CONCEPTOS DE BASES DE DATOS

Dato

- Hechos, ideas o conceptos que por si solos carecen de significado puesto que adquieren este cuando se les da una interpretación.
- Cantidad mínima de información, hechos sin evaluar, valor sin significado.

Banco de Dato

- Gran repositorio de información relevante para una organización. Esta información tiene la particularidad de estar en grandes cantidades.

Información

- Conjunto de datos interrelacionados entre sí. La información son los datos que han sido organizados y presentados de modo que los patrones que muestran sean claros.

Sistema de Información

- Conjunto de datos interrelacionados entre sí de forma que ante una entrada proporcionada por un mundo exterior produce una respuesta (salida). En un sentido más amplio, un sistema es un conjunto de componentes que interactúan entre sí para lograr un objetivo común para la toma de decisiones

Campo

- Conjunto de caracteres dentro de los que se tienen unidades de información, físicamente es la unidad mínima de almacenamiento.

Registro

- Conjunto de campos relacionados.

Archivo

- Un archivo es un conjunto de registros y se manejan de la siguiente manera:

Archivos secuenciales. Es la manera más simple de almacenar un conjunto de registros ordenados a manera de lista larga. La única forma de recuperar los datos es empezar al inicio del archivo y leer un registro después de otro, en secuencia, hasta llegar al que se está buscando.

Archivos secuénciales indexados. Es aquel archivo que está en secuencia de acuerdo con un campo específico y se le construye un índice, como el de un libro, con base en el mismo campo.

Base de Datos

- Conjunto de datos interrelacionados con cierto orden y guardados en algún lugar.

Base de Datos Relacional

- Conjunto de datos interrelacionados interrelacionados, almacenados más o menos permanentemente en una computadora; de modo que estos datos guardan independencia física y lógica, consistentes, íntegros y con redundancia controlada.

Una base de datos es una colección de datos de forma que:

a) Los datos son compartidos por diferentes usuarios y programas de aplicación; existe un mecanismo común para inserción, actualización, borrado y consulta de los datos.

b) Tanto los usuarios finales como los programas de aplicación no necesitan conocer los detalles de las estructuras de almacenamiento.

Ventajas de las bases de datos relacionales

- Puede reducirse la repetición de datos.
- Puede evitarse los errores al insertar datos, problemas de actualización y establecer restricciones en el borrado.
- Los datos pueden compartirse. No solo significa que las aplicaciones existentes pueden compartir los datos de la base de datos, sino también que es factible desarrollar nuevas aplicaciones que operen con los mismos datos almacenados.
- Control centralizado de la base de datos.
- Pueden aplicarse restricciones de seguridad.
  a) Asegurar que el único medio de accesar la base de datos sea a través de los canales establecidos.
  b) Definir controles de autorización para que se apliquen cada vez que se intente el acceso a datos sensibles.
- Es compacto. No hacen falta archivos de papeles que pudieran ocupar mucho espacio.
- Es rápido. La máquina puede obtener y modificar datos con mucha mayor velocidad que un ser humano, así es posible satisfacer con rapidez consultas de casos particulares, sin necesidad de búsquedas visuales o manuales que requieren mucho tiempo.
- Es menos laborioso. Se elimina gran parte del trabajo de mantener archivos a mano, las tareas mecánicas siempre serían mejor realizadas por las máquinas.
- Es actual. Se dispone en cualquier momento de información precisa y al día.

Objetivos.

- Proteger el valor de los datos
- Hacer que las fuentes de datos sean responsables de los cambios de las necesidades de información
- Habilitar que la organización que procesa datos logre un mejor control y seguimiento de sus planes de negocios y metas
- Reducir los costos de optimización del desempeño

Costos.

Los costos de establecer y operar en un ambiente de bases de datos incluyen:

- Costos de la Tecnología DBMS
- Costos de Operación de la Base de Datos
- Costos de Conversión de los datos y la lógica
- Costos de Planeación
- Costos de Riesgo

Evolución de las bases de datos.

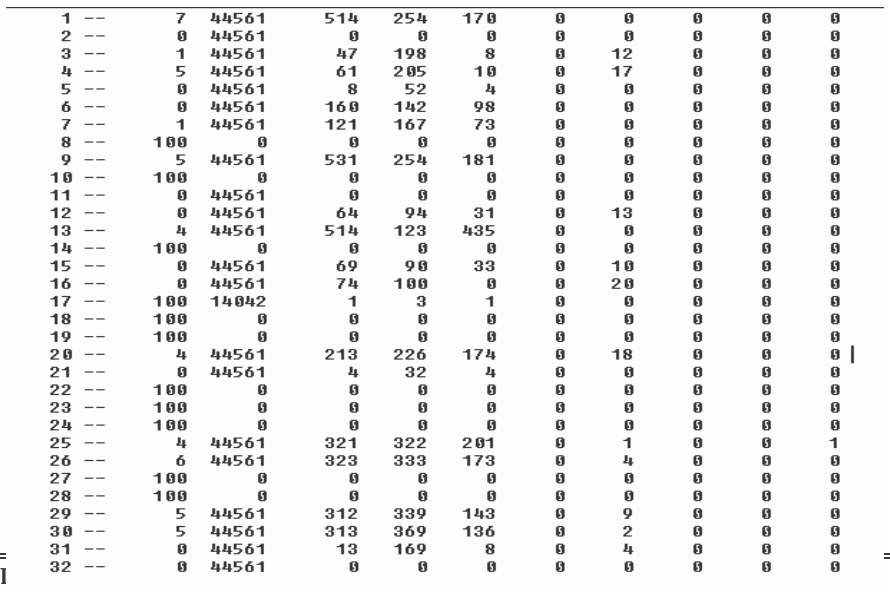
50s	Sistemas manejadores de archivos.
60s	Bases de Datos jerárquicas.
70	Bases de Datos reticulares (de red)
Finales 70s a 90s	Surgimiento y auge de Bases de Datos Relacionales.
Finales 90s	Surgimiento de Bases de Datos Orientadas a Objetos

Veremos a continuación una breve descripción de cada uno de los modelos de bases de datos arriba mencionados, a fin de ubicarnos evolutivamente en el ámbito del modelo relacional de bases de datos.

Sistemas Manejadores de Archivos.

Características:

- Datos separados y aislados.
- Exceso de redundancia.
- La estructura física de los datos era la misma que la lógica.
- Las aplicaciones eran dependientes de las estructuras de los archivos.
- Frecuentemente los archivos no eran compatibles con otras aplicaciones.
- Procesamiento por lotes (batch)



```
 1 --   7  44561  514  254  170   0   0   0   0   0
 2 --   0  44561    0    0    0   0   0   0   0   0
 3 --   1  44561   47  198    8   0  12   0   0   0
 4 --   5  44561   61  205   10   0  17   0   0   0
 5 --   0  44561    8   52    4   0   0   0   0   0
 6 --   0  44561  160  142   98   0   0   0   0   0
 7 --   1  44561  121  167   73   0   0   0   0   0
 8 --  100     0    0    0    0   0   0   0   0   0
 9 --   5  44561  531  254  181   0   0   0   0   0
10 --  100     0    0    0    0   0   0   0   0   0
11 --   0  44561    0    0    0   0   0   0   0   0
12 --   0  44561   64   94   31   0  13   0   0   0
13 --   4  44561  514  123  435   0   0   0   0   0
14 --  100     0    0    0    0   0   0   0   0   0
15 --   0  44561   69   90   33   0  10   0   0   0
16 --   0  44561   74  100    0   0  20   0   0   0
17 --  100 14042    1    3    1   0   0   0   0   0
18 --  100     0    0    0    0   0   0   0   0   0
19 --  100     0    0    0    0   0   0   0   0   0
20 --   4  44561  213  226  174   0  18   0   0   0
21 --   0  44561    4   32    4   0   0   0   0   0
22 --  100     0    0    0    0   0   0   0   0   0
23 --  100     0    0    0    0   0   0   0   0   0
24 --  100     0    0    0    0   0   0   0   0   0
25 --   4  44561  321  322  201   0   1   0   0   1
26 --   6  44561  323  333  173   0   4   0   0   0
27 --  100     0    0    0    0   0   0   0   0   0
28 --  100     0    0    0    0   0   0   0   0   0
29 --   5  44561  312  339  143   0   9   0   0   0
30 --   5  44561  313  369  136   0   2   0   0   0
31 --   0  44561   13  169    8   0   4   0   0   0
32 --   0  44561    0    0    0   0   0   0   0   0
```

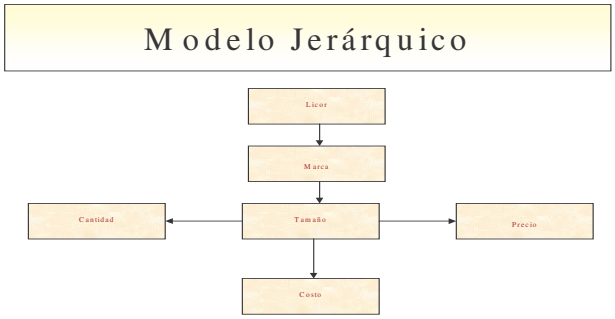
Bases de Datos Jerárquicas

Ventajas

- Relativa simplicidad y facilidad de uso.
- Familiaridad de los usuarios

Desventajas

- No modela de manera sencilla las relaciones uno a muchos
- Anomalías de inserción, borrado, actualización, consulta
- Un hijo solo puede tener un padre



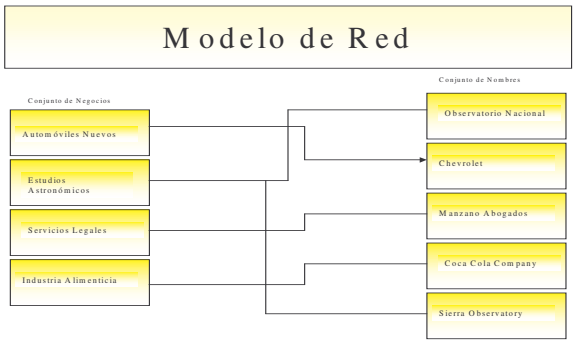
Base de Datos de Red o Reticulares

Ventajas

- La relación muchos a muchos se puede implantar fácilmente
- Esté modelo está respaldado por el grupo de tareas de Bases de Datos (DBTG) de CODASYL.

