* de *predicados* (operadores) aplicado y propio de las bases de datos relacionales.

NORMALIZACION

Para una explotación adecuada del Modelo Relacional es de suma importancia el diseño de base de datos, en este punto debe considerarse un aspecto fundamental alrededor de lo siguiente: ¿Cómo se decide qué relaciones se necesitan y qué atributos deben de tener?

Así entonces podríamos decir que un buen criterio de diseño sea el de "cada cosa en su lugar".

Ahora bien, aunque el diseño esté normalizado, no solamente se debe apoyar el diseño sobre la normalización, en otras palabras el asunto fundamental es que una relación dada, aunque esté normalizada puede poseer ciertas propiedades indeseables, y la teoría de la normalización permite reconocer tales casos e indica cómo tales relaciones se pueden convertir en una forma más deseable.

Y es preciso puntualizar que el modelo relacional solo se admiten relaciones normalizadas porque:

1.- La selección no impone ninguna restricción real sobre lo que puede representarse.

2.- La simplificación resultante en la estructura de los datos conduce a simplificaciones en otras áreas, en particular en los operadores del DML (lenguaje de manipulación de datos).

Al normalizar una base de datos, lo que se busca es que los datos no contengan redundancia entre los campos no llave, ahora bien, los datos normalizados, sujetos a posibles modificaciones estructurales por razones de desempeño, son el diseño final para las bases de datos relacionales. En realidad el proceso de normalización asegura que los campos de unión aparecerán en todas las tablas en que deban hacerlo con base en la relaciones entre los campos.

La entrada requerida por el proceso de normalización es una lista del conjunto de campos de datos y de las asociaciones entre ellos.

Tenemos que los objetivos de la normalización son:

1.- Hacer posible la representación de cualquier relación en una base de datos.

2.- Obtener algoritmos poderosos de recorrido de la base de datos sobre un conjunto muy simple de operaciones relacionales.

3.- Liberar a las relaciones de dependencias indeseables que provoquen inserciones, actualizaciones y eliminaciones.

4.- Reducir la necesidad de reestructurar las relaciones cuando se introducen nuevos tipos de datos.

5.- Hacer el conjunto de relaciones neutral a las estadísticas de consulta, donde esas estadísticas están sujetas a cambios con respecto al tiempo.

Antes de ver las formas de normalización, veamos los tipos de dependencia de datos con los que podemos encontrarnos en el proceso de normalización y que deberemos de considerar.

Dependencia Funcional.

Dada una relación R, el atributo Y de R es funcionalmente dependiente del atributo X en R si tiene asociado a él exactamente un valor de Y en R (en cualquier instante). Aquí no existe el requisito de que un valor dado de X aparezca solo en una tupla de R. Así entonces hay una definición alternativa:

Dada una relación R, el atributo Y de R es funcionalmente dependiente del atributo X de R sí y solo sí siempre que dos ocurrencias de R coincidan en sus valores de X, también coinciden en sus valores de Y.

Una dependencia funcional (DF) es una forma especial de la Restricción de Integridad.

Propiedades de las Dependencias Funcionales.

Proyectividad	Si $Y \subset X, \Rightarrow X \rightarrow Y$
Aditividad	Si $X \rightarrow Y$ y $X \rightarrow Z \Rightarrow X \rightarrow Y \cup Z$
Transitividad	Si $X \rightarrow Y$ y $Y \rightarrow Z \Rightarrow X \cup Z \rightarrow Y$
Ampliación	Si $X \rightarrow Y$ y $Z \subset Y \Rightarrow X \cup U\ Z \rightarrow Y$
Descomposición	Si $X \rightarrow Y$ y $Z \subset Y \Rightarrow X \rightarrow Z$

Dependencia Parcial.-

Sean $f : A_1, A_2, \ldots, A_n \rightarrow B$

$g : A_1, A_2, \ldots, A_n \rightarrow B$ con $m < n$

tales que

$f(a_1, a_2, \ldots, a_n) = g(a_1, a_2, \ldots, a_n)$

tal que para $a_i \in A$, $i=1,\ldots,m$

en este caso se dice que B es parcialmente dependiente de $A_1, A_2, \ldots, A_n$: (esto es, que los atributos $A_{m+1}, \ldots, A_n$ sobran para determinar funcionalmente a B).

Si no existe g con esa propiedad, se dice que B es totalmente dependiente de $A_1, A_2, \ldots, A_n$.

Dependencia Funcional Completa - El atributo Y es funcionalmente dependiente, en forma completa del atributo X_j, si es funcionalmente dependiente de X y no depende funcionalmente de ningún subconjunto propio de X, es decir, no existe un subconjunto

propio X' de los atributos que constituyen a X tales que Y sea funcionalmente dependiente de X'.

Ahora bien, si Y es funcionalmente dependiente de X, pero no en forma completa, entonces X debe ser compuesto.

El reconocimiento de las dependencias funcionales es importante para la comprensión del significado o semántica de los datos.

Descomposición sin pérdidas.- Dada una relación R con los posibles atributos compuestos A,B, y C que satisfagan la DF R.A → R.B., R se puede "descomponer sin pérdidas" en sus proyecciones R1(A.B) y R2(A.C). Como no se pierde ninguna información que se pueda derivar de la estructura original también se puede derivar de la estructura nueva. Sin embargo, la recíproca no es verdadera: la estructura nueva puede contener información que no se podía representar en la original. Por tanto, la estructura nueva es algo mas fiel al mundo real.

Entonces:

Sea R una relación con conjuntos de atributos

$a = a_1 \cup a_2 \cup ... \cup a_n = (\ <a_1>, <a_2>,....,<a_n>)$

$= R\ [a_1\ ],\ R[a_2\ ],.....,R[a_n\ ]$ es una descomposición sin pérdidas de R si:

$R[a1\ ] * R[a2\ ] *...* R[a_n\ ] = R.$

Teorema de la Descomposición.-

Sea R una relación con conjuntos de atributos a.

Sean X, $Y \in a$ y sea $Z = a - (X \cup Y)$

Si $X \rightarrow Y, \Rightarrow R1 = R\ [X \cup\ Y]$ y $R2 = R\ [X \cup Z]$

La obtención de R1 y R2 en vez de R constituye una descomposición sin pérdidas.

Aunque, una nueva reunión de las proyecciones puede hacer que la relación R reaparezca conjuntamente con algunas ocurrencias adicionales "espúrias". No puede generar jamás nada menor que R.

Una descomposición sin pérdidas garantiza que la reunión genera exactamente la relación original R. Una descomposición que no sea sin pérdidas pierde información en el sentido de que la reunión puede generar un subconjunto de la relación original R, y no hay manera de saber cuales ocurrencias del superconjunto son espúrias o genuinas.

Dependencia Multivaluada.- Las dependencias multivaluadas DMVs son una generalización de las dependencias funcionales (es decir, una DF es un caso especial de una DMV).

Dada una relación R con atributos A,B y C, la dependencia multivaluada $R.A \rightarrow\!\!\!\rightarrow R.B$,

se cumple en R si y sólo si el conjunto de valores de B que corresponden a un par (valor de A, valor de C) dado en R depende tan solo del valor de A y es independiente del valor de C.

Definidas de este modo las dependencias multivaluadas, solo pueden existir si la relación R tiene al menos tres atributos.

Dada la relación R(A,B,C), la DMV R.A $\rightarrow\!\!\rightarrow$ R.B se cumple sí y sólo sí también se cumple la DMV R.A $\rightarrow\!\!\rightarrow$ R.C. Las dependencias multivaluadas siempre van en parejas de esta manera. Y entonces se expresan en una sola proposición, usando la notación: R.A $\rightarrow\!\!\rightarrow$ R.B | R.C

Teorema (Fagin).- La relación R, con atributos A,B y C, se pueden descomponer sin pérdidas en sus dos proyecciones R1 (A,B) y R2 (A,C) sí y sólo sí la dependencia multivaluada A $\rightarrow\!\!\rightarrow$ B | C se cumple en R.

Dependencia de Reunión.- Se puede ver que existen relaciones que no se pueden descomponer sin pérdidas en dos proyecciones, pero sí en tres o más. Este fenómeno lo observaron por primera vez Aho Beeri y Ullman*.

Si se tiene que ABC es la reunión de sus tres proyecciones: AB, BC y CA, es equivalente a decir que:

si la pareja a_1, b_1 aparece en AB,
la pareja b_1, c_1 aparece en BC
y la pareja a_1, c_1 aparece en CA
entonces la tercia a_1, b_1, c_1 aparece en ABC

ya que ésta aparece en la reunión de AB, BC y CA. Puesto que a_1, b_1 aparece en AB sí y sólo sí aparecen juntos en ABC, esto es válido también para b_1, c_1 y , a_1. Esta proposición es la siguiente restricción sobre ABC:

Sí $a_1, b_1, c_2, a_2, b_1, c_1, a_1, b_2, c_1$ aparecen en ABC, entonces $a1, b1, c1$ también aparecen en ABC.

Esta restricción aunque raramente se da, se satisface sí y sólo sí, la relación que se trabaja es la de alguna de sus proyecciones, y se le denomina dependencia de reunión *(DR). En general, la relación R satisface la DR (X, Y, ..., Z) Sí y sólo sí, es la reunión de sus proyecciones sobre (X, Y, ..., Z), donde éstas son subconjuntos del conjunto de atributos de R.*

Ya vistos los tipos de dependencia, es posible entender mejor el proceso de normalización, que da una metodología para el diseño de bases de datos no redundantes, fáciles de manejar y sin pérdida de información.

* Beeri y Ullman, publicado en "The Theory of Joins in Relacional Database" ACM Transaction on Database Systems 4, num. 3, september, 1979.

Primera Forma Normal. 1FN

Los datos en primera forma normal tienen la propiedad de que cada anotación de datos, o valor de campo, debe ser indivisible (atómica).

Cada anotación de campo de cada registro está constituido por un solo elemento de datos no subdivisibles.

La representación de los datos en la primera forma normal no es por si misma, útil como un arreglo para el control de la redundancia, sino solo el primer paso para seguir normalizando.

Esencialmente, se requiere que todo atributo de la relación este basado sobre dominios simples, es decir una relación esta en primera forma normal si todo atributo de la relación está basado sobre un dominio simple.

Ejemplo:

En el registro de monumentos arqueológicos muebles se manejan piezas que son presdtdas a diferentes custodios, con el siguiente esquema:

R (No. pieza, tipo, cultura, región custodio)

Custodio (No. custodio, nombre custodio)

El esquema no está en IFN porque el atributo Custodio, no está definido en dominios simples ya que él mismo es una relación.

Asf entonces se define una nueva relación R' que incorpore los componentes del atributo custodio en la relación R, obteniendo el siguiente esquema, que ya está en IFN:

R'(No. pieza, No. Custodio, Nom. Custodio, tipo, cultura, región)

R'	No.Pieza	No.Custodio	Nom.Custodio	Tipo	Cultura	Región

Segunda Forma Normal. 2FN

Después de poner los datos en 1FN, es posible que sigan habiendo datos muy redundantes, en este punto la metodología se orienta al problema de qué buscar en la estructura de datos y qué modificar para reducir la redundancia.

El diseño se dividirá en más tablas, cada una de las cuales tiene la propiedad de que su llave entera se necesita para definir cada uno de sus campos no llave.

Dentro de una llave específica, ningún campo no llave está definido por una parte de la llave solamente.

Varios campos se duplican en este proceso, que solo aparecían una vez como campo en la 1FN. Pero este tipo de duplicación de campo es necesario en esta etapa entre aquellos campos que participan como campos llave.

La reducción consiste en reemplazar las relaciones por proyecciones adecuadas; el conjunto de estas proyecciones es equivalente a la relación original, en el sentido de que la relación original se puede recuperar siempre formando la reunión natural de estas proyecciones, de manera que ninguna información se pierda en el proceso.

Continuando con el ejemplo anterior, si se relacionan los conjuntos de atributos que provocan la dependencia parcial, se puede aplicar el teorema de la descomposición como sigue:

X={No.Pieza} Y={Tipo, Cultura, Región}

y como $X \subseteq K$ = {No.Pieza, No. Custodio, Nom. Custodio}

tenemos

$R_1 = R[X \cup Y] = R$[No.Pieza, Tipo, Cultura, Región]

$R_2 = R[X \cup Z] = R$[No.Pieza, No.Custodio, Nom.Custodio]

Por elTeorema:

$R_1 * R_2 = R \Rightarrow R_1, R_2$ están en 2FN, es decir ahora tenemos:

R_1	No.Pieza	Tipo	Cultura	Región
R_2	No.Pieza	No.Custodio	Nom.Custodio	

Tercera Forma Normal. 3FN

Una relación esta en tercera forma normal si y solo si esta en segunda forma normal, y todo atributo no primo es dependiente no transitivamente de la llave primaria.

