Ingeniería en Desarrollo de Software
3er semestre

Programa de la asignatura:
Diseño de bases de datos

Unidad 3. Implementación de bases de datos

Clave:

Ingeniería:	TSU:
15142315	**16142315**

Universidad Abierta y a Distancia de México

Unidad 3. Implementación de bases de datos

Presentación de la Unidad

En este tramo final de la asignatura Diseño de bases de datos iniciarás tu aprendizaje con temas de gran interés, tales como el álgebra relacional, que se refiere a operadores que en ella se utilizan para la mejor administración de las bases de datos, y el cálculo relacional, equivalente al álgebra relacional, pero basado más en la lógica.

Durante la unidad recordarás un poco de lo referente al modelo relacional, y aunado a ello, aprenderás a implementarlo una vez que conozcas la definición de tuplas y esquemas. Otro punto de suma importancia es conocer cómo crear bases de datos y cómo realizar consultas en ellas a través del uso de comandos de SQL, los cuales ayudarán a determinar si tu base de datos es útil para satisfacer las peticiones formuladas por el usuario. Finalmente, conocerás los dispositivos de almacenamientos más comunes y útiles para guardar o respaldar bases de datos.

Propósito

Al término de esta unidad lograrás:

- Comprender los operadores del álgebra relacional y las operaciones lógicas del cálculo relacional necesarias para complementar las bases de datos.
- Realizar bases de datos y consultas de las mismas mediante el uso de comandos SQL, tomando en cuenta los dispositivos de almacenamiento necesarios para el resguardo de las bases de datos.

Competencia específica

Implementar bases de datos para administrar información de un caso específico mediante el uso de los diferentes modelos de manipulación de bases de datos.

3.1. El álgebra y el cálculo relacional

Al estudiar las unidades anteriores, te has dado cuenta que es importante contar con bases de datos (BD) para organizar la información y cuál es la forma adecuada de generarlas; además de ello es de suma importancia que conozcas la formas apropiadas para obtener la información de dichas BD, pues aunque la información se encuentre almacenada, a veces resulta necesario consultar datos específicos.

Primero es importante considerar que una base de datos es un conjunto de información que se encuentra organizada en forma de tablas, pero de nada sirve tener esta información si no se le puede consultar o extraer los datos que se necesitan, aunque éstos se encuentren en diferentes tablas.

Para poder obtener o consultar datos de una BD, en la actualidad existen lenguajes de programación llamados *de consultas*, los cuales verás a detalle más adelante, estos lenguajes se fundan en la combinación de diferentes tablas conocidas como relaciones.

Es importante que conozcas de forma sencilla qué es el álgebra relacional y que es el cálculo relacional, así como la diferencia que hay entre ellos. Dichas definiciones se muestran a continuación:

El *álgebra relacional* muestra paso a paso cómo se obtiene información a partir de tablas relacionadas; por tal motivo se le conoce como de tipo procedimental; mientras que el *cálculo relacional* indica lo que se quiere saber, pero no muestra cómo es posible obtenerlo, y por tal motivo se le denomina de tipo declarativo.

Para entender un poco más lo referente al álgebra relacional, lee las siguientes definiciones:

- **Operandos**: son los valores asignados.
- **Operadores**: se entienden como los procedimientos que generan nuevos operandos o valores.

Para aplicar estos conceptos recuerda que una base de datos se forma de tablas relacionadas entre sí, de manera que a los operadores del álgebra son a los que en álgebra relacional se les denomina *relaciones*.

Para comprender qué es el álgebra relacional, es válido describirla como el conjunto de operaciones que se aplican sobre relaciones en una BD.

Con todo lo descrito anteriormente, es posible resumir que:

- Álgebra relacional es un lenguaje procedimental, es decir, que indica qué y cómo.
- Cálculo relacional es un lenguaje que no es procedimental, indica qué, pero no cómo obtenerlo.

A continuación se muestran algunos ejemplos de lenguajes de consulta que se basan en el álgebra relacional y cálculo relacional para la obtención de dichos datos:

- El SQL (Structured Query Languaje): este lenguaje está basado en el álgebra relacional.
- El QBE (Query Bye Example): este lenguaje está basado en el cálculo relacional.

Las definiciones vistas en este tema sobre álgebra y cálculo relacional son el punto de partida para que puedas comprender la funcionalidad que tienen al momento de consultar datos de una BD, y más aún: cómo se expresan o se generan para para ser interpretadas por los lenguajes de consultas como SQL y QEB.

3.1.1. Una ventaja revolucionaria en la manipulación de datos

Imagina una empresa grande, cualquiera de ellas maneja enormes volúmenes de información, además de que ésta tiene que organizarse con la finalidad de poder consultar datos específicos, por ejemplo:

Un banco necesita registrar cuántos clientes tiene, en cuántos lugares del país y del extranjero, si poseen una o varias cuentas, con uno o miles de movimientos cada una de ellas, o quizá sólo se interesa conocer cuánto quedó a deber un cliente el mes de junio del año pasado. O a lo mejor resulta que ese cliente que ya existe desea abrir una cuenta más de crédito a su nombre, pero desea que se agrupe con sus otras cuentas para conservar sus antecedentes.

Hoy en día la existencia de lenguajes de manipulación de datos DML (*Data Manipulation Language*) permite realizar acciones como consultas, modificación e inserción de datos en un conjunto de información comúnmente llamado base de datos.

Las consultas forman la parte más importante dentro de la manipulación de datos, ya que existe gran variedad de información almacenada en distintas tablas dentro de la base de datos, la principal función de una consulta es extraer la información y la posible relación que existe con otros datos.

Gracias a lenguajes como el DML, se tiene una gran ventaja que vino a revolucionar la manera de manipular la información, simplificando y haciendo la tarea muchísimas veces más rápida y confiable, sin dejar atrás los lenguajes formales tales como el álgebra relacional y el cálculo relacional.

A continuación, estudiarás cada uno de los lenguajes y verás la forma tan sencilla de obtener la información que desees de una base de datos.

3.1.2. Álgebra relacional

Como ya se mencionó, el álgebra relacional es el conjunto de operaciones que con base en las relaciones que existen entre tablas, son útiles para la obtención de cierta información en una BD.

El estudio y la comprensión de este subtema serán de utilidad para que posteriormente comprendas cómo funciona una consulta de datos en SQL.

El álgebra relacional es un lenguaje formal que muestra técnicas fundamentales para la extracción de información de una base de datos. A continuación se mostrarán las técnicas mencionadas:

Operadores básicos

- Unión
- Diferencia
- Producto cartesiano
- Selección
- Proyección

Operadores derivados

- Intersección
- Join
- División

A continuación se explica cada una de estas técnicas y cómo funcionan, todo esto para que diferencies en qué caso te conviene aplicar cada una de ellas al momento de consultar información de una BD.

De cada una de las técnicas se muestra un ejemplo de tablas y, posteriormente, cómo quedaría ésta al aplicar dicha técnica.

Unión

Como su nombre lo dice, este operador une o junta en una misma relación las tuplas o registros de dos relaciones diferentes. Éstas se encuentran condicionadas a que sean compatibles, es decir, que tengan el mismo tipo de atributos, y se expresa de la siguiente manera.

Tabla clientes contado

ID	NOMBRE	EDAD
98	Juan López	33
99	Arturo Jiménez	23

Tabla clientes crédito

ID	NOMBRE	EDAD
14	Mónica Ramos	25
15	David Hernández	36

clientes contado ∪ clientes crédito

ID	NOMBRE	EDAD

98	Juan López	33
99	Arturo Jiménez	23
14	Mónica Ramos	25
15	David Hernández	36

Diferencia

Es la resta de dos relaciones o tablas, generando una nueva relación donde de la primera se quitan aquellos registros que coincidan. Dicho de otra manera: se quedan aquellos registros que están en la primera pero no en la segunda.

De la misma manera que en caso de la unión, ambas tablas deben tener los mismos atributos para que sean posibles.

Tabla clientes

ID	NOMBRE	EDAD
98	Juan López	33
99	Arturo Jiménez	23
100	Jaime Reyes	38

Tabla cliente contado

ID	NOMBRE	EDAD
98	Juan López	33
15	Margarita Juárez	42

clientes – contado

ID	NOMBRE	EDAD
99	Arturo Jiménez	23
100	Jaime Reyes	38

contado – clientes

ID	NOMBRE	EDAD
15	Margarita Juárez	42

Producto cartesiano

Define una relación que es la unión de cada fila de una entidad con la fila de otra entidad.