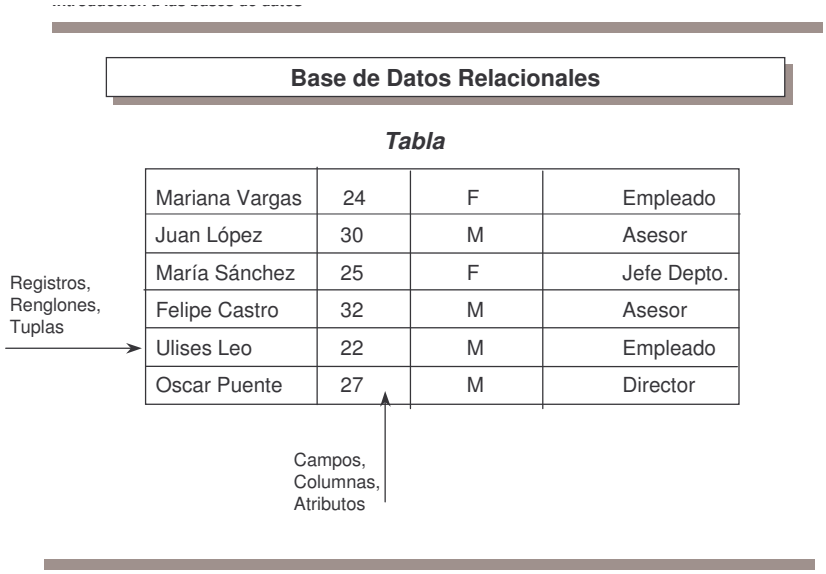
Desventajas

- Demasiado complejo, dificultad para definir nuevas relaciones
- Difícil de mntener y gran desperdicio de recursos.



Bases de Datos Relacionales.

En 1968 en el IBM Research Laboratory en San José, California, con un modelo abstracto de información se inició el trabajo que dio como resultado el modelo de datos relacional. El objetivo del trabajo era encontrar un fundamento teórico de los diferentes aspectos de un DBMS completamente ajeno de los aspectos de un proceso físico dentro de una máquina o CPU en particular. Este modelo es el que actualmente se conoce como Modelo Relacional o Estructura de Datos Relacionales. Una de las partes más importantes de este modelo es la introducción del concepto de tener en la estructura misma operadores lógicos. Esta fue el álgebra que Codd definió para operar estas estructuras. Estos operadores son los que se utilizan para toda la manipulación de datos en una estructura relacional. El Dr. Edgar Codd diseña un tipo de base de datos que se conecta más fácilmente a la realidad. Un concepto que la teoría relacional pone de relieve, es el concepto de dominio. Es una base de datos donde todos los datos visibles al usuario están organizados como tablas de valores, y en donde todas las operaciones de la base de datos operan sobre estas tablas. Poseen un modelo y lenguaje independiente de la implementación para operar en ella.



El primer nombre que recibió el lenguaje manejador de datos fue ALPHA y posteriormente cambió a SQL (Structured Query Language). El trabajo en referencia propició que se empezara a diseñar y desarrollar sistemas basados en la teoría del modelo relacional.

Cada evento se almacena sólo una vez, dando consistencia y ahorrando recursos y es fácil de cambiar. Conjunto de tablas de dos dimensiones (relaciones) con tuplas (renglones) y atributos (columnas) que tienen integridad de datos, es decir, sus datos deben ser precisos y consistentes.

1985 Definición de las doce reglas de Codd para un DBMS relacional.

1972 Definición de las formas normales.

CONCEPTOS ASOCIADOS A BASES DE DATOS

🕮 Redundancia

Repetición de la información que le es perjudicial a la base de datos. Esto sucede cuando en las Bases de Datos cada aplicación tiene sus propios archivos privados, esto provoca considerable redundancia en los datos almacenados, con el consecuente desperdicio de espacio de almacenamiento.

Puesto que los datos son requeridos por múltiples aplicaciones, con frecuencia se registran en múltiples archivos de datos. En la mayoría de los casos los datos se almacenan repetidamente. Esta situación, llamada redundancia de los datos, conduce a muchos problemas que tienen que ver con la integridad de los datos.

🕮 Consistencia

Es la capacidad que tiene la base de datos para garantizar las reglas del negocio. Desde luego, una base de datos que se halle en estado de inconsistencia puede suministrar información incorrecta o contradictoria.

🕮 Integridad

La integridad es la facultad de poder implementar los mecanismos necesarios para que los datos guarden consistencia.

Integridad implica asegurar que lo que se trata de hacer es correcto. El problema de la integridad radica en asegurar que la información de la base de datos sea correcta. La inconsistencia entre dos entradas que representan al mismo "hecho" es un ejemplo de falta de integridad (que, por supuesto, solo ocurre si existe redundancia en los datos almacenados). Es conveniente señalar que la integridad de los datos es más importante en un sistema de base de datos que un sistema de archivos privados, precisamente por que el primero se comparte y porque sin procedimientos de validación adecuados es posible que un programa con errores genere datos incorrectos que afecten a otros programas que utilicen esa información.

Dentro del concepto de integridad, intervienen sus diferentes tipos, los cuales son:

❖ Integridad referencial

Este tipo de integridad es el que nos permite lograr que los datos al estar relacionados, guarden ciertas restricciones, las cuales son:

1. On Delete Cascade. (Borrado en Cascada). Este tipo de restricción nos dice que si al padre se le borra, los datos hijos relacionados también se borrarán.
2. On Delete Restrict. (Borrado en Restricción). Si existe un padre, este no se podrá borrar hasta no borrar los datos hijos relacionados.
3. On Update Cascade. (Actualización en Cascada). Muchas veces se querrá modificar información en un tabla, es decir actualizar; en este caso si el padre tiene datos hijos relacionados, se podrá hacer la actualización sin ningún problema.

4. On Update Restrict. (Actualización en Restricción). Si el padre tiene datos hijos relacionados, no se podrán hacer las actualizaciones hasta liberar a sus datos hijos relacionados.

❖ **Integridad de tabla**

Cada tabla dentro de una base de datos debe tener un hombre único e irrepetible, que nos permita identificarla de manera única.

❖ **Integridad de campo**

Cada campo que este dentro de una tabla deberá contener valores atómicos, es decir, que la información contenida sea la mínima para el ambiente de la base de datos.

❖ **Integridad de llave primaria**

Una llave es el identificador único de un registro. Por lo cuál deberá de cuidar tres características muy importantes:

- Not Duplicate. (No Duplicarse) Esto es que el valor que contenga una llave primaria no debe duplicarse con el mismo valor en otro registro.
- Not Null. (No Nula) Garantizar que siempre tenga un valor la llave primaria.
- Not Change. (No Cambie) Al ya tener un valor asignado, la llave primaria por ningún motivo podrá modificar dicho valor.

❖ **Integridad definida por el usuario**

Las cuales comprenderán las reglas que se definan para la base de datos activa.

📖 **Seguridad**

La seguridad implica asegurar que los usuarios están autorizados para llevar acabo lo que tratan de hacer. El problema de la seguridad tiene muchos aspectos, entre ellos los siguientes.

- Aspectos legales, sociales y éticos (por ejemplo, ¿tiene la persona que solicita el crédito de un cliente, digamos, derecho legal a obtener la información solicitada?).
- Controles físicos (por ejemplo, ¿deberá cerrar o resguardar de alguna otra manera el cuarto de computadoras?).
- Cuestiones de política interna (por ejemplo, ¿cómo decide la empresa propietaria del sistema, quiénes pueden tener acceso a qué?).
- Problemas de operación (por ejemplo, si se utiliza un sistema de contraseñas, ¿cómo se mantienen en secreto las contraseñas?, ¿con qué frecuencia se cambian?).
- Controles de equipo (por ejemplo, ¿posee el CPU características de seguridad tales como claves para la protección de las áreas de almacenamiento o un modo de operación privilegiado?).

Entre los tipos de seguridad existentes se encuentras tres niveles:

❖ **Nivel 1. Seguridad del Manejador de Bases de Datos.**

El software que maneja las bases de datos, deberá de tener diferentes usuarios y controlar el acceso a estos usuarios con un nombre y un password, logrando así un filtro de las personas autorizadas y aquellas que no los son.

- **Nivel 2. Seguridad de la Base de Datos.**

Una vez validado el usuario, se destinará que Bases de Datos puede o no tener acceso; cuales estarán disponibles para ese usuario en particular.

- **Nivel 3. Seguridad de Objetos**

Cada usuario de la base de datos, tendrá la característica de que ya autenticado por el Manejador de Base de Datos y posicionado en una Base de Datos en específico; éste tendrá permisos diferentes sobre los objetos (triggers, stored procedures, tables, indesx, etc)

Independencia Lógica de los Datos

Se puede cambiar la estructura datos almacenados, sin afectarlos. Además de cambiarse los programas aplicativos que acceden a las base de datos sin que esto altere a las base de datos.

Independencia Física de los Datos

Las aplicaciones permanecen inalteradas sin importar los cambios efectuados en el almacenamiento o en los métodos de acceso.

