UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES

CAMPUS ARAGÓN

OPCIÓN DE TITILACIÓN:

SEMINARIOS Y CURSOS DE ACTUALIZACIÓN Y CAPACITACIÓN PROFESIONAL

T E S I N A

PARA OBTENER EL GRADO DE:
INGENIERO EN COMPUTACIÓN

P R E S E N T A

Omar Rafael Chong Badillo

A S E S O R
Enrique García Guzmán

MÉXICO 2009

IMPLEMENTACIÓN DE UNA BASE DE DATOS

INGENIERO EN COMPUTACIÓN

PRESENTA:

OMAR RAFAEL CHONG BADILO
MATRICULA 09906299-3

ASESOR:

ENRIQUE GARCÍA GUZMÁN

MEXICO D.F., ENERO 2009

Deseo agradecer...

Profundamente a la casualidad que la vida me otorgó al haberme puesto en un hogar maravilloso al nacer, el cual recuerdo ahora de manera nostálgica.

Sin el apoyo en todo sentido de mis padres y hermanos, el placer cotidiano de vivir sería simple monotonía.

Es difícil imaginar cómo sería el andar cotidiano sin recordar su comprensión, su apoyo inmenso y su amor.

Gracias a mis padres, hermanos y seres queridos por compartir y dedicar gran parte de sus vidas conmigo y por darme aliento para la ardua tarea de caminar hacia la perspectiva de un nuevo día; de verdad serán inolvidables.

Quiero agradecer muy especialmente a Jocelyn, mi querida niña, que durante bastante tiempo tuvo la paciencia suficiente para apoyarme profundamente, para darme su comprensión, su cariño y su amor. Gracias por hacer de esos momentos un verdadero vivir.

Gracias a todos por hacer de esto algo especial, que me ha costado mucho sacrificio y que me ha dejado un gran aprendizaje, pero sobre todo que me ha hecho salir adelante como persona.

Chong Badillo Omar Rafael

ÍNDICE

INTRODUCCIÓN

El mundo en el que nos desenvolvemos y los estilos de vida que se desarrollan en las enormes urbes, nos conducen necesariamente a optimizar infinidad de elementos como tiempos, espacios, recursos y una de las fuentes más importantes de vida y desarrollo humano: la información. La transmisión de información y el conocimiento que se va adquiriendo, son un punto de partida para los avances en todas las áreas de interacción humana y la evolución de los pueblos. Nuestras vidas giran entorno a la información que recibimos por tal motivo es necesario que ésta cuente con una serie de características que nos permitan utilizarla, debe ser fidedigna, oportuna y suficiente. Sin embargo, es infinita la información que se obtiene del entorno, por tal motivo es necesario sistematizarla a través de distintas herramientas, una de ellas es la base de datos.

Las bases de datos constituyen una herramienta básica para el desarrollo del conocimiento en distintos campos como la ciencia, la medicina, el comercio, el hogar, el deporte, etcétera, en fin, su utilidad resulta básica para el desarrollo humano.

Las bases de datos son conjuntos de información perteneciente a un mismo contexto y almacenada sistemáticamente para su posterior uso, esto permite sea manejada de manera más clara, sencilla y ordenada, obteniendo así solo la información relevante.

La explotación óptima de las bases de datos depende de dos elementos básicos: la herramienta de explotación y el diseño de la base de datos, en este trabajo se abordan los métodos y lenguajes de explotación estándares para bases de datos, así como las premisas para realizar un diseño de bases de datos con calidad.

Actualmente contamos con herramientas de explotación de bases de datos muy sofisticadas, así como con sistemas manejadores de bases de datos que son capaces de administrar varios terabytes de información, por tal motivo en este trabajo retomaremos SQL (Standard Query Language) Server para demostrar como es el ambiente de una base de datos en un SMBD (Sistema Manejador de Base de Datos), además de hacer los tipos básicos de consultas con el lenguaje SQL, sin embargo, es necesario contar con una base de datos bien diseñada, para que la explotación no resulte complicada e ineficiente, aunque se utilice el sistema de explotación más sofisticado. El presente trabajo tiene como objetivo primordial demostrar la funcionalidad de las bases de datos y su manipulación, así como dejar ver el manejo óptimo de éstas.

Dentro de los alcances de este trabajo se encuentra poder definir y generar bases de datos que faciliten su explotación y que cumpla con las características de un buen diseño de bases de datos.

El documento de divide en cinco capítulos, el primero nos presenta los conceptos fundamentales, es decir, los componentes básicos que conforman una base de datos; en el segundo capítulo se menciona que es el Sistema Manejador de Base de Datos (SMBD) y su evolución desde los años 60' haciendo hincapié en la participación del Dr. Codd en estas investigaciones, se menciona la importancia del enfoque relacional y los elementos que lo conforman, así como se describen los principios indispensables en el diseño de una base de datos. El tercer capítulo nos menciona los modelos por los cuales se desarrolla una base de datos: el modelo lógico y el modelo físico, dejando ver al primero como el punto de partida para la implementación de una base de datos en los distintos tipos de SMBD y al segundo como el estado de representación de la base de datos en un lenguaje particular para generarse en un SMBD especifico, se habla de la normalización como una serie de reglas que sirven para ayudar a minimizar las dificultades del diseñador de una base de datos. En el cuarto capítulo se describe el lenguaje utilizado para manipular en un SMBD una base de datos y los tipos de sentencias que utiliza. Finalmente en el quinto capítulo se construye con todos los elementos analizados una base de datos con el lenguaje SQL.

CAPÍTULO I

FUNDAMENTOS DE BASE DE DATOS

IMPLEMENTACIÓN DE UNA BASE DE DATOS

En este apartado revisaremos los conceptos básicos asociados con las bases de datos, que son el fundamento sobre el cual se desarrollarán los demás temas.

1. BASE DE DATOS

Una base de datos es una colección de datos agrupados con cierto orden, lógica y coherencia, que persisten por un periodo relativamente largo. Típicamente las bases de datos están orientadas a un tema en particular, por ejemplo:

- Base de datos de indicadores económicos
- Base de datos de resultados electorales preliminares
- Base de datos de contabilidad
- Etcétera.

Toda base de datos se divide en elementos más simples que denominaremos tablas, a su vez las tablas se dividen en campos o columnas y los datos que se graban en las columnas son conocidos como registros o renglones. En la Figura 1 se grafican los elementos mencionados con anterioridad de una base de datos.

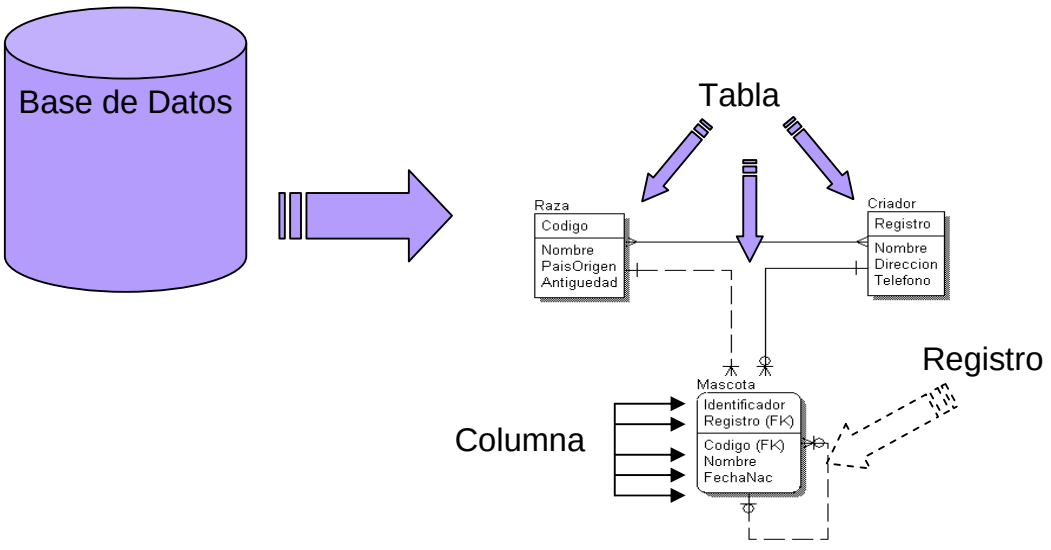


Figura 1. Componentes de una Base de Datos

En el paradigma relacional al cual nos enfocaremos mas adelante se establece la siguiente equivalencia:

Definición Genérica	Paradigma Relacional
Tabla	Entidad
Campo o Columna	Atributo
Registro o renglón	Tupla

1.1 DATOS

Ahora bien, comencemos por definir lo que es un dato: el dato es la unidad mínima de información que expresa una característica asociada a una entidad u objeto.

De la definición anterior podemos deducir ejemplos de dato, tales como color, edad, peso, antigüedad, temperatura, clave, etcétera.

Una característica importante de los datos es su relevancia, que es la relación que guardan con respecto al tema que se intenta describir. Podemos decir que un objeto tiene asociada una cantidad prácticamente infinita de datos, la mayoría de los cuales pueden resultar irrelevantes para el tema rector de la base de datos que deseamos construir, es aquí donde radica la importancia de evaluar la relevancia de los datos, para tomar la decisión acerca de incluirlos en nuestro modelo o dejarlos fuera.

Por ejemplo, el nombre de una persona y la antigüedad en su trabajo actual resultan relevantes para una base de datos de tarjetas de crédito, en cambio, para la base de datos de estadísticas de desempleo del INEGI, el primer dato es irrelevante, mientras que el segundo sí resulta de utilidad.

Es importante que al tomar datos o características de un objeto para incluirlos en nuestra base de datos, se tome en cuenta la orientación, objetivo o tema que rige su construcción.

Ahora bien, los datos de manera aislada difícilmente nos darán una utilidad real, cuando los datos se relacionan cobran un significado más profundo y generan información, de la cual podemos obtener mayor utilidad, tanto para la operación cotidiana como para la toma de decisiones.

1.2 TABLA

La tabla es la estructura que contiene uno o más datos y que está compuesta por columnas y renglones. El nombre de una tabla debe reflejar la relación que guardan los datos que contiene y debe ser congruente con algún estándar de nomenclatura definido para ello. Se recomienda que el nombre sea un sustantivo en singular.

Código #	Nombre	País Origen	Antigüedad
1	Rafael Medina	Alemania	10/10/1820
2	Carlos Torres	México	05/11/1600

La tabla debe poseer una llave de identificación, llave primaria, que evite la duplicidad de la información que contiene. La llave primaria de una tabla puede estar formada por una o más columnas de la tabla, existen casos en los que la llave primaria se forma con la totalidad de sus columnas.

1.3 COLUMNA

Las columnas están formadas de dos elementos, su nombre y los valores que se registran en ella.

Código #	Nombre	País Origen	Antigüedad
1	Rafael Medina	Alemania	10/10/1820
2	Carlos Torres	México	05/11/1600

Los nombres de las columnas deben ser representativos de los datos que se registrarán en ella y deben obedecer a un estándar de nomenclatura.

1.4 RENGLÓN

Un renglón se forma por los valores registrados en cada una de las columnas contenidas en una tabla. Como se mencionó anteriormente, los renglones en una tabla no deben estar duplicados.

El conjunto de renglones dan la profundidad de la tabla, que es un elemento básico en la estimación del espacio físico necesario para alojar la base de datos.

CAPÍTULO II

ANÁLISIS Y DISEÑO DE UNA BASE DE DATOS

2. SISTEMA MANEJADOR DE BASE DE DATOS (SMBD)

Un sistema manejador de bases de datos, SMBD por sus siglas en inglés, es una poderosa herramienta que permite crear y administrar grandes cantidades de información, de manera segura, eficiente y dotándole de persistencia por periodos largos.

Existen tres capacidades básicas que todo SMBD debe cumplir:

1. Almacenamiento persistente. De manera análoga a un sistema de archivos, un SMBD soporta el almacenamiento de cantidades muy grandes de datos, que existen independientemente del uso que haga de ellos cualquier proceso que los acceda. Sin embargo, un SMBD va más allá de la flexibilidad de un simple sistema de archivos porque provee eficiencia en el acceso a las estructuras que almacenan cantidades considerables de datos.

2. Interfaz de programación. Un SMBD debe permitir a un usuario o un programa de aplicación, el acceso y modificación de datos a través de un lenguaje de consulta. Nuevamente, el SMBD aporta mayor flexibilidad que la simple lectura / escritura de un sistema de archivos.

3. Administración de las transacciones. Un SMBD soporta el acceso concurrente a los datos, es decir, acceso simultáneo de distintas transacciones al mismo tiempo. Para evitar las consecuencias indeseables del acceso simultáneo, el SMBD debe tener la capacidad de aislamiento, la apariencia de que las transacciones se ejecutan una a la vez y la atomicidad, esto es, el requerimiento de que las transacciones se ejecuten completas o se reversen en caso de no completarse. Una última característica relacionada con las transacciones es la durabilidad, interpretada como la capacidad de recuperación de pérdidas debidas a errores de distintos tipos.

2.1 EVOLUCIÓN DE LOS SMBD

Los primeros SMBD aparecieron al final de los 60s como un desarrollo de los sistemas de archivos. Estos sistemas ya proveían algunas características como la capacidad de almacenamiento y la protección de uso indebido o no autorizado, pero no garantizaban que la información no respaldada no fuese susceptible de perderse o que se pudiese tener acceso a información cuya localización exacta se desconociera.

Por otra parte, estos sistemas de archivos no soportaban directamente un lenguaje para consultas y modificaciones a los datos contenidos en los archivos.

Finalmente, los SMBD basados en sistemas de archivos, no satisfacían el control concurrente de usuarios simultáneos ni protegían los datos de corrupción debida a accesos accidentales.
Posteriormente, en los 70s y antes de la publicación de sus doce reglas en 1984, el Dr. Codd, investigador de IBM, sentó las bases de lo que serían los SMBD relacionales, provocando un cambio sustancial en la manera en que se manejaban las bases de datos hasta ese momento.

Codd propuso que los SMBD deberían presentar los datos como una vista de relaciones o tablas sobre la cual se pudiesen elaborar consultas con una respuesta eficiente. El usuario no requería conocer los detalles del almacenamiento físico de los datos para poder acceder a ellos.

La arquitectura de un sistema de base de datos está influenciada por el sistema informático que soporta la instalación del SMBD, lo que reflejará muchas de las características propias del sistema subyacente en el SMBD.

La aparición de las redes de computadores permitió separar tareas en un esquema cliente – servidor; con la aparición del multiprocesamiento, el procesamiento paralelo dentro del computador permitió acelerar algunas de las tareas de la base de datos, así como la posibilidad de ejecutar más transacciones por segundo.

De acuerdo al uso que los sistemas hacen de la tecnología, se pueden distinguir los siguientes tipos de SMBD.

- **Centralizados o Monousuario.** Se ejecuta en un único sistema computacional sin tener que interactuar con otros equipos. Generalmente no tienen control de concurrencia y sus sistemas de recuperación son precarios.

- **Cliente-Servidor.** Existen dos partes conceptuales, el servidor que atiende peticiones, valida accesos y administra transacciones y el cliente, que puede ser más de uno de forma simultánea, en él se encuentra la interfaz a través de la cuál el usuario accede a la base de datos.

- **Motores de base de datos multiprocesos.** Cada vez que un usuario se conecta a la base de datos, ésta inicia una nueva instancia de la aplicación de base de datos. Con el fin de coordinar a muchos usuarios que acceden los mismos conjuntos de datos, estos ejecutables trabajan con un coordinador global de tareas que planifica operaciones para todos los usuarios.

- **Motores de base de datos multihilos.** Abordan el problema del acceso multiusuario de una manera distinta, pero con principios similares. En lugar de confiar en que el sistema operativo comparta los recursos de procesamiento, el motor toma la responsabilidad por sí mismo, lo que en la práctica se asocia a una mejor portabilidad del sistema. Las ventajas de este

tipo de motores radican en una mayor eficiencia en el uso de recursos para determinadas plataformas.

- **SMBD Paralelos.** Constan de varios procesadores y varios discos conectados a través de una red de interconexión de alta velocidad (Cluster, NAS o SAN). Su rendimiento se mide con base en la productividad (throughput), que se entiende como el número de tareas que pueden completarse en un intervalo de tiempo determinado y el tiempo de respuesta (response time), que es la cantidad de tiempo que necesita para completar una única tarea a partir del momento en que se envíe.

- **SMBD Distribuidos.** Aquí la base de datos se almacena en varios equipos que se pueden comunicar a su vez por distintos medios de comunicación (desde redes de alta velocidad a líneas telefónicas). No comparten memoria ni discos y sus tamaños pueden variar tanto como sus funciones, pudiendo abarcar desde PC hasta grandes sistemas. Se denomina con el término de emplazamientos o nodos a todos aquellos equipos que pertenecen a un sistema distribuido.

2.2 REGLAS DE CODD

Con la numerosa aparición de SMBD en los 80s, comenzaron a surgir algunos que solamente almacenaban los datos en tablas sin tener la capacidad de asignar una llave primaria que evitara la duplicidad de la información y que garantizara el acceso a los registros requeridos.

Por lo anterior y para estandarizar los SMBD relacionales, en 1984 Codd emitió doce reglas que un verdadero sistema relacional debería cumplir.

Debido a la complejidad de las reglas, algunas de ellas no han podido ser cumplidas por los SMBD, pero estos sistemas se consideran más relacionales en la medida que cumplan con una mayor cantidad de las reglas definidas.

A continuación se enumeran y explican las doce reglas definidas por Codd.

Regla 0

Para que un SMBD se denomine relacional, debe usar sus capacidades relacionales exclusivamente para gestionar la base de datos.

Regla 1. De la Información

Toda la información en una base de datos relacional se representa explícitamente en el nivel lógico exactamente de una manera: con valores en tablas.

Regla 2. Del Acceso Garantizado

Para todos y cada uno de los datos (valores atómicos) de un SMBD relacional, se garantiza que son accesibles a nivel lógico utilizando una combinación de nombre de tabla, nombre de columna y valor de clave primaria.