Forma Normal de Boyce /Codd. FNBC

La definición de FNBC es conceptualmente más sencilla que la de la 3FN, en el sentido de que no hace ninguna referencia explícita a la 1FN y 2FN como tales, ni a los conceptos de dependencia completa y transitiva.

Se acuerda llamar determinante (funcional) a un atributo, tal vez compuesto, del cual depende funcionalmente en forma completa algún otro atributo.

Entonces se define a FNBC como:

Una relación R esta en FNBC si cada determinante es una llave candidata. Para esta forma se habla en términos de llaves candidatas, no solo de llaves primarias. Como en 3FN no se maneja satisfactoriamente el caso de una relación que posea dos o más llaves candidatas compuestas y traslapadas y aunque la FNBC es más restrictiva que la 3FN, sigue siendo cierto que cualquier relación se puede descomponer sin pérdidas en un conjunto equivalente de relaciones en FNBC.

Dos llaves candidatas se traslapan si comprenden dos o más atributos cada una y si tienen algún atributo en común.

Del esquema en 2FN vemos que no está en 3FN porque una de sus relaciones no lo está, a saber R_2 porque:

No.Custodio→→Nom.Custodio

Aplicando el Teorema de la Descomposición

R_2(No.Pieza, No.Custodio, Nom.Custodio)

obtenemos:

R_3(No.Pieza, No.Custodio) y R_4(No.Custodio, Nom.Custodio)

que constituyen una descomposición sin pérdidas de R_2, además R_3 y R_4 están en 3FN y dado que R_1 ya lo estaba, el modelo formado por R_1, R_3 y R_4 queda constituído por relaciones en 3FN y FN/BC, ya que todos los atributos no primos dependen completamente de cada llave:

R_1	No.Pieza	Tipo	Cultura	Región
R_3	No.Pieza	No.Custodio	} Descomposición sin	
R_4	No.Custodio	Nom.Custodio	} pérdidas de R_2	

Cuarta Forma Normal 4FN.

Una relación R está en cuarta forma normal 4FN si y solo si, siempre que exista una DMV en R.(por ejemplo A→→B) todos los atributos de R también son funcionalmente dependientes de A (es decir A→x para todos los atributos x de R.).

En otras palabras, las únicas dependencias (DFs y DMVs) en R son de la forma K→x (esto es, una dependencia funcional de una llave candidata K a algún otro atributo x).

Ahora bien, la 4FN es estrictamente más fuerte que la FNBC, es decir cualquier relación puede descomponerse sin pérdidas en un conjunto equivalente de relaciones en 4FN.

Ejemplo:

Para las visitas guiadas en un Museo, supóngase que se dá una relación *no normalizada* que contiene la información acerca de visita, guía, sala. Cada registro en la relación se compone de un idioma, más un grupo de repetición de nombres de guías, más un grupo de repetición de salas, según la siguiente figura:

VGS

Visita	Guía	Sala
Español	g1	mexica
	g2	Golfo
	g3	
Inglés	g4	maya
		Sureste

El significado de un registro específico en esta relación es que la visita indicada la puede dar cualquiera de los guías indicado y en ella se utilizan todas las salas indicadas.

Se supone que, para una visita dada, puede existir cualquier número de guías correspondientes y cualquier número de salas correspondientes; además suponemos que las guías y las salas son independientes entre sí (es decir, independientemente de quien dé la visita es un idioma dado, se utilizan las mismas salas). También supondremos que un guía o sala específica puede estar asociado con cualquier número de visitas.

Para convertir esta estructura a una forma normalizada equivalente vemos que:

a) En los datos no existen dependencias funcionales por lo tanto no se puede descomponer la estructura en proyecciones, por lo que la única operación normalizante sería la de "aplanar" la estructura resultando algo como:

VGS

Visita	Guía	Sala
Español	g1	Mexica
Español	g1	Golfo
Español	g2	Mexica
Español	g2	Golfo
Español	g3	Mexica
Español	g3	Golfo
Inglés	g4	Maya
Inglés	g4	Sureste

b) El significado de la relación normalizada VGS es la siguiente: una ocurrencia <v,g,s>, aparece en VGS sí y solo sí la visita v la da el guía g y usa la sala s como referencia. Notemos que, para una visita dada aparecen todas las combinaciones posibles de guía y sala, es decir VGS satisface la restricción.

Si las dos ocurrencias <v,g1,s1> y <v,g2,s2> aparecen, entonces las ocurrencias <v,g1,s2> y <v,g2, s1también aparecen.

Claramente VGS contiene mucha redundancia, lo que ocasiona problemas con las operaciones de actualización. Sin embargo VGS está en FNBC porque es "toda llave" y no hay determinantes funcionales.

En VGS, las dificultades se deben al hecho de que los guías y las salas son independientes entre sí, esta situación mejoraría si VGS se reemplazara por sus dos proyecciones VG (visita, guía) y VS (visita, sala), ya que VG y VS son "solo llave" y por lo tanto ambos están en FNBC, de la siguiente forma:

VG

Visita	Guía
Español	g1
Español	g2
Español	g3
Español	g4

VS

Visita	Sala
Español	Mexica
Español	Golfo
Inglés	Maya
Inglés	Sureste

Esta forma resulta de las proyecciones obtenidas por las Dependencias Multivaluadas:

VGS.Idioma→→VGS.Guía

VGS.Idioma→→VGS.Sala

Se vé pues que VGS no está en 4FN, porque comprende una DMV que no es una DF, sin embargo las proyecciones VG y VS están en 4FN, pues con ella se eliminó la redundancia por proyecciones VG y VS.

Quinta Forma Normal. 5FN

En 4FN se ha supuesto de modo implícito que la sola operación necesaria o utilizable en el proceso de descomposición es el reemplazo de una relación por sus dos proyecciones.

Pero existen relaciones que no se pueden descomponer sin pérdidas en 2 proyecciones, pero que se puede descomponer sin pérdidas en 3 (o más). Así entonces es posible y tal vez deseable descomponer una relación en componentes más pequeñas a saber, en las proyecciones especificadas por la dependencia de reunión.

Y se dice que una relación R esta en 5FN *(Forma normal de Proyección-Reunión* (FN/PR) sí y solo sí toda dependencia de reunión en R está implicada por las llaves candidatas de R.

Ejemplo:
Considérese la relación ABC como sigue:

ABC

A	B	C
a1	b1	c2
a1	b2	c1
a2	b1	c1
a1	b1	c1

Esta relación es "toda llave" y no comprende DF_s ni DMV_s y por tanto están en 4FN.

Si descomponemos ABC en sus tres proyecciones tenemos:

AB

A	B
a1	b
a1	b2
a2	b1

BC

B	C
b1	c2
b2	c1
b1	c1

CA

C	A
c2	a1
c1	a1
c1	a2

El efecto de reunir AB y BC sobre B genera una copia de ABC original mas un renglón espurio, a saber:

A	B	C	
a1	b1	c2	
a1	b1	c1	
a1	b2	c1	
a2	b1	c2	← espurio
a2	b1	c1	

Al reunir AB y BC sobre B con CA sobre ella misma se elimina el renglón espurio obteniéndose ABC original.

ABC

A	B	C
a1	b1	c2
a1	b1	c1
a1	b2	c1
a2	b1	c1

Se observa que ABC, con su dependencia de reunión (AB, BC, y CA) se puede descomponer en tres. La pregunta sería ¿Debe ser descompuesta en tres? La respuesta sería tal vez sí, pues las operaciones de actualización para ABC causan problemas que se eliminan cuando es "descompuesta en tres".

La 5FN es la última forma normal con respecto a la proyección y la reunión. Ya que si una relación está en 5FN, las únicas descomposiciones válidas son las que se basan en las llaves candidatas (de tal forma que cada proyección se compone de una o más llaves candidatas junto con cero o más atributos distintos).Así entonces la que se da es una técnica de descomposición sin pérdidas.

La idea básica de la metodología de diseño es que se comience con alguna relación dada, junto con una declaración de ciertas restricciones (DFs, DMVs, DRs) se reduzca en forma sistemática esa relación a un conjunto de relaciones que sean equivalentes a la original, pero en alguna forma preferibles a ella, usando las restricciones para guiarse en el proceso de reducción.

Entonces se tiene que:

- Tomar proyecciones de la relación original en 1FN para eliminar cualesquiera DFs NO COMPLETAS. Esto generará un conjunto de relaciones en 2FN.
- Tomar proyecciones de esta relaciones en 2FN para eliminar cualesquiera

DEPENDENCIAS TRANSITIVAS. esto generará un conjunto de relaciones en 3FN.

- Tomar proyecciones de estas relaciones en 3FN para eliminar cualesquiera DEPENDENCIAS FUNCIONALES donde el DETERMINANTE NO SEA UNA LLAVE CANDIDATA, esto generará un conjunto de relaciones en FNBC.

- Tomar proyecciones de estas relaciones en FNBC para eliminar cualesquiera DEPENDENCIAS MULTIVALUADAS que no sean dependencias funcionales, generará un conjunto de relaciones en 4FN. (aunque en la práctica se suelen eliminar esta DMVs antes de aplicar los pasos anteriores).

- Tomar proyecciones de éstas relaciones en 4FN para eliminar cualesquiera DEPENDENCIAS DE REUNION que no sean implicadas por las llaves candidatas (sí las hay), generará un conjunto de relaciones en 5FN.

El objetivo general del proceso de normalización es reducir la redundancia, y por lo tanto, evitar ciertos problemas con las operaciones de actualización.

Aunque se debe subrayar que los pasos de normalización tan solo son pautas, algunas veces hay buenas razones para no normalizar "hasta el fin".

PARTE II

SQL (Lenguaje Estructurado de Consulta)

SQL se originó de un proyecto de la IBM de San José Research Laboratory, dirigido por E.F. Codd en 1969 y 1970, del cual se publicaron documentos que introducían al concepto de bases de datos relacionales, como SQUARE y SEQUEL, lenguajes de interfaz para una base de datos relacional, dado a conocer como sistema R y que apareció a mitad de los 70's. SEQUEL se desarrolló hasta llegar a ser SQL a fines de los 70's, utilizándose como lenguaje en DB2 (primera base de datos relacional de IBM), naturalmente se instrumentó para máquinas IBM, pero rápidamente se adaptó para microcomputadora DEC por Oracle Corporation & Tecnology Relational. Desde entonces SQL ha tenido aplicaciones en diferentes computadoras como un lenguaje de interfaz de base de datos relacionales, y en los últimos 3 años ha sido aplicado también en infinidad de microcomputadoras.

En febrero de 1987, la American National Standards Institute (ANSI) adoptó SQL como un lenguaje estándar para el acceso a los sistemas manejadores de bases de datos relacionales, posteriormente la International Standars Organization (ISO) hizo lo mismo, lo cual ha sido reconocido por muchas de las compañías dedicadas a la elaboración de software creando productos basados en éste.

SQL es un lenguaje "no-procedural" diseñado para la creación, mantenimiento y manipulación de una base de datos relacional.

Este lenguaje se fundamenta en la teoría matemática de conjuntos, que se muestra por sí misma en la sintaxis del lenguaje e implementa el Algebra y el Cálculo Relacional. SQL expone y permite la fácil manipulación de datos en grandes grupos así como también para elementos individuales, por lo que dependerá de la destreza que el usuario desarrolle, para obtener resultados óptimos, al aplicar este lenguaje.

En cuanto a su tecnología, SQL ha desarrollado terminología simple y accesible, que facilitan al usuario su manejo y el entendimiento de ideas del modelo de conjuntos en el cual éste lenguaje se basa. Un archivo se concibe como un arreglo bidimensional formado por renglones y columnas. Los archivos serán referidos como tablas, donde los renglones son datos individuales y serán referidos también como ocurrencias. Las columnas de una tabla son datos individuales contenidos en un registro y serán llamados campos o atributos.

Entre las grandes ventajas que SQL ofrece al usuario, se tienen la consulta de datos que en cualquier momento se requiera y la capacidad para realizar reportes. Es decir, un usuario de SQL puede formular eficientemente análisis sofisticado de sus datos, así como obtener su información en múltiples tablas a la medida; también está presente la capacidad para reducir los datos por grupo y conjuntos de operaciones, además de

cálculos estadísticos simples. Para obtener reportes por columnas que generan múltiples tablas con diversos tipos de ordenación de SQL, definitivamente, dependerá del conocimiento o destreza del usuario y será muy fácil para quien tiene inclinación matemática.