Tabla clientes

ID	NOMBRE	EDAD
98	Juan López	33
99	Arturo Jiménez	23

Tabla productos

IDpro	DESCRIPCION	UNIDAD
23	Manzana	Kg
24	Salsa roja	Pz
25	Azúcar	Kg

Resultado de aplicar el operador de producto cartesiano:

IDpro	DESCRIPCION	UNIDAD	ID	NOMBRE	EDAD
23	Manzana	Kg	98	Juan López	33

24	Salsa roja	Pz	99	Arturo Jiménez	23
25	Azúcar	Kg	98	Juan López	33
23	Manzana	Kg	99	Arturo Jiménez	23
24	Salsa roja	Pz	98	Juan López	33
25	Azúcar	Kg	99	Arturo Jiménez	23

Selección

Se extrae una nueva tabla con los mismos atributos que cumplen con la condición especificada. Y se simboliza con σ.

Tabla clientes

ID	NOMBRE	EDAD
98	Juan López	33
99	Arturo Jiménez	23
14	Mónica Ramos	25
15	David Hernández	36

σ edad>= 34 años (clientes)

ID	NOMBRE	EDAD
15	David Hernández	36

Proyección

Produce una tabla como resultado con los atributos que se especifican eliminando las filas duplicadas; su símbolo es π.

Tabla clientes

ID	NOMBRE	EDAD
98	Juan	33
99	Arturo	23
14	Mónica	25
15	David	36
33	Juan	25

π nombre, edad (clientes)

NOMBRE	EDAD
Juan	33
Arturo	23
Mónica	25
David	36
Juan	25

π nombre (clientes)

NOMBRE
Juan
Arturo
Mónica
David

Intersección

Es el resultado de aquellos registros que se encuentran en ambas tablas; su símbolo es ∩.

Tabla clientes contado

ID	NOMBRE	EDAD
98	Juan López	33
99	Arturo Jiménez	23

Tabla clientes crédito

ID	NOMBRE	EDAD
14	Mónica Ramos	25
15	David Hernández	36
99	Arturo Jiménez	23

clientes contado ∩ clientes créditos

ID	NOMBRE	EDAD
99	Arturo Jiménez	23

***Join* (unión natural)**

Es el resultado de unir dos relaciones, combinando los atributos de las dos. Para lograr este *join* las entidades deben tener un atributo en común, y se simboliza *.

Tabla empleados

ID	NOMBRE	EDAD	IDP
98	Juan López	33	98
99	Arturo Jiménez	23	99
14	Mónica Ramos	25	99
15	David Hernández	36	114

Tabla puestos

IDP	PUESTO
98	Almacenista
99	Vendedor

empleados * puestos

ID	NOMBRE	EDAD	IDP	PUESTO
98	Juan López	33	98	Almacenista

	99	Arturo Jiménez	23	99	Vendedor
	14	Mónica Ramos	25	99	Vendedor

División o cociente
Es el conjunto de atributos en dos relaciones en donde la segunda relación está incluida en la primera. Y se representa con ÷.

Tabla clientes

ID	EDAD
98	33
99	23
14	25
15	33

Tabla edad promedio

EDAD
23
33

clientes ÷ edad promedio

ID
98
99
15

Retomando todo lo anterior y ya conociendo todas las técnicas que existen en el álgebra y en qué momento se puede aplicar cada una de ellas, te darás cuenta que se convierten en una herramienta básica para la manipulación de los datos en una BD, también piensa que se pueden combinar entre ellas y aplicar una sobre otra, haciendo la obtención de la información más simple y sobre todo rápida para llegar al resultado.

3.1.3. Cálculo relacional

A continuación verás cómo a través del cálculo relacional puedes expresar lo que deseas obtener de tus bases de datos. Estudiar este tema tiene la finalidad de que al trasladarlas a un lenguaje como el SQL, lo hagas de una manera fácil y rápida, sabiendo que la información que muestres sea exactamente la que quieres saber.

Recuerda que el álgebra relacional sólo muestra lo que se desea saber sobre ciertos datos específicos en una consulta, no dice cómo obtenerlo, es la representación gráfica sobre lo que se dice con palabras. Para entender este tema se expondrá en dos partes: cálculo de tuplas y cálculo de dominios.

Cálculo de tuplas:

Para entender qué es el cálculo de tuplas, recuerda que una *tupla* es un registro o una fila de una tabla en una base de datos. Se utilizarán los siguientes símbolos para expresar el cálculo de tuplas:

t	Es la variable que representa lo que se desea saber.
P(t)	Es la fórmula o forma en la que se va a obtener.
¬	Negación.
←	Implicación.
∧	Conclusión: conocido también como "*and*" o "y".
∨	Disyunción: conocido también como "or" u "o".
∀	Cuantificador universal, también conocido "para todo".
∃	Cuantificador existencial, también usado para "existe al menos un".

La forma de expresar una consulta en cálculo de tuplas está dada por:

{ t / P (t)}

De manera que, suponiendo una base de datos de una empresa que se dedica al diseño de sistemas, y que tiene dos tablas de empleados (una contiene los datos de los programadores y otra los de analistas) y que la consulta que se desea es obtener todos los empleados de la empresa, dicha base se expresaría de la siguiente forma:

{t| (t ∃ Programadores) ∨ (t ∃ Analistas) }

La anterior consulta define los trabajadores que existen; se quiere saber cuáles son programadores y cuáles analistas.

Ahora, si de esta misma base de datos se necesita saber los empleados que hacen las dos funciones, es decir, los que son programadores y analistas al mismo tiempo, la expresión sería la siguiente:

{t| (t ∃ Programadores) ∧ (t ∃Analistas) }

Otro caso: se quiere saber los empleados que son programadores, pero no analistas:
{t| (t ∃ Programadores) ∧¬ (t ∃Analistas) }

Ahora que ya sabes cómo expresar consultas a modo de cálculo de tuplas, vas a revisar la siguiente técnica para expresar consultas a modo de cálculo de dominio.

Cálculo relacional de dominios

El cálculo relacional de dominios también expresa consultas a través de símbolos, pero en lugar de hacerlo sobre las tuplas, registros o renglones, lo hace mediante dominios, mejor entendidos como atributos o encabezados de columnas.

Su representación es de la siguiente manera:

{<x1, x2,…,xn> | P(x1,x2,…,xn)}

donde X1 a Xn son los atributos que se desea conocer al momento de hacer la consulta y P(x1 a Xn) son los atributos de las tablas existentes.

La técnica de cálculo de dominios se apoya con el uso de los símbolos de cálculo de tuplas, pero usando dominios.
A continuación se muestra un ejemplo de uso de la técnica de cálculo de dominios, en donde se tiene una base de datos con las siguientes tablas y atributos:

Obtener los alumnos con promedio superior a 4:

{<codigo,nombre,edad,promedio> |(t ∃alumnos) ∧ (promedio >4)}

Ahora que ya conoces la forma de hacer operaciones con algebra relacional y la manera de representarlas con cálculo relacional, ya estás listo(a) para comprender cómo convertir estas consultas a un código en un lenguaje y ser aplicados en una base de datos; sin embargo, antes se comentará acerca de la dificultad que existe para el uso del álgebra y el cálculo relacional.

3.1.4. La dificultad relativa del álgebra relacional y el cálculo relacional

A continuación se presenta una explicación de cómo se muestra la información en las bases de datos y cómo intervienen el álgebra relacional y el cálculo relacional, con base en consultas para poder acceder a ciertos datos de la información almacenada.

El poder representar todas las tablas de una base de datos y sus atributos, y sobre todo las relaciones, es muy fácil si se trata de una base de datos pequeña, pero recuerda que no siempre va a ser así, pues existen bases de datos tan complejas que resulta prácticamente imposible realizar estas consultas en papel; lo valioso de haber tratado estos ejemplos consiste en que te ayudarán a razonar qué es lo que quieres en el álgebra y cómo reescribirlo en el cálculo.

Los *lenguajes de consulta* son los que solicitan información de la base de datos. Y si dominas éstos, realizarás estas consultas de manera más fácil y directamente en el sistema, pero debes saber que los lenguajes de consulta como el SQL, están en niveles superiores a los lenguajes de programación.

Los lenguajes de consulta se clasifican como procedimentales o no procedimentales. Los lenguajes procedimentales son en los cuales el usuario dice al sistema que lleve a cabo una serie de operaciones en la base de datos. En los lenguajes no procedimentales, el

usuario describe la información deseada, sin dar un procedimiento concreto para obtener esa información, de la misma manera que lo hacen el álgebra y el cálculo relacional.