REGLAS DE CODD

Regla 0

For any system that is advertised as, or claimed to be, a relational Data Base management system, that system must be able to manage Data Bases entirely through its relational capabilities.

"Cualquier DBMS que proclame ser relacional, deberá manejar, completamente, las Bases de Datos por medio de sus capacidades relacionales"

Regla 1 (The Information Rule)

All information in a relational Data Base is represented explicity at the logical level and in exactly one way by values in tables.

"Toda la información dentro de una base de datos relacional se representa de manera explícita a nivel lógico y exactamente de una sola manera, como valores en una tabla"

Regla 2 (Guaranteed Access Rule)

Each and every datum (atomic value) in a relational Data Base is guaranteed to be logically accesible by resorting to a combination of table name, primary key value and column name.

"Se garantiza que todos y cada uno de los datos (valor atómico) en una Base de datos relacional pueden ser leídos recurriendo a una combinación de nombre de la tabla, valor de la llave primaria y nombre de la columna"

Regla 3 (Systematic Treatment of Null Values)

Null values (distinct from the empty character string or string of blank characters and distinct from zero or any other number) are supported in fully relational DBMS for representing missing information and inapplicable information in a systematic way, independient of data type.

"En un DBMS totalmente relacional se soportan los valores nulos (que son distintos de una cadena de caracteres vacía o de una cadena con caracteres en blanco o de cero o cualquier otro número). Para representar información faltante o no aplicable de una forma consistente independientemente del tipo de dato"

Regla 4 (Dynamic on-line Catalog Based on the Relational Model)

The DataBase description is represented at the logical level in the same way as ordinary data, so that authorized users can apply the same relational language to its interrogation as they apply to the regular data.

"La descripción de la base de datos se representa en el nivel lógico de la misma forma que los datos ordinarios, de tal suerte que los usuarios autorizados puedan aplicar el mismo lenguaje relacional para consultarla, que aquel que emplean con sus datos habituales"

Regla 5 (Comprehensive Data Sublanguage)

Deberá contener un Sublenguaje de datos completo llamado SQL (Structured Query Lenguaje) que comprenda como estándar lo siguiente:

- Definición de datos
- Definición de vistas
- Manipulación de datos
- Restricciones de integridad (manejo)
- Autorización
- Inicio y fin de una transacción

Regla 6 (View Updating Rule)

All view that are theoretically updatable are also updatable by the system.

"Todas las vistas que teóricamente sean actualizables, deben ser actualizadas por medio del sistema"

Regla 7 (High-Level Insert, Update and Delete)

The capability of handling a base relation or a derived relation as a single operand applies not only to the retrieval of data but also to the insertion, update and deletion of data.

"La posibilidad de manejar una relación base o una relación derivada como un solo operador se aplica a la lectura, inserción, modificación y eliminación de datos"

Regla 8 (Physical Data Independence)

Application program and terminal activity remain logically unimpared whenever any changes are made in either storage representations or access methods.

"Los programas aplicativos y la actividad en terminales no deberán ser afectados por cambios en el almacenamiento físico de los datos o en los métodos de acceso"

Regla 9 (Logical Data Independence)

Application program and terminal activities remain logically unimpared when information preserving changes of any kind that theoretically permit unimpairment are made to the bade tables.

"Los programas aplicativos y la actividad en terminales no deberán ser afectados por cambios de cualquier tipo, que preserven la información y que teóricamente permitan la afectación, en las tablas base."

Regla 10 (Integrity Independence)

Integrity constraints specific to a particular relational Data Base must be definable the relational data sublanguage and storable in the catalog, not in the applications programs.

"Las restricciones de integridad de una Base de datos deberán poder definirse con el mismo sublenguaje de datos relacional y deberá almacenarse en el catálogo, no en los programas aplicativos."

Regla 11 (Distribution Independence)

A relational DBMS has distribution independence.

"Un DBMS relacional tiene independencia de distribución, esto es si se manipulan varios fragmentos de la misma base en difrentes medios de almacenamiento, tener la capacidad como si fuera una sola"

Regla 12 (Nonsubversion Rule)

If a relational system has a low level languaje (single record at a time), that low level cannot be used to subvert or bypass the integrity rules and constrainsts expressed in the higher level relational language.

"Si un sistema relacional tiene un lenguaje de bajo nivel (que opere un registro cada vez), ese lenguaje no deberá poder emplearse para subvertir las reglas de integridad y las restricciones expresadas en el lenguaje relacional de alto nivel."

DEFINICIÓN DE SISTEMA MANEJADOR DE BASES DE DATOS (DBMS)

Entre la base de datos física (es decir, los datos tal y como están almacenados en la realidad) y los usuarios del sistema, existe un nivel de programas, denominado, *manejador de bases de datos (MBD)* o, en la mayoría de los casos, el *sistema administrador de bases de datos* DBMS (Data Base Management System).

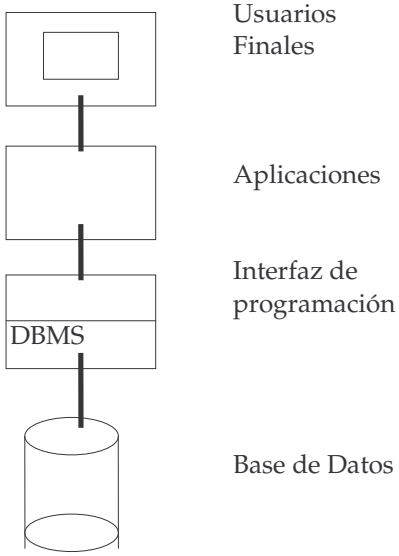
El DBMS maneja todas las solicitudes de acceso a las Bases de Datos formuladas por los usuarios, para la adición y eliminación de archivos (tablas), la obtención y puesta al día de los datos de esos archivos o tablas, etc. Todas estas posibilidades están incluidas en el DBMS. Así, una de las funciones generales del DBMS es distanciar a los usuarios de la base de datos de detalles al nivel del equipo (de manera muy similar a la forma como los sistemas de lenguajes de programación evitan a los programadores de aplicaciones la necesidad de ocuparse de detalles al nivel de la máquina), y hace posible sus operaciones (como por ejemplo las operaciones de SQL en un manejador relacional).

MANEJADOR DE BASE DE DATOS RELACIONAL (RDBMS)

Software que provee el mecanismo para definir, actualizar y accesar datos en una base de datos. Provee independencia de datos para programas y mantiene el control de redundancia.

Es el conjunto de programas que maneja todo acceso a la base de datos. Conceptualmente lo que sucede es lo siguiente:

a) Un usuario solicita acceso, empleando algún sublenguaje de datos determinado
b) El DBMS interpreta esa solicitud y la analiza
c) El DBMS ejecuta las operaciones necesarias sobre la base de datos almacenada



El Sistema de Administración de Datos Relacionales (RDBMS) es el software que maneja todos los accesos a la base de datos. En términos conceptuales, lo que sucede es lo siguiente:

1. Un usuario emite una solicitud de acceso, utilizando algún lenguaje de manipulación de datos específico;
2. El RDBMS intercepta la solicitud y la interpreta;
3. El RDBMS inspecciona por turno el esquema externo, la correspondencia externa-conceptual, el esquema conceptual, la correspondencia conceptual-interna y la definición de la estructura de almacenamiento, y
4. El RDBMS realiza las operaciones necesarias sobre la base de datos almacenada.

Por ejemplo, considérese lo que interviene en la recuperación de una ocurrencia de un registro externo específico. En general se necesitarán campos de varias ocurrencias de registros conceptuales. Cada ocurrencia de un registro conceptual a su vez, puede requerir campos de varias ocurrencias de registros almacenados. Por tanto al menos desde el punto de vista conceptual, el DBMS debe recuperar todas las ocurrencias requeridas de registros almacenados, construir las ocurrencias necesarias de registros conceptuales y luego construir la ocurrencia requerida del registro externo. En cada etapa pueden necesitarse conversiones de tipos de datos o de otra clase.

La descripción anterior presupone que el proceso completo es interpretativo, lo cual a menudo implica un desempeño bastante deficiente. En la práctica, por supuesto, algunas veces las solicitudes de acceso serán compiladas de antemano, evitándose así los costos de interpretación.

COMPONENTES DEL RDBMS

Elementos de un DBMS Relacional:

🕮 Lenguaje de Definición de Datos o DDL

Permite la definición o descripción de los objetos de la base de datos. Puede usarse para crear, alterar o borrar relaciones (tablas), vistas, restricciones de integridad (por ejemplo, llaves primarias y llaves foráneas), etc.

El RDBMS debe ser capaz de aceptar definiciones de datos (esquemas externos, el esquema conceptual, el esquema interno y todas las correspondientes asociadas) en versión fuente y convertirlas en la versión objeto apropiada. Define los objetos (tablas, tipos de datos, índices, reglas, defaults, vistas, triggers, procedimientos almacenados)

Un ejemplo es SQL:

Create
Alter
Drop

🕮 Lenguaje de Control de Datos o DCL

Permite la asignación de los diferentes permisos sobre los objetos existentes en una base de datos. Permite la definición de lo que los usuarios pueden hacer dentro de la base de datos.

Un ejemplo es SQL:

Grant
Revoke

Lenguaje de Manipulación de Datos o DML

Apoya el manejo o procesamiento de los objetos de la base de datos. Puede usarse para leer (consultar), modificar, borrar, o agregar tuplas (renglones) a las relaciones existentes. Una de las primeras funciones de los DBMS es la de soportar un DML en el cual el usuario pueda formular comandos que permitan manipular los datos. Los DML se distinguen por sus sublenguajes de recuperación subyacentes; se pueden distinguir dos tipos de DML, el procedural y el no procedural. La principal diferencia entre ambos es que en los lenguajes procedurales se tratan los registros individualmente, mientras que en uno no procedural se opera sobre un conjunto de registros.