Regla 3. Tratamiento Sistemático de Valores Nulos

Los valores nulos (que son distintos de la cadena vacía, blancos, 0, etcétera) deben soportarse en los SMBD totalmente relacionales, para representar información desconocida o no aplicable de manera sistemática, independientemente del tipo de dato.

Regla 4. Catálogo Dinámico en Línea Basado en el Modelo Relacional

La descripción de la base de datos (metadatos) se representa a nivel lógico de la misma manera que los datos normales, de modo que los usuarios autorizados pueden aplicar el mismo lenguaje relacional a su consulta, como lo hacen con los datos normales.

Regla 5. Del Sublenguaje de Datos Completo

Un sistema relacional debe soportar varios lenguajes y varios modos de uso de terminal (rellenar formularios, por ejemplo). Sin embargo, debe existir al menos un lenguaje cuyas sentencias sean expresables, mediante una sintaxis bien definida, como cadenas de caracteres y que sea capaz de soportar:

- Definición de datos
- Definición de vistas
- Manipulación de datos (interactiva y por programa)
- Limitantes de integridad
- Limitantes de transacción (iniciar, realizar, deshacer)

Regla 6. De Actualización de Vistas

Todas las vistas, que son teóricamente actualizables, se pueden actualizar por el sistema.

Regla 7. Inserción, Actualización y Borrado de Alto Nivel

La capacidad de manejar una relación base o derivada como un solo operando, se aplica no sólo a la recuperación de los datos (consultas), si no también a la inserción, actualización y eliminación de datos.

Regla 8. Independencia Física de Datos

Los programas de aplicación y actividades de terminal, permanecen inalterados a nivel lógico cuando quiera que se realicen cambios en las representaciones de almacenamiento o métodos de acceso.

Regla 9. Independencia Lógica de Datos

Los programas de aplicación y actividades del terminal permanecen inalterados a nivel lógico cuando quiera que se realicen cambios a las tablas base, siempre y cuando se preserve la información.

Regla 10. Independencia de Integridad

Los limitantes de integridad específicos para una determinada base de datos relacional deben poder ser definidos en el sublenguaje de datos relacional, y almacenables en el catálogo, no en los programas de aplicación.

Regla 11. Independencia de Distribución

Una SMBD relacional tiene independencia de distribución.

Regla 12. Regla de la No Subversión

Si un sistema relacional tiene un lenguaje de bajo nivel (un registro de cada vez), ese bajo nivel no puede ser usado para evitar (subvertir) las reglas de integridad y los limitantes expresados en los lenguajes relacionales de más alto nivel (una relación, conjunto de registros, de cada vez).

2.3 EL ENFOQUE RELACIONAL

El surgimiento del enfoque relacional se dio en la década de los 70's y se consolidó en la década de los 80's. Actualmente, este enfoque es el predominante en el diseño y operación de la mayoría de los DBMS comerciales, manteniéndose vigente debido a que ha perfeccionado su operación y ha evolucionado de acuerdo a las necesidades de los usuarios.

El enfoque relacional establece equivalencias entre el enfoque tradicional de bases de datos y su propuesta, definiendo una tabla o entidad como una relación, una columna como un atributo de esa relación y un registro como una tupla de la relación.

Para diseñar una base de datos debemos comenzar por identificar la información que deberá incluirse y cuáles son las relaciones entre los componentes de esa información, este es un principio de diseño que se denomina fidelidad y que definiremos más adelante. Asimismo, la estructura de la base de datos o esquema de la base de datos debe definirse de acuerdo a algún tipo de lenguaje o estándar.

Para el diseño de bases de datos, basado en el enfoque relacional, existe una serie de protocolos que deben cumplirse, uno de ellos es el modelo Entidad – Relación, que se describe a continuación.

2.4 MODELO ENTIDAD-RELACIÓN

El modelo de datos entidad - relación (E/R) esta basado en una percepción del mundo real consistente en objetos básicos llamados entidades y de relaciones entre estos objetos. Se desarrollo para facilitar el diseño de base de datos permitiendo especificación de un esquema de la empresa que representa la estructura lógica completa de una base de datos. El modelo de datos E/R es uno de los diferentes modelos de datos semánticos; el aspecto semántico de modelo yace en la representación del resultado de los datos. El modelo E/R es extremadamente útil para hacer corresponder los significados e interacciones de las nuevas empresas del mundo real con un esquema conceptual. Debido a esta utilidad muchas herramientas de diseño de bases de datos se basa en los conceptos del modelo E/R.

El modelo entidad relación es un modelo esencialmente gráfico, en el que se utilizan cuadros para definir a las entidades y flechas de distintos tipos para definir las relaciones entre las entidades.

2.4.1 ELEMENTOS

Existen tres elementos básicos en un diagrama Entidad – Relación (E/R):

Entidad. Es una abstracción de algún objeto que tiene relación con la base de datos que estamos diseñando.

Ejemplos de Entidad (Figura 2) son los siguientes:

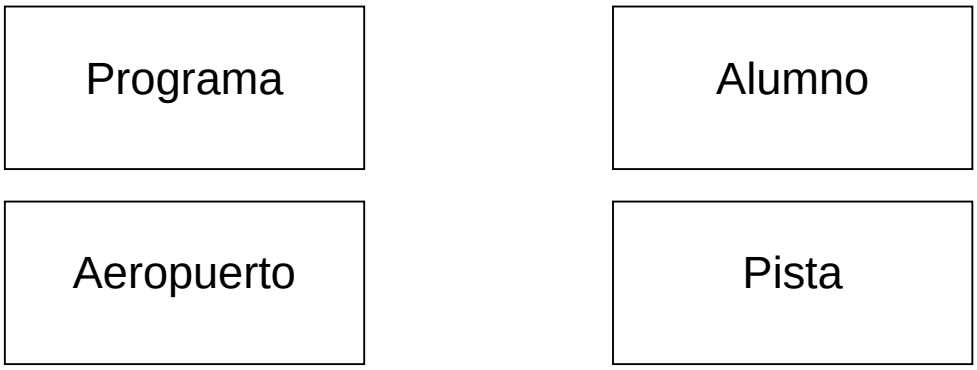


Figura 2. Ejemplos de Entidades

Atributo. Un atributo es una característica o propiedad asociada a una entidad. Los atributos pueden tomar valores de un determinado dominio, el cual representa al conjunto de valores que el atributo puede adoptar. Ejemplo de los atributos se observan en la Figura 3.

Los atributos se representan gráficamente como óvalos asociados a las entidades por líneas sólidas, por ejemplo:

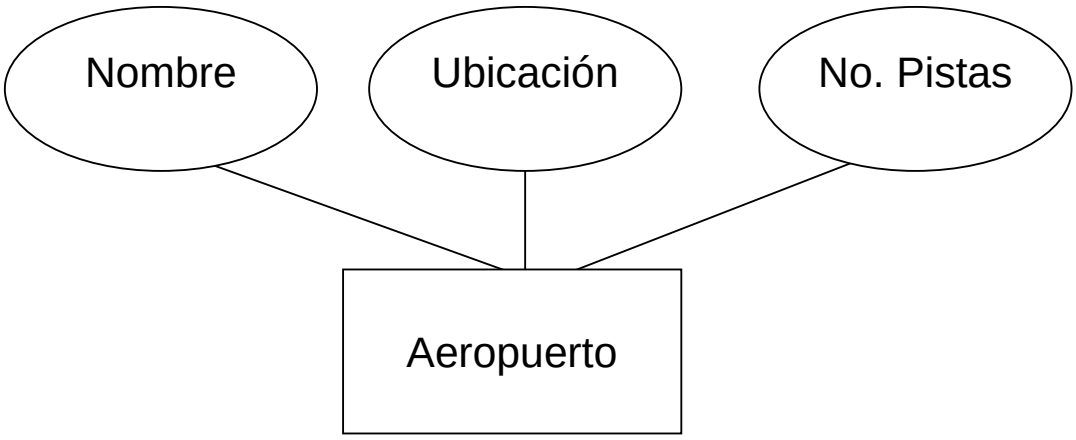


Figura 3. Atributos de una Entidad

Relaciones. Son conexiones entre una o más entidades. Cuando la relación se da entre una entidad consigo misma, se dice que es una relación recursiva.

Una relación binaria es aquélla que se establece entre dos entidades, como la entidad Aeropuerto y la entidad Pista de la Figura 3 y la relación Estelar y Película de la Figura 4.

Figura 4. Relación Binaria

Relaciones múltiples. Se caracterizan por conectar más de una entidad. Por ejemplo, se observa en la Figura 5 que para la producción de una película tendríamos:

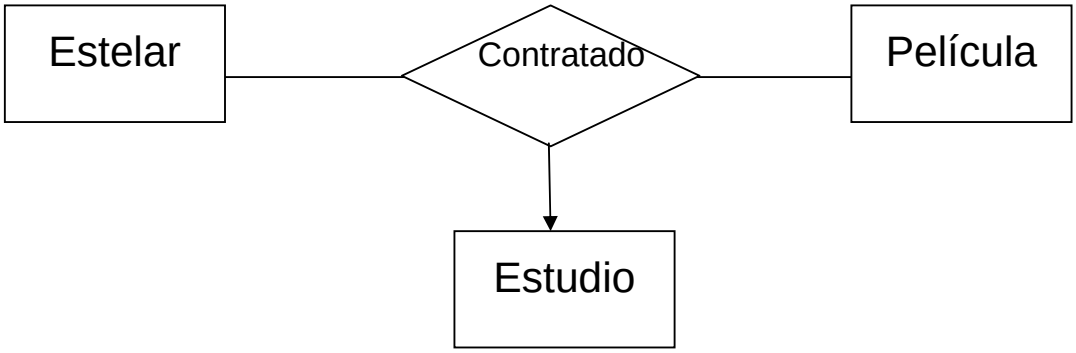


Figura 5. Relación Múltiple

Otra característica de las relaciones que debemos tomar en cuenta es la cardinalidad de la relación. Dependiendo del número de elementos que se relacionen de una entidad a otra, se definirá la cardinalidad en alguna de las siguientes tres categorías:

- **Relación Muchos a Uno**, se deberá leer como que un elemento de la entidad A, sólo se relaciona con un máximo de un elemento de la unidad B, mientras que un elemento de la entidad B puede estar relacionado con muchos elementos de la entidad A.

• **Relación Uno a Uno**, en ambos sentidos, de la entidad A a la B y de la B a la A, cada elemento se relaciona con un máximo de un elemento de la otra entidad.

• **Relación Muchos a Muchos,** este tipo de relación admite la existencia de que un elemento de la entidad A pueda relacionarse con muchos elementos de la entidad B y viceversa.

• **Relación requerida u opcional,** cada una de las tres relaciones anteriores puede tener la característica de ser requerida u opcional. Cuando es requerida significa que en la entidad referenciada deberá existir al menos un elemento con el que se relacione. Cuando es una relación opcional, el número de elementos puede ir desde cero hasta los que la relación indique.

Tomando como ejemplo la Figura 4 una película puede tener un máximo de un estelar, sin embargo un estelar puede ser estelar en muchas películas.

En la relación múltiple ejemplificada en la Figura 5, se debe entender que para una película y un estelar particular, puede existir un máximo de un estudio en el cual el estelar esté contratado para la película: en este caso también deberá leerse como que, un estudio puede contratar a varios estelares para una película y que un estelar puede contratarse con un estudio para más de una película.

En la Figura 6 se debe entender que un Área superior (de una empresa), tiene muchas Áreas subordinadas y que un Área subordinada tiene, como máximo, un Área superior. Existe otro tipo de relaciones que conectan a las entidades como subclases, este tipo de relaciones se conoce como relaciones *Es Un(a)*, donde se expresa especialización de las entidades y atributos comunes en una entidad padre. Un ejemplo de lo anterior se muestra en la Figura 6.

Como puede verse, la relación *Es Un* se representa a través de un triángulo cuya base está del lado del las subclases.

Las relaciones Muchos a Muchos deben resolverse mediante la conversión de la relación en una entidad, debido a que una relación de muchos a muchos no puede mapearse en el modelo físico, por lo que se considera una anomalía.

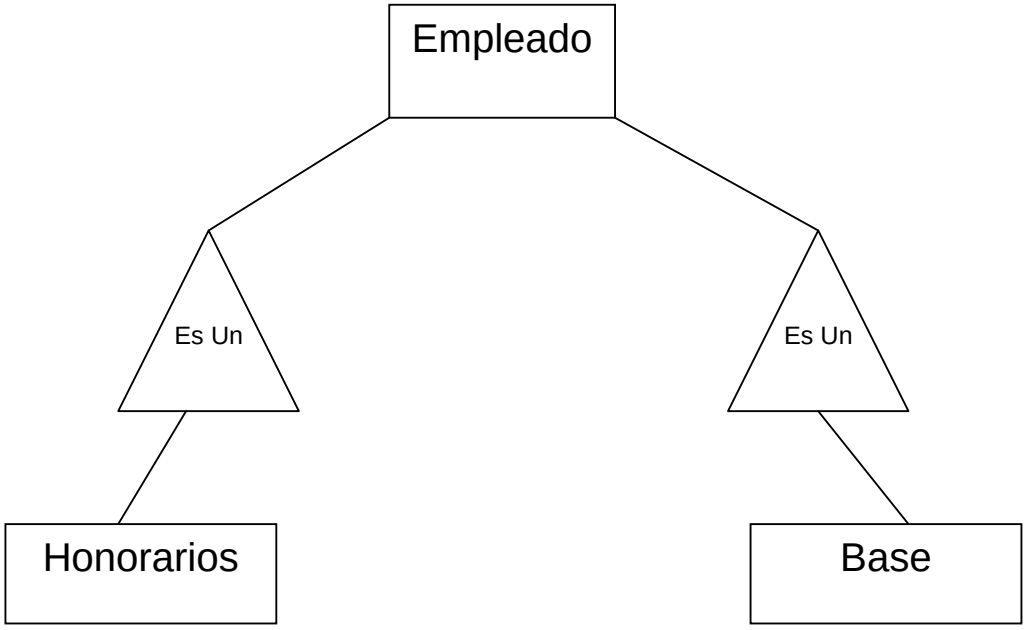


Figura 6. Subclases

2.4.2 PRINCIPIOS DE DISEÑO

Para dotar de calidad nuestros diseños relacionales, es necesario seguir algunos principios de diseño que pueden orientar mejor nuestro trabajo. A continuación se mencionan los principios indispensables de diseño en el enfoque relacional.

Fidelidad. Se refiere al hecho de incluir sólo elementos que se encuentren dentro de la definición de la aplicación, es decir, una entidad sólo debe tener asociados atributos que representen propiedades que dicha entidad pueda tener en el mundo real. Por ejemplo, la entidad Mascota no debería tener asociado el atributo Número de Licencia de Conducir, o bien, si estamos modelando una base de datos de nómina, la entidad Empleado no requiere incluir su frecuencia cardiaca ya que, aunque es un atributo del empleado, para efectos del objetivo que persigue una base de datos de este tipo, la frecuencia cardiaca es un dato irrelevante.

Otro ejemplo de una falta a este principio es el descrito en la figura 4, ya que en el mundo real una película tiene varios actores estelares y un actor puede ser estelar en muchas películas, por lo que la relación correcta debería ser una relación muchos a muchos (aunque después se resolviera con la conversión de esa relación en una entidad).

Eliminación de la redundancia. No debemos duplicar información en las entidades, es decir, si en el ejemplo de la figura 5 colocamos el atributo Nombre del estudio en la entidad Película y también lo colocamos en la entidad Estudio, no caeríamos en ninguna falta si evaluamos cada entidad de manera individual, no obstante, al evaluar el modelo en su conjunto

nos percataremos de que el atributo mencionado se duplica en dos entidades y que pertenece, por su naturaleza, a la entidad Estudio, mientras que la entidad Película puede acceder a él a través de la relación Contratado.

Simplicidad. Es necesario mantener el modelo simple y restringir las abstracciones a los fines prácticos que queremos representar. Para fines prácticos, resulta poco operante realizar modelos demasiado complejos que otros miembros del equipo de trabajo no comprendan o cuyo mantenimiento posterior se vuelva demasiado pesado.

Selección de las relaciones adecuadas. Aunque dos entidades pueden relacionarse de muchas formas, se debe seleccionar la relación que represente de una forma más eficaz la relación de las entidades, en congruencia con el propósito de la aplicación.

Seleccionar el tipo de elemento adecuado. Debe seleccionarse la manera más adecuada de representar los elementos que nos encontramos en la realidad, con los tipos de elementos reconocidos por el modelo E/R.

Por ejemplo se podría prescindir de la entidad Estudio de la figura 5, si agregamos el atributo Nombre del Estudio a la entidad Película y mantener la relación entre Estelar y Película, sin embargo, esta acción nos generaría redundancia en la entidad película.

En este caso, al seleccionar el elemento más adecuado se deberá tener en cuenta las ventajas y desventajas de cada decisión.

CAPÍTULO III

DESARROLLO DE UNA BASE DE DATOS

3. MODELADO DE REGLAS

En el mundo real y para representar de una manera más fiel la operación de las aplicaciones y las restricciones del negocio, debemos incluir reglas a nuestro modelo.

La cantidad y tipo de reglas que se incorporen al modelo no puede decidirse de manera arbitraria, siempre deberá estar supeditada a las directrices que guíen la operación del negocio.

Las reglas en el modelo E/R se pueden clasificar en los siguientes tipos:

Llave primaria. Está formada por el atributo o combinación de atributos que identifican de manera única cada tupla de la entidad, esto es, no pueden existir dos tuplas con los mismos valores en la llave primaria, aunque varíen en los valores de sus demás atributos. Cuando la llave primaria está formada por más de un atributo, se dice que es una llave compuesta. Esto no significa que una entidad tenga varias llaves: se trata de una sola llave compuesta por varios atributos.

En el caso de llaves compuestas, es aceptable que dos tuplas compartan valores en algunos de los atributos que forman la llave, pero no en la totalidad de ellos.