3.2. Implementación relacional

En importante que recuerdes un poco sobre cómo se hace una base de datos en el modelo relacional. El modelo relacional se considera un modelo lógico de no muy alto nivel, ya que está orientado a los registros, las tablas y las relaciones entre ellas. Por su parte el sistema gestor de bases de datos es en donde se va a manipular ésta, y hay muchos en el mercado.

Otro aspecto a considerar es la diferencia entre el modelo relacional y el modelo entidad-relación, los cuales se representan de manera diferente.

A continuación se te presentarán algunos términos, que si bien ya estudiaste, servirán para reforzar el conocimiento:

Entidad: es el conjunto de todas las filas de una relación y sus atributos.
Atributo: son todas las columnas de una relación.

En el modelo entidad-relación se utilizan diferentes símbolos: círculos para los atributos, cuadrados para las entidades, y rombos para las relaciones.

Al transformar del modelo entidad-relación a relacional debes seguir una serie de reglas para no perder la conexión entre las tablas, como lo pudiste ver en la unidad 2. Es necesario dominar este tema para implementar la estructura de dichas tablas, así como la manera en la que éstas se convierten de entidad-relación a relacional; si no es así, antes de proseguir realiza una nueva lectura, ya que es la base para comprender lo que estudiarás a continuación.

3.2.1. Modelo relacional

Para poder aplicar consultas, las cuales tienen su origen en el álgebra y el cálculo relacional, hay que tener presente el modelo relacional, con el fin de que se pueda dar una secuencia y se entienda cómo aplicar consultas, sobre todo cuando se involucran más de una tabla, ya sea que estén relacionadas o no.

Ya que el modelo relacional no utiliza diagramas para representar su información, es posible apoyarse con el uso de tablas (esto es opcional, y podrás encontrarte con autores que manejan diferentes formas de representarlo).

El siguiente ejemplo muestra cómo de un determinado modelo entidad-relación se puede conseguir un modelo relacional:

Tabla alumno

Matrícula	Nombre	Dirección	Teléfono
15245	Juan Pérez	Calle los Olmos no. 25	7777777

Tabla materia

Clave materia	Nombre materia	Duración materia
A23	Algebra	75 horas
A24	Español	80 horas

Tabla relación materia-alumno

Matrícula	Clave materia	Calificación
15245	A23	80
15245	A24	100

Cada una de estas tablas proviene de ciertas entidades con sus respectivos atributos, y considerando el tipo de relación que tengan, se generará la base de datos.

Una vez que recordaste cómo pasar de un modelo entidad-relación a uno relacional y que de éste ya conociste el origen de su nombre y cómo se representa, en el siguiente tema estudiarás específicamente lo que son las tablas y los esquemas.

3.2.2. Definición tablas y esquemas

Es importante hablar de tablas, debido a que con base en ellas se definirán éstas y sus atributos que tendrá la BD que desees crear; aunado a ello, es importante hablar de esquemas, ya que muestran de forma clara las tablas relacionadas en la base de datos.

De este modo una tabla es un grupo de información clasificada de acuerdo al contenido de la misma, y por tanto hay diferentes formas de representarla, la más simple es la que hasta ahora has estudiado; ve como ejemplo la siguiente tabla.

Matrícula	Nombre	Dirección	Teléfono
15245	Juan Pérez	Calle los Olmos no. 25	7777777

Dentro de las tablas, también se encuentra información sobre la manera en que se relaciona con otras, dando origen a una nueva forma de representar una base de datos a través de un esquema, el cual es usado en el momento de programarla en su lenguaje, y

por ello es conveniente asignarle un nombre para posteriormente hacer referencia al mismo. Existen convencionalismos para hacerlo y están dados por lo que a continuación se presenta:

- Nombres en minúsculas para las relaciones
- Sin espacios entre nombres, se pueden usar – o _
- Nombres en minúsculas para los esquemas de las relaciones, pero comenzar en mayúsculas

Se muestra un ejemplo de lo anterior:

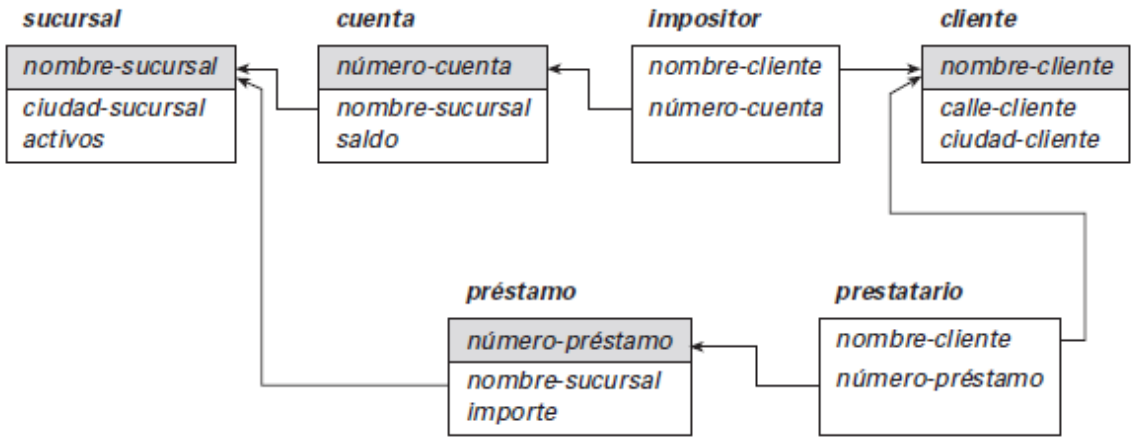
```
Esquema-Alumno(matrícula,nombre,dirección, fecha-de nac)
```

Se denota que *alumno* es una relación de *esquema-alumno mediante alumno*. En general, los esquemas de las relaciones incluyen una lista de los atributos y de sus dominios correspondientes.

El esquema también puede representarse gráficamente a través de cuadros, donde cada uno de éstos representa una tabla y se encuentra subrayado o resaltado; la clave principal de cada tabla en dicho esquema se denota en la clave que pasa como foránea o secundaria a la siguiente, para no perder la relación entre tablas; asimismo se representa con flechas cómo se relacionan cada una de éstas. Se debe colocar el nombre de cada tabla en la parte superior.

Silberschatz y Korth (2002, p. 58) afirman que “un esquema de bases de datos, junto con las dependencias de clave primaria y externa, se puede mostrar gráficamente mediante diagramas de esquema”. A continuación se presenta una figura que muestra el diagrama de esquema aplicado a un ejemplo del sector bancario. Cada relación aparece como un cuadro, con los atributos listados dentro de él y el nombre de la relación sobre él. Si hay atributos con clave primaria, una línea horizontal cruza el cuadro con los atributos clave primaria listados sobre ella. Las dependencias de clave externa aparecen como flechas desde los atributos clave externa de la relación referenciada, a la clave primaria de la relación referenciada.

Silberschatz y Korth (2002, p. 58) advierten que “no hay que confundir un diagrama de esquema con un diagrama E-R. En particular, los diagramas E-R no muestran explícitamente los atributos clave externa, mientras que los diagramas de esquema sí”.



Esquema banco
Tomado de Silberschatz y Korth (2002).

Los términos presentados anteriormente te ayudan a entender cómo se estructura de principio a fin una base de datos. El esquema es el que da forma gráfica, presenta las relaciones que existen entre las tablas de una base de datos, mismas que son útiles para cuando se quiere consultar información.

3.2.3. Manipulación de datos

Una vez revisados los temas sobre tablas y esquemas, puedes comenzar con los lenguajes de consulta que tienen su origen en el álgebra y el cálculo relaciona. El lenguaje de consulta estructurado SQL (Structured Query Language) es un lenguaje de base de datos utilizado por Microsoft Jet.

Está formado por instrucciones que se componen de cláusulas, comandos y operadores. Estos elementos se combinan en las instrucciones para crear, actualizar y manipular las bases de datos. Los comandos SQL se clasifican en dos tipos:

- Los DDL (Lenguaje de Definición de Datos), que permiten crear y definir nuevas bases de datos, campos e índices.
- Los DML (Lenguaje de Manipulación de Datos), que permiten generar consultas para ordenar, filtrar y extraer datos.

En la siguiente tabla se mencionan y describen algunos de los comandos de SQL.