El DBMS debe ser capaz de atender las solicitudes del usuario para extraer, y quizá poner al día, datos que ya existen en la base de datos, o para agregar en ella datos nuevos. Dicho de otro modo, el DBMS debe incluir un componente procesador de lenguaje de manipulación de datos.

En general, las solicitudes en el DML pueden ser "planeadas o no planeadas":

- Una solicitud planeada es aquella cuya necesidad se previó mucho tiempo antes de que tuviera que ejecutarse por primera vez.
- Una solicitud no planeada es una consulta *ad hoc*, es decir, una solicitud cuya necesidad no se previó, sino que surgió de improviso.

Un ejemplo es SQL:

Insert
Update
Select
Delete

Diccionario de Datos o DD

Es una base de datos por propio derecho, una base de datos que contiene "datos acerca de datos" (es decir, descripciones de otros objetos del sistema, y no tan solo "datos en bruto"). En particular, todos los diversos esquemas (externo, conceptual e interno), se almacenan físicamente en el diccionario, tanto en forma fuente como en forma objeto. Un diccionario amplio incluirá también las referencias cruzadas que indican, por ejemplo que partes de datos utiliza cada programa, que informes necesita cada departamento, etc. De hecho, el diccionario puede integrarse a la base de datos que describe, y por tanto, incluir su propia descripción. Debe ser posible consultar el diccionario de la misma manera que cualquier otra base de datos, de modo que, por ejemplo, el DBA pueda describir con facilidad que programas tienen probabilidad de ser afectados por un cambio propuesto al sistema.

El diccionario de datos almacena información acerca de la estructura de la base de datos y la información de autorización, como las restricciones de la clave.

En un sistema de bases de datos relacional se necesita saber información acerca de los datos y sus relaciones. Esta información se denomina *diccionario de datos, o catálogo de sistema*.

Entre los tipos de información que el sistema debe almacenar están:

- Los nombres de las relaciones.
- Los nombres de los atributos de cada relación.
- Los dominios de los atributos.
- Los nombres de las vistas definidas en la base de datos y la definición de esas vistas.
- Las restricciones de integridad de cada relación (por ejemplo, las restricciones de clave).

Va a tener la información de todos los objetos de la base de datos. Sus principales funciones son las siguientes:

- Describe todos los elementos en el sistema (flujo de datos, almacenes de datos, procesos).
- Los elementos se centran en los datos y en la forma en que están estructurados.
- Comunica los mismos significados para todos los elementos del sistema.
- Documenta las características del sistema.
- Facilita el análisis de los detalles para evaluar las características y determinar cómo deben realizarse los cambios.
- Localiza errores y omisiones en el sistema

Además de esto es recomendable que en la mayoría de los sistemas se conserven los siguientes datos:

- Nombre de los usuarios autorizados.
- Información contable acerca de los usuarios.

En los sistemas que utilizan estructuras altamente sofisticadas para almacenar relaciones, pueden conservarse datos estadísticos y descriptivos acerca de las relaciones:

- Número de tuplas de cada relación.
- Método de almacenamiento utilizado para cada relación (por ejemplo, agrupado o sin agrupar).

Es importante almacenar la información de los índices de cada una de las relaciones:

- Nombre del índice.
- Nombre de la relación que se indexa.
- Atributos sobre los que está el índice.
- Tipo de índice.

Toda esta información constituye, de hecho, una base de datos en miniatura. Generalmente es preferible almacenar los datos acerca de la base de datos en la misma base de datos, ya que ocasionalmente en otros sistemas se almacena esta información en otro lado.

DEFINICIÓN DEL ADMINISTRADOR DE LA BASE DE DATOS

DBA (Data Base Administration)

Persona que tiene el control centralizado sobre el sistema de base de datos y que controla tanto los datos como los programas que tienen acceso a ellos.

Funciones:

- Decidir el contenido de la base de datos
- Crear la estructura de almacenamiento y los métodos de acceso
- Modificar la base de datos o la descripción de la organización física
- Otorgar permisos de acceso y prioridades a los diferentes usuarios
- Especificar las limitaciones de integridad
- Ser el enlace con los usuarios
- Definir estrategias para respaldo y recuperación.

En el medio de la base de datos están involucrados muchos usuarios, por lo que es absolutamente necesaria una función que pueda analizar las diferentes necesidades y resolver conflictos de intereses. Se debe establecer una función administrativa permanente para coordinar y llevar a cabo todos los pasos de diseño, implantación y mantenimiento de una base de datos integrada.

La administración de la base de datos es, así una función compuesta por gente responsable de proteger un valioso recurso: los datos. En el medio convencional del procesamiento de datos, un programador de aplicaciones "posee" un archivo de datos. Los usuarios "protegen" sus datos para evitar que otros los usen.

La era de la base de datos ha eliminado la idea de "propiedad" individual. A la persona encargada de la función de la administración de la base de datos se le llama administrador de la base de datos (DBA Data Base Administrator). El administrador de la base de datos no es el "propietario" de los datos, sino el "protector" de ellos.

Ya que al programador de aplicaciones se le "quita" el control directo sobre los datos, pierden el sentimiento de contacto personal y responsabilidad de éstos. Esta falta de contacto fuerza a la empresa a desarrollar procedimientos para asegurar que la integridad de los datos no quede comprometida. Este objetivo deberá estar coordinado con la función de administración de la base de datos.

La administración de la base de datos es una función que proporciona servicios a los usuarios de la base de datos. El DBA protege al recurso llamado "datos". La función del DBA deberá ser situada lo bastante alto en la jerarquía de la organización para tener autoridad, así como responsabilidad, sobre las estructuras de datos y su acceso. La persona a cargo de la función del DBA también debe conocer la forma en que trabaja la empresa y cómo usa los datos. Aunque es aconsejable que el DBA sea técnicamente competente, es más importante que conozca la empresa y que tenga la capacidad y responsabilidad suficientes para interactuar con la gente y proponer alternativas a los procedimientos normales.

El DBA tiene que coordinar las funciones de recopilación de información acerca de los datos y diseñar, implantar y mantener la base de datos y su seguridad. La misión del DBA no termina con la implantación de la base de datos. Una de las principales funciones del DBA consiste en poner atención tanto a las futuras como a las presentes necesidades de información de la empresa, por lo que el diseño de las bases de datos deberá ser tan flexible, o independiente de los datos, como sea posible.

EL ENFOQUE RELACIONAL

DISEÑO DE BASES DE DATOS

Un DBMS utiliza un modelo de datos para definir la estructura fundamental de las bases de datos. Consta de tres modelos. Es una representación abstracta de los datos, define la forma en que los datos elementales son organizados y relacionados. Puede hacer uso de varias representaciones.

🕮 ModeloConceptual

En el modelo conceptual se va a realizar un análisis de datos, así mismo se consideran las aplicaciones existentes y potenciales. Define y modela aspectos importantes de la información que el negocio necesita saber o tener y las relaciones entre dicha información. Es el primer proceso del modelo top-down para el desarrollo de bases de datos. Se ejecuta durante las fases de análisis y estrategia del ciclo de desarrollo del sistema y debe basarse en la visión del usuario acerca del proceso a automatizar y los datos implicados. Para ello es posible auxiliarse de las siguientes herramientas:

- Pseudocódigo
- Diagramas de flujo (estructuras de control)
- Diagramas de flujo de datos
- Diagramas entidad relación
- Diagramas de estructura (módulos)
- Diseño orientado a objetos
- Manuales de procedimientos
- Manuales de organización
- Árboles de decisión
- Tablas de decisión

Son las unidades básicas para construir cualquier base de datos que se ajuste al modelo.