Los usuarios familiarizados con manejo de bases de datos relacionales, se sentirán rápidamente como en su casa al trabajar el conjunto de modelos o ejemplos en los que se basa esta tecnología. Muchos sistemas usan una sintaxis idéntica a la instrumentada en DB2, las principales diferencias son las opciones y comandos que proveen una habilidad adicional, mejor dicho variaciones en la sintaxis. Esta similitud permite a los usuarios moverse más fácilmente de un sistema a otro. Una persona que ha sido entrenada para utilizar el SQL en una computadora personal, no tiene que ser instruida para utilizarlo en una supermicro o una minicomputadora. Esta similitud ha sido formalizada como un ANSI estándar para SQL.

De hecho, los principios del SQL han llegado a ser uno de los más grandes espacios para trabajar en computadoras conectadas con arquitecturas disímiles, de múltiples manufacturas, dentro de redes integradas, además, las grandes compañías han hecho el esfuerzo para adquirir tecnología como una influencia unificadora alrededor de sus trabajos de computación. El mecanismo para complementar esto último, ha sido soportado o justificado por múltiples fabricantes de hardware, por niveles de distribución de SQL y considerando la intercomunicación, lo que se justifica como parte integral de estos productos.

Los usuarios pueden ver hacia futuro una gran gama de productos adicionales y producto "puente" para hacerse de una conexión entre el SQL y algunos de los paquetes populares que actualmente existen.

SQL (El lenguaje).

El SQL como un lenguaje está convenientemente dividido en 3 partes: el DDL,DCL y DML.
DDL (Lenguaje de Definición de Datos) provee de los medios para crear y alterar la estructura de una base de datos.
DCL (Lenguaje de Control de Datos) esta relacionado con el acceso que controla los datos en un contexto multiusos.
DML (Lenguaje de Manipulación de Datos) permite la facilidad para consultar, añadir y cambiar datos, esto último es el componente usado invariablemente por la literatura cuando se dan ejemplos para mostrar que tan simple es usar el SQL.

A continuación se profundiza más en ellos.

DDL

Consiste en un conjunto de declaraciones para crear y alterar la estructura de una tabla de datos. Una vez creada la tabla y desplegada se pueden añadir o modificar columnas, pero no borrarlas, empecemos observando los tipos de elementos de datos básicos soportados por el SQL. Mientras que diferentes instrumentaciones pueden dar mas variaciones, los tipos básicos de datos son: CHAR, VARCHAR, LONG, NUMBER, DATE.

El CHAR y VARCHAR se utilizan para representar cadenas de caracteres y si son representaciones variables en el tamaño de las cadenas se hace por VARCHAR. En algunas implementaciones el CHAR y VARCHAR se tratarán idénticamente. El tamaño máximo de una cadena debe ser declarada, y la mayoría de las instrumentaciones deben tener un límite máximo de 240 a 255. En un campo de 20 caracteres, como puede ser un apellido debe ser declarado ya sea como CHAR(20) o VARCHAR(20).

LONG.- Se utiliza para representar bloques de tamaño variable de un texto y tiene como instrumentación específica un tamaño máximo en miles de caracteres. La mayoría de las instrumentaciones planean restricciones extras en las columnas LONG por lo tanto no debe verse como un equivalente a VARCHAR. En particular las columnas LONG generalmente están restringidas para formar parte de criterios de investigación de consulta.

La palabra clave NUMBER puede ser utilizada por si misma para designar un entero o también puede ser utilizada con un tamaño decimal opcional, esto es similar al declarar un campo numérico en Dbase III.Diferentes instrumentaciones pueden corresponder a una lectora de tipos numéricos en adición al título genérico de número. NUMBER se

adecua él solo a muchos casos a pesar de que su permanencia sea benéficamente significativa. La documentación del sistema debe ser revisada cuidadosamente para ver si este es el caso.

Ejemplo:

La fecha puede ser utilizada como un tipo de dato ya que esto permite que la fecha aritmética pueda ser utilizada convenientemente en consultas.

Declaración. Significado.

NUMBER(4) 4 dígitos enteros.

NUMBER(7,2) 7 dígitos con 2 decimales como (99999.99)

NUMBER Entero con tamaño definido en el sistema.

Debe tenerse mucho cuidado al utilizar sistemas con tamaños establecidos ya que pueden ser demasiado grandes. Las extravagancias de la fecha tienden a ser instrumentaciones especificas en términos de un formato establecido y un rango permitido.

RELACIONES.-Una Tabla es la construcción básica de bloques de base de datos SQL. Contienen los datos actualizados de lo que los usuarios quieren guardar y manipular, los comandos básicos para crear una nueva tabla son muy fáciles: CREATE TABLE. El tipo de datos para cada columna se especifica como parte de la definición de la tabla.

Para ilustrar el lenguaje SQL, se plantea el siguiente problema:

En el Museo "X" se tienen en bodega diferentes bienes arqueológicos muebles, que tienen un cierto movimiento en el sentido de que son solicitados, por otros Museos o Centros Regionales, para exhibirse en sus salas. Para su traslado es necesario fijar un valor monetario, aunque por su origen no lo tengan, con la finalidad de asignar una prima de seguro en la eventualidad de algún siniestro.

Haciendo una rápida aplicación de SQL, nos referiremos al "Control de Bienes Arqueológicos Muebles" como punto de referencia. Con sus aplicaciones, este control consta de 4 tablas: cédula, solicitud, monumentos y custodios.

Empecemos por definir la tabla "cédula" para nuestro "control de bienes arqueológicos muebles", para hacer las cosas un poco más fáciles guardaremos las palabras claves del SQL como en el caso anterior:

```
CREATE TABLE Cédula
(pieza CHAR(5) NOT NULL,
tipo CHAR(25) NOT NULL,
cultura CHAR(25),
región CHAR(15),
```

edo_conservación CHAR(8),

valor CHAR(8),

época CHAR(15));

En este ejemplo las palabras claves NOT NULL significan que los campos de pieza y tipo, no pueden permanecer vacíos (vacío es un valor posible para todos los campos aún numéricos), para los campos numéricos vacíos o nulos es diferente tener un valor a tener un valor de cero. Además se puede decir si tiene un valor intencional de cero como opuesto a no tener nunca un campo en él.

Queremos ahora tener acceso a la tabla de cédula por una columna pieza; conociendo esto, frecuentemente daremos de alta un índice permanente en esa columna, esto se hace con la siguiente declaración o enunciado.

CREATE UNIQUE INDEX pieza index ON cédula (piezaASC);

CREATE INDEX ya es suficiente, la palabra UNIQUE indica que los valores duplicados no están permitidos en la columna pieza. La palabra clave ASC seguida de pieza indican que la columna debe ser guardada en orden ascendente. DESC es también utilizada para indicar orden descendente.

También quisiéramos tener acceso a esta tabla por tipo (nadie tiene la identificación de las piezas a la mano).

CREATE INDEX piezatipo ON pieza (tipoASC) COMPRESS;

esta vez no insistimos en que sea única alguna pieza, puede tener múltiples asignaciones. Introducimos la palabra clave COMPRESS que indica que el valor del tipo no será copiado dentro del INDEX, los apuntadores serán mantenidos fuera de los valores de tipo en la tabla de cédula. Esta última opción dará una solución rápida y se ahorrará espacio.

ALTER TABLE cédula MODIFY (piezaCHAR(6));

ALTER TABLE cédula ADD(valorNUMBER(8,2) NOT NULL, seguroNUMBER(8,2));

Hemos modificado(el registro pieza), complementado(el registro valor) y añadido(el registro seguro) con 2 nuevos comandos diferentes.

La tabla de solicitud contendrá una fila para cada "control de bienes arqueológicos muebles". No guardaremos ninguna información en esta tabla de la tabla de "cédula", con excepción de la de pieza. La columna de pieza además relacionará filas correspondientes de las 2 tablas. Uno de los principios del diseño de bases de datos relacionales, como se vió en la normalización, es el evitar guardar copias duplicadas de datos cuando sea posible. Así que si es posible inferir: tipo, cultura, región, edo_conservación y valor de la pieza, entonces no hay razón para duplicar los valores de otros campos en la tabla de "cédula".

Uno de los artificios de la base de datos relacional está basado en el mismo principio. Nos gustaría poder colocar cualquier cantidad de diferentes monumentos de nuestro inventario en el mismo control de bienes arqueológicos muebles. Para complementar esto, separamos los "monumentos" que han sido prestados, en una tabla separada: CUSTODIOS.

```
CREATE TABLE solicitud(
custodionum NUMBER(5) NOT NULL,
pieza CHAR(8) NOT NULL,
iniciales CHAR(3) NOT NULL,
fecha DATE);
CREATE TABLE custodio(
custodionum NUMBER(5) NOT NULL,
existencia CHAR(7) NOT NULL,
cantidad NUMBER(3) NOT NULL);
```

Observemos que estos comandos de la columna custodionum aparecen en ambas tablas y son utilizadas para reunirlas. La "pieza" también aparece en la tabla de "solicitud" y se utiliza para asociar los monumentos individuales con el tipo del pieza, su cultura y la información de valor tomada del archivo de "cédula". Estamos omitiendo aun varios monumentos de nuestro control de bienes arqueológicos muebles original, según se muestra en la figura 1: la descripción de monumentos, en primera instancia, como nosotros esperamos otorgar en custodio cada monumento muchas veces, no hay necesidad de repetir esta información constantemente y la colocamos en una tabla separada, que contiene solo la información de los monumentos que ya han sido dados en custodia.

```
CREATE TABLE monumentos(
existencia CHAR(7) NOT NULL,
descripcion CHAR(35) NOT NULL,
cantidad NUMBER(4) NOT NULL,
valor NUMBER(7,2) NOT NULL);
```

Antes de que nuestra tabla definitiva esté completa, tenemos que considerar que estamos utilizando solo una de las mejores instrumentaciones, tenemos grupos que sirven para nuestro uso y aparecen en apéndice A.

Solicitud #: 99999	fecha:99/99/99		Custodio: 99999
Cant	descripción	valor	cant asegurada
999	xxxxxxxxxxxxxxxxxxxxxxxxx	99999.99	99999.99.
999	xxxxxxxxxxxxxxxxxxxxxxxxx	99999.99	99999.99
Total		99999.99	99999.99

Figura 1. Formato de control de bienes arqueológicos muebles, dados en custodia para una base de datos simple.

DML

Empecemos por ver lo referente a la facilidad de manipular datos con SQL.

Por conveniencia subdividiremos el DML en 3 áreas:

1). Consulta.
2). Generación de reportes a medida.
3). Creación y actualización de datos.

1) Consulta.- Para generar un listado simple de recuperación que contenga todos los datos de una simple entrada solamente daremos la instrucción:

```
SELECT *
FROM monumentos;
```

Esto producirá un listado de todos los monumentos en nuestro inventario. El asterisco (*) en el SELECT cierra las solicitud para usar solamente todas las columnas en la tabla, y el FROM cierra los identificadores de la tabla que estén en uso. Esta simple consulta dirá entonces que liste todas las columnas de la tabla "monumentos".

Para quitar lo más sofisticado y tener un listado solamente de existencias, descripción y valor, la instrucción será:

```
SELECT existencia, descripción, valor
FROM monumentos;
```

El listado omitirá la columna cantidad (en existencia) y tendremos una simple lista de valores. Por supuesto se quisiera tener una lista exacta de los monumentos prestados. Esto nos llevaría a crear una lista de monumentos cuyo valor sea mayor que $100,000.00, ya que el valor mínimo de cada pieza es éste.

```
SELECT Existencia,descripción,valor
FROM monumentos
WHERE valor > 100,000;
```

Ahora hemos introducido la cláusula WHERE, la cual nos permite poner condiciones sobre los renglones seleccionados por la consulta. Las posibles variaciones de la cláusula WHERE pueden conformar un monumento por sí mismo. En suma, para los operadores usuales de comparación (>, <, =, ≠) , los operadores booleanos (AND, OR, NOT) y paréntesis para indicar un orden deseado de evaluación son las condiciones que van seguidas de la cláusula WHERE.

El Código:

```
SELECT pieza,tipo
```

```
FROM cédula
WHERE seguro > 40,000 AND custodio='NL';
```

Selecciona todos los cédula de Nuevo León que han rebasado la suma asegurada por $ 40,000. Y

```
SELECT cantidad,existencia,descripción,valor
FROM custodio,monumentos
WHERE custodio.existencia = monumentos.existencia;
```

Crea la tabla original de nuestro control de bienes arqueológicos muebles, esto es solo para ilustrar una simple reunión.

Aquí se muestra información de dos tablas diferentes y usando solamente la columna existencia de la tabla de custodio y existencia de la columna de la tabla de monumentos para decidir cual renglón de las 2 tablas será combinado.