Comandos DLL	***Comandos DML***
CREATE: utilizado para crear nuevas tablas	**SELECT**: utilizado para consultar registros de la base de datos que satisfagan un criterio determinado
DROP: utilizado para eliminar tablas	**INSERT**: utilizado para cargar lotes de datos en la base en una única operación
ALTER: utilizado para modificar las tablas, agregando campos o cambiando la definición de éstos	**UPDATE**: utilizado para modificar los valores de los campos y registros especificados
	DELETE: utilizado para eliminar registros de una tabla de una base de datos

Las cláusulas son condiciones de modificación utilizadas para definir los datos que desea seleccionar o manipular; a continuación se muestran algunas de ellas:

- **FROM**: utilizada para especificar la tabla de la cual se van a seleccionar los registros
- **WHERE**: utilizada para especificar las condiciones que deben reunir los registros que se van a seleccionar
- **ORDER BY**: utilizada para ordenar los registros seleccionados de acuerdo con un orden específico

A continuación se muestra una instrucción utilizando el comando *create*, para crear una base de datos:

Createdatabase alumnos: esta instrucción crea la base de datos “alumnos”.

Operadores lógicos

- **AND**: el resultado debe cumplir ambas expresiones antes y después del *and* para dar el resultado deseado.
- **OR**: el resultado está en función de si una u otra expresión se cumple.
- **NOT**: el resultado invierte la condición de la expresión.

Operadores relacionales

<menor que,	*a<b*	*a* es menor que *b*

>mayor que	*a>b*	*a* es mayor que *b*
<>distinto de	a<> a	*a* es distinto que *b*
<= menor que o igual	a<=*b*	*a* es menor o igual que *b*
>= mayor que o igual	a>=*b*	*a* es mayor o igual que *b*
BETWEEN: utilizado para especificar un intervalo de valores		

Funciones

Las funciones de agregado se usan dentro de una cláusula s*elect* en grupos de registros para devolver un único valor que se aplica a un grupo de registros.

- **AVG**: utilizada para calcular el promedio de los valores de un campo determinado
- **COUNT**: utilizada para devolver el número de registros de la selección
- **SUM**: utilizada para devolver la suma de todos los valores de un campo determinado
- **MAX**: utilizada para devolver el valor más alto de un campo especificado
- **MIN**: utilizada para devolver el valor más bajo de un campo especificado

La sintaxis básica para crear una base de datos en SQL es:

CREATE DATABASE nombre_base de datos

Cuando ya ha sido creada la base de datos, lo siguiente es hacer las tablas, para lo cual se debe conocer primero los tipos de datos que se pueden utilizar, y estos varían dependiendo del manejador de bases de datos que se use; sin embargo la mayoría reconoce estos tipos de datos.

Tipo de de dato	Características
VARCHAR (tamaño)	Almacena cadenas de caracteres de una longitud variable. La longitud máxima son 4 000 caracteres.
CHAR (tamaño)	Almacena caracteres con una longitud fija de 2 000 caracteres el máximo.
NUMBER (precision,escala)	Almacena datos numéricos, tanto enteros como decimales, con o sin signo. Indica: • Precisión: el número máximo de dígitos que va a tener el dato

	• Escala: el número de dígitos que puede haber a la derecha del punto decimal
LONG	Almacena cadenas de caracteres de longitud variable. Puede almacenar hasta 2 Gb de información
DATE	Almacena información de fechas y horas. De forma predeterminada almacena un dato con el siguiente formato: siglo/año/mes/dia/hora/minutos/segundos Este formato se puede cambiar con otros parámetros.

Una vez que sabes los tipos de datos se muestra un ejemplo de cómo se crea una tabla:

CREATE TABLE nombre_tabla (nombre_columnatipo_columna [cláusula_defecto] [vínculos_de_columna][, nombre_columnatipo_columna [cláusula_defecto] [vínculos_de_columna] ...]
[, [vínculo_de tabla] ...])

nombre_columna: es el nombre de la columna que compone la tabla.

tipo_columna: es la indicación del tipo de dato que la columna podrá contener.

cláusula_defecto: indica el valor de defecto que tomará la columna si no se le asigna uno explícitamente en el momento en que se crea la línea. La sintaxis que hay que usar es la que se indica en el ejemplo siguiente:

Antes de crear tablas debes conocer cómo crear las claves primarias y las claves foráneas que vienen de otra relación.

- La definición de la llave primaria:

  PRIMARY KEY (columna1 [, columna2 ...])

- Las definiciones de las llaves externas:

  FOREIGN KEY (columna1 [, columna2 ...]) definiciones_de_referencia

Cuando ya se tienen las tablas creadas, lo siguiente es el manejo o manipulación de los datos, cómo agregarlos, eliminarlos y modificarlos:

Para aclarar mejor el uso de la instrucción CREATE TABLE, se mostrarán algunas órdenes que implementan la base de datos ejemplificada.

CREATE TABLE Ventas (
ID INTEGER PRIMARY KEY,
nombreclienteCHAR(18) NOT NULL
);

Esta instrucción crea la tabla Ventas y sólo contiene el atributo ID de tipo *integer* y nombrecliente de tipo *char* con 18 caracteres, *id* es la clave primaria, y se está poniendo como requisito que el nombre de cliente sí tenga datos, al no permitir nulos.

CREATE TABLE libro (ID INTEGER PRIMARY KEY,
título VARCHAR(160) NOT NULL,
publicación INTEGER NOT NULL,
volumen VARCHAR(16),
series VARCHAR(160),
edición VARCHAR(16),
notas VARCHAR(255)
);

Para borrar tablas se utiliza el comando de SQL DROP TABLE nombre_tabla. Si la tabla tiene datos, éstos van a ser borrados permanentemente. Una vez que la tabla se borra no se puede recuperar.

Para agregar datos a las tablas se utiliza la siguiente instrucción:

INSERT INTO nombre_tabla[(lista_campos)]
VALUES (lista_valores)

nombre_tabla es el nombre de la tabla en la que se tiene que incluir la nueva línea.

lista_campos es la lista de los nombres de los campos a los que hay que asignar un valor, separados entre sí por una coma. Los campos no incluidos en la lista tomarán su valor por defecto o NULL si no lo tienen; de no hacerlo se estaría incurriendo en un error. En caso de que no se especifique la lista, habrá que especificar los valores de todos los campos de la tabla.

La sintaxis básica de una consulta de selección es la siguiente:

SELECT Campos FROM Tabla;

“Campos” es la lista de campos que se desean recuperar y “Tabla” es el origen de los mismos, por ejemplo:

SELECT Nombre, Teléfono FROM Clientes;

Usando estas consultas, y combinando con los operadores lógicos, relacionales y las funciones, podrás lograr consultas más avanzadas. Una consulta es usada con vistas y éstas serán aplicadas en el siguiente tema.

3.2.4. Definición de vistas

Las vistas fueron creadas para encapsular las consultas y para que los usuarios no tengan el acceso a las mismas por cuestiones de seguridad; así sólo algunos de los datos quedarán ocultos para ciertos usuarios.

Por ejemplo: un alumno que desea saber el número de calificación de cierta materia debería ver una relación descrita, en el álgebra relacional, mediante:

πnombre-alumno, numero-calif_materia (alumno x calif_materia)

Además de las consideraciones sobre la seguridad, puede que se desee crear un conjunto personalizado de relaciones que se adapte mejor que el modelo lógico a la intuición de un usuario concreto.

Definición de vistas. Las vistas se definen con la instrucción **createview**. Se tomará en cuenta que para definir o crear una vista, se debe asignar un nombre a ésta e indicar la consulta que la calculará. La forma de la instrucción **view** es:

Createview *view_name* **as <**expresión de consulta**>** donde:

- **Createview** es la expresión que crea la vista.
- View_name *representa el nombre de la vista que se creará.*
- **<expresión de consulta>** se refiere a la consulta que guardará y ejecutará la vista.

Un ejemplo: imagina una vista en una base de datos de una determinada biblioteca. La vista se llamará autores_mexicanos y se quiere que anide una consulta que muestre algunos atributos de la tabla autores cuyo país sea México, se expresaría:

Créate view autores_mexicanos **as**

(**Select** id_autor, nombre_autor

From autores

Where país = “Mexico”)

Cuando se define la vista, ésta puede ser utilizada en cualquier momento, mientras se haga referencia a ella, pero se debe de tener cuidado cuando se modifiquen las relaciones de las tablas a las que hace referencia la vista, debido a que puede generar errores en la consulta interna.

Por tal motivo, el sistema gestor de vistas guarda sólo la vista creada y no el resultado que arroja cuando se manda llamar.