En el modelo E/R las llaves se representan subrayando el o los atributos que formen parte de ella.

Para evitar inconsistencias en la llave primaria se recomienda que, en medida de lo posible, sólo estén compuestas de datos de tipo numérico.

Valor único. Además de la llave primaria, existen otros atributos que por cuestiones del negocio o de la operación, deben presentar valores únicos en algunos atributos. Tal es el caso del número de seguridad social, el registro federal de contribuyentes, el CURP o el número de cuenta escolar, los cuales típicamente son alfanuméricos e irrepetibles.

Integridad referencial. Es una regla que garantiza la consistencia de la información almacenada en nuestra base de datos, ya que asegura la correspondencia entre atributos referenciados, esto es, que el valor registrado en el atributo X de la entidad A corresponda al valor existente en la llave primaria de la entidad B, a la cual hace referencia. El atributo en la entidad A se denomina Llave foránea y en la entidad B debe ser siempre una Llave primaria. Este tipo de regla evita que existan apuntadores a valores inexistentes, aunque puede aceptar valores nulos en la entidad que hace referencia. Gráficamente la integridad referencial se representa como un semicírculo en el extremo de la relación:

Dominios. Es la regla que liga un atributo con un conjunto o rango específico de valores válidos. Existen dominios genéricos como

los números enteros, los reales, los caracteres, etcétera y dominios personalizados, que restringen aún más los valores aceptables de acuerdo con las directrices del negocio, por ejemplo, el estado de un empleado se puede representar con una letra, sin embargo, podemos restringir que sólo sean dos caracteres válidos, del conjunto total de caracteres: A (para Activo) o I (para Inactivo).

Generales. Son reglas arbitrarias que deben cumplir los valores de los atributos de una entidad y están dictados por el negocio o por políticas propias de cada aplicación. Que una factura sólo pueda llevar cierta cantidad de artículos, es una regla general del negocio, que al exceder cierto monto se deba pedir la autorización de un supervisor, es otra regla general del negocio.

3.1 ENTIDADES DÉBILES

Cuando la llave primaria de una entidad está formada, total o parcialmente, por atributos que pertenecen a otra entidad, se dice que es una entidad débil.

La causa de la existencia de las entidades débiles se deriva de dos fuentes: primero, las subclases o relaciones de tipo *Es Un(a)*, recordemos que la subclase de entidad se deriva de una entidad padre de la cual obtiene su llave primaria, o parte de ella; segundo, las entidades que resuelven relaciones de cardinalidad muchos a muchos, en este caso son entidades derivadas que se componen de las llaves primarias de las entidades que relacionaban, a veces es posible agregar atributos propios.

La composición de la llave primaria de una entidad débil siempre cumple las siguientes reglas:

1. Cero o más atributos propios de la entidad

2. Uno o más atributos llave provenientes de las entidades que le dieron origen, ya sea por una relación *Es Un(*a) o por la resolución de una relación muchos a muchos. Las relaciones Muchos a uno que recibe la entidad débil se llaman Relaciones de soporte.

Para considerar que una relación es de soporte, deberá cumplir con las siguientes características:

1. Debe ser una relación Muchos a uno de la entidad débil a la entidad fuerte.

2. Debe tener integridad referencial de la entidad débil a la entidad fuerte.

3. Los atributos que la entidad fuerte aporta a la entidad débil deben ser llave primaria de la entidad fuerte (definición de llave foránea).

4. Si la entidad fuerte es a su vez, una entidad débil, los atributos que aporta deberán pertenecer a la llave primaria de la entidad fuerte que le dio origen, esta regla aplica para todas las entidades débiles de manera recursiva.

5. Si existe más de una relación de soporte que entre a la entidad débil, cada una de esas relaciones aportará la llave primaria de la entidad fuerte como parte de la llave de la entidad débil. Generalmente, la llave primaria de una entidad débil es una llave compuesta.

3.2 MODELO LÓGICO

Una vez que se ha completado el modelo E/R, el siguiente paso es traducir ese modelo al modelo lógico que tiene por objeto dar una visión más cercana a lo que será la base de datos implementada físicamente en algún SMBD específico.

Es importante resaltar que a pesar de que SQL (Standard Query Language) define la forma en que se debe hacer la interacción con bases de datos relacionales, cada SMBD tiene particularidades en la implementación de SQL que lo diferencian de los demás SMBD.

El modelo lógico es el punto de partida para la implementación de la base de datos en los distintos tipos de SMBD. Para la traducción del modelo E/R en el modelo lógico se utilizan símbolos análogos.

Las entidades se siguen representando como una caja, con las siguientes características:

Nombre Entidad	
Atributo 1	Tipo de dato
Atributo 2	Tipo de dato
Atributo 3	Tipo de dato
…	…
Atributo n	Tipo de dato

Figura 7.

De la Figura 7 podemos resaltar varias características importantes:

1. Debemos contar con un estándar de nomenclatura.

2. El nombre de la entidad debe estar definido en singular y encuentra fuera de la caja de la entidad.

3. Los atributos no deben tener un nombre demasiado largo ni incluir caracteres especiales.

4. Se debe especificar un tipo de dato genérico, aunque este no se despliegue en el modelo:

Tipo de Dato	Significado
A(n)	Alfabético de n caracteres
N(n.m)	Numérico de n dígitos enteros y m decimales
Date time	Fecha y hora
B	Binario

5. Cuando el atributo es llave primaria o forma parte de la llave primaria, se coloca en la parte superior de la caja, separándose del resto de los atributos con una línea.

6. Los atributos que no forman parte de la llave primaria se colocan en la sección inferior de la caja.

La representación de las relaciones cambia también, las flechas que utilizábamos se modifican quedando como líneas simples para la relación de uno a uno y como flechas con punta triple para denotar una relación de muchos. Esta representación se conoce como IE (Information Engineering) y es la que utilizaremos en este curso porque es la de uso más extendido, aunque hay que mencionar que existen otras como la IDEF1X (Integration Definition for Information Modeling) que también es ampliamente aceptada.

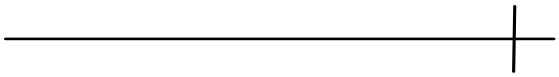
Las siguientes figuras muestran la representación gráfica de los diferentes tipos de relaciones existentes:

Herencia de llaves. Cuando una entidad hereda su llave a otra, como parte de su llave primaria, se usan líneas sólidas para representar la relación, independientemente de la cardinalidad que

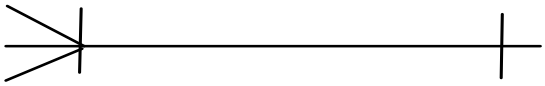
presenten, a este tipo de relaciones se les conoce como identificadas. Si la llave heredada no formará parte de la llave primaria de la entidad receptora, la línea se vuelve punteada y se le conoce como una relación no identificada.

Cardinalidad. Los tipos de cardinalidad vistos anteriormente se representan de la siguiente manera:

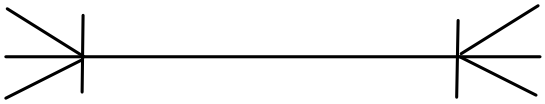
Relación Uno a Uno, la línea vertical indica la entidad que hereda su llave.



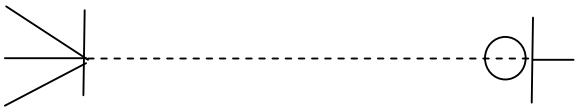
Relación Uno a Muchos, la punta triple debe ponerse del lado de la entidad que recibe el atributo heredado, sea parte de su llave primaria o sólo una llave foránea.



Relación Muchos a Muchos, existe una punta triple de ambos lados. Esta relación puede resolverse desde el modelo lógico, si se requiere agregar atributos propios a la entidad resultante de la relación, o bien, hasta el modelo físico.



Valores nulos. Cuando en la entidad que recibe el atributo referenciado puede no existir correspondencia hacia la entidad que hereda, se coloca un círculo en el extremo de ésta última, como se muestra en la siguiente figura:



Es importante mencionar que la lógica del modelo sólo permite que se representen valores nulos en relaciones no identificadas, debido a que ninguna llave primaria permite el uso de valores nulos, ya sea en su totalidad o en alguno de los atributos que la integran, en caso de llaves primarias compuestas.

La siguiente figura muestra un ejemplo de un modelo lógico de datos, en el cual se resaltan todos los elementos explicados anteriormente y algunos adicionales. (Figura 8)

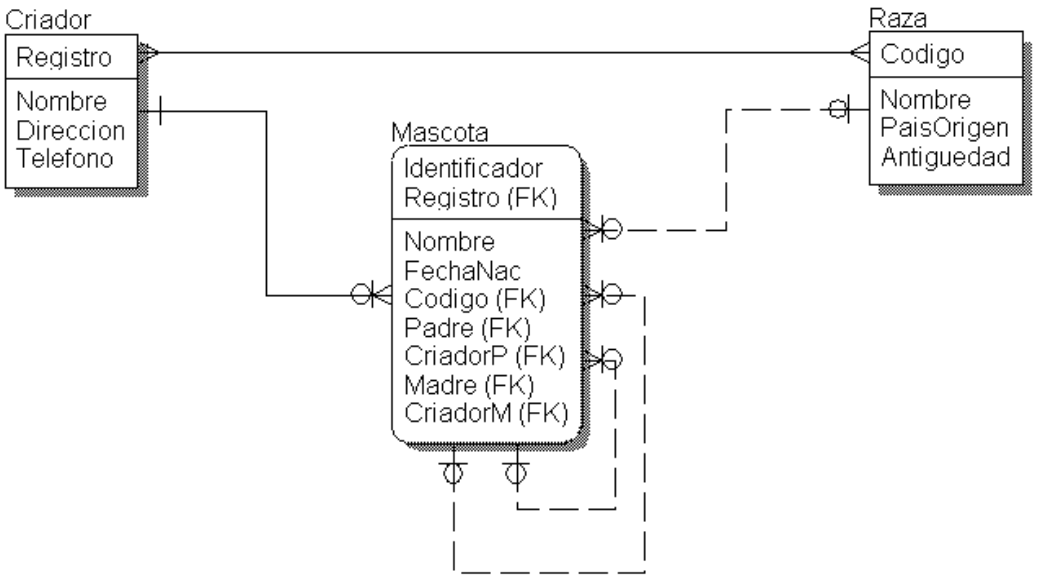


Figura 8.

En el modelo tenemos tres entidades, con una relación muchos a muchos entre Raza y Criador, la cual dará origen a una nueva entidad en el modelo físico.

La relación de Raza a Mascota indica que pueden existir varias mascotas con la misma raza, pero que una mascota siempre va a tener sólo una raza. Como la línea es punteada entendemos que el atributo Código de Raza no formará parte de la llave primaria de Mascota, esto es, sólo es una llave foránea en Mascota.

Por su parte, Mascota tiene dos relaciones recursivas consigo mismo, que indica que tanto el padre como la madre de una mascota pueden estar registrados en la misma tabla y que esa información no será parte de la llave primaria de mascota, por ser una relación no identificada. A este tipo de relaciones también se les conoce como relaciones de Rol, porque el nombre de los atributos no se puede duplicar y se reasigna de acuerdo al rol que juegan, esto es, Identificador y Registro cambian a Id_Padre y Reg_Padre, respectivamente, para referenciar recursivamente al registro del padre de la mascota; de la misma manera Id_Madre y Reg_Madre hacen referencia recursiva hacia el registro de la madre de la mascota.

El modelo relacional no permite la existencia de relaciones “Muchos a Muchos”, por lo que se deben resolver mediante una entidad intermedia.

Por último, la relación entre Criador y Mascota indica que un criador puede tener muchas mascotas registradas y que la mascota también es identificada por el número de registro del criador, es decir, forma parte de su llave primaria.

3.3 MODELO FÍSICO

El modelo físico es la representación de la base de datos en un lenguaje particular para generarse en un SMBD específico. Es importante mencionar que, aunque metodológicamente se debe tener el antecedente de los modelos E/R y lógico, es posible diseñar directamente el modelo físico, sin la existencia de modelos previos.

La representación gráfica de este modelo es muy similar a la del modelo lógico, con la diferencia de que en este se resuelven las relaciones de muchos a muchos y se visualizan los tipos de datos específicos de cada SMBD.

A continuación veremos en la Figura 9 el modelo lógico de la figura traducida a algunos de los SMBD más comerciales, nótese que la única variación se presenta en los tipos de dato que se manejan.

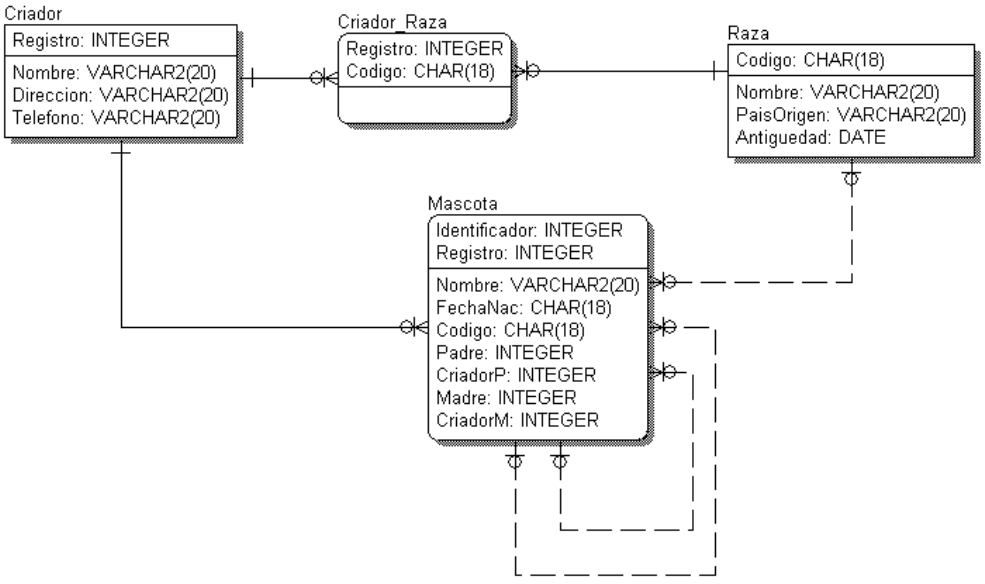


Figura 9.

Como puede observarse, la entidad que resuelve la relación muchos a muchos toma automáticamente el nombre de las dos entidades que relacionaba y está compuesta por la llave de cada una de ellas: es una entidad débil, como explicamos en secciones anteriores.

3.4 REGLAS DE INTEGRIDAD REFERENCIAL

Además de la integridad de datos que se debe tener al interior de las tablas y que está definida por las llaves primarias, los dominios y las demás reglas que se han explicado anteriormente, la integridad referencial asegura el mantenimiento de esta integridad al exterior de la tabla y la consistencia de la información entre tablas referenciadas.

La integridad referencial asegura que los valores registrados en los atributos definidos como llave foránea, tengan correspondencia con los valores definidos como llave primaria en la tabla a la que se haga referencia, dotando de calidad a la información almacenada. Existen diferentes estrategias para lograr la integridad referencial entre tablas, a continuación se explicará cada una de ellas:

Integridad referencial por restricción. Esta estrategia no permite eliminar registros en la tabla padre si existen registros coincidentes en la tabla hija, esto es, cuando se quiere eliminar un registro, cuya llave primaria es una llave foránea en otras tablas, el SMBD no permite la eliminación. Esta estrategia se instrumenta en relaciones de herencia de llaves, es decir, cuando la llave heredada no sólo es un atributo más en la tabla hija, sino que se vuelve parte de su llave primaria.

Integridad referencial en cascada. Esta estrategia consiste en que cuando se elimina un registro en la tabla padre, cuya llave primaria es una llave foránea en otras tablas, los registros correspondientes en las tablas hijas son eliminados también. Un ejemplo de ello es la eliminación de una tupla de la tabla Raza, con esta acción se eliminarán automáticamente las tuplas de las tablas Raza_Criador y Mascota, que tengan correspondencia con la raza eliminada. Un efecto similar existirá si eliminamos una tupla de Criador. Por el contrario, si eliminamos una tupla en Mascota, no se eliminarán ni la raza ni el criador correspondientes de las tablas Raza y Criador.

I**ntegridad referencial por omisión (valor default).** Podría ocurrir que, por reglas del negocio, se asigne un valor por omisión para aquéllas tuplas cuyo valor de llave foránea se elimine, en este caso el valor por omisión también deberá corresponder a una tupla existente en la tabla padre. Por ejemplo, el negocio podría decidir que si se elimina una raza, las mascotas de dicha raza no se eliminen sino que se les asigne un valor por omisión cuyo significado sea algo como 'Por definir', hasta que se acepte otro estándar de raza válido.

Esta estrategia normalmente se delega a la aplicación y no al SMBD.

Integridad referencial con valores nulos. Esta última estrategia consiste en que no se eliminen las tuplas correspondientes, ni se asigne un valor por omisión, sino que se asigne un valor nulo en el atributo de la llave foránea que fue eliminada.

El valor nulo no debe confundirse con un espacio en blanco o con un cero, el valor nulo es un valor vacío. En el ejemplo anterior en lugar de asignar un valor por omisión en Mascota, simplemente se coloca un valor nulo hasta que se defina la raza de las mascotas que se quedaron sin una raza específica.

La decisión de adoptar una u otra regla, para garantizar la integridad en las tablas, depende de la naturaleza de la relación que exista y de las reglas del negocio.