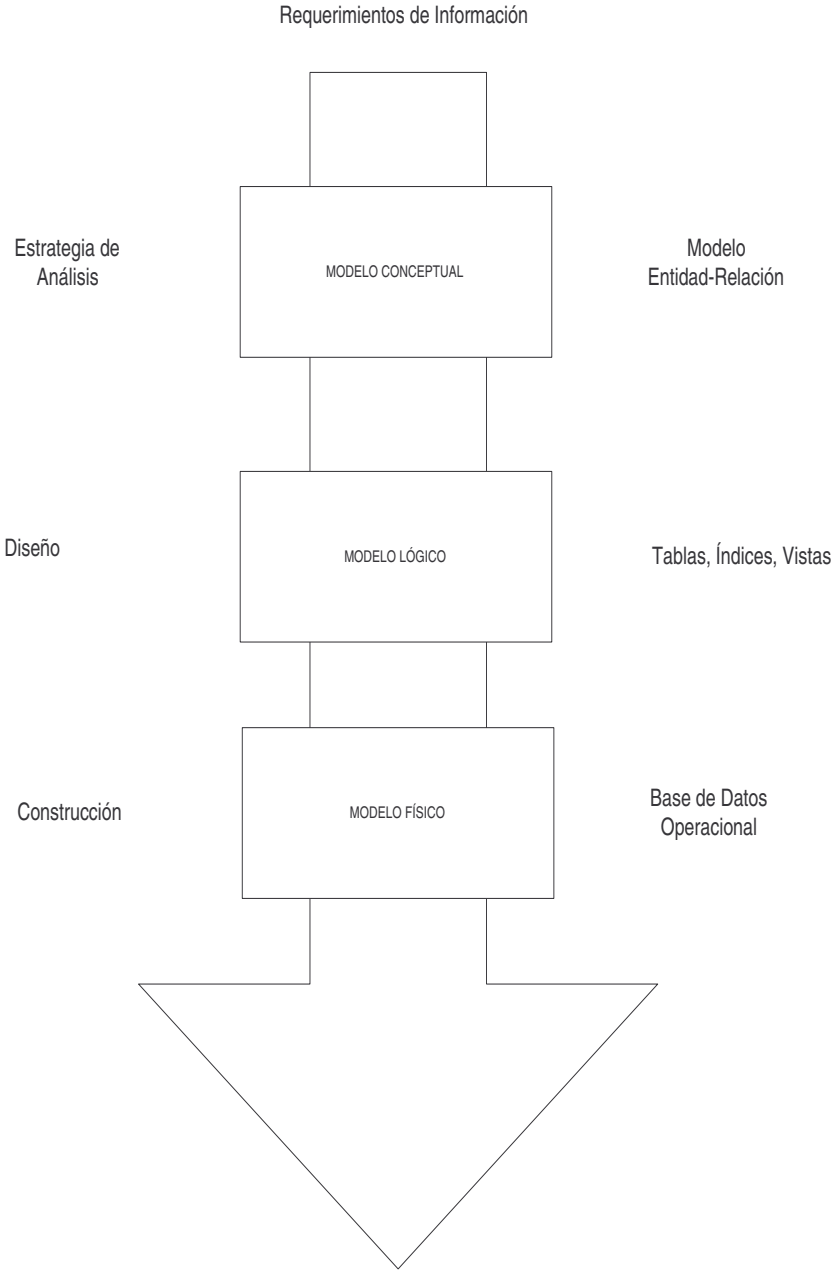
🕮 Modelo Lógico

Requerimientos y procedimientos impuestos por un DBMS. Colección de reglas generales de integridad, las cuales limitan el conjunto de casos de esos tipos de objetos que pueden aparecer en forma legal en cualquier base de datos que se ajuste al modelo (considerando los enfoques relacional (contiene entidades, atributos y relaciones), jerárquico o de red).

🕮 Modelo Físico

Colección de operadores, aplicables a esos casos de objetos para obtener información y para otros propósitos. Especifica cómo se almacenarán los datos, el espacio que será ocupado, métodos de acceso rápido a los datos, etcétera.

MODELO TOP-DOWN DE DISEÑO DE BASES DE DATOS



Componentes del Modelo Relacional.

Entidad
Es una persona, lugar, evento o un objeto identificado en forma única y del cual se registra información y que además cae dentro del alcance del sistema por lo que éste debe mantener, correlacionar y desplegar información. Estas pueden ser de dos tipos:

- Tangibles. Empleados, alumnos, piezas, artículos.
- Intangibles. Un suceso, actividad, la cuenta de un cliente.

Relaciones
Dentro de el enfoque relacional una tabla es conocida como una **Relación**. Una relación es una el conjunto de tuplas(registros) y atributos(campos); y tiene las siguientes propiedades:

- Cada columna contiene valores relativos al mismo atributo, y cada valor de una columna de la tabla debe ser simple (un solo valor)
- Los atributos están en desorden.
- Las tuplas están en desorden
- Las tuplas no ser repiten.

BECARIOS

B1	Lucero	14	Informática
B2	Alfonso	13	Informática
B3	César	12	Ing. Computación
B4	América	11	Arquitectura
B5	Carolina	10	Informática

TUPLAS

ATRIBUTOS

Atributos.
Son las características propias de la entidad, la misma posee atributos básicos que la caracterizan. Los atributos se modelan como columnas de la entidad. La forma de diferenciar las entidades es por medio de atributos, y cada una de ellas debe tener por lo menos un atributo diferente.

Tupla
Conjunto de valores que componen un renglón de la relación. Es equivalente a una instancia de un registro. Es el renglón n de una tabla.

Grado de una tupla.
Número de atributos que tiene una tupla (n de una n-tupla)

Cardinalidad.
Número de tuplas de una relación.

Dominio
Conjunto de valores válidos para un atributo.

BECARIOS

CLAVE	NOMBRE	PROMOCION	CARRERA
B1	Lucero	14	Informática
B2	Alfonso	13	Informática
B3	César	12	Ing. Computación
B4	América	11	Arquitectura
B5	Carolina	10	Informática

Cardinalidad

Grado

Terminología

TERMINO RELACIONAL	EQUIVALENTES
Relación	Tabla
Tupla	Fila o registro
Cardinalidad	Número de filas
Atributo	Columna o campo
Grado	Número de columnas
Llave Primaria	Identificador único de tupla
Dominio	Conjunto de valores legales

TIPO DE LLAVES

- **Llave primaria**. El atributo que identifica de manera única a un registro o renglón de la tabla.
- **Llave candidata**. Atributo o conjunto de atributos que podrían servir como llaves primarias.
- **Llave secundaria.** Todas aquellas llaves candidatas que no se eligieron como llave primaria. Son llaves que tiene todas las características para ser primarias, pero que por una razón o por otra no fueron tomadas como tales, ya que hubo otra (s) que cumplían mejor con ese objetivo.
- **Llave extranjera o foránea.** Una llave foránea (**FK**) es la llave primaria de otra tabla y al mismo tiempo forma parte de otra tabla como atributo.

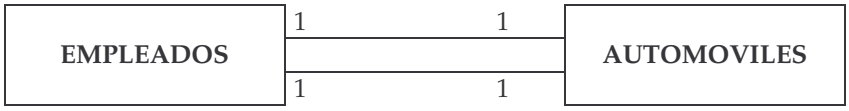
ASOCIACIONES

Una asociación es la unión o enlace de dos o más relaciones, las cuales se encuentran dentro del alcance del sistema, y por ello el sistema debe mantener, correlacionar u desplegar información

Generalmente las asociaciones requieren de al menos dos relaciones (tablas). Existen 3 tipos de asociaciones:

Uno a Uno (1:1)

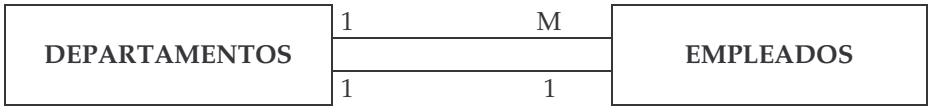
Las ocurrencias de una relación se pueden asociar solo a una ocurrencia de la otra relación. Para los siguientes ejemplos de asociaciones se considerará a una empresa 'X', supóngase que como una prestación de la empresa es proporcionar un automóvil a cada uno de sus empleados, en la asociación uno a uno teniendo las relaciones EMPLEADOS y AUTOMOVILES la asociación se establece de la siguiente manera:



Al modelar este tipo de asociaciones hay que hacerlo cuidadosamente, de manera que los valores nulos se minimicen o se eviten totalmente.

Uno a Muchos (1:M)

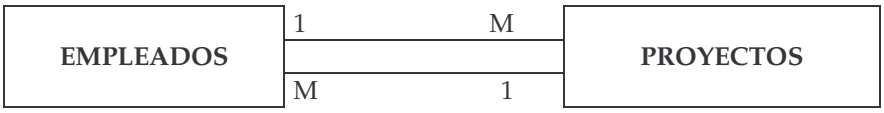
Se dice que una asociación entre relaciones es uno a muchos, si las ocurrencias de una relación están relacionadas con muchas ocurrencias de otra relación. Continuando con el mismo ejemplo, la empresa cuenta con "n" departamentos y de igual manera "m" empleados; para llevar el control del Departamento de Personal, si se consideran las relaciones DEPARAMENTOS y EMPLEADOS, su asociación uno a muchos es la siguiente:



🕮 Muchos a Muchos (M:M)

Ocurre cuando se asocian una ocurrencia en una relación, con muchas ocurrencias en la otra relación y viceversa.

La empresa es una firma de consultoría que elabora proyectos de outsourcing para otras empresas, por ello, cada empleado tiene asignado a su cargo varios proyectos. Teniendo la relaciónes EMPLEADOS consideramos una nueva entidad llamada PROYECTOS para tener un control de las personas que tienen asignado un proyecto. La asociación que se establece es la siguiente:



La asociación que aquí se presenta es de MUCHOS a MUCHOS donde un empleado puede tener a su cargo más de un proyecto, y un proyecto estará cargo de más de un empleado. Para relacionar dos tablas de una asociación M:M, se emplea una tercer relación que comúnmente se llama asociativa o transitiva, y ésta no representa una relación (Tangible o Intangible), sino una asociación entre relaciones.

Esta relación transitiva forma su llave primaria de la correspondiente a las dos relaciones principales de que proviene. Una relación transitiva es una relación en la cual todos los componentes de su llave primaria son llaves foráneas; además esta tabla puede contener más atributos que la caracterizan.

DIAGRAMA ENTIDAD-RELACIÓN

El diagrama entidad-relación (ER) se utiliza como una herramienta de comunicación entre los analistas y diseñadores de sistemas y los usuarios finales durante las fases de análisis de requerimientos y de diseño conceptual debido a que es simple y fácil de entender.

Conceptos Básicos del Diagrama Entidad-Relación

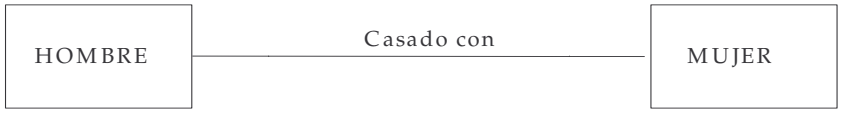
El modelo de datos entidad-relación se basa en una percepción de un mundo real que consiste en un conjunto de objetos básicos llamados entidades y relaciones.

Entidad. Es el objeto principal del cual se tiene que almacenar información, normalmente denotando una persona, lugar, cosa o evento. En un diagrama entidad-relación (DER) las entidades se representan con un rectángulo; un sustantivo en español corresponde al nombre de la entidad en el DER. Tienen propiedades o atributos.