Tomando en cuenta las tablas siguientes, mediante SQL se tiene la creación de vistas de la siguiente manera:

Una tabla, con múltiples vistas

Tabla empleados

No. empleado	Nombre	Puesto	Fecha-nac	Salario	Extras	No. depto
2334	Juan	Supervisor	17-dic-80	6000	0	45
5677	Alejandra	Secretaria	20-Feb-81	4000	2000	46
3456	Arturo	Supervisor	23-Ago-75	7000	0	56

Vista de supervisores

Nombre	Puesto
Juan	Supervisor
Arturo	Supervisor

Vista de secretarias

Nombre	Puesto	Salario
Alejandra	Secretaria	4000

¿Cómo crear, nombrar y consultar vistas?

```
SQL> CREATE VIEW      secretaria     AS
       SELECT         nombre, puesto, salario
       FROM           Empleados
       WHERE      Puesto= 'secretaria';
```

Se puede utilizar cualquier *select* que contenga un *order by* cuando se crea una vista. Para consultar una vista, se hace un *select* como si la vista fuera una tabla.

SQL> SELECT * **FROM** Secretaria;

Creando *vistas* con *alias* de nombres de columnas

A menos que se especifique lo contrario, la vista heredan los nombres de las columnas de la tabla de la que se deriva. Para dar a la vista nombres de columnas diferentes a los de la tabla base, se utiliza la siguiente sintaxis:

CREATE VIEW nombre_vista (alias, alias,...) AS query

SQL> CREATE VIEW MYDEPT
(PERSON,TITLE,SALARY)
AS SELECT nombre, puesto, salario
FROM Empleado
WHERE No. depto=10;

Razones para copiar tablas y vistas

- Respaldar tablas y vistas originales
- Dar una copia a alguien más
- Hacer cambios tentativos
- Archivarlas antes de borrarlas

Borrado de vistas

Cuando no se necesitan o simplemente se quiere eliminar una vista previamente ejecutada, se requiere de ciertos comandos, como los que a continuación se muestran.

Para borrar vistas

El comando de SQL es DROP VIEW nombre_vista. Las tablas en las que se basa la vista no se alteran. Las vistas pueden llegar a simplificar el uso de consultas y hacen que la información de cómo elaborar la consulta no esté al 100% disponible para el usuario al permitir renombrarla con un nombre específico.

3.3. Implementación relacional con consulta gráficos

El lenguaje de consulta (QBE), por su importancia, es uno de los primeros lenguajes de consulta gráficos con sintaxis mínima para los sistemas de bases de datos; este lenguaje es considerado como la versión gráfica para la implementación relacional de bases de datos.

De acuerdo con Ramez y Shamkant (2007, p. 903), las consultas QBE de recuperación,

> se especifican rellenando una o más filas de las plantillas de las tablas. En una consulta de una sola relación, introducimos constantes o *elementos de ejemplo* (un término QBE) en las columnas de la plantilla de esa relación. Un elemento de ejemplo simboliza una variable de dominio y se especifica como un valor de ejemplo precedido por el carácter de subrayado U. Además, se introduce un prefijo P (denominado operador de punto P) en determinadas columnas para indicar que nos gustaría imprimir (o visualizar) los valores de esas columnas para nuestro

resultado. Las constantes especifican valores que deben ser exactamente iguales a los presentes en esas columnas.

3.3.1. Definición de lenguaje de consulta de gráficos

Actualmente existen herramientas de gran utilidad para la gestión de modelos relacional, lo cual ayuda que la implementación sea simple y sencilla de poderse realizar.
Query By Example, por sus siglas en ingles QBE, es la versión gráfica del lenguaje de datos relacional, como es *Structure Query Lenguaje* (SQL).

El funcionamiento básico de QBE permite analizar y convertir las acciones de los usuarios en las declaraciones expresadas en un lenguaje de manipulación de bases de datos. Actualmente QBE es un lenguaje compatible con varias bases de datos relacionales tales como:

- Visual Query By Example
- MSQL
- Microsoft Access

Estos lenguajes están soportados para la facilidad de manipulación de consultas, de tal forma la interfaz de usuario del QBE y SQL interactúan para obtener el modelo relacional gráfico. Estos programas hacen por sí solos el esquema de la base de datos y con clic se pueden seleccionar claves primarias y arrastrar para que se conviertan en claves foráneas. Y mediante la selección de lo que se desea saber a través de ayuda y asistentes o simplemente de manera gráfica, se obtiene la consulta.

QBE también está soportado por otras compañías. Borland ofrece PARADOX como una versión de funcionalidad completa del QBE, el cual incluye la definición de tablas. A pesar de que la sintaxis es ligeramente diferente de la de IBM, el lenguaje tiene mucha similitud. Lotus 123 también ofrece un lenguaje simple de consultas de base de datos, que usa un enfoque parecido al de QBE, debido a que el QBE permite que los usuarios no técnicos formulen fácilmente sus consultas.

3.3.2. Manipulación de datos

La mayoría de los lenguajes estructurados de consultas son textuales, por lo que al formular una solución ésta debe ser creada por una cadena de caracteres. A diferencia de ellos, QBE es un lenguaje gráfico, el cual estructura las soluciones de consultas a través de representaciones gráficas de las tablas de la base de datos en cuestión. La similitud de QBE con SQL es que se basan en el desarrollo de expresiones condicionadas, funciones integradas, agrupación, entre otras, por tal motivo los resultados de los valores de datos

serán los mismos en ambos lenguajes. Debido a que la mayoría de los sistemas con QBE utilizan las bases de datos que han sido desarrolladas utilizando las instrucciones de SQL, se tienen diferentes acciones para el manejo de las bases de datos tales como:

Consultas simples: ésta es una de las instrucciones básicas para el inicio de la manipulación de los datos, las cuales sólo afectan no más de una tabla dentro de la base de datos. Para el desarrollo de esta consulta, el usuario solicita el gráfico de la relación existente. Debido a que la mayor parte de las órdenes definen instrucciones SQL, no se analizan las definiciones de datos en QBE.

QBE: como un lenguaje gráfico, su principal objetivo es ofrecer medios para que el usuario interactúe de forma gráfica con la base de datos, esto se muestra diseñando una estructura de las tablas.

Las *consultas simples* son aquellas que afectan a sólo una tabla dentro de la base de datos; son utilizadas para ilustrar la estructura básica de SQL.

El siguiente ejemplo muestra una consulta simple utilizando la sentencia *select* de SQL. Consulta: ¿quiénes son clientes de convenio?

```
SELECT    nomcliente, tipocliente
FROM        Clientes
WHERE  tipocliente = 'Convenio'
```

Resultado: la siguiente imagen muestra el resultado de la consulta simple realizada.

	nomcliente	tipocliente
1	COMISIÓN FEDERAL DE ELECTRICIDAD	Convenio
2	BANRURAL	Convenio
3	SAGRADO	Convenio
4	ZAMORA	Convenio
5	SERDÁN CLAUDIA	Convenio
6	ABASTOS	Convenio
7	MARTHA ZUNO PÉREZ	Convenio
8	SERDÁN JORGE	Convenio
9	JACONA	Convenio
10	SÚPER	Convenio
11	SAN JOSÉ	Convenio
12	AVENIDA	Convenio
13	CALVARIO	Convenio
14	ALMACÉN	Convenio
15	PUEBLO	Convenio

Resultados de consulta simple

En la consulta ilustrada se puede observar que se utilizaron tres cláusulas básicas usadas en SQL:

- **SELECT**: enlista las columnas que se desean observar en el resultado de la consulta, esto tiene que ser siempre columnas de una tabla relacional.

- **FROM**: enlista una o más tablas que son referenciadas en la consulta.
- **WHERE**: es utilizado como condicionante; de esta forma, es la *condición* para seleccionar las filas de la tabla que se dan en la cláusula *from*.