Supongamos que queremos saber el monto del valor total de las piezas dadas en custodia para septiembre, las órdenes son:

```
SELECT SUM(cantidad*valor)
FROM custodio, monumentos, solicitud
WHERE custodio.existencia = monumentos.existencia
AND custodio.custodionum = solicitud.custodionum
AND (solicitud.fecha BETWEEN '09/01/90' AND
'09/30/90');
```

Aquí combinamos información de las 3 tablas usando las 2 primeras partes de la cláusula WHERE. El conjunto operador SUM es usado en la cláusula SELECT para tomar los valores de todos los renglones seleccionados, y el operador BETWEEN es introducido en la última parte de la cláusula WHERE. La entrada y salida de esta consulta será un número singular el cual es el valor total de las piezas dadas en custodia en septiembre '90.

Poniendo en marcha lo que resta de nuestra forma de control de bienes arqueológicos muebles, para calcular el total monetario de los bienes dados a determinado custodio, usaremos:

```
SELECT SUM(cantidad*valor)
INTO variable de memoria
FROM custodio, monumentos, solicitud
WHERE custodio.existencia = monumentos.existencia
AND custodio.custodionum = solicitud.custodionum;
```

La cláusula INTO es usada para guardar el resultado en una variable de memoria. Esto es una instrumentación-independiente, ya que se define arbitrariamente un valor temporal. Si la instrumentación que se está usando soporta esto, entonces el total solicitado se da instantáneamente, en base a los valores de variable de memoria.

Para el total de suma asegurada de estos bienes:

```
SELECT 1.06 *SUM(cantidad*valor)
FROM custodio,monumentos,solicitud
WHERE custodio.existencia = monumentos.existencia
AND custodio.custodionum = solicitud.custodionum;
```

Supongamos que se quiere contar cómo algunas órdenes serán embarcadas a diferentes estados. Esto puede realizarse con:

```
SELECT custodionum, COUNT(*)
FROM cédula, solicitud
WHERE cédula.pieza = solicitud.pieza
GROUP BY custodionum;
```

Aquí se introducen más elementos del DML, el operador COUNT y la cláusula GROUP BY. COUNT simplemente cuenta el número de renglones que serán seleccionados. GROUP BY es el medio para decirle al manejador de en base datos que agregue el dato en un campo particular.

Si se quiere consultar, se puede hacer lo siguiente:

```
SELECT pieza,tipo,cultura,región, valor,custodionum
FROM cédula
WHERE pieza IN(SELECT pieza
FROM solicitud
WHERE fecha< 06/01/90')
ORDER BY valor;
```

Esta consulta da una lista de envío de todas las piezas que tienen una o más solicitudes antes del 1o. de junio, 1990. Cada pieza aparecerá después sin tomar en cuenta cuantas solicitudes tiene, y el listado será desplegado en orden de valor.

Estos ejemplos son una breve vista a la capacidad de consulta de SQL.

Conociendo el manejo de DDL y DML, y combinando comandos de ambos, se pueden generar:

2).- Reportes.- En algunos casos, se deseará un reporte completo en adición a la consulta, este es un camino para organizar encabezados y títulos de columnas, los comandos DDL permiten la elaboración sofisticada de reportes.

Creando y actualizando la base de datos.- El comando para agregar un nuevo renglón a la tabla es INSERT.

```
INSERT INTO monumentos
VALUES ('2','vasija de barro','1','150,000');
```

la forma básica del comando es el nombre de la tabla, donde serán insertados, y la lista de valores para las columnas de la tabla.

Si se quisiera una tabla conteniendo sólo las piezas aseguradas, se puede obtener con:

```
INSERT INTOpréstamos(piezanúmero,prestadoa,seguro)
SELECT pieza,tipo,valor)
FROM cédula
WHERE valor > 100000;
```

lo que tenemos es que otra tabla llamada "préstamos", fue previamente reemplazada por un comando CREATE TABLE y que éste tiene columnas similares al de la tabla "cédula", pero con diferentes nombres.

3)Actualización.- Se pueden actualizar las sumas aseguradas apropiadamente de manera individual como:

```
UPDATE cédula
SET seguro=200000
WHERE tipo='vaso obsidiana';
```

o como un grupo:

```
UPDATE cédula
SET seguro=200000
WHERE valor > 40000;
```

la cláusula SET puede aceptar expresiones así como también valores constantes:

```
UPDATE cédula
SET valor=.06*valor
WHERE región='Altiplano central';
```

Vistas.- Habiendo desarrollado la estructura de procedimiento de la base de datos, el departamento de catálogo ahora informa que no quieren tener acceso a través del departamento de inventarios, a la información de préstamos, contenida en la tabla cédula. El DDL de SQL da la forma de poder hacer esto sin reorganizar los datos, vía vistas. Una vista es un simple término, un medio de creación de tablas virtuales. Por consiguiente, se puede lograr el requerimiento hecho por el Departamento de Catálogo a través del comando:

```
CREATE VIEW Catálogo
SELECT pieza,tipo,cultura,región,,valor,seguro,custodio
FROM cédula;
```

Aparecerá una tabla llamada "catálogo" en el sistema idéntico a "cédula", excepto por los 2 renglones que no fueron seleccionados. Cada vez que "cédula" es actualizado, "catálogo" aparecerá para ser actualizado automáticamente. Se puede usar una vista para combinar información desde dos o más tablas.

```
CREATE VIEW embarqueSELECTc ustodionum,pieza,tipo,
cultura,región,valor,seguro,edo_conservación,
```

FROM solicitud,cédula
WHERE solicitud.pieza=cédula.pieza;

La cláusula WHERE es usada para conectar las dos tablas.

En esencia la funcionalidad de la declaración SELECT puede ser usada para crear vistas de la base de datos. Estas vistas pueden abarcar múltiples tablas. Aún más, después de creada una vista puede ser manejada como una tabla normal en términos del comando UPDATE, y puede ser usada como objeto del comando INSERT las columnas NOT NULL de la tabla son parte de la vista.

Cuando un comando UPDATE es aplicado para múltiples tablas, la columna apropiada en la línea sobre las tablas son actualizadas. Las vistas pueden ser una herramienta poderosa para crear un medio ambiente simplificado, así como también para aplicar en forma clara los comandos de seguridad de los datos, y explotar la gran capacidad para capturar consultas que son usadas varias veces.

DCL

Los comandos de seguridad básica provistos para un administrador de base de datos (DBA) son GRANT y REVOKE. Como puede indudablemente suponerse GRANT permite el acceso y REVOKE lo impide. Continuando con nuestro sistema de control de bienes arqueológicos muebles, el Administrador de Base de Datos puede darle al usuario "X" completa autoridad dentro de la tabla de monumentos, y aun puede habilitar el acceso a otros usuarios por:

GRANT ALL ON monumentos TO X WITH GRAN OPTION;

X puede decidir que persona estaría capacitada para leer la tabla, y el comando solución:

GRANT SELECTION monumentos TO PUBLIC;

En suma, para conceder los privilegios, SELECT permite leer, pero no cambiar una tabla, el administrador de la base de datos puede permitir cambiar, borrar, indexar, insertar y actualizar en alguna combinación. Adicionalmente la operación UPDATE puede tener una lista de columnas:

GRANT UPDATE (cultura,región,edo_conservación,época) ON cédula TO Juárez;

Este comando no permite a Juárez tener facultad para actualizar la pieza, tipo, seguro o custodio. Por esto Juárez solo puede mejorar o corregir la descripción de la pieza.

En suma, para operar en tablas, GRANT puede ser usado para SELECT, DELETE, INSERT; UPDATE pertenece a una base de datos VIEW. La facilidad de combinación de GRANT con VIEW, efectivamente esta dada por la facultad del Administrador de Base de Datos, para controlar el acceso a los datos en el nivel individual de columnas y

renglones.

La facultad de los comandos CONNECT, RESOURCE y el Administrador de Base de Datos esta disponible, cuando esto se da, el usuario solo requiere de un password. CONNECT tiene la facultad básica para recorrer la base de datos, RESOURCE permite la creación de objetos en la base de datos, y el Administrador de Base de Datos permite el acceso a todo lo autorizado por él mismo.

GRANT CONNECT,RESOURCE TO usuario IDENTIFIED by password;

Esto es un comando básico para agregar un nuevo usuario a la base de datos del SQL. En el proceso, instrumentaciones individuales pueden agregarse pero no dentro de los procedimientos del SQL.

El comando REVOKE opera análogamente y no requiere un password para ser especificado.

```
REVOKE CONNECT FROM usuario;
REVOKE UPDATE ON monumentos FROM usuario;
```

Con todo esto podemos decir que SQL es un rico y poderoso lenguaje para la manipulación, definición y control de bases de datos, sin embargo, al aplicarlo se han detectado serias faltas, las cuales a continuación se detallan.

A.- LA DUPLICACION DE RENGLONES. La proyección no elimina renglones duplicados. Cada una de las funciones estadísticas construidas dentro del DBMS tendrá dos caminos:

a) Que trate cada renglón como esté (sólo uno), ignorando algún grado de duplicación.

b) Que trate cada renglón como si éste ocurriera *n* veces, donde *n* es el grado de duplicación de este renglón.

Las relaciones donde se permite la duplicidad de renglones se denominan "relaciones sucias" cuando se trabaja con relaciones "no-sucias" no importa el orden en que se utilicen los operadores, pero, si se permite la duplicidad de renglones, el orden de los operadores afectará el resultado.

E. F. Codd propone solucionar este problema:

1. Dando a conocer, a los usuarios, que el soporte para la duplicidad de renglones está en marcha para suprimirlo tal vez para 1992.

2. Instalando, en alguna nueva realización, un switch de doble posición (i.e., un bit que controle el DBA) y así permitir al DBMS operar alternativamente en dos modos respecto de los renglones duplicados:

a) aceptando éstos como están, o

b) rechazándolos

3. Suprimiendo completamente el soporte para renglones duplicados dentro de una relación y mejorando el optimizador acordemente.

Considerando la pérdida de integridad en la base de datos, es mejor prevenir que corregir. Por esta razón, el DBMS verificará que los renglones duplicados no sean generados cuando se ejecute un operador que pueda producirlos. Tres de los operadores que se definen para remover renglones duplicados son: PROYECCION, UNION y ACTUALIZACION, incluyendo la carga inicial.

Se puede agregar el comando CONTROL DUPLICATE ROWS, para el DBA, como una característica adicional. Usando este comando, los renglones duplicados son removidos sin pérdida de información.

B. CONFUSION DEL LENGUAJE

La anidación en SQL causa problemas por factores humanos en el lenguaje, ya que, distintos usuarios harían una misma consulta utilizando el lenguaje en distinta manera, en consecuencia se obtienen resultados diferentes, por lo que es necesario hacer consultas cuidadosamente y revisar si el sistema cuenta con una versión SQL de anidación permitida.

C. SOPORTE INADECUADO PARA TRES Y CUATRO VALORES LOGICOS.

Este problema se presenta cuando se tienen valores faltantes en la base de datos, por ejemplo:

teniendo las siguientes condiciones:

FECHA90-12-31 donde la columna FECHA tiene valores de tipo DATE, y CUENTA < 400 donde la columna CUENTA tiene valores NUMERICOS;

se requiere una consulta con la combinacion de ambas columnas. Para obtener un tercer valor lógico, considerando que faltan valores para fecha o cuenta, se puede aplicar la condición:

CUENTA < 400 OR FECHA > 90-12-31

El DBMS reconoce verdadera la expresión, dada la combinación: M ó T, T ó M , M ó M, donde T es *verdadero* y M es el tercer valor lógico posible *quizás*, originado por valores faltantes. Esto sucede así cuando el DBMS soporta tres valores lógicos, si carece de este soporte debe pedirse:

. los valores de la llave primaria de esta orden, para las que la fecha > 90-12-31 es verdadera.

.. los valores de llave primaria para esta orden en la que la CUENTA < 400, sea verdadera, y

... la UNION de los dos conjuntas generados por los dos pasos anteriores.

Para la corrección de esta falta Codd propone:

Que se trabaje (los distribuidores de software) para la introducción del soporte de cuatro valores lógicos.

Mientras estas tres fallas son corregidas por los distribuidores de software, hay algunos pasos que pueden seguirse para proteger las bases de datos y, en consecuencia, los negocios.

Primero, hay que evitar la duplicidad de renglones dentro de las relaciones en todo momento, insistiendo en que:

. Debe especificarse exactamente una llave primaria para cada relación de la base de datos.

.. La palabra clave DISTINCT debe ir seguida de la palabra clave SELECT en todos los comandos SQL que incluyan la cláusula SELECT.

... La palabra clave ALL nunca se acompañará de alguna UNION.

Segundo, deben evitarse las versiones de SQL que contengan declaraciones con la versión anidada.