3.5 NORMALIZACIÓN

La normalización es el proceso mediante el cual se trasforman datos complejos a un conjunto de estructuras de datos mas pequeñas, que además de ser mas simples y mas estables, son mas fáciles de mantener. También se puede entender la normalización como una serie de reglas que sirven para ayudar a los diseñadores de bases de datos a desarrollar un esquema que minimice los esquemas de lógica, cada regla esta basada en la que le antecede. La normalización se adopto porque el viejo estilo de poner todos los datos en un solo lugar, como un archivo o una tabla de la base de datos, era ineficiente y conducía a errores de lógica cuando se trataba de manipular los datos. La normalización también hace las cosas fáciles de entender, los seres humanos tenemos la tendencia de simplificar las cosas al máximo, lo hacemos con casi todo, desde los animales hasta con los automóviles, vemos una imagen de gran tamaño y la hacemos más simple agrupando cosas similares juntas. Las guías que la normalización provee crean el marco de referencia para simplificar una estructura de datos complejo.

Otra ventaja de la normalización de bases de datos es el consumo de espacio, una base de datos normalizada ocupa menos espacio en un disco que una normalizada, hay menos repetición de datos, lo que tiene como consecuencia un mucho menor uso de espacio en disco.

El proceso de normalización tiene un nombre y una serie de reglas para cada fase, esto puede parecer un poco confuso al principio, pero poco a poco se va entendiendo el proceso, así como las razones para hacerlo de esta manera.

3.6 FORMAS NORMALES

Dentro de la literatura se han definido diversas formas normales, pero las más difundidas se ilustran en la siguiente Figura:

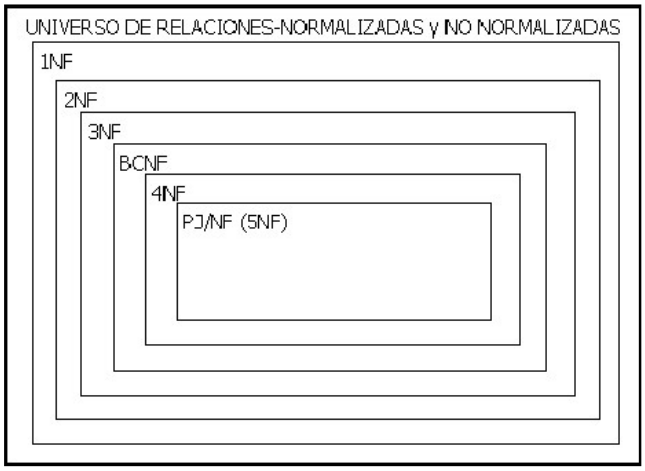


Figura 10. Formas Normales

3.6.1 PRIMERA FORMA NORMAL

Una relación esta en PRIMERA FORMA NORMAL (1NF) si todos los dominios base de los campos tienen valores atómicos. Es decir un campo solo tiene un valor y no un conjunto de valores. Tomemos como ejemplo el caso de una relación para llevar la información de materias y calificaciones de materias de alumnos, un posible diseño se ilustra en la Tabla 1:

MATRÍCULA	CALIFS
331678	CB-001 10, MA-001 9
337890	F-001 7, H-002 8
337777	CB-002 7, CS-056 8
446789	MA-031 7, CB-072-8, H-002 9

Tabla 1: Ejemplo Materias y Calificaciones de Alumnos

Observando la relación nos damos cuenta que el dominio del campo MATRICULA si cumple la característica de tener valores atómicos, pero el campo CALIFS tiene el problema de que el dominio utilizado proporciona conjuntos de valores, es decir no esta en 1NF. Un posible mejor diseño sería cómo se ilustra en la Tabla 2:

MATRÍCULA	CALIF
331678	CB-001 10
331678	MA-001 9
337890	F-001 7
337890	H-002 8
337777	CB-002 7
337777	CS-056 8
446789	MA-031 7
446789	CB-072 8
446789	H-002 9

Tabla 2: Mejor Diseño de Materias y Calificaciones de Alumnos

Aun así observamos que este diseño adolece del problema de que todavía el campo CALIF es un conjunto de valores (una clave de materia y una calificación) es decir todavía no esta en 1NF. Un mejor diseño sería como se ilustra en la tabla 3:

MATRÍCULA	CLAVE	CALIF
331678	CB-001	10
331678	MA-001	9
337890	F-001	7
337890	H-002	8
337777	CB-002	7
337777	CS-056	8
446789	MA-031	7
446789	CB-072	8
446789	H-002	9

Tabla 3: Ejemplo de 1NF de Materias y Calificaciones de Alumnos

Revisando esta relación observamos que todos los dominios de los campos proporcionan valores atómicos, es decir ninguno de los campos tiene un conjunto de valores; por lo tanto esta relación ya esta en 1NF.

3.6.2 SEGUNDA FORMA NORMAL

Una relación está en 2NF si está en 1NF y además cumple la condición de que cada atributo no llave depende de la llave primaria.

La llave primaria es el CONJUNTO MÍNIMO DE ATRIBUTOS que cumplen la característica de UNICIDAD y que se escogió para ser la LLAVE. Si existe más de un CONJUNTO MÍNIMO DE ATRIBUTOS QUE CUMPLEN LA UNICIDAD, al conjunto escogido como LLAVE se le llama LLAVE PRIMARIA y a los demás conjuntos se les llama LLAVES CANDIDATO.

Supongamos que tenemos una relación que permite llevar la información de los proveedores de una empresa junto con las partes y cantidad en las que son suministradas, se ilustra en la Tabla 4:

NUM_PROV	NOM_PROV	CIUDAD	ENTREGA	NUM_PARTE	CANTIDAD
1	JUAN	ACAPULCO	2	P1	100
1	JUAN	ACAPULCO	2	P2	100
2	PEDRO	MERIDA	4	P1	200
2	PEDRO	MERIDA	4	P3	100
2	PEDRO	MERIDA	4	P5	300
3	ANA	TIJUANA	5	P3	200
3	ANA	TIJUANA	5	P4	100
3	ANA	TIJUANA	5	P5	300
4	RENE	DF	1	P1	200
5	PATRICIA	REYNOSA	2	P1	100
5	PATRICIA	REYNOSA	2	P4	200
6	RENATA	OAXACA	2	P5	100

Tabla 4: Proveedores, Partes y Cantidades Suministradas

Considerando que la llave primaria es NUM_PROV, NUM_PARTE, tendríamos las siguientes relaciones de dependencia entre los campos no llave y los campos de la llave candidato, como se ilustra en la Figura 11 siguiente:

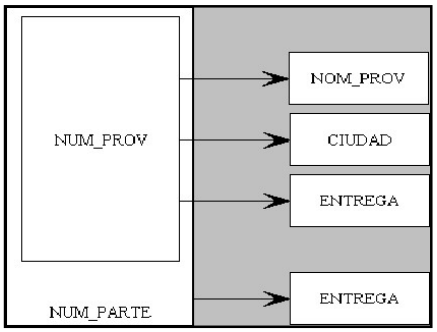


Figura 11. Modelo de Num_prov y Num_parte

NUM_PROV	NUM_PARTE	CANTIDAD
1	P1	100
1	P2	100
2	P1	200
2	P3	100
2	P5	300
3	P3	200
3	P4	100
3	P5	300
4	P1	200
5	P1	100
5	P4	200
6	P5	100

Tabla 5: Información Partes

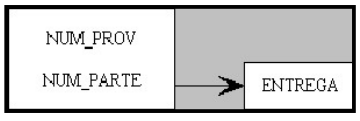


Figura 12. Campos de Partes

3.6.3 TERCERA FORMA NORMAL

Una relación está en 3NF si esta en 2NF y además se cumple que todos los atributos de la relación no dependen transitivamente de la llave primaria. Es decir no se da la situación de que un campo dependa de la llave y otro dependa del primero. Tomemos como ejemplo la tabla 5 de proveedor anterior:

NUM_PROV	NOM_PROV	CIUDAD	ENTREGA
1	JUAN	CAJAMARCA	2
2	PEDRO	CHIMBOTE	4
3	ANA	TRUJILLO	5
4	RENE	LIMA	1
5	PATRICIA	IQUITOS	2
6	RENATA	CUZCO	2

Tabla 6: Información Proveedores

Pero analizando en detalle cada atributo de la relación nos dimos cuenta que el atributo ENTREGA depende realmente de CIUDAD y no directamente del NUM_PROV, es decir tenemos la siguiente relación de dependencia, como se ilustra en la Figura 13:

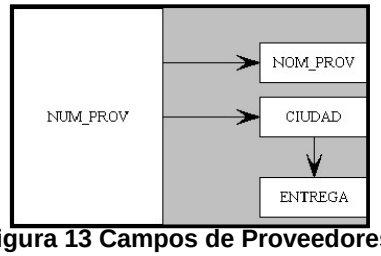


Figura 13 Campos de Proveedores (3NF)

O más claramente, como se ilustra en la Figura 14:

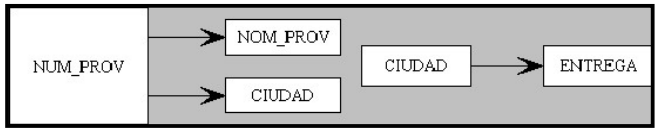


Figura 14. Otra forma de mostrar los Campos de Proveedores

Para normalizar esta relación se realizan dos proyecciones teniéndose, como se ilustra en las tablas:

NUM_PROV	NOM_PROV	CIUDAD
1	JUAN	CAJAMARCA
2	PEDRO	CHIMBOTE
3	ANA	TRUJILLO
4	RENE	LIMA
5	PATRICIA	IQUITOS
6	RENATA	CUZCO

Tabla 7

CIUDAD	ENTREGA
CAJAMARCA	2
CHIMBOTE	4
TRUJILLO	5
LIMA	1
IQUITOS	2
CUZCO	2

Tabla 8

3.6.4 FORMA NORMAL DE BOYCE-CODD (BCNF)

Para ilustrar la necesidad de una forma normal adicional daremos un ejemplo, el cual se ilustra en la Figura 15:

BASE DE DATOS DE ALUMNOS-PROFESORES-MATERIAS.

Supongamos que deseamos llevar la información de que materias lleva cada alumno y que profesor se las imparte. Se sabe además que:

- Un alumno lleve una o más materias.
- Un profesor imparte sólo una materia.
- Una materia puede ser impartida por uno o más profesores.
- En una materia puede haber inscritos uno o más alumnos.

Un posible diseño sería (considerando que ya se definieron previamente los campos de cada entidad):

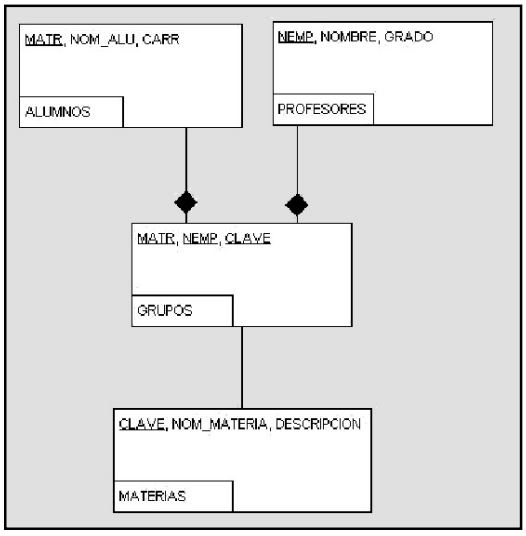


Figura 15. Bases de Datos de Alumnos-Profesores-Materias

Pero como un profesor sólo imparte una materia no es posible que un alumno lleve dos ó mas materias con el mismo profesor, de donde concluimos que no se requiere que la llave primaria de la relación GRUPOS sea MATR+NEMP+CLAVE sino solamente MATR+CLAVE ó MATR+NEMP. Es decir que tenemos dos llaves candidato MATR+CLAVE y MATR+NEMP.

De forma que dos mejores diseños se ilustran en las figuras 16 y 17:

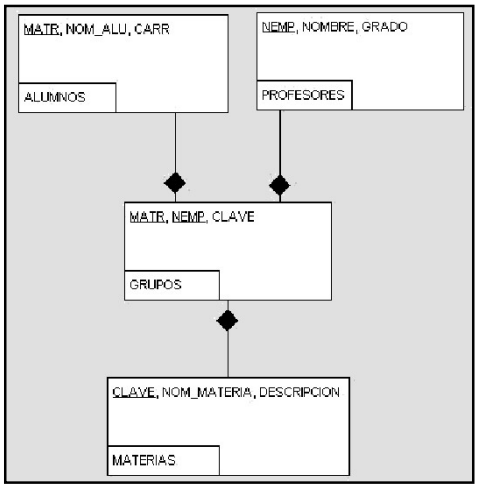


Figura 16. Bases de Datos Alumnos-Profesores-Materias

ó

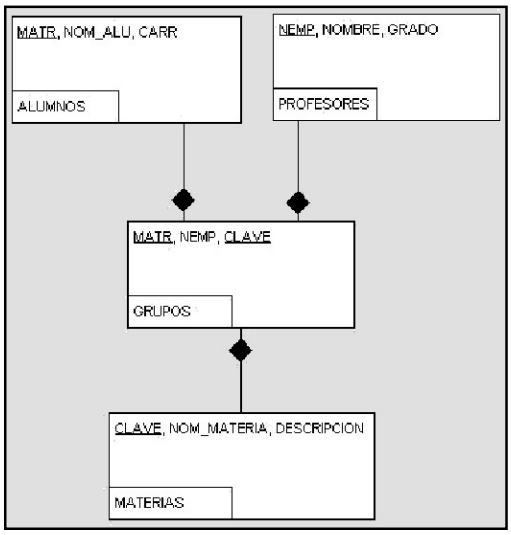


Figura 17. Base de Datos de Alumnos-Profesores-Materias

Construyendo los diagramas de dependencia funcional para la relación GRUPOS de los dos diseños tenemos la figura 18:

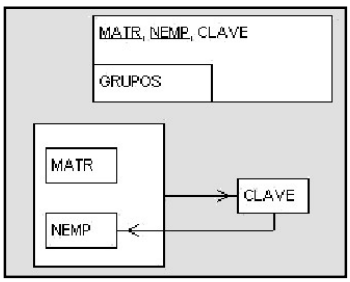


Figura 18. Dependencia Funcional para la Relación de Grupos

NOTA: La flecha de NEMP a CLAVE indica que el profesor determina la materia, puesto que solo imparte una.

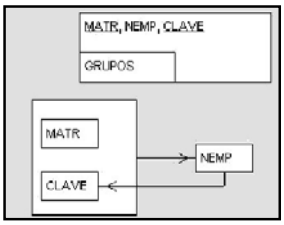


Figura 19. Dependencia Funcional de la Relación de Grupos

En cualquiera de los dos casos podemos observar que la relación GRUPOS está en 3NF porque todos los campos no llave no dependen transitivamente de la llave primaria.

Pero aún así la relación tiene ciertas anomalías, por ejemplo, si solo se tiene a un alumno inscrito en una materia impartida por un profesor, al dar de baja a dicho alumno, se pierde la información de que materia imparte dicho profesor.

Para evitar este tipo de anomalías se ha definido la BCNF que dice:

- Una relación está en BCNF si cada determinante es una llave candidato.
- Un DETERMINANTE es cualquier atributo o conjunto de atributos del cuál otro atributo es dependiente funcionalmente en forma completa.

Analizando el diagrama de dependencias y las llaves candidato tenemos lo siguiente:

LAS LLAVES CANDIDATO SON:

- MATR+CLAVE
- MATR+NEMP
- LOS DETERMINANTES SON:
- MATR+CLAVE
- MATR+NEMP
- NEMP (Puesto que determina la CLAVE).

De donde concluimos que la relación GRUPOS no está en BCNF.

Para poner a GRUPOS en BCNF tenemos que analizar sus posibles proyecciones, como se ilustra en la tabla 9:

Proyección	Comentario
GRUPOS [MATR, CLAVE]	Se pierde la información de que profesor le da clase a un alumno, puesto que hay varios profesores para una materia.
GRUPOS [MATR, NEMP]	OK, pero se requiere adicionalmente la proyección GRUPOS [NEMP, CLAVE] para tener la información original.
GRUPOS [NEMP, CLAVE]	OK, pero se requiere la proyección GRUPOS [MATR, NEMP] para recuperar la información original.

Tabla 9: Grupos en BCNF

De lo anterior concluimos que la relación grupos debe descomponerse en las proyecciones: GRUPOS [MATR, NEMP] y GRUPOS [NEMP, CLAVE], teniendo el siguiente diseño de nuestra base de datos, que se ilustra en la figura:

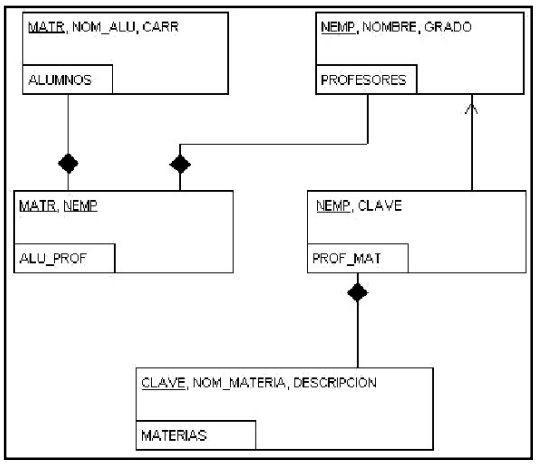


Figura 20. Base de Datos de Alumnos-Profesores-Materias

De donde podemos observar que ALU_PROF ya está en BCNF(es toda llave) y que PROF_MAT ya está en BCNF puesto que solo existe un determinante (que es NEMP).

Adicionalmente se puede verificar que la anomalía de perder la información de que materia da un profesor al borrar al único alumno que estaba inscrito se ha evitado.

NOTA: GENERALMENTE se cumple que una relación no está en BCNF si las llaves candidatos se traslapan.

3.6.5 CUARTA FORMA NORMAL (4NF)

Existen algunas relaciones que a pesar de estar en BCNF presentan cierto grado de redundancia. Lo cuál a su vez tiene como consecuencia que tengamos anomalías al realizar altas o cambios.

Para ilustrar esto, consideremos que en una universidad se desea llevar el control de materias, profesores y libros de texto.