Otra variante de la consulta anterior se expresa cuando se toman todas las columnas de la tabla; el símbolo "*" en la cláusula *select* significa que muestre la fila completa. Así se tiene lo siguiente.
Consulta: ¿quiénes son los clientes de convenio?

```
Select *

From Cliente

Where tipocliente = 'convenio'
```

Resultado: la siguiente imagen muestra el resultado de la consulta simple mostrando todos los campos de la tabla.

idcliente	fechaalta	tipocliente	rfcalfa	rfcfecha	rfchomoclave	nomcliente	domicilio1	domicilio2	telefono	contacto
2	2006-04-02 10:20:43.013	Convenio				COMISIÓN FEDERAL DE ELECTRICIDAD				
3	2006-04-02 10:20:56.483	Convenio				BANRURAL				
5	2006-04-02 10:21:16.247	Convenio				SAGRADO				
7	2006-04-02 10:21:33.543	Convenio				ZAMORA				
8	2006-04-02 10:21:43.623	Convenio				SERDÁN CLAUDIA				
14	2006-04-02 10:23:56.090	Convenio				ABASTOS				
15	2006-04-02 10:24:08.857	Convenio				MARTHA ZUNO PÉREZ				
16	2006-05-23 15:50:31.373	Convenio				SERDÁN JORGE				
4	2006-04-02 10:21:08.420	Convenio				JACONA				
6	2006-04-02 10:21:26.013	Convenio				SÚPER				
9	2006-04-02 10:21:51.293	Convenio				SAN JOSÉ				
10	2006-04-02 10:21:58.903	Convenio				AVENIDA				
11	2006-04-02 10:22:06.733	Convenio				CALVARIO				
12	2006-04-02 10:22:16.873	Convenio				ALMACÉN				
18	2006-05-28 11:35:11.140	Convenio				PUEBLO				

Resultados de consulta simple mostrando todos los campos de la tabla

A continuación otro ejemplo de consulta:
Consulta: ¿qué productos tienen precio entre $150 y $200?

```
SELECT ean, nomproducto, preproducto, precfarmacia, descuento, fechaoferta, precpublico
FROM Ofertas
WHERE (precpublico>= 150) AND (precpublico<= 200)
ORDER BY nomproducto DESC
```

ean	nomproducto	preproducto	precfarmacia	descuento	fechaoferta	precpublico
7501328975825	WINTOMYLON B...	0MG TAB C30 ...	144.69	9.00	2006-06-02 10:...	179.70
7703332003628	UREADERM LOC...	G ...	157.60	16.00	2006-06-02 10:...	197.00
7501250828350	RIMSALIN AD 60...	G INY 2ML C3 ...	124.03	9.00	2006-06-02 10:...	156.00
7501250828169	RETODOL CPTO ...	SOL AMP C3 ...	144.70	9.00	2006-06-02 10:...	182.00
7501070634810	POSIPEN 500MG...	P C12 ...	137.54	9.00	2006-06-02 10:...	173.00
7501233251205	PENTREXYL 500...	CAP C28 ...	122.45	23.00	2006-06-02 10:...	155.00
7501109760572	MAGSOKON TRI...	ITRO C25 ...	125.25	16.00	2006-06-02 10:...	167.00
7501273500516	LIPODERM C CR...	0G ...	150.10	16.00	2006-06-02 10:...	190.00
7501054508588	LABELLO PAQ C...	AS 2 CRA 100ML...	162.95	25.00	2006-06-02 10:...	191.71
3400004574	GRIMAL 2MG SO...	0ML ...	133.39	20.00	2006-06-02 10:...	171.00
7501273500318	GLICODERM CR...	VITA 60G ...	149.12	16.00	2006-06-02 10:...	175.44
729513104011	GASA LEROY EX...	OB GDE C100 ...	131.35	50.00	2006-06-02 10:...	164.19
7501125101960	FLUCOXAN 150...	1 ...	122.25	9.00	2006-06-02 10:...	150.00
7501125103483	ENTEREX VLLA P...	400G ...	153.20	16.00	2006-06-02 10:...	185.70

Resultado de consulta simple con variación de operadores en la cláusula *where*

Resultado: la imagen anterior muestra el resultado de la consulta, nótese que agrega unas características adicionales a la cláusula *where*, tales como los operadores de comparación y el conector booleano *and*.

Aquí cabe destacar que existen seis operadores de comparación: = (igual), <> (no igual), < (menor que), > (mayor que), <= (menor e igual que), >= (mayor e igual que), éstos pueden ser utilizados para comparar las columnas con otras columnas; los conectores booleanos *and, or, not*, pueden ser utilizados para la creación de condiciones compuestas o para negar una condición. Los paréntesis también pueden usarse en la forma acostumbrada de los lenguajes de programación, donde son utilizados para agrupar condiciones.

También es común utilizar el operador *between* (entre) el cual indica que muestre sólo los datos comprendidos entre los valores indicados, y el operador *orderby,* el cual indica que los registros serán ordenados por nombre del producto y el operador *desc,* que ordena la información de forma descendente o *asc*, que lo hace ascendentemente. Esto se observa en el siguiente ejemplo:

```
SELECT *
FROM        Ofertas
WHERE   precpublicobetween 150 and 200
ORDER BY nomproducto DESC
```

Resultado: la siguiente imagen muestra el resultado de la consulta, con el operador *between*, utilizado para comparar los valores entre sí, el primero menor que el segundo.

ean	nomproducto	preproducto	precfarmacia	descuento	fechaoferta	precpublico
7501328975825	WINTOMYLON B...	0MG TAB C30 ...	144.69	9.00	2006-06-02 10:...	179.70
7703332003628	UREADERM LOC...	G ...	157.60	16.00	2006-06-02 10:...	197.00
7501250828350	RIMSALIN AD 60...	G INY 2ML C3 ...	124.03	9.00	2006-06-02 10:...	156.00
7501250828169	RETODOL CPTO ...	SOL AMP C3 ...	144.70	9.00	2006-06-02 10:...	182.00
7501070634810	POSIPEN 500MG...	P C12 ...	137.54	9.00	2006-06-02 10:...	173.00
7501233251205	PENTREXYL 500...	CAP C28 ...	122.45	23.00	2006-06-02 10:...	155.00
7501109760572	MAGSOKON TRI...	ITRO C25 ...	125.25	16.00	2006-06-02 10:...	167.00
7501273500516	LIPODERM C CR...	0G ...	150.10	16.00	2006-06-02 10:...	190.00
7501054508588	LABELLO PAQ C...	AS 2 CRA 100ML...	162.95	25.00	2006-06-02 10:...	191.71
3400004574	GRIMAL 2MG SO...	0ML ...	133.39	20.00	2006-06-02 10:...	171.00
7501273500318	GLICODERM CR...	VITA 60G ...	149.12	16.00	2006-06-02 10:...	175.44
729513104011	GASA LEROY EX...	OB GDE C100 ...	131.35	50.00	2006-06-02 10:...	164.19
7501125101960	FLUCOXAN 150...	1 ...	122.25	9.00	2006-06-02 10:...	150.00
7501125103483	ENTEREX VLLA P...	400G ...	153.20	16.00	2006-06-02 10:...	185.70

Resultado de consulta simple con variación de la cláusula *where*, utilizando el operador *between*

Ahora observa lo siguiente:

Consulta: indicar los productos con el código de presentación 01 y 02

```
SELECT ean, nomproducto, preproducto, precfarmacia, descuento, fechaoferta, precpublico, codpresentacion
FROM Ofertas
WHERE (codpresentacion IN ('01', '02'))
ORDER BY nomproducto DESC
```

ean	nomproducto	preproducto	precfarmacia	descuento	fechaoferta	precpublico	codpresentacion
7501328975405	WINTOMYLON P...	USP 150ML ...	102.82	9.00	2006-06-02 10:01:22.420	127.70	02
7501328975825	WINTOMYLON B...	0MG TAB C30 ...	144.69	9.00	2006-06-02 10:01:22.420	179.70	01
7501293200540	VITALAIF CAP C...	...	189.80	9.00	2006-06-02 10:01:22.420	241.78	01
7703332003628	UREADERM LOC...	G ...	157.60	16.00	2006-06-02 10:01:22.420	197.00	01
7705849004891	UMBRELLA PFS ...	60G ...	166.40	16.00	2006-06-02 10:01:22.420	208.00	02
7705849004907	UMBRELLA PFS ...	60G ...	166.40	16.00	2006-06-02 10:01:22.420	208.00	01
51131675988	ULTRA THON RE...	INSEC 170G ...	45.33	9.00	2006-06-02 10:01:22.420	56.66	01
7501293200564	TRIYOTEX 75MG...	P C30 ...	74.18	9.00	2006-06-02 10:01:22.420	94.50	01
729513146721	TOBILLERA ELA...	MIRREK CH ...	34.64	33.00	2006-06-02 10:01:22.420	43.30	01
729513146738	TOBILLERA ELA...	MIRREK MED ...	34.64	33.00	2006-06-02 10:01:22.420	43.30	02
7502012981108	TIRALECHE CRI...	L ROGER ...	35.95	16.00	2006-06-02 10:01:22.420	44.95	01
7501072300195	TING TENIS TCO...	ES 75G ...	28.99	9.00	2006-06-02 10:01:22.420	36.00	01
7501072300201	TING TENIS TCO...	ES 160G ...	40.26	9.00	2006-06-02 10:01:22.420	50.00	02
7501072300140	TING CRA 48G ...	...	54.75	9.00	2006-06-02 10:01:22.420	68.00	02
7501072300133	TING CRA 28G ...	...	38.64	9.00	2006-06-02 10:01:22.420	48.00	01
7501094911249	TESACOF INF S...	80MG ...	70.76	6.00	2006-06-02 10:01:22.420	89.00	01
7501094911232	TESACOF AD SO...	60MG ...	73.14	6.00	2006-06-02 10:01:22.420	92.00	02
7501095452130	TEMPRA INF JBE...	20ML ...	47.83	9.00	2006-06-02 10:01:22.420	60.55	02
7501254500900	SULFATIAZOL A...	IC PDA 20G ...	5.90	16.00	2006-06-02 10:01:22.420	7.40	01
7501228300390	STRESSTABS 60...	IERRO GRAG C3...	73.45	9.00	2006-06-02 10:01:22.420	92.39	02
7501228300383	STRESSTABS 60...	RAG C30 ...	73.45	9.00	2006-06-02 10:01:22.420	92.39	01
7501125109928	SOLDRIN OTI G...	10ML ...	46.62	16.00	2006-06-02 10:01:22.403	57.20	02

Resultado de consulta simple con variación de la cláusula *where* utilizando el operador *in*

En la imagen anterior se muestra la consulta utilizando el operador de comparación *in*. La cláusula *where* ha evaluado si el tipo de presentación de la fila se encuentra.