Relación. Representa la asociación o correspondencia entre 2 entidades o de una entidad consigo misma; es binaria, es decir, puede leerse en dos direcciones o sentidos. Asocia una entidad de un conjunto a una o varias entidades de otro. En un DER las relaciones se representan con líneas conectando las entidades relacionadas; normalmente un verbo corresponde a la relación. Formalmente, una asociación es un subconjunto del producto cartesiano de una lista de dominios:

Sintaxis:

Cada entidad 1 { Debe / Puede } ser Nombre de la relación { Uno o más / Uno y sólo uno } Entidad 2



Grado de una relación. Es el número de entidades asociadas en la relación. Una relación n-aria es de grado n. Las relaciones binarias (incluyendo las recursivas) y ternarias son casos especiales donde el grado es de 1, 2 y 3 respectivamente.

BINARIA

Cardinalidad

La cardinalidad de una relación especifica el tipo de asociación de las ocurrencias de las entidades de la relación. Los valores de la cardinalidad son de "uno" o "muchos". Los tipos básicos de cardinalidad son los siguientes: uno a uno, uno a muchos y muchos a muchos.

EJEMPLO:

Esto quiere decir que para cada proyecto sólo hay un empleado. Cada empleado trabaja en un sólo proyecto. (relación uno a uno).

Otro caso es que cada proyecto puede tener muchos empleados. Sólo se puede asociar un proyecto con cada empleado, pero varios empleados se asocian con el mismo proyecto.

Por último, se tiene que cada proyecto puede tener muchos empleados y cada empleado puede estar asociado con más de un proyecto.

Conversión del Diagrama Entidad-Relación al Modelo Relacional

De una manera muy simple se puede decir que las entidades del diagrama ER corresponden a las tablas del modelo relacional y que las relaciones del modelo ER, si tienen campos, también corresponden a tablas del modelo relacional.

Para realizar un buen diseño es necesario tomar en cuenta consideraciones como la cardinalidad y el tipo de relación. A continuación se describen los pasos y consideraciones a seguir.

1. Seleccione los actores (sustantivos) que intervienen en el sistema, sus posibles características y los procesos realizados entre ellos para el funcionamiento del sistema o proceso mayor.
2. Decida finalmente cuáles de todos los seleccionados deben ser empleados y cuáles no.

3. Cree su diagrama entidad-relación con las entidades (actores) y relaciones (procesos) identificados, incluyendo los atributos conservados de interés (características).
4. Resuelva las relaciones muchos a muchos que puedan existir.
5. Rastree cada una de las entidades; aquí le serán de utilidad las tablas de instancias. Estas deben ponerse en tercera forma normal.
6. Divida los datos en tablas. Desnormalice como sea necesario para desempeño.
7. Nombre las columnas en cada tabla, y decida que tipos de datos (y longitud si aplica) se deben utilizar.
8. Decida cuales columnas no deben permitir valores nulos y cuales no.
9. Decida si un default o regla se necesita para cada columna. Se debe tomar en cuenta la relación entre el estado nulo/no nulo de una columna y el default/regla.
10. Crear tipos de datos definidos por el usuario, para aquellas columnas que tengan características similares.
11. Crear las tablas.
12. Establecer las relaciones necesarias.
13. Definir las reglas y defaults.
14. Atar o vincular las reglas y defaults a las columnas apropiadas.
15. Cargar los datos.
16. Definir y crear los índices para incrementar el rendimiento.
17. Definir y crear las vistas y grupos de usuarios; y establecer los derechos de acceso pertinentes
18. Programación de transacciones o procedimientos almacenados necesarios.
19. Transformación de las entidades e interrelaciones a relaciones.
20. Definición de los tipos de datos para los atributos.
21. Definición de las llaves primarias.
22. Definición de transacciones por medio del álgebra relacional.

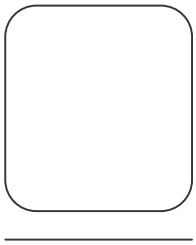
NOTACIÓN RELACIONAL CASE.

Una metodología incluye el conjunto de pasos secuenciales, herramientas, etcétera a emplear en el ciclo de desarrollo de una base de datos. Esta metodología debe incluir una notación especial, que permita representar simbólicamente una base de datos en un diagrama entidad relación.

DIAGRAMA ENTIDAD RELACIÓN.

Como en cualquier campo, un modelo representa parte de la realidad de forma abstracta y facilita la comprensión de un problema a resolver. En el ámbito de las bases de datos, existe un modelo conceptual de la bese de datos, denominado diagrama entidad relación. En él, se representan todas las entidades o "actores", sus atributos o "características" y las relaciones "procesos" existentes entre ellos, todo de manera gráfica para facilitar la comprensión del mismo.

Esta notación (CASE) incluye un conjunto de símbolos para la representación gráfica, mismos que se describen a continuación:

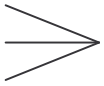


Entidad

Relación Debe

Relación Puede

Cardinalidad indica la forma en que se establece una relación:

Uno  Muchos

Los atributos en una entidad pueden ser de tres tipos:

O	Opcionales	Pueden ser valores nulos o desconocerse.
*	Obligatorios	No pueden ser valores nulos, siempre debe contarse con su valor.
#	PK	Llaves primarias por fuerza no nulas, únicas.

TABLAS DE INSTANCIAS.

Una tabla de instancias, es la "documentación" de las tablas que intervienen en nuestro diagrama entidad relación, facilitan la normalización y permiten verificar las características asignadas a cada entidad y relación.

A continuación un ejemplo de tabla de instancias:

AVION

Columnas	Id_avion	Capacidad	Modelo
Tipo de llave	PK		
NN/U	NN/U	N/NU	NN/UN
Ejemplos	1023	200	Boeing 747
	1024	200	Boeing 747
	1025		Boeing 747

NORMALIZACIÓN

La normalización es un proceso de descomposición sin pérdida, para lograr que nuestras bases de datos estén lo más óptimas posibles.

Una dependencia funcional se representa cuando los valores de un conjunto de atributos de una tupla determinan de manera única los valores de otro conjunto de atributos.

La teoría de normalización está basada en la observación de que cierto conjunto de relaciones presenta mejores propiedades en un medio de actualización, inserción y supresión, que las que presentan otros conjuntos de relaciones que contienen los mismos datos. Para seguir el proceso de normalización, es absolutamente necesario que el diseñador de la base de datos entienda la semántica de la información.

La razón de usar el procedimiento de normalización es asegurar que el modelo conceptual de la base de datos funcionará. Esto no significa que una estructura no normalizada no funcionará, sino que puede causar algunos problemas cuando los programadores de aplicación traten de modificar la base de datos.

Las formas normales son una serie de restricciones que se definen sobre las estructuras relacionales para evitar, como ya se señaló, anomalías al efectuar adiciones, eliminaciones o actualizaciones de tuplas. Con el fin se conseguir que una relación cumpla con una forma normal se efectúa un proceso de descomposición. Ésta implica dividir los atributos de una relación en dos subconjuntos (posiblemente con una intersección no vacía) sin que por ello se pierda alguna información contenida en la relación original.

Las formas de normalización fueron propuestas originalmente por Codd, en 1971 y 1972. Posteriormente varios investigadores continuaron trabajando en esta teoría y a lo largo del tiempo han surgido varias formas de normalización que complementan y refuerzan a las enunciadas por Codd.

Desnormalización

Las relaciones normalizadas evitan anomalías durante la modificación, es por eso que son preferibles que aquellas relaciones que no están normalizadas. Sin embargo existen ocasiones en las que esta normalización no es lo más conveniente o no merece la pena realizarla.

Pasos de la Normalización

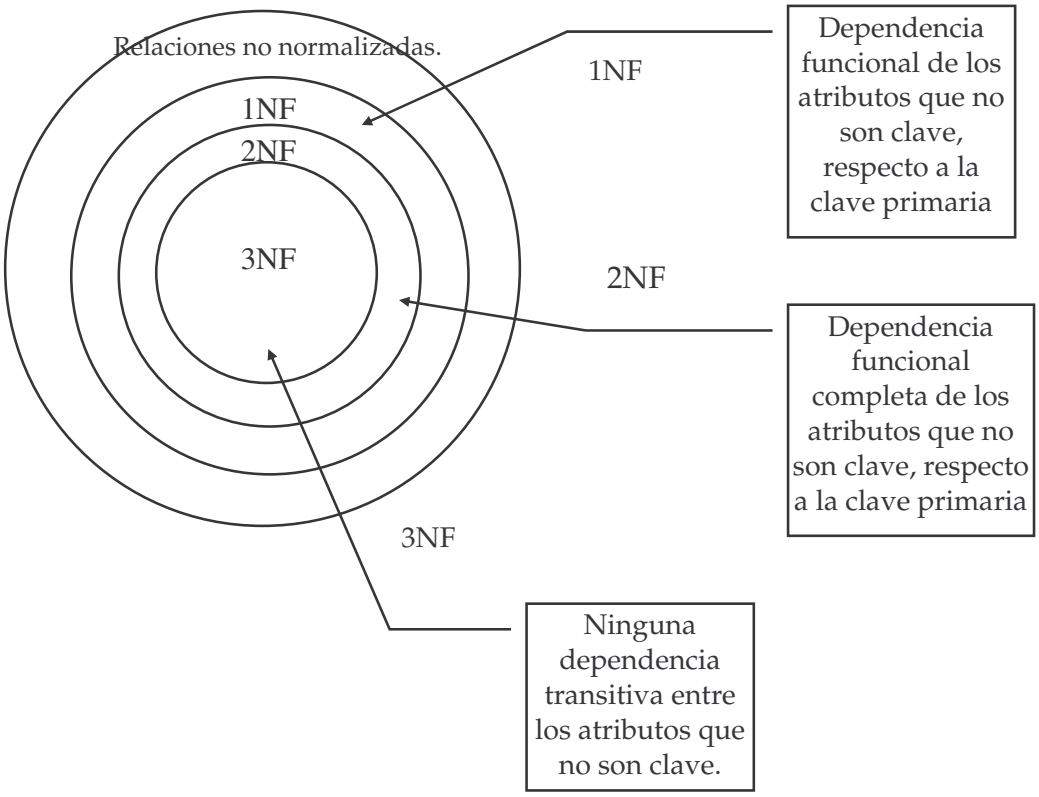
Una vez que se tiene el esquema relacional, se revisa que no haya ninguna relación no normalizada, esto es, con grupos repetitivos. Se eliminan todos los grupos repetitivos de esta relación, obteniendo un conjunto de relaciones en *primera forma normal (1FN).* Se eliminan las dependencias funcionales parciales, para obtener relaciones en segunda forma normal (2FN). Finalmente, se eliminan las dependencias transitivas, creando relaciones en *tercera forma normal (3FN).*

Primera Forma Normal (1FN).