Se tiene la siguiente información:

- Una materia puede ser impartida por uno o más profesores.
- Una materia tiene asociados uno o más libros de texto.
- Un profesor puede impartir una o más materias.
- Un profesor al impartir una materia usa todos los libros de texto de esa materia.
- Un libro de texto puede ser utilizado en una o más materias.

De este modo un posible diseño sería el que se ilustra en la Figura 21:

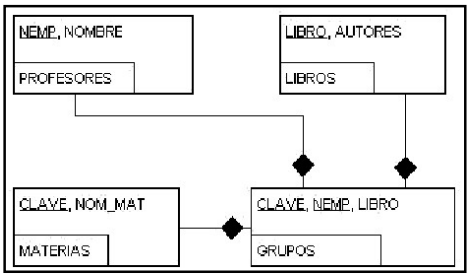


Figura 21. Base de Datos de Materias-Profesores-Libros de Texto

Analizando en detalle la relación GRUPOS nos podemos dar cuenta que se requiere que toda sea llave, puesto que:

- Un mismo libro se puede repetir para diferentes combinaciones de profesor y materia.
- Un mismo profesor se puede repetir para diferentes combinaciones de materias y libros.
- Una misma materia se puede repetir para diferentes combinaciones de profesor y libro.

En este sentido al ser la relación GRUPOS toda LLAVE ya está en BCNF pero tiene algunos problemas:

- Al dar de alta a un nuevo profesor para una materia tenemos el problema de dar de alta tantos registros como libros haya para la materia.
- Al dar de alta un libro para una materia, se tienen que dar de alta tantos registros como profesores haya de la materia.

La BCNF no nos ayuda a corregir estos problemas, por lo que se requiere una forma normal adicional. Una relación está en Cuarta Forma Normal (4NF) si al tener dependencias multivaluadas de la forma A-B todas las dependencias funcionales dependen de A. Es decir todas las dependencias multivaluadas son dependencias funcionales.

Para el caso de la relación analizada tenemos:

CLAVE - NEMP
CLAVE - LIBRO

Para este caso un mejor diseño separaría las dependencias multivaluadas dando la figura:

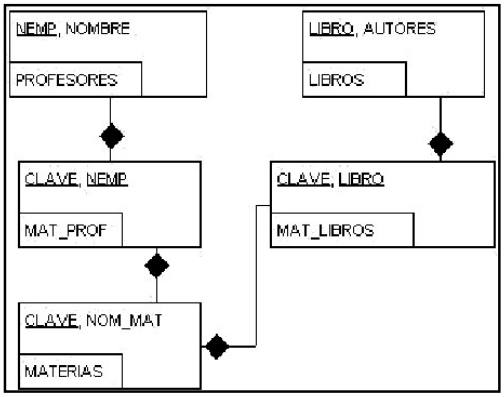


Figura 22. Base de Datos de Alumnos-Profesores-Libros de Texto

CAPÍTULO IV

LENGUAJE ESTRUCTURADO DE CONSULTAS (SQL)

4. LENGUAJE ESTRUCTURADO DE CONSULTAS (SQL)

Aunque el álgebra relacional y el cálculo relacional son lenguajes formales que proporcionan una notación concisa para la representación de consultas, son complejos para las personas que no están vinculadas con la lógica matemática y la notación de conjuntos, sin mencionar que son difíciles de implementar para los SMBD comerciales.

De lo anterior, surgió la necesidad de especificar un lenguaje sencillo de implementar y utilizar que además incluyera capacidades para definir estructuras de datos, modificación de datos y la especificación de restricciones de integridad.

SQL (Standard Query Language) surgió como el lenguaje estándar de base de datos relacionales que cumple con las exigencias de implementación y uso en los SMBD. Aunque hay varias versiones de SQL, todas parten de la versión original desarrollada en el laboratorio de investigación de San José, California (San José Research Center) de IBM. Este lenguaje, originalmente denominado Sequel, se implementó como parte del proyecto System R, a principios de los 70s y ha evolucionado hasta el SQL que conocemos actualmente.

SQL proporciona tres tipos de lenguajes diferentes: uno para especificar el esquema relacional, otro para expresar las consultas y actualizaciones de la base de datos y el último para definir los accesos y seguridad en la base de datos y sus tablas.

4.1 ESTANDARES

Existen 3 estándares aprobados de SQL:

SQL86. En 1986, ANSI (American National Standards Institute, Instituto Nacional Americano de Normalización) e ISO (International Standards Organization, Organización Internacional de Normalización) Publicaron esta norma de SQL como el estándar inicial de SQL, con un conjunto limitado de características y apenas con alguna restricción de integridad.

SQL89. En 1989 se publicó esta norma extendida para SQL, como resultado de una revisión mínima del estándar del 86. Añade llaves primarias y cierta integridad referencial. Definió la posibilidad de utilizarse a través de dos interfaces: interactivamente o dentro de programas de aplicación.

SQL92 (SQL2). Fue desarrollado por el comité técnico NCITS H2 sobre bases de datos, el cual desarrolla estándares para la sintaxis y semántica de los

lenguajes de bases de datos. SQL92 se diseñó para ser un estándar para los SMBD relacionales y actualmente es la versión más difundida ([ISO/IEC 1992] [ANSI 1992] [ISO/IEC 1994]).

Es una revisión completa del lenguaje que añade más potencia semántica en la definición de datos (restricciones de integridad, dominios, más tipos de datos, etcétera). Mejora el lenguaje de consulta (producto natural y externo, entre otros), elimina problemas de ortogonalidad y estandariza el SQL embebido.

La norma actual de SQL de ANSI/ISO es la SQL-92, aunque se debe tener en cuenta que algunas implementaciones de SQL en los SMBD comerciales pueden ser compatibles sólo con SQL-89 y no abarcar SQL- 92.

SQL (Structured Query Language) es un lenguaje para acceso a la información almacenada en bases de datos relacionales.

Cuando utilizas SQL puedes hacerlo desde un terminal sin necesidad de incluirlo previamente en un programa, esto lo hace útil tanto para programadores como para usuarios finales, dando la posibilidad de acceder a los datos mediante consultas abiertas.

Cuando ejecutes el procesamiento de las sentencias SQL este lo realiza el SGDB (Sistema de Gestión de Bases de Datos), las analiza, las traduce a operaciones con ficheros físicos, las ejecuta, y devuelve el resultado al programa o usuario que realizó la petición. El SGBD se va a encarga de coordinar todas la peticiones asegurando la integridad de los datos.

4.2 TIPOS DE SENTENCIAS

En SQL distinguimos tres tipos de sentencias:

-***DML*** (Data Manipulation Language).- Como su nombre indica son las sentencias SQL que permiten trabajar con los datos, seleccionarlos, actualizarlos, eliminarlos, agruparlos, etc…

-***DDL*** (Data Definition Language).- Son sentencias que crean o eliminan objetos del SGDB por ejemplo tablas, vistas, etc…

-***DCL*** (Data Control Language).- Se encarga de mantener la seguridad y el control de los datos.

4.3 SENTENCIAS DDL

En general, usamos SQL para acceder a los datos de unas tablas ya creadas por el DBA de la base de datos.

Pero en ocasiones podemos necesitar disponer de tablas de uso privado con datos exclusivamente interesantes para ti por tu trabajo.

Esto implica que tenemos que ser capaz de crear estas nuevas tablas, destruirlas cuando ya no las necesites, y autorizar a otros usuarios a utilizarlas si así lo estimamos oportuno. Todo ello puede hacerse con las sentencias DDL si el DBA nos ha otorgado previamente las autorizaciones correspondientes para ello.

CREATE TABLE

La ejecución de esta sentencia nos permite almacenar en el catálogo del SGBD la definición y el nombre de una tabla para que luego pueda ser referenciada.

Formato:

CREATE TABLE tabla (col-1 tipo-1 [not null]

[,col-2 tipo-2 [NOT NULL]]…)

donde col-1 y col-2 son los nombres asignados a las columnas de la tabla y tipo-1 y tipo-2 son sus respectivos tipos de datos (INTEGER, FLOAT, VARCHAR, TIME…).

Si en alguna columna se especifica NOT NULL, el SGBD no admitirá la inserción en la tabla de una fila con valor Nulo en esa columna.

Ejemplo:

Crear la tabla de clientes.

CREATE TABLE CLIENTES

(NUMCLI INTEGER NOT NULL

NOMBRE VARCHAR (20) NOT NULL

,REPCLI INTEGER NOT NULL

,LIMITECREDITO INTEGER NOT NULL)

DROP TABLE

Utilizaremos la sentencia **DROP TABLE** para destruir una tabla.

Formato:

DROP TABLE tabla

Esta sentencia borra del catálogo de SGBD la descripción de la tabla y desde ese momento no se admitirán referencias a ella.

Ejemplo:

Borrar la tabla de clientes.

DROP TABLE CLIENTES

GRANT

Como propietarios de una tabla podemos usar esta sentencia SQL para transmitir a otros sus autorizaciones sobre ella, pero éstos no podrán transmitirlas al no ser que especifiquemos como privilegio el mismo GRANT.

Formato (simplificado):

GRANT lista-privilegios

ON tabla

TO lista-usuarios WITH GRANT OPTION

Donde *lista-usuarios* es una lista de identificadores personales, separados por comas, de los usuarios a los que se va a autorizar, y *lista-privilegios* es una lista de palabras separadas por comas en la que se pueden especificar las siguientes: DELETE, INSERT, SELECT y UPDATE, etc... que indican que se autoriza a usar las correspondientes sentencias SQL sobre la tabla.

***Ejemplo* :**

Dar permisos de consulta y actualización sobre la tabla de clientes a los usuarios U1 y U2.

GRANT SELECT, UPDATE ON CLIENTES TO U1,U2

REVOKE

Es la sentencia con la que anulamos una autorización previamente concedida.

Formato (simplificado):

REVOKE lista-privilegios

ON tabla

FROM lista-usuarios

Donde *lista-usuarios* es una lista de identificadores personales, separados por comas, de los usuarios a los que se va a autorizar, y *lista-privilegios* es una lista de palabras separadas por comas en la que se pueden especificar las siguientes: DELETE, INSERT, SELECT y UPDATE, etc... que indican que se autoriza a usar las correspondientes sentencias SQL sobre la tabla

Ejemplo:

Quitar los permisos de consulta y modificación sobre la tabla de clientes a los usuarios U1, U2.

REVOKE SELECT,UPDATE ON CLIENTES TO U1,U2

4.4 SENTENCIAS DML

-***DML*** (Data Manipulation Language).- Como su nombre indica son las sentencias SQL que nos permiten trabajar con los datos: seleccionarlos, actualizarlos, eliminarlos, agruparlos, etc…

Las sentencias DML son *sentencias de manipulación de datos*.

¿Qué son las sentencias DML?

Las sentencias de manipulación de datos son de cuatro tipos y nos van a permitir realizar las siguientes actividades:

- Recuperar datos : SELECT

- Modificar datos: UPDATE

- Borrar Filas : DELETE

- Añadir Filas : INSERT

Sentencia SELECT (I)

La sentencia **SELECT** simple nos permite consultar los datos almacenados en una tabla de la base de datos.

Formato:

SELECT [DISTINCT] */<colum1,column2,...>

FROM <nombre-tabla>

[WHERE <condición>]

[GROUP BY <colum1,colum2,...>]

[HAVING <condición-selec-grupos>]

[ORDER BY <colum1[DESC],colum2[DESC]...>]

Cláusula **SELECT**:

En esta cláusula se especifican la lista de los campos separados por comas. Es suficiente con poner únicamente el nombre del campo, pero si en dos tablas existen campos con el mismo nombre se ha de preceder el nombre del campo por el de la tabla separados por un punto (nom_tabla.nom_campo).

-La ***palabra clave DISTINCT*** provocaría que la sentencia devolviese solamente las líneas que tengan todos las columnas especificadas en la sentencia SELECT distintas.

-El ***asterisco(*)*** traería todos los campos o columnas de las tablas indicadas en el FROM.

Cláusula **FROM**:

En esta cláusula se especifican la lista de los nombres de tablas separados por comas.

Cláusula "**WHERE** ":

Es una cláusula opcional en la que se incluyen las condiciones de filtrado de la consulta que se quiere realizar.

Cláusula "**GROUP BY** ":

Es una cláusula opcional de la sentencia SELECT que sirve para agrupar filas que tengan iguales valores en las columnas de agrupamiento.

Cláusula "**HAVING** ":

Es una cláusula opcional de la sentencia SELECT que sirve para descartar los grupos de filas. Una vez separados las filas en uno o varios grupos, se descartan aquellos que no satisfacen la condición.

Cláusula "**ORDER BY** ":

En esta cláusula opcional se le indica a la sentencia el orden en que se quiere recibir los datos. Las columnas por las que se quiera ordenar tienen que estar en la cláusula SELECT.

La palabra clave DESC devuelve los datos ordenados de forma decreciente.

Sentencia SELECT (II)

Funciones escalares sobre columnas:

Estas funciones nos permiten obtener un solo valor como resultado de aplicar una determinada operación a los valores contenidos en una columna, por ejemplo, la suma de todos ellos o su valor medio.

Algunos ejemplos de estas funciones son:

AVG: Calcula el valor medio de los valores de la columna.

MAX: Devuelve el máximo.

MIN: Devuelve el mínimo.

SUM: Calcula la suma.

COUNT: Halla cuántos valores hay en la colección.

Ejemplo:

Extraer de la tabla de empleados LAS VENTAS máximo y el mínimo para cada grupo de oficinas cuyo código es distinto de cero, agrupándolo por oficinas cuya máximas ventas sean superior a 20000.

```
SELECT OFICINA, MAX (VENTAS), MIN(VENTAS)

FROM EMPLEADOS

WHERE OFICINA <> 0

GROUP BY OFICINA

HAVING MAX(VENTAS)>20000

ORDER BY OFICINA;
```

Sentencia INSERT (I)

La sentencia **INSERT** es la sentencia con la que añadimos una o más filas completas a una tabla.

La especificaremos con uno de los dos formatos siguientes:

Formato 1:

```
INSERT INTO tabla[(col-1, col-2, ...)]

VALUES (valor-1,valor-2,...)
```

En este formato, tabla es el nombre de la tabla en la que se desea insertar, y col-1,col-2,etc. es una lista del nombre de columnas de esta tabla, no necesariamente todas ellas, ni en el mismo orden en que se han definido. Si se omite, se interpreta como si se especificara una lista incluyéndolas a todas y en dicho orden.

Cada uno de los elementos de la lista de valores valor1, valor2,... debe ser una constante o la palabra NULL o un registro especial, y debe haber tantos como en la lista de nombres de columnas.

Ejemplo 1:

Dar de alta un nuevo cliente en la tabla de CLIENTES.

INSERT INTO CLIENTES

VALUES (2124,10,'Pepe Perez',110,60000)

Sentencia INSERT (II)

Formato 2:

INSERT INTO tabla[(col-1, col-2, ...)]

Subselect

En este formato, subselect es una sentencia subordinada y por consiguiente su formato es el de un SELECT básico. Todas las filas que resulten de ejecutarla se insertan en la tabla.

El número de columnas del resultado del subselect debe ser igual al de nombres en la lista de columnas, asignándose la primera de ellas a la primera columna de la lista, la segunda a la segunda, etc.

Ejemplo 2:

Supongamos que tenemos una tabla de clientes2 en la que queremos meter los datos de los clientes que tenga un límite de crédito superior a 30000.

INSERT INTO CLIENTES2

SELECT *

FROM CLIENTES

WHERE LIMITECREDITO > 30000

Sentencia UPDATE

Con la sentencia **UPDATE** modificamos o actualizamos varias filas de una tabla.

Formato:

UPDATE tabla [nombre-local]

SET col-1 = expresión-1 [,col-2 = expresión-2]…

[WHERE predicado]

Actualiza todas las filas de la tabla mencionada detrás de UPDATE que cumplan la condición, modificando las columnas que se mencionen en la cláusula SET.

Se pueden actualizar parte o todas las columnas de los registros, al contrario de los que ocurre con la inserción y el borrado de filas.

Si se omite la cláusula WHERE se actualizan todas las filas.

Ejemplo:

Cambiar el límite de crédito del cliente Juan Bolto a 60000.

UPDATE CLIENTES

SET LIMITECREDITO=60000

WHERE NOMBRE = ‘JUAN BOLTO’

Sentencia DELETE

La sentencia **DELETE** nos permite borrar filas de una tabla.

Formato:

DELETE FROM tabla [nombre-local]

[WHERE condición]

Esta sentencia borra todas las filas que cumplan la condición expresada en la cláusula WHERE. No podemos borrar filas parcialmente, es decir, borramos filas completas.

Si se omite la cláusula WHERE se borran todas las filas de la tabla.

Ejemplo:

Borrar de la tabla de clientes a Juan Bolto.

DELETE FROM CLIENTES

WHERE NOMBRE = ‘JUAN BOLTO’

4.5 SENTENCIAS DCL

El DCL se encarga de mantener la seguridad y el control de los datos.

Existen tres sentencias para que podamos garantizar la seguridad e integridad sobre los datos de las tablas, cuando vayan a ser accedidos por varios usuarios a la vez, o por si cometemos algún error en los procesos de actualización de las tablas:

COMMIT

ROLLBACK

LOCK

COMMIT

Cada vez que se realiza alguna operación en la base de datos se realiza no sobre la tabla en sí, sino sobre una copia local de la misma. Así, si queremos que los resultados de la modificación se trasladen a la base de datos y perduren en el tiempo hay que confirmar dicha operación con el comando commit.

Esta sentencia provoca el fin de una unidad lógica de trabajo; dando por bien hechos todos los cambios en los datos desde el último punto de confirmación (también conocido como punto de sincronismo), es decir, confirmar las modificaciones realizadas que se van guardando en memoria.

Formato:

COMMIT WORK

ROLLBACK

Esta sentencia provoca el fin de una unidad lógica de trabajo ; pero en este caso se restablece la tabla a su estado anterior al último punto de confirmación, destruyendo los cambios hechos, es decir, deshace todas las modificaciones realizadas desde la última confirmación.