Las consultas de múltiples tablas son aquellas que implican la conexión de los datos a través de varias tablas. Para este tipo de consulta es común utilizar el operador *innerjoin*, a continuación se muestran algunos ejemplos utilizando este operador.

Consulta: ¿cuáles son los laboratorios de los productos asignados al laboratorio *WYETH, S.A. DE C.V.?*

Columna	Alias	Tabla	Salida	Tipo de orden	Criterio de ordenación	Filtro
idproducto		Productos	☑			
ean		Productos	☑			
IdLaboratorio		Productos	☑			
NomLaboratorio		Laboratorios	☑			= 'WYETH, S.A. DE C.V.'
Estandar		Laboratorios	☑			

Panel de criterios de la consulta realizada en SQL:

```
SELECT     Productos.*, Laboratorios.*
FROM       Productos INNER JOIN
             Laboratorios ON Productos.IdLaboratorio = Laboratorios.IdLaboratorio
WHERE     (Laboratorios.NomLaboratorio = 'WYETH, S.A. DE C.V.')
```

Existen varias sentencias en SQL, pero dentro de las básicas y simples también se tiene la sentencia *group by.* Esta clave de SQL es utilizada cuando se desea seleccionar múltiples columnas desde una o varias tablas. En esta sentencia se pueden utilizar operadores aritmeticos en la instrucción *select*. La sintaxis en SQL es tal como se muestra en el siguiente ejemplo:

```
SELECT "nombre1_columna", SUM("nombre2_columna")
FROM "nombre_tabla"
GROUP BY "nombre1-columna"
```

Operaciones que modifican la base de datos. Tanto QBE como SQL soportan tres tipos de operaciones que cambian la base de datos:

- ***Insert***: sirve para insertar registros o filas en la tabla de una base de datos. Un ejemplo común utilizando esta instrucción es el siguiente:

  ```
  insertintoClientes (cliente_id,nombre,paterno,materno,sucursal_id, ciudad)
  values ('0000099999','MARINA','HERNANDEZ','HERNANDEZ','00004',’Leon’),
  ('0000099998','CARLOS','VILLAVICENCIO','VAZQUEZ','00002',’León’)
  ```

('0000099997','MIGUEL','SALAZAR','GARCIA','00001',’Morelia’)

- ***Update***: esta instrucción actualiza los registros almacenados en la tabla de la base de datos; para esta instrucción se usa el siguiente ejemplo:

  Update clientes
  Set sucursal_id = ‘00001’
  Where ciudad = ‘Leon’
  En este ejemplo existen dos filas que satisfacen la condición de la cláusula *where* donde se actualizaría el valor de la sucursal a 00001.

- ***Delete*.** Esta instrucción comúnmente elimina los registros de una tabla dentro de la base de datos, se tiene como ejemplo la siguiente sentencia en SQL:

Delete from clientes
Where sucursal_id = ‘00001’

En este ejemplo se puede observar que se utiliza la instrucción *delete* haciendo referencia a la tabla de clientes, y se establece una condición, por lo que los registros a eliminar deberán de satisfacer esa condición.

Como has podido observar, crear una consulta para obtener exclusivamente lo que se quiere de una tabla es muy sencillo, y te puede liberar de horas de trabajo manual.

3.3.3. Definición y entrada de datos

Una vez que se ha conocido la forma de crear bases de datos, insertar datos y realizar consultas de dichos datos en formas diferentes, hay que conocer cómo se definen e ingresan éstas instrucciones en un determinado manejador de base de datos, en éste caso será SQL.

En SQL se usan términos como tabla, fila y columna para los términos relación, tupla y atributo del modelo relacional formal. Existe una gran variedad de comandos en SQL, tal como *create, que* es utilizado para hacer esquemas, tablas y dominios.

Un esquema de SQL se identifica con un nombre, el cual incluye un identificador de autorización para usuarios. Los elementos del esquema son las tablas, las restricciones, las vistas, los dominios y algunas otras estructuras.

Algunos de los comandos utilizados en SQL son:

- ***Create***: se usa para crear el objeto; puede ser una base de datos, una tabla, un esquema, vistas, etc. A continuación se muestra un ejemplo de creación de una base de datos.

  ```
  CREATE TABLE [dbo].[Bitacora](
        [IdBita] [int] IDENTITY(1,1) NOT NULL,
        [idbitacora] [int] NULL,
        [fechahora] [datetime] NOT NULL,
        [tipo] [char](3) NOT NULL,
        [mensaje] [char](250) NOT NULL,
        [status] [bit] NOT NULL,
  ```

- ***Inserinto****:* se utiliza para que una vez creada la base de datos, las tablas y los campos de cada uno, se inserten datos a cada campo de la tabla referenciada.

  INSERTINTO Clientes (id_cliente, nombre_cliente, tel_cliente) values ('08976', 'MariaMartinez', '78647907')

- ***Select***: selecciona un objeto dentro de la base de datos, al igual que tablas. Como un ejemplo comúnmente utilizado en select se tiene:

  ```
  SELECT idcliente, fechaalta, tipocliente, rfcalfa, rfcfecha, rfchomoclave,
  nomcliente, domicilio1, domicilio2, telefono, contacto, activo, comentario, imagen
  FROM        Clientes
  ```

SQL tiene varias características adicionales para el *desarrollo* e *implementación* de bases de datos; se enlistan a continuación:

- Técnicas de distintos lenguajes de programación que pueden incluir sentencias de SQL, las cuales son de utilidad para poder entrar a una o más bases de datos, como el SQL incrustado o dinámico o la conectividad de bases de datos abiertas ODBC (Open Data Base Connectivity).
- Todos los sistemas administradores de bases de datos relacional (RDMBS: Relational Data Base Management System), contienen los comandos de SQL, con un conjunto de comandos para especificar los parámetros de diseño físico de la base de datos.
- Controlador de transacciones: utilizado para especificar las unidades de procesamiento de bases de datos.
- Cuenta con estructuras de lenguaje comúnmente utilizadas para especificar la concesión y revocación de privilegios a los usuarios.
- Técnicas de actualización de base de datos que especifican acciones que son automáticamente ejecutadas por los eventos.
- Incorpora características de los modelos orientados a objetos, utilizados para mejorar los sistemas relacionales conocidos como de objetos relacionales.

En SQL se efectúan todas las instrucciones vistas en este tema, las cuales se pueden considerar las básicas para la creación, inserción y manipulación de datos. Es importante que recuerdes y pongas en práctica las instrucciones vistas cuando utilices SQL para la realización de tus bases de datos específicas.

3.4. Sistemas de bases de datos cliente/servidor

A continuación conocerás las diferencias que hay entre un cliente y un servidor, debido a que es de suma importancia identificarlos y diferenciarlos, además de saber en qué se enfocan respectivamente.

Un cliente es algo o alguien que solicita información o trabajo, y un servidor es quien la proporciona. En el ámbito de las redes, cliente/servidor es una relación que se establece entre una máquina o proceso que solicita información y una máquina o proceso que la proporciona. Esto se revisará más a detalle en el siguiente tema.

Un sistema de bases de datos basado en aplicaciones cliente/servidor permite extender las aplicaciones principales a una plataforma de equipos llamados *servidores*, cuya función es enlazar las computadoras-cliente o de sobremesa al servidor, donde el sistema de gestión de bases de datos funciona a través de la red y son llamadas al sistema.