Una relación normalizada es una relación que tiene sólo valores elementales (o simples) en la intersección de cada renglón y columna. Así, una relación normalizada no tiene grupos repetitivos.

Para normalizar una relación que contiene un sólo grupo repetitivo, se elimina el grupo repetitivo y se forman dos nuevas relaciones. *Una relación esta en primera forma normal si no contiene grupos repetitivos.*

La siguiente figura muestra las tres formas normales:



Una entidad R esta en Primera Forma Normal (1NF) si los valores, para cada atributo A ∈ R, son atómicos. Esto implica, que los valores en el dominio no deberán ser listas o conjuntos de valores.

EJEMPLO

La entidad GRUPO que se muestra en seguida no está en 1NF ya que contiene valores que son conjuntos de valores atómicos.

NOMBRE	MATERIA
GUADALUPE, JOSE, IVAN	CALCULO
MARIA, MATILDE	FISICA

Tabla 1. Entidad GRUPO

Para que estuviera en 1NF, tendría que cambiarse por la entidad mostrada a continuación:

NOMBRE	MATERIA
GUADALUPE	CÁLCULO
JOSE	CÁLCULO
IVAN	CÁLCULO
MARIA	FÍSICA
MATILDE	FÍSICA

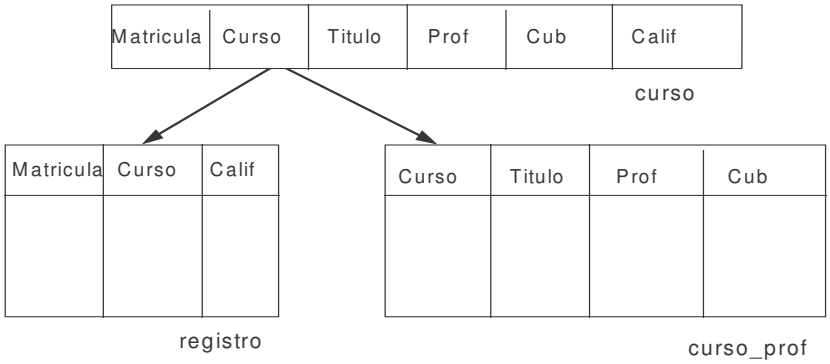
Tabla 2. Entidad GRUPO en 1NF.

Las actualizaciones representan un problema potencial si la entidad no está en 1NF. Supongamos que se quiere cambiar la primera entidad GRUPO del ejemplo, modificando la materia de Guadalupe a Física. El resultado de esta actualización es ambiguo. ¿Qué deberíamos hacer? ¿Mover a Guadalupe de un conjunto a otro o cambiar el valor CALCULO a FISICA? La misma situación aplicada a la segunda entidad GRUPO evita todo tipo de confusión.

Segunda Forma Normal (2FN)

Para eliminar anomalías de la primera forma normal, se deben eliminar las dependencias parciales. *Una relación está en segunda forma normal, si está en primera forma normal y se han eliminado las dependencias parciales.*

Se entiende por dependencia parcial cuando los atributos no llave dependen sólo de una parte de una llave compuesta.



El segundo paso de la normalización es establecer las claves y relacionarlas con los campos de datos. En la primera forma normalizada, el renglón entero de la tabla (tupla) depende de todos los campos de claves. En la segunda forma normalizada, se hace un intento de establecer los campos de datos que están relacionados con alguna parte de la clave completa. Si los campos de datos sólo dependen de una parte de la clave, la clave y los campos conectados a la clave parcial son susceptibles de separarse en registros independientes. La división de la primera tabla normalizada, en una serie de tablas en las que cada campo sólo depende de la clave completa se llama la segunda forma normalizada.

EJEMPLO:

MECANICOS

NumMec	NumCap	Capacidad	NomMec	EdadMec	NumTall	CdTaller	Calif
21	113	Carroceria	Pérez	55	52	Monterrey	3
35	113	Carroceria	Castro	32	44	D.F.	5
35	179	Motor	Castro	32	44	D.F.	1
50	179	Trans	Lopez	40	44	D.F.	2

PK

PRIMERA FORMA NORMAL

Tal cual se puede apreciar los datos de la tabla MECANICOS son muy redundantes. En este punto, la metodología se orienta al problema de cómo buscar en la estructura de datos y que modificar para disminuir la redundancia.

La combinación de campos **NumRec**, y **NumCap** es una llave PK válida para este archivo, lo cual significa "que cada campo no llave depende de la llave primaria"; aunque en ambas partes de la llave primaria compuesta son necesarias para definir el campo **Calif** sólo una de ellas se necesita para definir uno de los otros campos no llave (CatCap, NomMec, EdadMec, NumTall, CdTaller).

MECANICOS

NumMec	NomMec	EdadMec	NumTall	CdTaller
21	Pérez	55	52	Monterrey
35	Castro	32	44	D.F.
50	López	40	44	D.F

PK

CAPACIDAD

NumCap	CatCap
113	Carroceria

179	Motor

PK

CALIFICACION

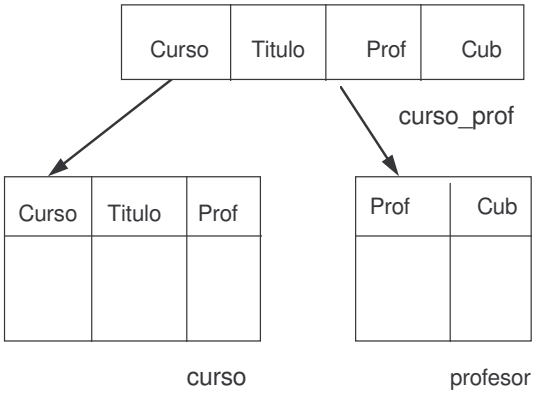
NumMec	NumCap	Calif
21	113	3
35	113	5
35	179	1
50	179	2

PK

Tercera Forma Normal (3FN)

Una relación está en tercera forma normal si está en segunda forma normal y no tiene dependencias transitivas. Esto es, cada atributo no llave depende totalmente de la llave primaria y no hay dependencias transitivas.

Una dependencia transitiva ocurre cuando un atributo no llave depende de uno o más atributos no llave.



EJEMPLO:

Las tablas **Capacidad** y la de **CALIFICACION** están libres de redundancia pues todos los campos no llave dependen de su llave primaria. Así, ambas tablas ya están en Tercera Forma Normal. En cambio la tabla **MECANICOS** no cumple esta regla, ya que NumRec que identifica a cada uno de los otros campos, pero dado, que la llave esta constituida por un campo, la regla de la segunda forma normal de dependencia de los campos no llave con respecto a la llave primaria no se cumple.

De esta manera la situación generada es que un campo no llave define a otros del mismo tipo. Esta forma de relación inesperada, es otro indicativo de que se están mezclando tipos fundamentalmente distintos de información en la misma tabla, acción que provoca la redundancia señalada.

MECANICOS

NumMec	NomMec	EdadMec
21	Pérez	55
35	Castro	32
50	López	40

TALLERES

NumTall	CdTaller
44	D.F.
52	Monterrey

PK

ALGEBRA RELACIONAL

Definición

El álgebra relacional consiste una colección de operaciones sobre relaciones donde cada operación toma una o más relaciones como sus operandos y produce otra relación como su resultado. Dado que el resultado de una operación del álgebra relacional es una relación, ésta a su vez puede ser sujeto de posteriores operaciones algebraicas.

En el álgebra relacional se consideran dos tipos de operadores:

- Los operadores tradicionales sobre conjuntos:
  unión, intersección, diferencia y producto cartesiano.
- Los operadores especiales:
  proyección, selección, "join" y división.

El álgebra relacional es un lenguaje relacionalmente completo. Esto significa que tiene la misma potencia de expresión que el cálculo relacional. Es decir, cualquier fórmula de álgebra relacional es expresable en una sentencia de cálculo relacional, y viceversa.

Por tanto, no es necesario inventar más operadores además de los de unión, intersección, diferencia, división, producto cartesiano, proyección, selección y yunción o join, a no ser que queramos dotar al álgebra de más potencia expresiva que el cálculo.

Operadores de Conjuntos.

Las bases de datos relacionales están basadas en el concepto matemático de relaciones entre conjuntos. Así las operaciones que se pueden efectuar entre relaciones son tanto las comunes a los conjuntos, unión, intersección, diferencia, producto cartesiano; como las específicas de las relaciones, selección, proyección, etc.

Si q, r y s son relaciones con todos los dominios iguales, esto es, con el mismo esquema, se les puede aplicar las operaciones típicas de conjuntos.

- **Unión**

$r \cup s$ Es la relación sobre los mismos dominios que contiene las eneadas que están en r, en s o en ambas. Esto es, son todos los elementos que se encuentran en el conjunto s o en el r. Todos los elementos.