Tercero, debe extremarse el cuidado al manejar relaciones cuyas columnas contienen valores de distinto tipo y, en lo posible, separar esta información en variables fácilmente indentificables por códigos que puedan reemplazarse posteriormente.

PARTE III

DB2

Las compañías dedicadas a la elaboración de software para bases de datos relacionales, han incluído en sus programas el SQL, por tratarse de un lenguaje estándar, beneficiando con ello a los usuarios de sus sistemas, al lograr que las tareas sean mas rápidas y eficientes.

En esta parte se presentan cuatro sistemas de Bases de Datos Relacionales, a saber: DB2, DBase IV, Informix SQL y Oracle Profesional. El objetivo es ver cómo están integrados y la forma en que utilizan al SQL.

DB2 fue el primer sistema manejador de base de datos relacional diseñado por IBM, su funcionamiento prácticamente está basado en SQL. Dado que anteriormente vimos como funciona SQL, solo indicaremos su uso en DB2.

Para definir los datos en DB2 se hace con las declaraciones SQL en donde se crea la tabla de datos dándole nombre a la tabla que va a ser creada, nombres de las columnas y los tipos de datos de esas columnas.

En DB2, después de haber creado las tablas y metido algunos registros en ella podemos empezar a hacer uso de ellos mediante las declaraciones de manipulación de datos SQL. Estas pueden ser usadas directamente desde el sistema es decir, en modo interactivo DB2I o insertando las declaraciones SQL dentro del programa de aplicación en algún lenguaje de alto nivel como: PL/1, Cobol, Fortran o Ensamblador, especificando que se trata de declaraciones SQL.

Las principales declaraciones son:

CREATE TABLE.- Cuyo efecto es crear una nueva tabla base.

Los datos pueden ser cargados vía la declaración INSERT de SQL.

Los tipos de datos que soporta DB2 son: INTEGER, FLOAT, CHAR(n), VARCHAR(n).

En DB2 una columna que puede aceptar valores nulos está físicamente representado en la base de datos almacenada por dos columnas, la columna de datos en sí misma y un indicador oculto de columna,

indicando que el valor de esta columna será ignorado, pero si el valor es cero se tomará como genuino, es decir., valor no nulo.

ALTER TABLE.- Con esta declaración la tabla puede ser modificada, pues se pueden agregar nuevas columnas.

DROP TABLE.- Suprime alguna tabla específica. La tabla base es removida desde el sistema. Todos los índices y vistas definidas en la tabla base son suprimidas automáticamente.

INDEX.- Como la tabla base, los índices son creados y destruidos usando las declaraciones DDL de SQL. CREATE INDEX y DROP INDEX son básicamente las únicas declaraciones SQL que se refieren a los índices; la decisión para usar o no un índice respondiendo a una requisición particular de SQL no es hecha por el usuario sino por el sistema optimizador.

El hecho de que las declaraciones DDL puedan ser ejecutadas en algún momento, hacen a DB2 un sistema muy flexible.

DML

DB2 utiliza las 4 declaraciones DML de SQL para manipular sus datos a saber: SELECT, UPDATE, DELETE e INSERT.

SELECT.- Especifica archivos desde (FROM) una tabla especifica cuando (WHERE) alguna condición es cierta. Con esta declaración se pueden hacer recuperaciones simples, en donde SQL no elimina duplicados, si no se pide explícitamente que se eliminen vía la palabra clave DISTINCT, se pueden recuperar expresiones y cuantificaciones usando la cláusula WHERE.

Unión de Consultas.- La habilidad para "unir" 2 o mas tablas es una de las características más poderosas de los sistemas de bases de datos relacionales. Una unión es una consulta en la cual los datos son recuperados desde mas de una tabla, esto se hace utilizando la declaración SELECT.

Construcción de Funciones.- La declaración SELECT, utiliza funciones para recuperar datos, estas son: COUNT, SUM, AVG, MAX, MIN.

Hay tres declaraciones de actualización, a saber:

UPDATE.- Cambia o modifica la tabla.
DELETE.- Borra datos de una tabla.
INSERT.- Agrega datos a una tabla.

Todas estas declaraciones pueden ir acompañadas de las cláusulas: WHERE, INTO, FROM, SET, GROUP BY.

La cláusula básica SELECT en si misma, incluye el uso de DISTINCT, contantes y expresiones.

La cláusula FROM, incluye el uso de Alias.

La cláusula WHERE incluye operadores de comparación, predicados de unión y operadores booleanos.

El uso de ORDER BY para ordenar la tabla que se requiera.

EL SISTEMA DE CATALOGO.

El sistema de catálogo puede concebirse como un sistema de base de datos que contiene información (algunas veces llamado descripción) acerca de varios objetos que son de interés para el mismo sistema, tales objetos pueden ser: tablas base, vistas, índices, usuarios, planes de aplicación, privilegios de acceso, etc.

Una de las características de un sistema relacional como DB2 es que, en tal sistema, el catálogo mismo es una relación o tabla. En DB2 el catálogo tiene de 20 a 25 tablas.

La forma simplificada contiene las siguientes tablas de catálogo:

SYSTABLES.- Esta tabla contiene un renglón para cada tabla (tabla base o vista) a la entrada del sistema. Para cada tabla, este da el nombre de la tabla, el nombre del usuario que creó la tabla' el número de columnas de la tabla, y algún otro tipo de información.

SYSCOLUMNS.- Esta tabla contiene un renglón para cada columna de toda tabla en la entrada del sistema. Para cada columna esta da el nombre de la columna, el nombre de la tabla de la cual forma parte esta columna, el tipo de datos de la columna y algunas otras cosas similares.

SYSINDEX.- Esta tabla contiene un renglón para todos los índices del sistema. Para cada índice, este da el nombre del usuario que creó el índice.

Ya que el catálogo esta formado por tablas, este puede ser consultado con algunas de las declaraciones SELECT de SQL como si se tratara de tablas ordinarias.

En DB2 el sistema de catálogo y la base de datos son consultadas directamente por el lenguaje de interface (SQL).El catálogo puede ser consultado con algunas de las declaraciones SQL SELECT. Sin embargo no puede ser actualizado usando UPDATE, DELETE e INSERT, sino que se actualiza con: CREATE que sería análogo a INSERT, DROP análogo a DELETE y ALTER análogo a UPDATE.

VISTAS.

Las vistas en DB2 son almacenados en el catálogo (en una tabla llamadas SYSVIEWS).

Se crea utilizando CREATE VIEW seguida del nombre y por la palabra clave AS que indica de que tabla será derivada la vista, ésta será una ventana dentro de la tabla real, y cumplirá lo especificado por las cláusulas FROM y WHERE, las operaciones de

actualización son usadas en forma similar, así entonces: Si una tabla base es suprimida, todas las vistas definidas que se derivaron de ella serán también suprimidas. En general se puede hacer cualquier tipo de actualización desde una vista, siempre y cuando se cumpla con las condiciones que fueron especificadas cuando fueron creadas.

SQL es siempre compilado en DB2, nunca interpretado, esto es, cuando las declaraciones en cuestión están sometiéndose interactivamente. En otras palabras, si el usuario mete, digamos una declaración SELECT desde la terminal, entonces esta declaración será compilada y el plan de aplicación generada por este, este plan será ejecutado, y finalmente, después de la ejecución, este plan será desechado.

DBASE IV

El primer producto de Ashton-Tate, DBII, fue escrito por Wyne Ratiliff e introducido en 1980. Posteriormente surgió DBASEIII que tiene la ventaja de memoria extensa de la PC-IBM, aquí fueron agregados nuevos comandos, se extendieron las especificaciones y en general fue mejorada la confiabilidad en este sistema. A principios de 1986 surge DBASEIII PLUS con comandos adicionales y menús principales. En 1988 DBASE evoluciona nuevamente para satisfacer las necesidades de desarrollo de los usuarios, con DBASE IV se ofrecen nuevas y formidables características.

DBASE IV funciona en dos modos: a).- el modo convencional de DBASE, el cual soporta todos los comandos de DBASE III PLUS y otros adicionales. b).- el modo SQL, el cual como se ha dicho anteriormente es estándar para todos los productos de bases de datos relacionales de hoy en día.

Este sistema soporta un extensivo numero de lenguajes de alto nivel lo que permite al usuario el desarrollo de programas de aplicación aproximadamente un 10% del tiempo en que se desarrollaría algún otro programa en lenguajes tales como Basic o C.

El acceso al sistema es desde un Centro de Control, desde aquí se pueden manejar archivos (crear, modificar, cerrar, agregar, editar, buscar), crear vistas, definir formatos, crear o generar reportes, correr programas, y para crear un nuevo programa se utiliza el Generador de Aplicaciones.

El Centro de Control funciona como un menú principal de alto nivel, permitiendo al usuario el acceso a trabajos dirigidos, los cuales son ventanas disponibles para el usuario, ya que desde ahí se pueden ver y editar datos, crear formatos y reportes, etc.

Una de las adiciones a la capacidad de DBASE III PLUS, es el agregar SQL. DBASE IV distingue los programas DBASE de SQL por la extensión del archivo que contiene el programa (.prg para DBASE IV, .prs para SQL), asimismo la extensión determinará el modo.

En el modo DBASE, todos los comandos de DBASE IV están disponibles. Se puede trabajar con archivos de la base de datos, formular consultas y vistas usando QBE, crear reportes y etiquetas, generar aplicaciones.

Se tienen catálogos DBASE para cada base de datos DBASE. Se puede seleccionar el catalogo que se quiera desde el Centro de Control en pantalla. Todos los catálogos contienen el nombre y descripción de los archivos, índices, vistas, (archivos QBE), formas, reportes, etiquetas, y programas de aplicación que contiene la base de datos. Se

pueden agregar nuevos archivos o removerlos desde el catalogo.

En este modo una vista se guarda en un archivo con las extensiones: .qbe o .vue. Esta puede ser consultada o editada con los comandos BROWSE y EDIT.

En el modo DBASE/SQL, se pueden utilizar comandos SQL, correr programas que contengan comandos SQL inmersos, aunque algunos de los comandos y funciones de DBASE IV no son accesibles en este modo.

Los catálogos son tablas dentro de la base de datos, las cuales contienen la información acerca de la base de datos, que ,D!KXC utiliza el sistema. Se puede crear una base de datos en el directorio que el usuario escoja, después de lo cual este directorio estar en uso para empezar a trabajar con esa base de datos, así entonces se pueden consultar las tablas de catalogo para obtener información acerca de la base de datos. Cuando se agregan objetos a la base de datos SQL, las tablas del catalogo son actualizadas por el sistema y así reflejar las modificaciones. La base de datos misma es listada por SQL, en las tablas del catalogo maestro SQL llamados Sysdb.dbf el cual esta en el directorio SQL.

Una vista es definida por la declaración CREATE VIEW la cual involucra una declaración SELECT. La definición de esta es guardada en las tablas de catalogo SQL, la cual solo puede ser accesada con comandos SQL, y puede ser usada para actualizar o borrar desde tablas, siempre y cuando se respeten las siguientes condiciones:

- En el caso de actualización, los campos a actualizar no pueden ser el calculo de un campo.

- La vista que se actualiza o borra no puede definirse utilizando un operador.

- Mas aún, la definición de la vista no será actualizable por la cláusula FROM, ni contendrá la palabra DISTINCT.

- La definición no contendrá una subconsulta, referida con la clausular FROM para alguna tabla de la cual es definida la vista.

El control de acceso a los datos se puede hacer por medio de vistas, con los privilegios de GRANT de SQL. Las vistas SQL no se usan como en DBASE IV para formatos, etiquetas y reportes porque estos requieren de un archivo especial DBASE, sin embargo se puede usar el comando SELECT con las cláusulas SAVE TO, etc., para crear un archivo DBASE y usarlo para formatos, etiquetas y reportes.

Todos los índices en este modo son "etiquetados" en un archivo .mdx. Se puede usar el comando CREATE INDEX para crear un índice. La diferencia del modo para usar índices es que: en el modo DBASE se puede seleccionar el índice que se quiere en la recuperación de la información de la base de datos, mientras que en el modo DBASE/SQL es seleccionado por el sistema.

En lo que se refiere a la programación, antes de elaborar programas SQL para DBASE

IV, es necesario considerar lo siguiente:

- Los programas y procedimientos en DBASE IV pueden contener todos los comandos y funciones de DBASE IV, pero no pueden contener algunas declaraciones SQL.

- Los programas y procedimientos SQL, pueden tener comandos SQL inmersos y pueden usarse (pero no todos) los comandos de DBASE IV.