Formato:

ROLLBACK WORK

LOCK TABLE

Para garantizar la integridad de los datos también tenemos la sentencia LOCK TABLE.

Dicha sentencia nos permitirá bloquear una tabla mientras estemos accediendo a ella, para así estar seguros de que los datos no son modificados por otros usuarios, mientras estamos accediendo nosotros a ellos.

Formato :

LOCK TABLE tabla

IN SHARE /EXCLUSIVE MODE

SHARE : bloqueamos la tabla en modo compartido, permitiendo a los demás usuarios acceder a los datos de esa tabla , pero sólo para lectura no modificación.

EXCLUSIVE: bloqueamos la tabla en modo exclusivo, de forma que los demás usuarios no puedan acceder a los datos de esa tabla ni siquiera para lectura.

El bloqueo durará hasta que acabe el programa o se produzca un COMMIT o un ROLLBACK.

CAPÍTULO V

CONSTRUCCIÓN DE LA BASE DE DATOS

5. CONSTRUCCIÓN DE LA BASE DE DATOS

Comienza la construcción de la base de datos:

Estas sentencias son necesarias para la utilización de la misma en el SMBD.

```
/****** Object:  Database CSS    Script Date: 21/11/2008 05:51:04 p.m. ******/
IF EXISTS (SELECT name FROM master.dbo.sysdatabases WHERE name = N'CSS')
        DROP DATABASE [CSS]
GO

CREATE DATABASE [CSS]  ON (NAME = N'CSS_Data', FILENAME = N'C:\BD\CSS_Data.MDF' , SIZE = 102, FILEGROWTH = 10%) LOG ON (NAME = N'CSS_Log', FILENAME = N'C:\BD\CSS_Log.LDF' , SIZE = 1, FILEGROWTH = 10%)
GO

exec sp_dboption N'CSS', N'autoclose', N'false'
GO

exec sp_dboption N'CSS', N'bulkcopy', N'true'
GO

exec sp_dboption N'CSS', N'trunc. log', N'true'
GO

exec sp_dboption N'CSS', N'torn page detection', N'false'
GO

exec sp_dboption N'CSS', N'read only', N'false'
GO

exec sp_dboption N'CSS', N'dbo use', N'false'
GO

exec sp_dboption N'CSS', N'single', N'false'
GO

exec sp_dboption N'CSS', N'autoshrink', N'true'
GO

exec sp_dboption N'CSS', N'ANSI null default', N'false'
GO

exec sp_dboption N'CSS', N'recursive triggers', N'false'
GO

exec sp_dboption N'CSS', N'ANSI nulls', N'false'
```

```
GO

exec sp_dboption N'CSS', N'concat null yields null', N'false'
GO

exec sp_dboption N'CSS', N'cursor close on commit', N'false'
GO

exec sp_dboption N'CSS', N'default to local cursor', N'false'
GO

exec sp_dboption N'CSS', N'quoted identifier', N'false'
GO

exec sp_dboption N'CSS', N'ANSI warnings', N'false'
GO

exec sp_dboption N'CSS', N'auto create statistics', N'true'
GO

exec sp_dboption N'CSS', N'auto update statistics', N'true'
GO

if( (@@microsoftversion / power(2, 24) = 8) and (@@microsoftversion &
0xffff >= 724) )
        exec sp_dboption N'CSS', N'db chaining', N'false'
GO

use [CSS]
GO

if exists (select * from dbo.sysobjects where id =
object_id(N'[dbo].[FK_Gpo_Acceso_Cve_Cve_Acceso]') and
OBJECTPROPERTY(id, N'IsForeignKey') = 1)
ALTER TABLE [dbo].[Gpo_Acceso_Cve] DROP CONSTRAINT
FK_Gpo_Acceso_Cve_Cve_Acceso
GO

if exists (select * from dbo.sysobjects where id =
object_id(N'[dbo].[FK_UserA_Cve_Acceso]') and OBJECTPROPERTY(id,
N'IsForeignKey') = 1)
ALTER TABLE [dbo].[UserA] DROP CONSTRAINT FK_UserA_Cve_Acceso
GO

if exists (select * from dbo.sysobjects where id =
object_id(N'[dbo].[FK_Solicitud_EQUIPO]') and OBJECTPROPERTY(id,
N'IsForeignKey') = 1)
ALTER TABLE [dbo].[Solicitud] DROP CONSTRAINT
FK_Solicitud_EQUIPO
GO
```

```
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[FK_Gpo_Acceso_Cve_Gpo_Acceso]') and OBJECTPROPERTY(id, N'IsForeignKey') = 1)
ALTER TABLE [dbo].[Gpo_Acceso_Cve] DROP CONSTRAINT FK_Gpo_Acceso_Cve_Gpo_Acceso
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[FK_UserG_Gpo_Acceso]') and OBJECTPROPERTY(id, N'IsForeignKey') = 1)
ALTER TABLE [dbo].[UserG] DROP CONSTRAINT FK_UserG_Gpo_Acceso
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[FK_UserA_NOPASA]') and OBJECTPROPERTY(id, N'IsForeignKey') = 1)
ALTER TABLE [dbo].[UserA] DROP CONSTRAINT FK_UserA_NOPASA
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[FK_UserG_NOPASA]') and OBJECTPROPERTY(id, N'IsForeignKey') = 1)
ALTER TABLE [dbo].[UserG] DROP CONSTRAINT FK_UserG_NOPASA
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[FK_Solicitud_Tipo_TIPO_SERVICIO]') and OBJECTPROPERTY(id, N'IsForeignKey') = 1)
ALTER TABLE [dbo].[Solicitud_Tipo] DROP CONSTRAINT FK_Solicitud_Tipo_TIPO_SERVICIO
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[FK_Solicitud_Detalle_Solicitud]') and OBJECTPROPERTY(id, N'IsForeignKey') = 1)
ALTER TABLE [dbo].[Solicitud_Detalle] DROP CONSTRAINT FK_Solicitud_Detalle_Solicitud
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[FK_Solicitud_DGT_Solicitud]') and OBJECTPROPERTY(id, N'IsForeignKey') = 1)
ALTER TABLE [dbo].[Solicitud_DGT] DROP CONSTRAINT FK_Solicitud_DGT_Solicitud
GO
```

```
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[FK_Solicitud_Tipo_Solicitud]') and OBJECTPROPERTY(id, N'IsForeignKey') = 1)
ALTER TABLE [dbo].[Solicitud_Tipo] DROP CONSTRAINT FK_Solicitud_Tipo_Solicitud
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[FK_Solicitud_Usu_Atn_Solicitud]') and OBJECTPROPERTY(id, N'IsForeignKey') = 1)
ALTER TABLE [dbo].[Solicitud_Usu_Atn] DROP CONSTRAINT FK_Solicitud_Usu_Atn_Solicitud
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[FK_Solicitud_DGT_Detalle_Solicitud_DGT]') and OBJECTPROPERTY(id, N'IsForeignKey') = 1)
ALTER TABLE [dbo].[Solicitud_DGT_Detalle] DROP CONSTRAINT FK_Solicitud_DGT_Detalle_Solicitud_DGT
GO

/****** Object: View dbo.v_Solicitud Script Date: 21/11/2008 05:51:09 p.m. ******/
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[v_Solicitud]') and OBJECTPROPERTY(id, N'IsView') = 1)
drop view [dbo].[v_Solicitud]
GO

/****** Object: Table [dbo].[Solicitud_DGT_Detalle] Script Date: 21/11/2008 05:51:09 p.m. ******/
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[Solicitud_DGT_Detalle]') and OBJECTPROPERTY(id, N'IsUserTable') = 1)
drop table [dbo].[Solicitud_DGT_Detalle]
GO

/****** Object: Table [dbo].[Solicitud_DGT] Script Date: 21/11/2008 05:51:09 p.m. ******/
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[Solicitud_DGT]') and OBJECTPROPERTY(id, N'IsUserTable') = 1)
drop table [dbo].[Solicitud_DGT]
GO

/****** Object: Table [dbo].[Solicitud_Detalle] Script Date: 21/11/2008 05:51:09 p.m. ******/
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[Solicitud_Detalle]') and OBJECTPROPERTY(id, N'IsUserTable') = 1)
drop table [dbo].[Solicitud_Detalle]
```

```
GO

/****** Object:  Table [dbo].[Solicitud_Tipo]    Script Date: 21/11/2008 05:51:09 p.m. ******/
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[Solicitud_Tipo]') and OBJECTPROPERTY(id, N'IsUserTable') = 1)
drop table [dbo].[Solicitud_Tipo]
GO

/****** Object:  Table [dbo].[Solicitud_Usu_Atn]    Script Date: 21/11/2008 05:51:09 p.m. ******/
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[Solicitud_Usu_Atn]') and OBJECTPROPERTY(id, N'IsUserTable') = 1)
drop table [dbo].[Solicitud_Usu_Atn]
GO

/****** Object:  Table [dbo].[Gpo_Acceso_Cve]    Script Date: 21/11/2008 05:51:09 p.m. ******/
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[Gpo_Acceso_Cve]') and OBJECTPROPERTY(id, N'IsUserTable') = 1)
drop table [dbo].[Gpo_Acceso_Cve]
GO

/****** Object:  Table [dbo].[Solicitud]    Script Date: 21/11/2008 05:51:09 p.m. ******/
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[Solicitud]') and OBJECTPROPERTY(id, N'IsUserTable') = 1)
drop table [dbo].[Solicitud]
GO

/****** Object:  Table [dbo].[UserA]    Script Date: 21/11/2008 05:51:09 p.m. ******/
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[UserA]') and OBJECTPROPERTY(id, N'IsUserTable') = 1)
drop table [dbo].[UserA]
GO

/****** Object:  Table [dbo].[UserG]    Script Date: 21/11/2008 05:51:09 p.m. ******/
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[UserG]') and OBJECTPROPERTY(id, N'IsUserTable') = 1)
drop table [dbo].[UserG]
GO
```

```
/****** Object:  Table [dbo].[Areas]    Script Date: 21/11/2008 05:51:09 p.m. ******/
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[Areas]') and OBJECTPROPERTY(id, N'IsUserTable') = 1)
drop table [dbo].[Areas]
GO

/****** Object:  Table [dbo].[Areas_BK]    Script Date: 21/11/2008 05:51:09 p.m. ******/
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[Areas_BK]') and OBJECTPROPERTY(id, N'IsUserTable') = 1)
drop table [dbo].[Areas_BK]
GO

/****** Object:  Table [dbo].[BITACO1]    Script Date: 21/11/2008 05:51:09 p.m. ******/
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[BITACO1]') and OBJECTPROPERTY(id, N'IsUserTable') = 1)
drop table [dbo].[BITACO1]
GO

/****** Object:  Table [dbo].[Cve_Acceso]    Script Date: 21/11/2008 05:51:09 p.m. ******/
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[Cve_Acceso]') and OBJECTPROPERTY(id, N'IsUserTable') = 1)
drop table [dbo].[Cve_Acceso]
GO

/****** Object:  Table [dbo].[EQUIPO]    Script Date: 21/11/2008 05:51:09 p.m. ******/
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[EQUIPO]') and OBJECTPROPERTY(id, N'IsUserTable') = 1)
drop table [dbo].[EQUIPO]
GO

/****** Object:  Table [dbo].[ESTADOS]    Script Date: 21/11/2008 05:51:09 p.m. ******/
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[ESTADOS]') and OBJECTPROPERTY(id, N'IsUserTable') = 1)
drop table [dbo].[ESTADOS]
GO

/****** Object:  Table [dbo].[Gpo_Acceso]    Script Date: 21/11/2008 05:51:09 p.m. ******/
```

```
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[Gpo_Acceso]') and OBJECTPROPERTY(id, N'IsUserTable') = 1)
drop table [dbo].[Gpo_Acceso]
GO

/****** Object:  Table [dbo].[NOPASA]    Script Date: 21/11/2008 05:51:09 p.m. ******/
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[NOPASA]') and OBJECTPROPERTY(id, N'IsUserTable') = 1)
drop table [dbo].[NOPASA]
GO

/****** Object:  Table [dbo].[Sys_Clave]    Script Date: 21/11/2008 05:51:09 p.m. ******/
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[Sys_Clave]') and OBJECTPROPERTY(id, N'IsUserTable') = 1)
drop table [dbo].[Sys_Clave]
GO

/****** Object:  Table [dbo].[TIPO_SERVICIO]    Script Date: 21/11/2008 05:51:09 p.m. ******/
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[TIPO_SERVICIO]') and OBJECTPROPERTY(id, N'IsUserTable') = 1)
drop table [dbo].[TIPO_SERVICIO]
GO

/****** Object:  Table [dbo].[USUARIOS]    Script Date: 21/11/2008 05:51:09 p.m. ******/
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[USUARIOS]') and OBJECTPROPERTY(id, N'IsUserTable') = 1)
drop table [dbo].[USUARIOS]
GO

/****** Object:  Table [dbo].[cata]    Script Date: 21/11/2008 05:51:09 p.m. ******/
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[cata]') and OBJECTPROPERTY(id, N'IsUserTable') = 1)
drop table [dbo].[cata]
GO

/****** Object:  Table [dbo].[catedo]    Script Date: 21/11/2008 05:51:09 p.m. ******/
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[catedo]') and OBJECTPROPERTY(id, N'IsUserTable') = 1)
```

```
drop table [dbo].[catedo]
GO

/****** Object:  Table [dbo].[delito]    Script Date: 21/11/2008 05:51:09 p.m. ******/
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[delito]') and OBJECTPROPERTY(id, N'IsUserTable') = 1)
drop table [dbo].[delito]
GO

/****** Object:  Table [dbo].[Areas]    Script Date: 21/11/2008 05:51:11 p.m. ******/
CREATE TABLE [dbo].[Areas] (
	[ID_AREA] [numeric](18, 0) IDENTITY (1, 1) NOT FOR REPLICATION NOT NULL ,
	[Status] [char] (1) NOT NULL ,
	[NOM_AREA] [char] (150) NOT NULL ,
	[NOM_AREA_C] [char] (15) NOT NULL ,
	[DG] [numeric](18, 0) NOT NULL ,
	[D] [numeric](18, 0) NOT NULL ,
	[SD] [numeric](18, 0) NOT NULL ,
	[J] [numeric](18, 0) NOT NULL ,
	[Folio] [numeric](18, 0) NOT NULL ,
	[Responsable] [char] (15) NOT NULL
) ON [PRIMARY]
GO

/****** Object:  Table [dbo].[Areas_BK]    Script Date: 21/11/2008 05:51:12 p.m. ******/
CREATE TABLE [dbo].[Areas_BK] (
	[ID_AREA] [numeric](18, 0) IDENTITY (1, 1) NOT NULL ,
	[Status] [char] (1) NOT NULL ,
	[NOM_AREA] [char] (150) NOT NULL ,
	[NOM_AREA_C] [char] (15) NOT NULL ,
	[DG] [numeric](18, 0) NOT NULL ,
	[D] [numeric](18, 0) NOT NULL ,
	[SD] [numeric](18, 0) NOT NULL ,
	[J] [numeric](18, 0) NOT NULL ,
	[Folio] [numeric](18, 0) NOT NULL
) ON [PRIMARY]
GO

/****** Object:  Table [dbo].[BITACO1]    Script Date: 21/11/2008 05:51:12 p.m. ******/
CREATE TABLE [dbo].[BITACO1] (
	[NOMBRE] [char] (45) NOT NULL ,
	[FECHAHORA] [datetime] NOT NULL ,
	[ENTIDAD] [numeric](2, 0) NULL ,
	[CONTROL] [decimal](18, 0) IDENTITY (1, 1) NOT NULL ,
	[TCPIPCLIENT] [char] (15) NULL
```

```
) ON [PRIMARY]
GO

/****** Object:  Table [dbo].[Cve_Acceso]    Script Date: 21/11/2008 05:51:12 p.m. ******/
CREATE TABLE [dbo].[Cve_Acceso] (
	[Cve_Acceso] [char] (20) NOT NULL ,
	[Descripcion] [char] (100) NOT NULL ,
	[Nivel1] [numeric](18, 0) NOT NULL ,
	[Nivel2] [numeric](18, 0) NOT NULL ,
	[Nivel3] [numeric](18, 0) NOT NULL ,
	[Nivel4] [numeric](18, 0) NOT NULL
) ON [PRIMARY]
GO

/****** Object:  Table [dbo].[EQUIPO]    Script Date: 21/11/2008 05:51:13 p.m. ******/
CREATE TABLE [dbo].[EQUIPO] (
	[Inventario] [char] (20) NOT NULL ,
	[Serie] [char] (20) NOT NULL ,
	[Descrip] [int] NOT NULL ,
	[Marca] [int] NOT NULL ,
	[Modelo] [int] NOT NULL ,
	[Accesorios] [varchar] (50) NOT NULL ,
	[Status] [int] NOT NULL ,
	[Nomb_Ins] [varchar] (20) NOT NULL ,
	[Fech_Ins] [datetime] NOT NULL
) ON [PRIMARY]
GO

/****** Object:  Table [dbo].[ESTADOS]    Script Date: 21/11/2008 05:51:13 p.m. ******/
CREATE TABLE [dbo].[ESTADOS] (
	[EDO] [numeric](18, 0) NOT NULL ,
	[ENTIDAD] [char] (30) NOT NULL ,
	[DIRECCION] [text] NOT NULL ,
	[DELEGADO] [char] (60) NOT NULL ,
	[TITULAR] [char] (60) NOT NULL ,
	[MICRO] [char] (30) NOT NULL ,
	[RED] [char] (30) NOT NULL ,
	[ENCARGADO] [text] NOT NULL ,
	[id_Area] [numeric](18, 0) NOT NULL
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO

/****** Object:  Table [dbo].[Gpo_Acceso]    Script Date: 21/11/2008 05:51:13 p.m. ******/
CREATE TABLE [dbo].[Gpo_Acceso] (
	[Cve_Grupo] [char] (20) NOT NULL ,
	[Descripcion] [char] (100) NOT NULL
```

```
) ON [PRIMARY]
GO

/****** Object:  Table [dbo].[NOPASA]    Script Date: 21/11/2008 05:51:14 p.m. ******/
CREATE TABLE [dbo].[NOPASA] (
	[LOGIN] [char] (30) NOT NULL ,