UNION: Construye una relación formada por todas las tuplas que aparecen en cualquiera de las dos relaciones especificadas o la suma de los elementos de dos conjuntos.

Ejemplo:

Sean r y s relaciones con esquema {A,B,C}

r:

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

s:

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

r ∪ s:

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

- **Intersección**

r ∩ s Es la relación que contiene las eneadas que están en r y en s. Esto es, construye una relación formada por aquellas tuplas que aparezcan en las dos relaciones especificadas. Indica todos los elementos que se encuentran tanto en el conjunto r como en el conjunto s, o bien, os elementos comunes a los dos conjuntos.

INTERSECCIÓN: Construye una relación formada por aquellas tuplas que aparezcan en las dos relaciones especificadas, es decir, son los elementos contenidos en ambos conjuntos.

Ejemplo:

Sean r y s relaciones con esquema {A,B,C}

r:

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

s:

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

r ∪ s:

A	B	C
a1	b1	c1

- **Diferencia**

r – s Es la relación con las eneadas que están en r pero no en s. Corresponde a obtener los elementos del conjunto r que no se encuentran en el conjunto s.

DIFERENCIA: Construye una relación formada por todas las tuplas de la primera relación que no aparezcan en la segunda de las dos relaciones especificadas o bien, los elementos que se encuentran en el primer conjunto pero no en el segundo.

Ejemplo:

Sean r y s relaciones con esquema {A,B,C}

r:

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

s:

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

r - s:

A	B	C
a1	b2	c1
a2	b1	c2

- **Producto Cartesiano**

rXq Obtiene todas las eneadas que se construyen concatenando cada eneada de r con otra de s. En este caso los dominios de r y s no tienen que ser los mismos. Ejemplo:

PRODUCTO CARTESIANO: A partir de dos relaciones especificadas, construye una relación que contiene todas las combinaciones posibles de tuplas, una de cada una de las dos, esto es, los pares ordenados.

r:

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

q:

D	E	F
D1	e1	f1
D2	e2	f1
D2	e2	f2

r X q:

A	B	C	D	E	F
a1	b1	c1	d1	e1	f1
a1	b1	c1	d2	e2	f1
a1	b1	c1	d2	e2	f2
a1	b2	c1	d1	e1	f1
a1	b2	c1	d2	e2	f1
a1	b2	c1	d2	e2	f2
a2	b1	c2	d2	e2	f1
a2	b1	c2	d2	e2	f2

DIVISION: Toma dos relaiones, una binaria y una unaria, y construye una relación formada por todos los valores de un atributo de la relación binaria que concuerda con todos los valores en la relación unaria.

De igual forma, si r y s son relaciones con todos los dominios iguales, esto es, con el mismo esquema, se les puede aplicar las operaciones relacionales.

Operadores Relacionales

- **Proyección**

Es una operación unaria. El resultado es un subconjunto de dominios, permite obtener subrelaciones de otras más grandes seleccionando algunos atributos. Se eliminan, luego, las eneadas repetidas. Extrae los atributos especificados de una relación dada:

Primero obtendríamos de r:

A	C
a1	c1
a2	c2
a2	c2

Y después eliminamos las eneadas repetidas obteniendo:

A	C
a1	c1
a2	c2

- **Selección o Restrict**

Produce un subconjunto de las eneadas de la relación que cumplen con una condición (simple o compuesta) sobre los valores para uno o varios de los atributos. Extrae las tuplas especificadas de una relación dada. Por ejemplo, seleccionemos los elementos de r donde existe c2:

r:

A	B	C
a2	b1	c2
a2	b1	c2

- **Unión o Reunión (JOIN)**

r*{condición}q Construye una relación formada por todas las eneadas que aparecen en cualquiera de las dos relaciones especificadas en que se cumple alguna condición en dominios comunes. Se obtiene concatenando una eneada de r con otra de q, de forma que cumpla con una condición en los dominios comunes. Si no hay dominios comunes, esta operación es un producto cartesiano. Sean q y r dos conjuntos como siguen:

REUNION (join): Apartir de dos relaciones especificadas, construye una relación que contiene todas las posibles combinaciones de tuplas, una de cada una de las dos relaciones, tales que las dos tuplas participantes en una combinación dada satisfagan alguna condición especificada.

q:

D	E	F
c1	e1	f1
c2	e2	f1
c2	e2	f2

r:

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

Luego, en la unión, con la condición de que C y D sean iguales tendríamos:

r∪q:

A	B	**C**	**D**	E	F
a1	b1	**c1**	**c1**	e1	f1
a2	b1	**c2**	**c2**	e2	f1
a2	b1	**c2**	**c2**	e2	f2

También es posible efectuar una reunión en atributos que tengan diferentes nombres (como en el ejemplo), pero el mismo dominio. En este caso se llama equireunión [Maier]. Si la condición es sobre atributos con nombres distintos, mismo dominio pero la condición no es sobre igualdades, se le llama reunión donde indica que tipo de condición se tiene (<, >, etc.).

DEFINICIÓN DE SQL

SQL es una herramienta para organizar, gestionar y recuperar datos almacenados en una base de datos. El SQL es un lenguaje para DBMS relacionales que puede ser usado como lenguaje de manipulación de datos y de definición de datos. SQL es parte integral de un sistema de gestión de base de datos, un lenguaje y una herramienta para comunicarse con el DBMS.

PAPEL DE SQL

- SQL es un lenguaje de consultas interactivas
- Lenguaje de Programación de Bases de Datos
- Lenguaje de Administración de Bases de Datos
- Lenguaje Cliente/Servidor
- Lenguaje de Bases de Datos Distribuidas
- SQL es un lenguaje de pasarela de BD

CARACTERÍSTICAS Y BENEFICIOS

- Independencia de los vendedores
- Portabilidad a través de sistemas informáticos
- Estándares SQL
- Apoyo de IBM
- Fundamento relacional
- Estructura de alto nivel en inglés
- Consultas interactivas ad hoc
- Vistas múltiples de datos
- Lenguaje completo de base de datos
- Definición dinámica de los datos
- Arquitectura Cliente/Servidor

SERVIDOR SQL

El servidor SQL maneja los datos y la memoria
- Maneja múltiples bases de datos y varios usuarios
- Mantiene el rastro de la locación actual de los datos en los discos
- Mantiene un mapeo de la descripción lógica de los datos para un almacenamiento físico de los mismos
- Mantiene los datos y procedimientos más utilizados en memoria

El servidor SQL entiende SQL
- Compila y ejecuta batches de instrucciones SQL
- Regresa los resultados a los programas clientes

Optimizador inteligente de consultas en base a su costo
- Automáticamente determina la forma mas eficiente de llevar acabo las tareas

Arquitectura de Procesos Sencillos/Multi-Hilos
- El servidor SQL maneja por si mismo a múltiples usuarios
- No depende del sistema operativo del host para llevar a cabo las tareas

- Mejora el desempeño, al librar al SO de varias tareas

Construido alrededor de Redes

- No es un producto de una sola máquina adaptado para red
- El diseño del servidor reduce el tráfico de la red significativamente.

SINTAXIS DE LOS PRINCIPALES COMANDOS SQL.

COMANDO **SELECT**
Permite elegir atributos específicos de una tabla.

SINTAXIS
SELECT atributo (s) FROM tabla (s);

CLÁUSULA **WHERE**
Sirve para establecer una condición para realizar una selección de tuplas específicas de una tabla.

SINTAXIS
SELECT atributo (s) FROM tabla (s)
WHERE atributo operador condición;

OPERADORES DE COMPARACIÓN

=	igual
!= <>	diferente
>	mayor que
<	menor que
>=	mayor o igual que
<=	menor o igual que

EXPRESIONES LÓGICAS

not	NO se cumple condición
and	condición 1 Y condición 2
or	condición 1 O condición 2

APLICACIÓN DEL NOT

between valor and valor	entre dos valores
in (lista)	dentro de una lista de valores
is null	su valor es nulo
like	busca un patrón de caracteres

COMANDO **CREATE**
Definición de tablas.

SINTAXIS

```
CREATE TABLE tabla (    atributo 1 (tipo (longitud)),
                        atributo 2 (tipo (longitud)),
```

etc.);

COMANDO **INSERT**
Insertar tuplas en una tabla

SINTAXIS
INSERT INTO tabla
(valor atributo 1, valor atributo 2, valor atributo n);

COMANDO **DELETE**
Borrar una tupla en una tabla
DELETE (ELIMINAR) Sirve para eliminar todos los registros de una tabla especificada que cumplan con la condición dada, su sintaxis es la siguiente:

DELETE
FROM
WHERE

SINTAXIS
DELETE tabla
WHERE atributo= condición;

COMANDO **UPDATE**
Cambio de un valor de un atributo en una tupla.

SINTAXIS
UPDATE tabla
SET atributo=nuevo valor
WHERE condición;

UPDATE (actualizar); Sirve para actualizar o modificar una tabla, por ejemplo la sintaxis para este ejemplo sería:

UPDATE Convenios SET Convenios.Vigencia="dos anos", Convenios.Ingresos="s"
WHERE ((Convenios.NodeRegistro="4143-001-5-I-95"));

BIBLIOGRAFÍA RECOMENDADA

Introducción a los sistemas de bases de datos
Date, Chris.J

Técnicas de Bases de Datos: Estructuración en Diseño y Administración
Shakuntala Atre
Ed. Trillas
México D.F. 1988.

Fundamentos de estructuras de datos relacionales
Adad, Rubén; et. al.
Ed. Megabyte

Aplique SQL
Groff, James R. y Weinberg Paul N.
Osborne McGraw-Hill, 1991

Organización de las Bases de Datos
Martin, James.
Prentice-Hall Hispanoamericana, S. A., 1991