- Algunos de los comandos de DBASE IV pueden ser encajados en declaraciones SQL pero no todas, como por ejemplo:

APPEND, BLANCK, FROM, MEMO .- Agrega registros, los importa desde otro archivo, agrega campos memo un campo existente.
ASSIST.- Llama al menú del sistema.
AVERAGE.- Calcula el promedio de valores de un campo.
BROWSE.- Llama consultas del sistema.
CALCULATE.- Calcula funciones financieras o estadísticas.
CHANGE.- Cambia campos y registros específicos.
CONTINUE.- Continua localizando el siguiente registro.

COPY.- Copia índices, memo, estructuras, arreglos, etc.
COUNT.- Cuenta registros desde un archivo.
CREATE.- Crea archivos, modifica o crea etiquetas, vistas, reportes.
DELETE.- borra registros desde un archivo.
DISPLAY STRUCTURE.- Despliega la estructura del archivo activo.
EDIT.- Despliega registros una a un tiempo.
EXPORT TO.- Exporta un archivo .dbf.
FIND.- Encuentra un registro.
GO/GOTO.- Posiciona el apuntador del registro en un archivo.
IMPORT FROM.- Importa un archivo.INDEX.- Crea un índice (.ndx o .mdx).
INSERT.- Inserta un registro en un archivo.
JOIN.- Combina registros desde dos archivos de la base de datos.
LABEL FROM.- Imprime etiquetas.
LIST STRUCTURE.- Lista o imprime el estado y parámetros del sistema.
LIST.- Lista/imprime campos y registros específicos.
LOCATE.- Localiza un registro y posiciona ahí el apuntador del registro.
MODIFY STRUCTURE.- Cambia la estructura de un archivo de la base de datos.
PACK.- Remueve registros para ser borrados.
RECALL.- Remarca registros para ser borrados.
REINDEX.- Reconstruye índices.
REPLACE.- Cambia los contenidos de los campos e un archivo.
REPORT FROM.- Despliega/imprime reportes tabulados de datos.
ROLLBACK.- Recupera un archivo para un estado de pretransacción.
SCAN.- Encuentra el siguiente registro de una búsqueda.SELECT.- Activa el área de trabajo especificada.
SET INDEX TO.- Cierra índices.
SET VIEW TO.- Activa una vista DBASE.SKIP.- Mueve el apuntador antes o

después de un registro.
SORT.- Crea una copia de una archivo de la base de datos arreglando los registros en un orden especificado.
SUM.- Hace una suma aritmética.
UPDATE.- Hace cambios en un archivo de la base de datos.
USE.- Activa un archivo de la base de datos.
ZAP.- Remueve todos los registros desde un archivo activo de la base de datos.

- Los programas DBASE IV pueden llamar a programas SQL y viceversa. DBASE IV encenderá automáticamente el modo, dependiendo de la extensión del programa o procedimiento. Sin embargo en la combinación de estos, en ambos modos, se debe considerar lo siguiente:a).- Si se crea una estructura en el modo DBASE, esta no será agregada al catalogo SQL, y SQL no tendrá acceso a la tabla resultante a menos de que se utilice el comando DBDEFINE para tenerlo en el modo DBASE/SQL.

b).- Las tablas del catalogo SQL no se actualizan estando en el modo DBASE.

Las ventajas de la programación con SQL inmerso son:a).- Una declaración SQL puede hacer tanto trabajo como algunos comandos de DBASE IV; la clave esta en que se disminuye la posibilidad de errores, y hace el código mas fácilmente entendible para programadores y no programadores.

b).- La programación con SQL inmerso es simple, porque se puede pensar en términos de operaciones de una o mas tablas.

c).- Las vistas y uniones son fáciles de organizar en SQL.

Lo recomendable será que se use el modo DBASE para crear pantallas a medida, generar reportes y etiquetas, mientras que el modos DBASE/SQL se use para una rápida y fácil definición y manipulación de datos.

COMANDOS DE DBASE IV:

ACCEPT Almacena una cadena de caracteres.
ACTIVATE Activa una barra de menú ya existente y la despliega
APPEND Agrega un registro al final de un archivo de una base de datos.
ASSIST Activa Dbase IV para Modo Centro de Control.
AVERAGE Hace una suma aritmética por medio de una variable de memoria.
BROWSE Funciona el manejador de menú, edita toda la base de datos en pantalla.

CALL Ejecuta archivos en memoria cargados con LOAD.
CANCEL Aborta el programa de ejecución.
CHANGE Edita archivos específicos en una base de datos.
CLEAR Limpia la pantalla y mueve el cursor hacia la parte inferior de ésta.
CLOSE Cierra objetos específicos.
COMPILE Compila un archivo fuente sin ejecutarlo.
CONTINUE Continua para buscar el siguiente
registro, usando condiciones especificadas en el comando locate.

CONVERT Agrega un campo al archivo de la base de datos corriente.
COPY Copia archivos, índices, memo, estructuras y arreglos.
EDIT Edita los contenidos de un registro de la base de datos.
EJECT Ejecuta un dispositivo de alimentación en la impresión.
ERASE Borra los archivos especificados desde el directorio.
EXIT Sale desde un loop DO WHILE sin considerar la condición específica.
FIND Posiciona el apuntador del registro en un índice de la base de datos para el primer registro que cumpla la condición especifica.
FUNCTION Identifica el comienzo de una función definida por el usuario.
GO Posiciona el apuntador en un registro especificado de la base de datos.
HELP Funciona el menú de ayuda de dBase IV.
INDEX ON Crea un índice para una base de datos sobre una llave especifica.
INPUT Permite la entrada de datos.
INSERT Inserta un nuevo registro dentro de la base

de datos en una posición especificada.
JOIN WITH Combina registros y campos desde 2 bases de datos para formar una tercera base de datos.
LIST Lista campos de registros, archivos, comandos previamente ejecutados, variables de memoria, índices, estructuras, usuarios.
LOAD Guarda archivos en memoria para ejecutarlos desde el comando CALL.
MODIFY Modifica un programa, archivo, consultas, reportes, pantallas, estructuras, vistas, etc.
PACK Remueve todos los registros en la base de datos que son marcados para borrarse.
PARAMETERS Especifica variables de memoria para ser usadas con el comando DO...WITH
PRIVATE Especifica variables de memoria para ser consideradas locales para el programa.
PROCEDURE Identifica a un procedimiento ejecutable dentro del programa o abre un procedimiento.
PROTECT Activa la seguridad del sistema.
QUIT Cierra todos los archivos y sale de Dbase IV.
RENAME Cambia el nombre de algún archivo.
RESUME Resume un programa suspendido.
RETRY Reejecuta el procedimiento de un comando que ha causado un error.
RETURN Regresa a un punto especificado.
SKIP Mueve el apuntador hacia atrás o adelante, según la expresión especificada.
SORT TO Crea una base de datos con registros ordenados de acuerdo a la especificación (ascendentes o descendentes).
WAIT Suspende la ejecución del programa hasta que una tecla es presionada.
ZAP Remueve todos los registros desde una base de datos.

ORACLE PROFESIONAL

Oracle provee un conjunto de programas funcionales para microcomputadoras, que se pueden usar como herramientas para construir estructuras y tareas de funcionamiento y aún más, se pueden crear complejas aplicaciones.

Un sistema manejador de bases de datos moderno, puede efectuar un extenso arreglo de trabajos. En general, éste actúa como una interface transparente entre el almacenamiento físico y la representación lógica de los datos. En la práctica, provee un conjunto de herramientas más o menos sofisticadas para obtener información. Estas herramientas se usan para:

- Definir una base de datos.
- Consultar la base de datos.
- Agregar, editar y borrar datos.
- Modificar la estructura de la base de datos.
- Asegurar los datos para acceso público.
- Exportar e Importar datos.

La persona responsable de la base de datos, administrador de la base de datos (DBA), tiene un conjunto de privilegios especiales que le dan control completo dentro de la base de datos. Los trabajos del DBA incluyen:

- Creación primaria de la base de datos.
- Modificación de la estructura de la base de datos.
- Respaldo y devolución de la base de datos.
- Monitoreo de eficiencia y funcionamiento de la base de datos.
- Transferencia de datos (entre la base de datos y archivos externos).
- Control y monitoreo del acceso de usuarios a la base de datos.
- Manipulación para localización física de la base de datos.

La complejidad de estas tareas para el sistema y el DBA, provoca la necesidad de un sistema central de documentación incorporado, dentro de algún manejador de base de datos relacional, este sistema es el Directorio de Oracle.

Fin de Usuario.- Poco complejo pero igualmente necesario e importante, usado para:

- Consultar la base de datos.
- Generar impresiones de salida.
- Compartir datos o asegurarlos de otros usuarios.
- Usar métodos de actualización fáciles para la base de datos.
- Definir aplicaciones desde el punto de vista del usuario.

Sistemas fomentadores.- Crean aplicaciones alineadas, desde una forma simple de entrada de datos, para aplicaciones que comprenden multiusuarios dependiendo del nivel y complejidad de las tareas que se tienen. El sistema fomentador:

- Presenta pantallas entendibles con las cuales se puede tener acceso a los datos.
- Edita y controla la entrada de usuarios a estas pantallas.
- Obtiene y presenta datos basados en la información del usuario.
- Muestra datos por renglón o columna.
- Procesa datos usando lógica condicional.
- Accesa a la base de datos desde lenguajes procedurales tradicionales.
- Permite el acceso a dispositivos periféricos.
- Intercomunica datos con otros paquetes.
- Tiene salida a otras plataformas de hardware como una aplicación de ampliación.

ACCESO DE DATOS.

SQL es el lenguaje de interface entre el usuario y las bases de datos de Oracle, todas las herramientas de acceso de Oracle están basadas en SQL std., además Oracle es compatible con IBM y sistemas manejadores de bases de datos como DB2 y otros sistemas basados en SQL std.

La habilidad para operar con SQL, es fundamental para Oracle, ya que los programas proveen un rico conjunto de extensiones para SQL, los cuales son llamados comandos SQL*PLUS.

El conjunto de comandos estándar de SQL están dentro de 4 categorías: DDL, DML, DCL y los comandos de consulta que se utilizan para obtener información desde la base de datos. Oracle usa SQL para permitir acceso, modificar y desplegar datos. Se incluyen herramientas adicionales, pero como se dijo antes, este conjunto de comandos es el fundamento sobre el que Oracle está construido.

MEDIO AMBIENTE DE ORACLE.

Oracle es un sistema modular que se compone de la base de datos Oracle y varios programas funcionales, estos componentes pueden ser vistos como herramientas, las cuales tienen un propósito específico y se pueden accesar o usar en forma independiente. Estas herramientas son:

- Manejador de base de datos.
- Acceso y Manipulación de datos.
- Programación.

Herramientas del manejador de bases de datos.-

Llamadas RDBMS por Oracle, incluyen el núcleo de programas del sistema manejador de base de datos Oracle; esta base de datos se asocia con tablas y vistas, las cuales son almacenadas en el directorio de datos de Oracle, y un grupo de "utilería" local, esencial

para el trabajo del DBA, y se conforma como sigue:

IOR Comienza, para e inicializa un sistema Oracle.
SGI Provee memoria global al sistema según el modo de empleo de la información.
CRT Ajusta pantallas desplegando características.
EXP Exporta datos desde la base de datos a un archivo.
IMP Importa datos desde un archivo a la base de datos.
ODL Carga datos desde formatos de archivos externos.
CCF Crea un archivo para almacenar información de la base de datos.

Directorio de Oracle.-

El directorio de datos del sistema Oracle es la documentación central. Aquí se almacena información relacionada con todo lo existente en el sistema de base de datos: nombre de usuarios, acceso a usuarios autorizados, nombres de tablas, nombres de atributos de tablas, información almacenada en tablas; también recopila datos para recuperar algún desastre, lo cual resultaría imposible de manejar por el DBMS sin esta herramienta.

Un directorio de datos puede estar pasivo o activo. Si está activo, significa que el sistema automáticamente actualiza el directorio, es decir, actualiza conforme se van haciendo cambios. Los programas de base de datos de Oracle entonces recurren al directorio como se va necesitando. Esto es preferible al modo pasivo, en el cual el sistema no usa el directorio, y el DBA es responsable de la actualización de datos. El DBA no necesita el directorio, sin embargo todos los usuarios se accesan a él para obtener más rápidamente su información.

Herramienta de acceso y manipulación de datos.-

Todas las herramientas de acceso y manipulación de datos Oracle tienen en común algo vital: se basan en ANSI std. SQL. Estos programas son la entrada a Oracle, las herramientas que pueden usarse para accesar y manipular datos, así como para las que asignan o usan aplicaciones, tienen un punto separador de entrada y una única aproximación al sistema Oracle, y son los siguientes:

SQL*Plus Permite el acceso directo a la base de datos con comandos SQL., comando desde DOS: C/SQLPLUS.