	[CLAVE] [char] (30) NOT NULL ,
	[NOMBRE] [char] (45) NOT NULL ,
	[PRIORIDAD] [decimal](18, 0) NOT NULL ,
	[ULTENT] [datetime] NOT NULL ,
	[RFC] [char] (13) NOT NULL ,
	[fechaini] [datetime] NOT NULL ,
	[fechafin] [datetime] NOT NULL ,
	[area] [numeric](18, 0) NOT NULL ,
	[activo] [decimal](1, 0) NOT NULL ,
	[TIPO] [decimal](2, 0) NOT NULL ,
	[CONTROL] [decimal](18, 0) IDENTITY (1, 1) NOT NULL ,
	[Cargo] [varchar] (40) NOT NULL
) ON [PRIMARY]
GO

/****** Object:  Table [dbo].[Sys_Clave]    Script Date: 21/11/2008 05:51:15 p.m. ******/
CREATE TABLE [dbo].[Sys_Clave] (
	[Id_clave] [int] IDENTITY (1, 1) NOT NULL ,
	[Nombre] [char] (20) NOT NULL ,
	[Clave] [int] NOT NULL ,
	[Descrip] [char] (80) NOT NULL ,
	[Id_modulo] [int] NOT NULL
) ON [PRIMARY]
GO

/****** Object:  Table [dbo].[TIPO_SERVICIO]    Script Date: 21/11/2008 05:51:16 p.m. ******/
CREATE TABLE [dbo].[TIPO_SERVICIO] (
	[CLAVE] [char] (10) NOT NULL ,
	[DESCRIPCION] [varchar] (60) NOT NULL ,
	[ID_Area] [int] NOT NULL
) ON [PRIMARY]
GO

/****** Object:  Table [dbo].[USUARIOS]    Script Date: 21/11/2008 05:51:17 p.m. ******/
CREATE TABLE [dbo].[USUARIOS] (
	[RFC] [char] (13) NOT NULL ,
	[NOMBRE] [char] (30) NOT NULL ,
	[CA_APEPA] [char] (30) NOT NULL ,
	[CA_APEMA] [char] (30) NOT NULL ,
	[NOMBREC] [char] (90) NOT NULL
```

```
) ON [PRIMARY]
GO

/****** Object:  Table [dbo].[cata]    Script Date: 21/11/2008 05:51:19 p.m. ******/
CREATE TABLE [dbo].[cata] (
	[nombre] [char] (40) NOT NULL ,
	[ca_apepa] [char] (40) NOT NULL ,
	[ca_apema] [char] (40) NOT NULL ,
	[nombrec] [char] (90) NOT NULL ,
	[rfc] [char] (13) NOT NULL ,
	[lugar_nac] [char] (40) NOT NULL ,
	[CONTROL] [decimal](18, 0) IDENTITY (1, 1) NOT NULL ,
	[fecha_nac] [datetime] NOT NULL
) ON [PRIMARY]
GO

/****** Object:  Table [dbo].[catedo]    Script Date: 21/11/2008 05:51:20 p.m. ******/
CREATE TABLE [dbo].[catedo] (
	[nombre] [char] (30) NOT NULL ,
	[edo] [numeric](18, 0) NOT NULL
) ON [PRIMARY]
GO

/****** Object:  Table [dbo].[delito]    Script Date: 21/11/2008 05:51:21 p.m. ******/
CREATE TABLE [dbo].[delito] (
	[delito] [char] (110) NULL ,
	[numdelito] [decimal](3, 0) NOT NULL ,
	[prescrip1] [decimal](10, 0) NULL ,
	[prescrip2] [decimal](10, 0) NULL
) ON [PRIMARY]
GO

/****** Object:  Table [dbo].[Gpo_Acceso_Cve]    Script Date: 21/11/2008 05:51:21 p.m. ******/
CREATE TABLE [dbo].[Gpo_Acceso_Cve] (
	[Cve_Grupo] [char] (20) NOT NULL ,
	[Cve_Acceso] [char] (20) NOT NULL
) ON [PRIMARY]
GO

/****** Object:  Table [dbo].[Solicitud]    Script Date: 21/11/2008 05:51:22 p.m. ******/
CREATE TABLE [dbo].[Solicitud] (
	[Control] [int] IDENTITY (1, 1) NOT NULL ,
	[Folio] [int] NOT NULL ,
	[Status] [int] NOT NULL ,
	[Usu_Rec] [char] (15) NOT NULL ,
```

```
	[Fec_Rec] [datetime] NOT NULL ,
	[Usu_Sol] [varchar] (100) NOT NULL ,
	[Micro] [char] (15) NOT NULL ,
	[Id_Area_Sol] [int] NOT NULL ,
	[Descrip_Sol] [text] NOT NULL ,
	[Id_Area_Seguimiento] [int] NOT NULL ,
	[Usu_Conocimiento] [char] (15) NOT NULL ,
	[Fec_Conocimiento] [datetime] NOT NULL ,
	[Fec_Atencion] [datetime] NOT NULL ,
	[Fec_Propuesta] [datetime] NOT NULL ,
	[Usu_Autoriza] [varchar] (100) NOT NULL ,
	[Usu_Autoriza_Cargo] [varchar] (100) NOT NULL ,
	[Id_Inv] [char] (20) NOT NULL ,
	[Fec_Termino] [datetime] NOT NULL ,
	[Descrip_Termino] [text] NOT NULL ,
	[Usu_Supervisa] [char] (15) NOT NULL ,
	[Fec_Supervisa] [datetime] NOT NULL ,
	[Evaluacion] [int] NOT NULL ,
	[Descrip_Evaluacion] [text] NOT NULL ,
	[Rep_Sitio] [int] NOT NULL ,
	[Rep_DSTHS] [int] NOT NULL ,
	[Obser_DSTHS] [text] NOT NULL
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO

/****** Object:  Table [dbo].[UserA]    Script Date: 21/11/2008 05:51:22 p.m. ******/
CREATE TABLE [dbo].[UserA] (
	[Login] [char] (30) NOT NULL ,
	[Cve_Acceso] [char] (20) NOT NULL
) ON [PRIMARY]
GO

/****** Object:  Table [dbo].[UserG]    Script Date: 21/11/2008 05:51:23 p.m. ******/
CREATE TABLE [dbo].[UserG] (
	[LOGIN] [char] (30) NOT NULL ,
	[Cve_Grupo] [char] (20) NOT NULL
) ON [PRIMARY]
GO

/****** Object:  Table [dbo].[Solicitud_DGT]    Script Date: 21/11/2008 05:51:23 p.m. ******/
CREATE TABLE [dbo].[Solicitud_DGT] (
	[Id_Orden] [int] NOT NULL ,
	[Control] [int] NOT NULL ,
	[Inventario] [varchar] (20) NOT NULL ,
	[Fech_Envio] [datetime] NOT NULL ,
	[Observa] [text] NOT NULL ,
	[Asesor] [varchar] (100) NOT NULL ,
```

```
	[Reparacion_DGT] [int] NOT NULL ,
	[NombIns] [char] (30) NOT NULL ,
	[FechIns] [datetime] NOT NULL
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO

/****** Object:  Table [dbo].[Solicitud_Detalle]    Script Date: 21/11/2008 05:51:24 p.m. ******/
CREATE TABLE [dbo].[Solicitud_Detalle] (
	[Id_Equipo_Detalle] [int] IDENTITY (1, 1) NOT NULL ,
	[Control] [int] NOT NULL ,
	[Lista] [varchar] (50) NOT NULL ,
	[Clave] [int] NOT NULL
) ON [PRIMARY]
GO

/****** Object:  Table [dbo].[Solicitud_Tipo]    Script Date: 21/11/2008 05:51:25 p.m. ******/
CREATE TABLE [dbo].[Solicitud_Tipo] (
	[Id_Solicitud_Tipo] [int] IDENTITY (1, 1) NOT NULL ,
	[Control] [int] NOT NULL ,
	[Clave] [char] (10) NOT NULL ,
	[Clave_bk] [char] (10) NOT NULL
) ON [PRIMARY]
GO

/****** Object:  Table [dbo].[Solicitud_Usu_Atn]    Script Date: 21/11/2008 05:51:26 p.m. ******/
CREATE TABLE [dbo].[Solicitud_Usu_Atn] (
	[Id_Solicitud_Usu_Atn] [int] IDENTITY (1, 1) NOT NULL ,
	[Control] [int] NOT NULL ,
	[Login] [char] (15) NOT NULL
) ON [PRIMARY]
GO

/****** Object:  Table [dbo].[Solicitud_DGT_Detalle]    Script Date: 21/11/2008 05:51:27 p.m. ******/
CREATE TABLE [dbo].[Solicitud_DGT_Detalle] (
	[Id_Solicitud_DGT_Detalle] [int] IDENTITY (1, 1) NOT NULL ,
	[Id_Orden] [int] NOT NULL ,
	[Lista] [varchar] (50) NOT NULL ,
	[Clave] [int] NOT NULL
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[Areas] WITH NOCHECK ADD
	CONSTRAINT [PK_AREAS] PRIMARY KEY  CLUSTERED
	(
		[ID_AREA]
	)  ON [PRIMARY]
```

```
GO

ALTER TABLE [dbo].[Cve_Acceso] WITH NOCHECK ADD
	CONSTRAINT [PK_Cve_Acceso] PRIMARY KEY  CLUSTERED
	(
		[Cve_Acceso]
	)  ON [PRIMARY]
GO

ALTER TABLE [dbo].[EQUIPO] WITH NOCHECK ADD
	CONSTRAINT [PK_EQUIPO] PRIMARY KEY  CLUSTERED
	(
		[Inventario]
	)  ON [PRIMARY]
GO

ALTER TABLE [dbo].[ESTADOS] WITH NOCHECK ADD
	CONSTRAINT [PK_ESTADOS] PRIMARY KEY  CLUSTERED
	(
		[EDO]
	)  ON [PRIMARY]
GO

ALTER TABLE [dbo].[Gpo_Acceso] WITH NOCHECK ADD
	CONSTRAINT [PK_Gpo_Acceso] PRIMARY KEY  CLUSTERED
	(
		[Cve_Grupo]
	)  ON [PRIMARY]
GO

ALTER TABLE [dbo].[NOPASA] WITH NOCHECK ADD
	CONSTRAINT [PK_NOPASA] PRIMARY KEY  CLUSTERED
	(
		[CONTROL]
	)  ON [PRIMARY]
GO

ALTER TABLE [dbo].[Sys_Clave] WITH NOCHECK ADD
	CONSTRAINT [PK_Sys_Clave] PRIMARY KEY  CLUSTERED
	(
		[Id_clave]
	)  ON [PRIMARY]
GO

ALTER TABLE [dbo].[TIPO_SERVICIO] WITH NOCHECK ADD
	CONSTRAINT [PK_TIPO_SERVICIO] PRIMARY KEY  CLUSTERED
	(
		[CLAVE]
	)  ON [PRIMARY]
GO
```

```
ALTER TABLE [dbo].[USUARIOS] WITH NOCHECK ADD
	CONSTRAINT [PK_USUARIOS] PRIMARY KEY  CLUSTERED
	(
		[RFC]
	)  ON [PRIMARY]
GO

ALTER TABLE [dbo].[cata] WITH NOCHECK ADD
	CONSTRAINT [PK_cata] PRIMARY KEY  CLUSTERED
	(
		[CONTROL]
	)  ON [PRIMARY]
GO

ALTER TABLE [dbo].[Gpo_Acceso_Cve] WITH NOCHECK ADD
	CONSTRAINT [PK_Gpo_Acceso_Cve] PRIMARY KEY  CLUSTERED
	(
		[Cve_Grupo],
		[Cve_Acceso]
	)  ON [PRIMARY]
GO

ALTER TABLE [dbo].[Solicitud] WITH NOCHECK ADD
	CONSTRAINT [PK_REPORTE] PRIMARY KEY  CLUSTERED
	(
		[Control]
	)  ON [PRIMARY]
GO

ALTER TABLE [dbo].[UserA] WITH NOCHECK ADD
	CONSTRAINT [PK_UserA] PRIMARY KEY  CLUSTERED
	(
		[Login],
		[Cve_Acceso]
	)  ON [PRIMARY]
GO

ALTER TABLE [dbo].[UserG] WITH NOCHECK ADD
	CONSTRAINT [PK_UserG] PRIMARY KEY  CLUSTERED
	(
		[LOGIN],
		[Cve_Grupo]
	)  ON [PRIMARY]
GO

ALTER TABLE [dbo].[Solicitud_DGT] WITH NOCHECK ADD
	CONSTRAINT [PK_Solicitud_DGT] PRIMARY KEY  CLUSTERED
	(
		[Id_Orden]
```

```
	) ON [PRIMARY]
GO

ALTER TABLE [dbo].[Solicitud_Detalle] WITH NOCHECK ADD
	CONSTRAINT [PK_Solicitud_Detalle] PRIMARY KEY CLUSTERED
	(
		[Id_Equipo_Detalle]
	) ON [PRIMARY]
GO

ALTER TABLE [dbo].[Solicitud_Tipo] WITH NOCHECK ADD
	CONSTRAINT [PK_Solicitud_Tipo] PRIMARY KEY CLUSTERED
	(
		[Id_Solicitud_Tipo]
	) ON [PRIMARY]
GO

ALTER TABLE [dbo].[Solicitud_Usu_Atn] WITH NOCHECK ADD
	CONSTRAINT [PK_Solicitud_Usu_Atn] PRIMARY KEY CLUSTERED
	(
		[Id_Solicitud_Usu_Atn]
	) ON [PRIMARY]
GO

ALTER TABLE [dbo].[Solicitud_DGT_Detalle] WITH NOCHECK ADD
	CONSTRAINT [PK_Solicitud_DGT_Detalle] PRIMARY KEY
CLUSTERED
	(
		[Id_Solicitud_DGT_Detalle]
	) ON [PRIMARY]
GO

ALTER TABLE [dbo].[Areas] ADD
	CONSTRAINT [DF_AREAS_Status] DEFAULT (1) FOR [Status],
	CONSTRAINT [DF_AREAS_NOM_AREA] DEFAULT ('') FOR
[NOM_AREA],
	CONSTRAINT [DF_AREAS_NOM_AREA_C] DEFAULT ('') FOR
[NOM_AREA_C],
	CONSTRAINT [DF_AREAS_DG] DEFAULT (0) FOR [DG],
	CONSTRAINT [DF_AREAS_D] DEFAULT (0) FOR [D],
	CONSTRAINT [DF_AREAS_SD] DEFAULT (0) FOR [SD],
	CONSTRAINT [DF_AREAS_J] DEFAULT (0) FOR [J],
	CONSTRAINT [DF_AREAS_Folio] DEFAULT (0) FOR [Folio],
	CONSTRAINT [DF_Areas_Responsable] DEFAULT ('') FOR
[Responsable]
GO

ALTER TABLE [dbo].[BITACO1] ADD
	CONSTRAINT [DF_BITACO1_FECHAHORA] DEFAULT ('') FOR
[FECHAHORA],
```

```
	CONSTRAINT [CONTROL] PRIMARY KEY  NONCLUSTERED
	(
		[CONTROL]
	)  ON [PRIMARY]
GO

ALTER TABLE [dbo].[Cve_Acceso] ADD
	CONSTRAINT [DF_Cve_Acceso_Cve_Acceso] DEFAULT ('') FOR
[Cve_Acceso],
	CONSTRAINT [DF_Cve_Acceso_Nivel1] DEFAULT (0) FOR [Nivel1],
	CONSTRAINT [DF_Cve_Acceso_Nivel2] DEFAULT (0) FOR [Nivel2],
	CONSTRAINT [DF_Cve_Acceso_Nivel3] DEFAULT (0) FOR [Nivel3],
	CONSTRAINT [DF_Cve_Acceso_Nivel4] DEFAULT (0) FOR [Nivel4]
GO

ALTER TABLE [dbo].[EQUIPO] ADD
	CONSTRAINT [DF_EQUIPO_SERIE] DEFAULT ('') FOR [Serie],
	CONSTRAINT [DF_EQUIPO_DESCRIP] DEFAULT (0) FOR [Descrip],
	CONSTRAINT [DF_EQUIPO_MARCA] DEFAULT (0) FOR [Marca],
	CONSTRAINT [DF_EQUIPO_MODELO] DEFAULT (0) FOR [Modelo],
	CONSTRAINT [DF_EQUIPO_ACCESORIOS] DEFAULT ('') FOR
[Accesorios],
	CONSTRAINT [DF_EQUIPO_Status] DEFAULT (0) FOR [Status],
	CONSTRAINT [DF_EQUIPO_Nomb_Ins] DEFAULT ('') FOR
[Nomb_Ins]
GO

ALTER TABLE [dbo].[ESTADOS] ADD
	CONSTRAINT [DF_ESTADOS_EDO] DEFAULT (0) FOR [EDO],
	CONSTRAINT [DF_CAT_EDOS_ENTIDAD] DEFAULT ('') FOR
[ENTIDAD],
	CONSTRAINT [DF_CAT_EDOS_DIRECCION] DEFAULT ('') FOR
[DIRECCION],
	CONSTRAINT [DF_CAT_EDOS_DELEGADO] DEFAULT ('') FOR
[DELEGADO],
	CONSTRAINT [DF_CAT_EDOS_TITULAR] DEFAULT ('') FOR
[TITULAR],
	CONSTRAINT [DF_CAT_EDOS_MICRO] DEFAULT ('') FOR [MICRO],
	CONSTRAINT [DF_CAT_EDOS_RED_VIT] DEFAULT ('') FOR [RED],
	CONSTRAINT [DF_CAT_EDOS_ENCARGADO] DEFAULT ('') FOR
[ENCARGADO],
	CONSTRAINT [DF_ESTADOS_id_Area_Sol] DEFAULT (0) FOR
[id_Area]
GO

ALTER TABLE [dbo].[Gpo_Acceso] ADD
	CONSTRAINT [DF_Gpo_Acceso_Cve_Grupo] DEFAULT ('') FOR
[Cve_Grupo],
	CONSTRAINT [DF_Gpo_Acceso_Descripcion] DEFAULT ('') FOR
[Descripcion]
```

```
GO

ALTER TABLE [dbo].[NOPASA] ADD
	CONSTRAINT [DF_NOPASA_fechaini] DEFAULT ('') FOR [fechaini],
	CONSTRAINT [DF_NOPASA_fechafin] DEFAULT ('') FOR [fechafin],
	CONSTRAINT [DF_NOPASA_area] DEFAULT (0) FOR [area],
	CONSTRAINT [DF_NOPASA_Cargo] DEFAULT ('') FOR [Cargo],
	CONSTRAINT [IX_NOPASA] UNIQUE  NONCLUSTERED
	(
		[LOGIN]
	)  ON [PRIMARY]
GO

ALTER TABLE [dbo].[TIPO_SERVICIO] ADD
	CONSTRAINT [DF_TIPO_SERVICIO_CLAVE] DEFAULT ('') FOR
[CLAVE],
	CONSTRAINT [DF_TIPO_SERVICIO_DESCRIPCION] DEFAULT ('')
FOR [DESCRIPCION],
	CONSTRAINT [DF_TIPO_SERVICIO_Area] DEFAULT (0) FOR
[ID_Area]
GO

ALTER TABLE [dbo].[USUARIOS] ADD
	CONSTRAINT [DF_USUARIOS_RFC] DEFAULT ('') FOR [RFC],
	CONSTRAINT [DF_USUARIOS_NOMBRE] DEFAULT ('') FOR
[NOMBRE],
	CONSTRAINT [DF_USUARIOS_CA_APEPA] DEFAULT ('') FOR
[CA_APEPA],
	CONSTRAINT [DF_USUARIOS_CA_APEMA] DEFAULT ('') FOR
[CA_APEMA],
	CONSTRAINT [DF_USUARIOS_NOMBREC] DEFAULT ('') FOR
[NOMBREC]
GO

ALTER TABLE [dbo].[catedo] ADD
	CONSTRAINT [DF_catedo_nombre] DEFAULT ('') FOR [nombre],
	CONSTRAINT [DF_catedo_edo] DEFAULT (0) FOR [edo]
GO

ALTER TABLE [dbo].[Gpo_Acceso_Cve] ADD
	CONSTRAINT [DF_Gpo_Acceso_Cve_Cve_Grupo] DEFAULT ('')
FOR [Cve_Grupo],
	CONSTRAINT [DF_Gpo_Acceso_Cve_Cve_Acceso] DEFAULT ('')
FOR [Cve_Acceso]
GO

ALTER TABLE [dbo].[Solicitud] ADD
	CONSTRAINT [DF_Solicitud_Folio] DEFAULT (0) FOR [Folio],
	CONSTRAINT [DF_Solicitud_Status] DEFAULT (0) FOR [Status],
```

```
        CONSTRAINT [DF_REPORTE_Usu_Rec] DEFAULT ('') FOR [Usu_Rec],
        CONSTRAINT [DF_REPORTE_Fec_Rec] DEFAULT ('') FOR [Fec_Rec],
        CONSTRAINT [DF_REPORTE_Usu_Sol] DEFAULT ('') FOR [Usu_Sol],
        CONSTRAINT [DF_REPORTE_Micro] DEFAULT ('') FOR [Micro],
        CONSTRAINT [DF_REPORTE_AREA_CSS] DEFAULT (0) FOR [Id_Area_Sol],
        CONSTRAINT [DF_REPORTE_Descrip_Sol] DEFAULT ('') FOR [Descrip_Sol],
        CONSTRAINT [DF_REPORTE_Id_Area_Atencion] DEFAULT (0) FOR [Id_Area_Seguimiento],
        CONSTRAINT [DF_Solicitud_Usu_Atencion_Cap] DEFAULT ('') FOR [Usu_Conocimiento],
        CONSTRAINT [DF_Solicitud_Fec_Conocim] DEFAULT ('') FOR [Fec_Conocimiento],
        CONSTRAINT [DF_REPORTE_Fec_Atencion] DEFAULT ('') FOR [Fec_Atencion],
        CONSTRAINT [DF_REPORTE_Fec_Propuesta] DEFAULT ('') FOR [Fec_Propuesta],
        CONSTRAINT [DF_REPORTE_Usu_Autoriza] DEFAULT ('') FOR [Usu_Autoriza],
        CONSTRAINT [DF_REPORTE_Usu_Autoriza_Cargo] DEFAULT ('') FOR [Usu_Autoriza_Cargo],
        CONSTRAINT [DF_REPORTE_Id_Inv] DEFAULT (0) FOR [Id_Inv],
        CONSTRAINT [DF_REPORTE_Fec_Termino] DEFAULT (0) FOR [Fec_Termino],
        CONSTRAINT [DF_REPORTE_Descrip_Termino] DEFAULT ('') FOR [Descrip_Termino],
        CONSTRAINT [DF_REPORTE_Usu_Superviza] DEFAULT ('') FOR [Usu_Supervisa],
        CONSTRAINT [DF_REPORTE_Fec_Superviza] DEFAULT ('') FOR [Fec_Supervisa],
        CONSTRAINT [DF_REPORTE_Evaluacion] DEFAULT (0) FOR [Evaluacion],
        CONSTRAINT [DF_REPORTE_Descrip_Evaluacion] DEFAULT ('') FOR [Descrip_Evaluacion],
        CONSTRAINT [DF_Solicitud_Rep_Sitio] DEFAULT (0) FOR [Rep_Sitio],
        CONSTRAINT [DF_Solicitud_Rep_DSTHS] DEFAULT (0) FOR [Rep_DSTHS],
        CONSTRAINT [DF_Solicitud_Obser_DSTHS] DEFAULT ('') FOR [Obser_DSTHS]
GO

 CREATE INDEX [IX_Solicitud] ON [dbo].[Solicitud]([Id_Area_Seguimiento], [Status]) ON [PRIMARY]
GO
```

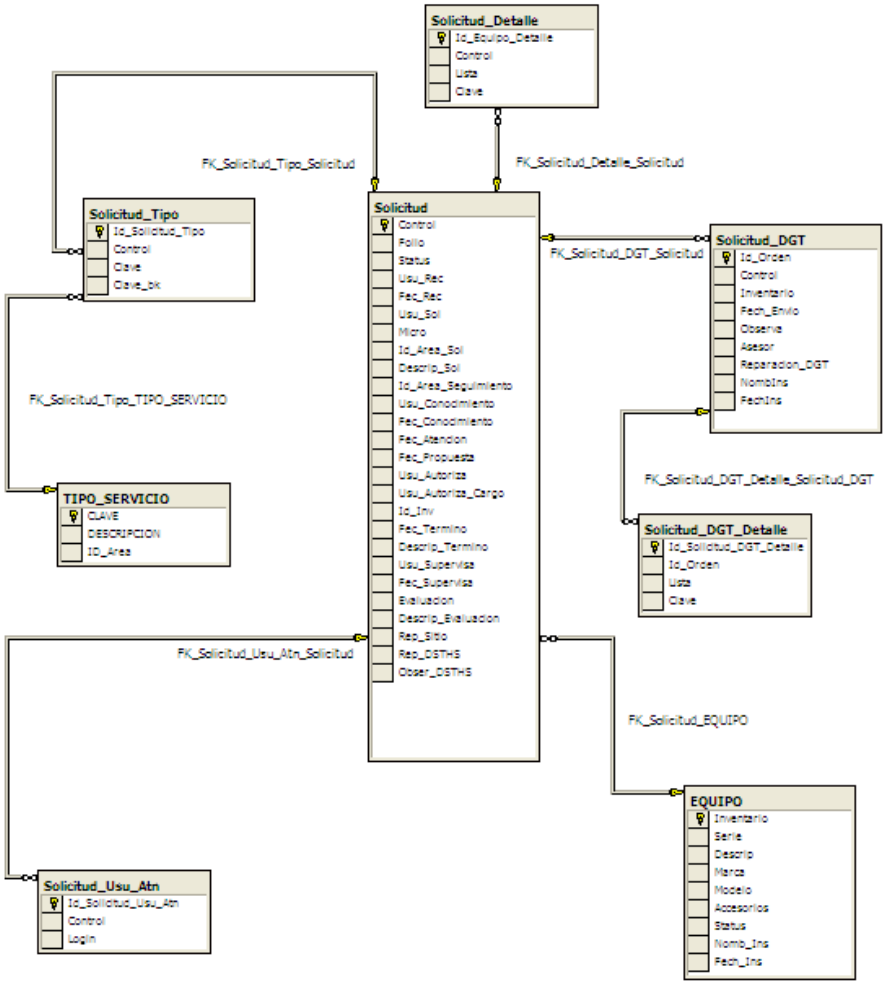
```
ALTER TABLE [dbo].[UserA] ADD
	CONSTRAINT [DF_UserA_Login] DEFAULT ('') FOR [Login],
	CONSTRAINT [DF_UserA_Cve_Acceso] DEFAULT ('') FOR [Cve_Acceso]
GO

ALTER TABLE [dbo].[UserG] ADD
	CONSTRAINT [DF_UserG_LOGIN] DEFAULT ('') FOR [LOGIN],
	CONSTRAINT [DF_UserG_Cve_Grupo] DEFAULT ('') FOR [Cve_Grupo]
GO

ALTER TABLE [dbo].[Solicitud_Tipo] ADD
	CONSTRAINT [DF_Solicitud_Tipo_Folio] DEFAULT (0) FOR [Control],
	CONSTRAINT [DF_Solicitud_Tipo_Clave] DEFAULT ('') FOR [Clave],
	CONSTRAINT [DF_Solicitud_Tipo_Clave_bk] DEFAULT ('') FOR [Clave_bk]
GO

 CREATE INDEX [IX_Solicitud_Tipo_Folio] ON [dbo].[Solicitud_Tipo]([Control]) ON [PRIMARY]
GO

ALTER TABLE [dbo].[Solicitud_Usu_Atn] ADD
	CONSTRAINT [DF_Solicitud_Usuarios_Folio] DEFAULT (0) FOR [Control],
	CONSTRAINT [DF_Solicitud_Usuarios_Login] DEFAULT ('') FOR [Login],
	CONSTRAINT [IX_Solicitud_Usu_Atn_Control_Login] UNIQUE NONCLUSTERED
	(
		[Control],
		[Login]
	)  ON [PRIMARY]
GO

ALTER TABLE [dbo].[Gpo_Acceso_Cve] ADD
	CONSTRAINT [FK_Gpo_Acceso_Cve_Cve_Acceso] FOREIGN KEY
	(
		[Cve_Acceso]
	) REFERENCES [dbo].[Cve_Acceso] (
		[Cve_Acceso]
	),
	CONSTRAINT [FK_Gpo_Acceso_Cve_Gpo_Acceso] FOREIGN KEY
	(
		[Cve_Grupo]
	) REFERENCES [dbo].[Gpo_Acceso] (
		[Cve_Grupo]
	)
```

```
GO

ALTER TABLE [dbo].[Solicitud] ADD
	CONSTRAINT [FK_Solicitud_EQUIPO] FOREIGN KEY
	(
		[Id_Inv]
	) REFERENCES [dbo].[EQUIPO] (
		[Inventario]
	)
GO

ALTER TABLE [dbo].[UserA] ADD
	CONSTRAINT [FK_UserA_Cve_Acceso] FOREIGN KEY
	(
		[Cve_Acceso]
	) REFERENCES [dbo].[Cve_Acceso] (
		[Cve_Acceso]
	),
	CONSTRAINT [FK_UserA_NOPASA] FOREIGN KEY
	(
		[Login]
	) REFERENCES [dbo].[NOPASA] (
		[LOGIN]
	)
GO

ALTER TABLE [dbo].[UserG] ADD
	CONSTRAINT [FK_UserG_Gpo_Acceso] FOREIGN KEY
	(
		[Cve_Grupo]
	) REFERENCES [dbo].[Gpo_Acceso] (
		[Cve_Grupo]
	),
	CONSTRAINT [FK_UserG_NOPASA] FOREIGN KEY
	(
		[LOGIN]
	) REFERENCES [dbo].[NOPASA] (
		[LOGIN]
	)
GO

ALTER TABLE [dbo].[Solicitud_DGT] ADD
	CONSTRAINT [FK_Solicitud_DGT_Solicitud] FOREIGN KEY
	(
		[Control]
	) REFERENCES [dbo].[Solicitud] (
		[Control]
	)
GO
```

```
ALTER TABLE [dbo].[Solicitud_Detalle] ADD
	CONSTRAINT [FK_Solicitud_Detalle_Solicitud] FOREIGN KEY
	(
		[Control]
	) REFERENCES [dbo].[Solicitud] (
		[Control]
	)
GO

ALTER TABLE [dbo].[Solicitud_Tipo] ADD
	CONSTRAINT [FK_Solicitud_Tipo_Solicitud] FOREIGN KEY
	(
		[Control]
	) REFERENCES [dbo].[Solicitud] (
		[Control]
	),
	CONSTRAINT [FK_Solicitud_Tipo_TIPO_SERVICIO] FOREIGN KEY
	(
		[Clave]
	) REFERENCES [dbo].[TIPO_SERVICIO] (
		[CLAVE]
	)
GO

ALTER TABLE [dbo].[Solicitud_Usu_Atn] ADD
	CONSTRAINT [FK_Solicitud_Usu_Atn_Solicitud] FOREIGN KEY
	(
		[Control]
	) REFERENCES [dbo].[Solicitud] (
		[Control]
	)
GO

ALTER TABLE [dbo].[Solicitud_DGT_Detalle] ADD
	CONSTRAINT [FK_Solicitud_DGT_Detalle_Solicitud_DGT] FOREIGN
KEY
	(
		[Id_Orden]
	) REFERENCES [dbo].[Solicitud_DGT] (
		[Id_Orden]
	)
```

5.1 ESQUEMA DE LA BASE DE DATOS

Este es el esquema final de nuestra base de datos:



CONCLUSIÓN

Una base de datos representa para nuestra sociedad una herramienta útil e indispensable, si de organizar información se trata; es un elemento con el cual se optimizan tiempos, espacios, recursos, y que sin lugar a dudas facilita sistematizar la infinita información con la que se cuenta. Contar con una base de datos resulta beneficioso de distintas formas, ya que brinda al usuario rapidez y confiabilidad en la búsqueda de datos, propicia un entorno conveniente y eficiente para las personas que lo usan en la recuperación y almacenamiento de la información, brindan seguridad de los datos almacenados, en caso de caídas del sistema o intentos de acceso sin autorización y logran tener una visión más profunda de los datos que se manejan.

El diseño llevado a cabo en una base de datos permitirá tener la información de una manera más organizada y sin redundancia, es decir, no se repetirán los datos y se obtendrán sólo los necesarios, elemento indispensable e importante de las bases de datos.

El sistema manejador de datos es de suma importancia ya que se interactuara con la base para poder hacer consultas de la misma, las cuales nos ayudaran a traer datos específicos o cualquier combinación que se quiera hacer entre las tablas de la base de datos, eso quiere decir que tendremos un mejor manejo y una mejor extracción de la información.

El lenguaje utilizable en la manipulación de una base de datos es el SQL, este nos permite interactuar con la base de datos desde nuestro mismo manejador, manipular la información y realizar las consultas pertinentes; fue por medio de la implementación de la base de datos como reafirmamos la importancia del lenguaje como favorecedor para explicarnos el manejo de un base de datos.

Una más de las situaciones que logre corroborar al diseñar la base de datos fue la importancia que contar con los conocimientos básicos, la estructura y planeación de una base de datos, ya que todas ellas permitirá al diseñador conformar una base de datos eficiente y útil al usuario.

Considero que el objetivo planteado al comienzo de este trabajo fue logrado, ya que, se manifestó la funcionalidad de las bases de datos por medio del manejador y las pruebas realizadas, además de generar una base de datos con las características necesarias para obtener un manejo adecuado y explotación favorable de ella.

BIBLIOGRAFÍA

1. SILBERSCHTZ, Abraham, KORTH, Henry F. y SUDARSHAN, *Fundamentos de Bases de Datos*, 4ª edición, Mc Graw Hill, 2002, España

2. WIEDERHOLD, Gio, Diseño de Base de Datos 2ª edición, Mc Graw Hill

3. Adoración de Miguel Castaño, Mario G. Piattini Velthuis, Universidad Carlos III de Madrid, Fundamentos y modelos de Base de Datos, Edición RA-MA, 1997

4. RIVERO E. Cornelio, Base de Datos Relacionales, 2ª edición, Paraninfo, 1992

5. El Modelo de Datos Relacional en http://macine.epublish.cl/tesis/index.html

6. Bases de Datos en http://es.tldp.org/Tutoriales/NOTAS-CURSO-BBDD/notas-curso-BD

7. Las 12 reglas de Codd que determinan la fidelidad de un Sistema Relacional al Modelo Relacional en http://petra.euitio.uniovi.es/docencia/cursos/tercero/sis.ges.bas.dat/apuntes/12codd98.html

8. Sistemas de Gestión en Bases de Datos en http://petra.euitio.uniovi.es/docencia/cursos/tercero/sis.ges.bas.dat/apuntes/PASQLw60.html

9. Instituto Tecnológico de la Paz en http://www.itlp.edu.mx/publica/tutoriales/basedat1

10. Tutorial SQL en http://www.w3schools.com/sql/sql alter.asp

11. Teoría de la Normalización en http://www.inf.udec.cl/~basedato/apunte/capitulo5/capitulo5.html

12. www.mygnet.net/manuales/sql/normalizacion_de_base_datos.1626