Comandos SQL*PLUS:

APPEND Agrega texto para el fin de la línea corriente en el buffer corriente.
BREAK Especifica los eventos que causarán un rompimiento y como SQL actúa cuando un rompimiento ocurre.
CHANGE Cambia el contenido del buffer corriente de la línea corriente.
CLEAR Limpia definiciones de break, texto del buffer corriente, definición de columnas y otros.
COLUMN Especifica como va a ser formateado en un reporte una columna y encabezados de columnas.
COMMIT Hace permanentes los cambios hechos en la base de datos desde el

último COMMIT.
COMPUTE Funciona operaciones en grupos de renglones seleccionados.
CONNECT Registra entrada de usuarios y rompimientos dentro de Oracle.
COPY Copia datos desde una base de datos SQL a otra.
DEFINE Designa un uso de variable y asigna este a un valor de caracter.
DEL Borra la línea corriente.
DESCRIBE Despliega una breve descripción de una tabla.
DOCUMENT Empieza un bloque de documentación en un archivo de comandos.
EDIT Llama al editor de textos estándar del sistema.
EXIT Termina SQL*PLUS y regresa el control al sistema operativo.GET Carga archivos dentro del buffer corriente.
HELP Despliega información acerca de los comandos SQL y SQL*PLUS.
INPUT Agrega nuevas líneas, después la línea corriente en el buffer corriente.
PAUSE Despliega un mensaje y entonces espera hasta que la tecla ENTER es oprimida.
QUIT Termina SQL*PLUS y regresa el control al sistema operativo (equivalente a EXIT).
REMARK Empieza un comentario en un archivo de comandos.
ROLLBACK Regresa hacia atrás y desecha los cambios hechos a la base de datos, después del COMMIT más reciente.
RUN Despliega y corre los comandos en el buffer SQL.
SAVE Salva los contenidos del buffer corriente en la base de datos o en un archivo del sistema operativo.
SET Ds un parámetro SQL para un valor especificado.
SHOW Despliega establecimiento de un parámetro SQL o propiedades del SQL*PLUS.
SPOOL Maneja el despliegue de salida desde un
archivo del sistema o al sistema de impresión.
START Ejecuta los contenidos de un archivo de comandos almacenados como un sistema operativo de archivos.
TIMING Ds el análisis de funcionamiento de comandos de archivo y comandos SQL.
TTITLE Hace que SQL despliegue un título al final de cada página de salida.
UNDEFINE Borra la definición de una variable de usuario definida.

SQL*Forms Ofrece un camino fácil al usuario para crear y usar formatos., comando desde DOS: C/SQLFORMS y C/RUNFORM.

COMANDOS SQL*FORMS

SQL*FORMS es más que un generador de pantallas. Se pueden crear formatos de estructura compleja, sin los lenguajes tradicionales de programación, elimina algunos pasos y usa una aproximación más visual.

CREATE Crea una nueva forma (bloque).
MODIFY Modifica una forma existente.

LIST Selecciona una forma desde un listado de las formas.
RUN Corre la forma.
DEFINE Define etiquetas, encabezados y comentarios.
LOAD Carga una nueva forma desde la base de datos.
FILE Mantiene la forma activa, pero almacenada.
GENERATE Genera la forma
DROP Suprime un bloque especificado.
FIELDS Lista los campos en el bloque especificado.
DEFAULT Crea una forma por default.
NEXT Selecciona el siguiente bloque en la forma.
PREVIOUS Selecciona el bloque anterior en la forma.

SQL*Reportwriter Da al usuario formatos de salida ya creados desde DOS: C/SQLREP.

COMANDOS SQL*REPORTWRITER.

Es el complemento de SQL*Plus, con él se pueden incorporar más de una consulta dentro del reporte.

Utilizando comandos de SQL se pueden obtener los reportes deseados, las opciones que utiliza son las siguientes:

ACTION con la cual se elige:
NEW Crea un nuevo reporte.
OPEN Abre un reporte existente.
COPY Copia un reporte existente.
RENAME Renombra un reporte existente.
DROP Suprime un reporte existente desde la base de datos.
EXECUTE Genera y ejecuta un reporte si es necesario.
QUIT Se sale de ACTION y regresa el menú.

Opción QUERY, Utilizando la declaración SELECT de SQL se escoge la consulta.

Opción FIELD Después de nombrar el reporte y haber metido la consulta, será necesarios un formato de salida, de tal manera que resulte un reporte presentable, es decir aquí se utilizará diciendo que encabezados y pies de página llevará el reporte, esto es con:

Field name.- Nombre del campo
Source column.- Columna que se consulta.
Group.- Grupo en el cual éste será incluido.
Label.- Texto que describirá la etiqueta usada.
Data type.- Tipo de columna (NUM, DATE, CHAR) no puede ser cambiada. Field width.- Despliega la anchura del campo.
Display format.- Máscara para números o columna de fecha.
Relative position.- Posición relativa del resto de los campos en éste grupo.
Lines before.- Número de líneas que preceden al campo.

Spaces before.- Número de espacios que preceden al campo.
Align.- Alineación (derecha, centro, izquierda).
Skip.- Una x suprime este despliegue en el reporte.
Repeat.- Un campo x que será repetido como una etiqueta para cada texto en el tablero.
Function.- Una de las 12 funciones que pueden ejecutar un cálculo.
Reset group.- El grupo que determinará el cálculo para cero.

Opción SET. Con ella se pueden agregar o modificar el formato de los textos del reporte. El primer campo en la pantalla es el objeto, los objetos y tipos posibles para el reporte son:

Report título de página.- Imprime en una página separada antes del reporte.
Report encabezado del reporte.- Imprime en la parte superior de la primera página del reporte.
Page header.- Imprime en la parte superior de cada página.
Group header.- Imprime una vez arriba un grupo.
Group heading.- Imprime una vez para cada columna.
Group body.- Define la estructura de cada registro en el reporte.
Group subfooter.- Imprime después de la columna o renglón de un reporte cruzado.

Group footers.- Imprime una vez después un grupo.
Page footer.- Imprime al final de cada página.
Reportfooter.- Imprime una vez fuera del fin del reporte.
Report trailer page.- Imprime fuera en una página separada después del reporte.

SQL*Menú Proporciona un camino para integrar una aplicación usando menús., comando desde DOS: C/SQLMENU.

SQL*MENU es un poderoso y flexible menú. Usándolo se puede desarrollar seguridad rápidamente, para presentación de formas, reportes y consultas creadas con Oracle, usando parte del menú, se pueden crear aplicaciones que están estrechamente enlazadas con las demás herramientas de Oracle.

Herramienta de programación.-

Las series de interfases de Programación Oracle son muy importantes. Oracle puede trabajar con Cobol y C. Tiene un precompilador para convertir las declaraciones de SQL, al código de los lenguajes de alto nivel y de esta manera compilar finalmente desde el lenguaje utilizado. En Oracle, todos los datos son almacenados y desplegados en tablas, ya que es un sistema de base de datos relacional.

Una vista es una tabla derivada que se puede crear con el propósito de visualizar en pantalla. Aunque ésta se percibe como una tabla "real", dentro de la base de datos es solamente una definición de almacenamiento. Por esta razón, las vistas son conocidas como una "tabla virtual", mientras que las tablas derivadas de ellas son llamadas tabla base. Una vista puede ser una combinación de dos tablas base o un subconjunto de una

tabla base. Se pueden hacer vistas para limitar el acceso a las tablas y para hacer más cortas ciertas bases de datos.

Dado que Oracle es un sistema relacional, los datos almacenados pueden ser conectados con varias tablas para incrementar su uso y obtener duplicados, utilizando: SELECTION, PROYECTION y JOIN.

INFORMIX-SQL.

Informix-SQL es un sistema manejador de bases de datos. Se conforma con programas útiles o módulos diseñados para mejorar las tareas de manejo de datos.

Un buen sistema manejador de bases de datos puede reducir sustancialmente la cantidad de tiempo requerido para organizar, modificar y recuperar información, la cual puede sumarse, agruparse y formatearse fácilmente y con poca probabilidad de errores.

Con Informix-SQL, se mejoran por ejemplo, las siguientes tareas:

a). Creación de bases de datos y tablas.
b). Diseño de formatos en pantalla.
c). Captura y modificación de datos con el uso de formatos de pantalla.
d). Carga de datos desde sistemas operadores de archivos.
e). Consulta desde formatos de pantalla o con un lenguaje interactivo de consulta.
f). Elaboración de reportes con formatos a medida.
g). Diseño de un menús adecuados para incluir elementos de informix-SQL a sistemas útiles, y otros programas.

RDSQL.

RDSQL es el lenguaje interactivo de consulta para Informix, y se usa para seleccionar el dato que se quiere en un reporte.

Es una extensión de SQL desarrollado por IBM para creación y consulta de bases de datos.

La adición de RDS al lenguaje permite: cargar y descargar tablas de bases de datos, dentro de sistemas de archivos; cambiar bases de datos, nombres de tablas y columnas. Otras extensiones incrementan la funcionalidad de las declaraciones del SQL estándar.

Se puede procesar una declaración a un tiempo en RDSQL, o se pueden procesar una secuencia de declaraciones separadas por (;).

En RDSQL (;) es un separador, no un terminador de declaraciones. No es necesario si solamente se realiza una declaración.

Para trabajar con RDSQL, se selecciona la opción del lenguaje de consulta en el Menú de informix, si ésta no es la base de datos activa, informix despliega en pantalla la base de datos seleccionada y despliega el menú de RDSQL.

Mientras se trabaje con Informix, la última secuencia de declaraciones, si las hay, son llamadas declaraciones actuales.

El Menú RDSQL tiene 10 opciones:

1). NEW Permite que se introduzca un nuevo conjunto de declaraciones actuales. Las declaraciones previas se perderán cuando se seleccione NEW OPTION, se ha de salvar con la opción SAVE.

2). RUN - Procesa las declaraciones actuales.

3). MODIFY.- Despliega el editor RDSQL con las declaraciones actuales, se pueden corregir errores de sintaxis, hacer cambios o agregar declaraciones

4).- USE-EDITOR.- Permite el uso de un editor del sistema con las declaraciones actuales en el buffer del editor. El editor por default (en sistema UNIX) o EDLIN (en sistema DOS), puede cambiar el editor por default por el conjunto de variables de DBEDIT.

5).- OUTPUT.- Envía los resultados procesando la declaración actual a la impresora, un sistema de archivos, o (en sistema UNIX solamente) otro programa.

6).- CHOOSE.- Permite seleccionar los contenidos de un comando de archivo, así como la declaración corriente. RDSQL presenta una lista de todos los archivos de comandos (archivos con la extensión .sql) en el directorio activo. Si esta no es la base de datos en uso, la lista contendrá todos los comandos de archivo localizados en el directorio activo y el directorio que contiene a la base de datos en uso.

7).- SAVE..- Salva la declaración actual de un comando de archivo. RDSQL agrega la extensión .sql al nombre del archivo que se elige.

8).- INFO.- Despliega la información de una tabla en la base de datos en uso.

9).- DROP.- Suprime un comando de archivo RDSQL desde la base de datos.

10.- EXIT.- Regresa al Menú de informix.

Las declaraciones RDSQL son usadas para crear y manipular bases de datos relacionales. Las bases de datos son creadas con subdirectorios del directorio activo. El nombre del directorio es el nombre de la base de datos, con la extensión .dbs.

El subidrectorio de la base de datos contiene 19 sistemas de catálogos. Informix usa el sistema de catálogos para mantener la trayectoria de tablas, columnas, índices, vistas, sinónimos y los permitidos en cada base de datos. El subdirectorio de base de datos solo contiene las tablas que constituyen la base de datos. Cada una de estas tablas está representada por archivos de datos y archivos indexados con la extensión .dat y .idx respectivamente.

DECLARACIONES:

Cinco diferentes tipos de declaraciones en RDSQL son usadas con informix.

1.- Definición de datos.

2.- Manipulación de datos.

3.- Acceso a los datos.

4.- Integridad de los datos.

5.- Auxiliares.

Declaración de datos.- Para crear y suprimir en una base de datos: tablas, ventanas, índices y declaraciones que modifican o renombran tablas o columnas, solamente se requieren las declaraciones antes de manipular los datos de una base de datos existente.

CREATE DATABASE crea un directorio de base de datos, dentro del sistema de catálogos y hace la nueva base de datos al propio tiempo que la activa.

DATABASE selecciona una base de datos y la hace la base de datos activa. Esta no puede ser mas que una base de datos en uso a un tiempo.

CLOSE DATABASE cierra los archivos de la base de datos activa y no permite su uso. Solamente 3 declaraciones de RDSQL son permitidas cuando esta no es la base de datos activa.

CREATE DATABASE
DATABASE
DROP DATABASE
DROP DATABASE suprime todas la tablas, índices, sistema de catálogos, y el subdirectorio de la base de datos.
CREATE TABLE crea una tabla y define las columnas y sus tipos de datos.
ALTER TABLE agrega y suprime columnas desde una tabla y modifica tipos de datos de columnas.
RENAME TABLE cambia el nombre de una tabla.
DROP TABLE suprime todos los datos e indexados de una tabla y cancela sus entradas en el sistema de catálogos.
CREATE VIEW diseña una tabla seleccionada desde renglones y columnas de tablas y vistas existentes, así como los cambios de datos de una línea en una tabla, aunque la ventana no esta construida sobre ellos.
DROP VIEW suprime la definición de la vista desde el sistema de catálogos, junto con algunas vistas que se le relacionan.
DROP SYNONYM suprime un sinónimo desde el sistema de catálogos.
CREATE SYNONYM define un nombre alternativo para una tabla o una vista. Un sinónimo es efectivo solamente para el usuario que creó éste y puede ser suprimido solamente por él. De modo que si alguna persona quiere usar un sinónimo deberá crear el propio.
RENAME COLUMN cambia el nombre de una columna.
CREATE INDEX crea un índice en una o más columnas de una tabla.
DROP INDEX suprime un índice.
UPDATE STATISTICS agrega datos estadísticos en tablas en el sistema de catálogos.

Declaración de Manipulación de Datos.

Las declaraciones mas frecuentemente usadas son las declaraciones de manipulación de datos:

DELETE borra uno o mas renglones de una tabla.
INSERT agrega uno o mas renglones a una tabla.
SELECT recupera datos dentro de una base de datos.
UPDATE Modifica los datos en uno o más renglones de una tabla.
SELECT es la declaración más compleja pero también las más importante, ya que a con ella se pueden realizar las consultas.

Declaración de Acceso de Datos.

Estas son efectivas solamente en sistemas multi-usuarios.

RDSQL provee las siguientes declaraciones afectando el acceso a los datos:
GRANT concede el acceso a base de datos especificando a los usuarios permitidos o si es público.
REVOKE modifica el acceso a base de datos de usuarios privilegiados o públicos.
LOCK TABLE limita el acceso a la tabla para el usuario corriente o permite a otros usuarios solamente leer la tabla. Usar esta declaración únicamente para hacer cambios mayores desde una tabla en un contexto multi-usuario y cuando la interacción simultánea con la tabla por otro usuario interfiera. Esta declaración disminuye la accesibilidad a la base de datos porque previene que otros usuarios tengan acceso a la tabla.
Si se hicieron transacciones a la base de datos, tendrá que hacerse una declaración de BEGIN WORK antes de tomar como salida la declaración LOCK TABLE.
UNLOCK TABLE devuelve el acceso a una tabla previamente vista.

Declaraciones de Integridad de datos.

RDSQL prevé la integridad de los datos con la instrumentación de los conceptos de rastreo y transacción. Un rastreo es un archivo que contiene una historia de adiciones, borrados, actualizaciones y manipulación de una tabla de una base de datos. En la ocurrencia de un sistema fracasado, se puede usar el rastreo para facilitar la reinstalación de una tabla desde una copia backup de la tabla y un archivo de rastreo.

Una transacción es una serie de operaciones (declaraciones en RDSQL) en una base de datos que va a ser completada como una unidad simple de trabajo o no. Por ejemplo, las transacciones son necesarias en el campo de mantenimiento de libros contables, cuando múltiples operaciones en varias cuentas diferentes hechas como una unidad o los libros estarán fuera de balance. Usando Informix se pueden archivar los datos íntegramente manteniendo un archivo de todas las operaciones que modifican los contenidos de una base de datos. Se tiene la opción para asumir estos cambios en la base de datos o para regresar a las acciones y permitir la base de datos intocable. Pero el rastreo y transacción afecta el funcionamiento del sistema y el almacenamiento. Por esta razón se tendrá cuidado y consideración antes de usarlos.

Las siguientes declaraciones son usadas en proceso de mantenimiento de la integridad de datos:

CREATE AUDIT crea un archivo rastreador.

DROP AUDIT suprime un archivo rastreador desde la base de datos.
RECOVER TABLE recupera una tabla de base de datos desde una copia backup y un archivo rastreador.
BEGIN WORK marca el principio de una transacción.
COMMIT WORK marca el fin de una transacción autorizando todos los cambios en la base de datos desde la última declaración de BEGIN WORK. Libera todos los renglones y cierra tablas.
ROLLBACK WORK marca el fin de una transacción, revocando todos los cambios de la base de datos desde la última declaración BEGIN WORK.
START DATABASE inicia una nueva transacción recorriendo archivos.
ROLLFORWARD DATABASE usa una transacción recorriendo archivos para restaurar una base de datos desde la última copia backup.

Declaraciones auxiliares.

Las siguientes declaraciones son extensiones RDS al SQL.:

CHECK TABLE revisa los índices de una tabla y determina si están dañadas.
REPAIR TABLE repara los índices de una tabla si es que están dañados.
LOAD lee un archivo dentro de una base de datos en sistema formato ASCII.
UNLOAD escribe los contenidos de una tabla de base de datos en texto ASCII para un sistema de archivo.
INFO despliega la información de ayuda para la base de datos, en el sistema de catálogos.
OUTPUT dirige la entrada y salida de una declaración SELECT desde un sistema de archivos, a la impresora.

A manera de resumen, en esta parte, tenemos que mientras Informix y Oracle utilizan SQL, como el lenguaje de interfase entre el usuario y la base de datos, DB2 utiliza completamente SQL y DBASE IV tiene otra forma de funcionamiento, donde el uso de SQL es opcional (aunque su uso es recomendable para disminuir posibles errores y hacer códigos más entendibles para operarios que sepan programar o no).

Por otro lado, es importante mencionar que Informix y Oracle funcionan para diversas plataformas, lo que les permite ser muy transportables. Esto no sucede con DBase IV y DB2.

CONCLUSIONES

Refiriéndonos a las bases de datos relacionales, su estructura permite al usuario ver sus datos de una manera clara, ya que su manipulación, definición y control puede hacerse eficiente y se logra tener una base de datos sin redundancia, i,e., no hay porque duplicar datos, en este sentido se ha avanzado grandemente pues se reduce el espacio de almacenamiento de ellos y en consecuencia el tiempo de máquina para su localización y lo que se quiera hacer con ellos.

En Informix existe RDSQL que es una extensión de SQL para crear y consultar las bases de datos.

En Oracle todas las herramientas de acceso están basadas en SQL, y tiene un conjunto de extensiones para SQL. Utiliza comandos SQL para permitir el acceso, modificar y desplegar datos, aunque dispone de herramientas compuestas de bases de datos y programas funcionales que se pueden usar en forma independiente, además es compatible con DB2 y otros sistemas basados en SQL.

DB2 esta completamente basado en SQL, por lo que siempre es compilado nunca interpretado.

DBASE IV, tiene dos modos de acceso a las bases de datos, a saber, el modo tradicional Dbase y el modo Dbase/SQL en el cual son integrados los comandos de SQL para una definición y rápida manipulación de los datos, y es para lo que se recomienda su uso.

Actualmente no podemos decir que un sistema manejador de Base de Datos es mejor que otro si este juicio se hace fuera de contexto, ya que para optar por cualquiera de ellos se deben considerar las características que más se adecúen al uso que se pretenda dar a las bases de datos a manejar y que pueden ser entre otras:

a)Requerimiento mínimo de equipo.

b)Capacidad de interactuar con otros lenguajes.

c)Recuperación de archivos y tablas.

d)Sistema multi-usuarios.

e)Existencia de usuarios privilegiados.

f)Modo interactivo.

g) Modo compilado.

h) Costo.

CARACTERISTICAS	DB2	DBASEIV	INFORMIX-SQL	ORACLE PROFESIONAL
Requerimiento Mínimo de Equipo	Micro	Micro	Micro(Funcionan también en máquinas de más capacidad)	Micro
Capacidad de interactuar con otros lenguajes	PL/I, Cobol, Fortran Ensamblador	C,Basic,Pascal	C,Cobol,ADA Fortram	Cobol, C
Recuperación de archivos y tablas	sí	sí	sí	sí
Sistema multi-usuarios	no	no	sí	sí
Usuarios privilegiados	sí	sí	sí	sí
Modo interactivo	sí	sí	sí	sí
Modo compilado	sí	no	sí	no
Costo *	4	3	1	2

Lo importante es que, no conociendo las bases de SQL y dado que la mayoría de los Sistemas Manejadores de Bases de Datos está tendiendo a utilizarlo, el familiarizarnos con este tipo de SMBD resulta muy importante.

SQL es un paso muy grande en el desarrollo de software para los sistemas manejadores de bases de datos relacionales, ya que provee un eslabón para acceder a los datos y es una herramienta útil para quienes manejan grandes volúmenes de datos utilizados para diferentes propósitos ya que su manejo llega a ser sencillo, y esta es probablemente una de las razones por las cuales diferentes compañías, dedicadas a la elaboración de paquetería para bases de datos relacionales han ido adoptando cada vez mas SQL dentro de sus sistemas manejadores de bases de datos, por lo que se puede esperar que en los 90's SQL llegue a ser adoptado en más sistemas, con lo que probablemente se logre que los usuarios puedan utilizar diferente paquetería sin tener que aprender su uso partiendo desde cero.

* La variación del costo está dado en función del más alto (1) al más bajo (4).

APENDICE A

COMANDOS DE SQL

Comandos de Manipulación de datos (DML):

DELETE Borra uno o más renglones desde una tabla o una vista.
INSERT Agrega nuevos renglones a una tabla o vista.
SELECT Funciona una consulta y selecciona renglones y columnas desde una o más tablas o vistas.
UPDATE Cambia los valores de los campos en una tabla o vista.

Comandos de Definición de datos (DDL):

ALTER PARTITION Agrega un archivo a la base de datos.
ALTER SPACE Altera una definición de espacio. ALTER TABLE Agrega una columna, o la redefine en una tabla existente.
COMMENT Inserta un comentario en el directorio acerca de una columna o tabla.
CREATE CLUSTER Crea un cluster el cual puede contener dos o más tablas.
CREATE INDEX Crea un índice para una tabla.
CREATE SYNONIM Crea un sinónimo para el nombre de una tabla o vista.
CREATE VIEW Define una vista de una o más tablas o de otra vista.
DROP Suprime un cluster, tabla base, índice, sinónimo, vista, desde la base de datos.
RENAME Cambia el nombre de una tabla, vista o sinónimo.
VALIDATE INDEX Checa la integridad del índice de una tabla.

Comandos de Control de Datos (DCL):

AUDIT Rastrea el uso de una tabla, vista, sinónimo
GRANT Concede privilegios en el sistema a usuarios sobre objetos de la base de datos.
LOCK TABLE Permite el acceso a la tabla para múltiples usuarios mientras se mantenga la integridad de las tablas.
NOAUDIT Regresa el efecto del AUDIT previo, parcial o completamente.
REVOKE Revoca privilegios de usuarios o bases de datos del sistema.

BIBLIOGRAFIA

INTRODUCCION A LAS BASES DE DATOS
Mark L. Guillenson
Edit. McGraw Hill, 1987.

AN INTRODUCTION TO DATABASE SYSTEMS
C.J. Date, Vol. I.
Fourth Edition
Edit. Addison Wesley
USA, 1987.

ANALISIS Y DISEÑO DE SISTEMAS DE COMPUTACION (Notas de Curso).
Luis F. Castro, Miguel A. Guzmán, Jorge Lozano,
Américo Vargas.
Depto. Ingeniería Eléctrica
UAM-Iztapalapa, México, D.F.
Cap. IV, PP. 1-23
Abril, 1988.

AN INTRODUCTION TO SQL
Edward A. Dowgiallo
Micro/Systems
(artículo), Sept/1988
pp. 24-45

MASTERING DBASE IV PROGRAMMING
Towsend, Carl
Edit. SYBEX Inc.
USA, 1989.

DBASE IV SQL
Jack L. Hurch, & Caroly J. Hurch
Edit. Asthon-Tate Corporation
California USA, 1989.

UNDERSTANDING ORACLE
G. Lateer
Edit. SYBEX,
California, USA, 1989.

INFORMIX-SQL
Relational Database Management System. Manual Reference
Jul. 1987
. Printed in USA. Informix Software, Inc.

THE RELATIONAL MODEL FOR DATABASE MANAGEMENT: VERSION 2
E.F.Codd
Edit. Addison-Wesley
USA. 1990.