Una de las principales ventaja que ofrece este tipo de esquema de sistemas cliente/servidor es compartir recursos y accesos al sistema de base de datos, mediante una *interfaz* gráfica de usuario, para darle más poder y facilidad a la presentación de los datos, consultas, etc. Las bases de datos pueden ser ubicadas en servidores y la información puede ser consultada de manera remota o local.

Los dos términos vistos en este apartado son el punto de partida para conocer el concepto y enfoque de cliente/servidor.

3.4.1. Conceptos de cliente/servidor

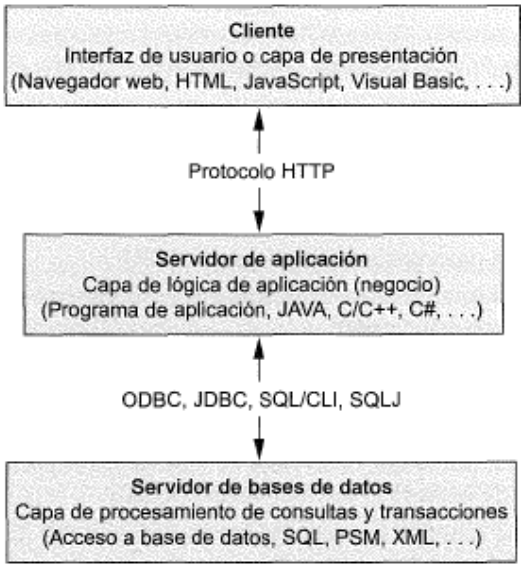
En este tema estudiarás la plataforma que sirve como mediador entre la base de datos y sus datos, conectada por medio de una red de computadoras.
El concepto cliente/servidor se conoce como aquella plataforma donde el medio de comunicación se realiza por la red de computadoras, donde se ofrecen los servicios por medio de un equipo dedicado (servidor). Los equipos o terminales de cómputo conectados a la red reciben el nombre de cliente.
La principal ventaja de la plataforma cliente/servidor es la optimización de los servicios para establecer fluidez en las transacciones de entrada y salida de datos, por medio de

interfaces gráficas de usuarios que otorgan medios orientados al usuario para facilitar el acceso a la base de datos.

Un Sistema Gestor de Bases de Datos (SGBD) también brinda interfaces de usuario que permiten visualizar y responder a las entradas de los usuarios. La arquitectura cliente/servidor se compone de tres niveles o capas:

- **Capa de presentación (cliente)**: esta capa otorga al usuario la interfaz para poder interactuar con el sistema de gestión de base de datos, las cuales se utilizan como conexiones con la aplicación desarrollada. Comúnmente se utilizan navegadores web o aplicaciones desarrolladas en distintos lenguajes de programación, tales como HTML, JAVA, C++, Visual Studio, etc. En esta capa se gestionan las entradas, salidas y navegación, aceptando comandos de usuario.
- **Capa de aplicación (lógica de aplicación)**: esta capa se utiliza para la programación lógica de las aplicaciones que tendrán acceso al sistema de gestión de bases de datos, El acceso a la información o consultas a la base de datos se realiza mediante la interface de la aplicación desarrollada.
- **Servidor de bases de datos**: en esta capa se controlan todas las consultas y peticiones de actualizaciones procedentes de la capa de aplicación y es la encargada de procesar las solicitudes para el envío de los resultados.

La siguiente imagen muestra un esquema general de las diferentes capas que componen una arquitectura cliente/servidor:



Arquitectura cliente/servidor de tres capas
Tomada de Ramez y Shamkant (2007).

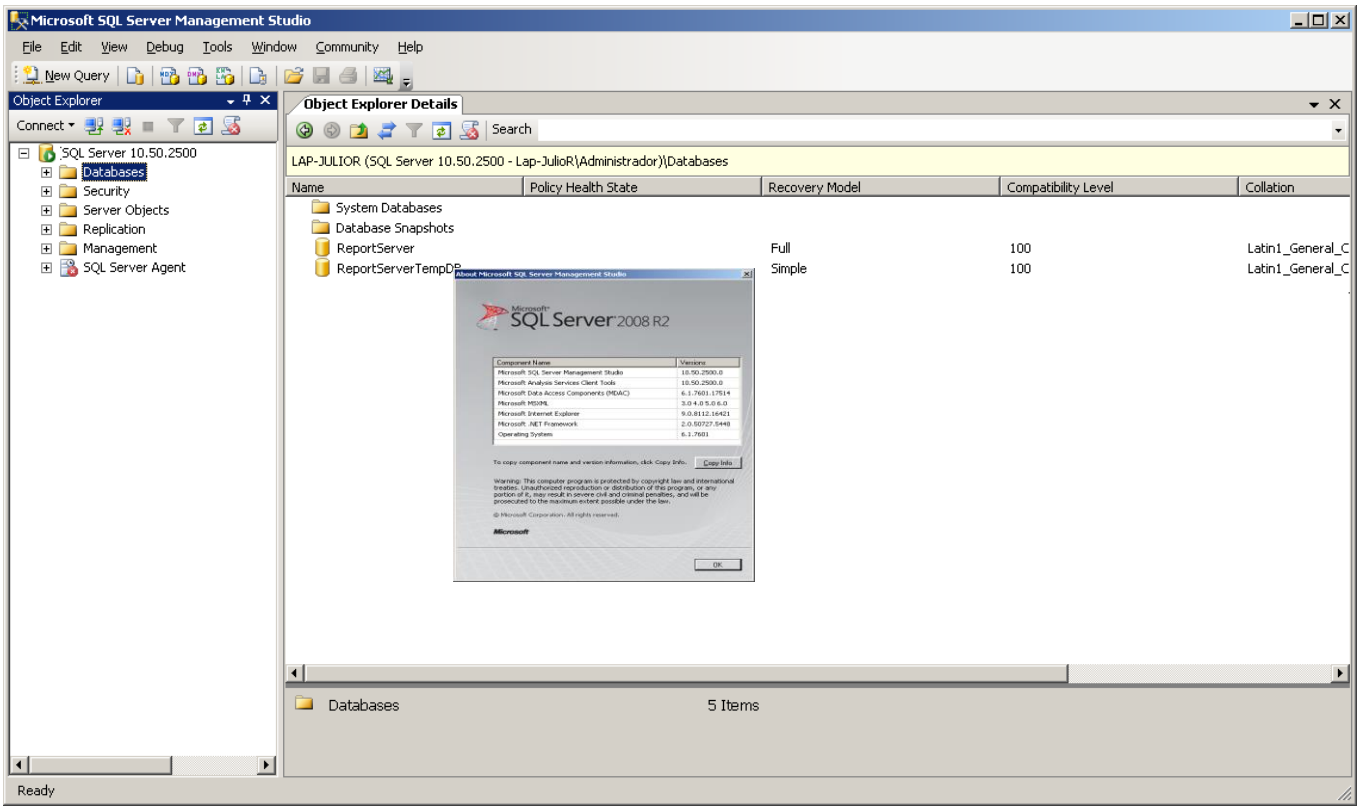
La arquitectura cliente/servidor es de suma importancia conocerla y entenderla para cuando pongas en práctica la realización y manipulación de base de datos, sea más fácil entender esta plataforma donde se procesan las instrucciones que indica el usuario.

3.4.2. Base de datos servidor

Comúnmente para la operatividad de un Database Management System (DBMS), o "Sistema Manejador de Bases de Datos" en español, se incluye la centralización a nivel de servidor de base de datos. El servidor de aplicaciones es el que debe formular las consultas SQL y realizar la conexión con el servidor de bases de datos con lo que se requiera.

SQL es un lenguaje de consultas relacional y estructurado, el cual también hace referencia a un diccionario de datos que incluye información acerca de la distribución de los mismos entre los distintos servidores SQL.

La siguiente imagen muestra un servidor de base de datos en SQL



Servidor de base de datos en SQL Server 2008 R2
Tomada de Microsoft SQL Server Management Studio (2008).

Acabas de conocer qué es un servidor de base de datos o manejador de sistemas de base de datos, el cual es una aplicación que puede variar según la que selecciones, pero en esencia, funciona de forma similar, sólo debes de familiarizarte con la plataforma elegida.

3.4.3. Manipulación y programación del servidor de datos

Como parte de la manipulación del servidor de base de datos, existen estructuras de control que permiten ser ejecutadas sin interrupciones dentro de un conjunto de lotes de instrucciones. Esas estructuras de control son parte de los lenguajes de flujo de control e incluyen instrucciones condicionales e instrucciones para hacer *bucles*.

Además de estas características, también se brindan procedimientos almacenados que son programas escritos, los cuales son compilados en algún lenguaje de programación y se guardan para ser ejecutados con eficiencia.

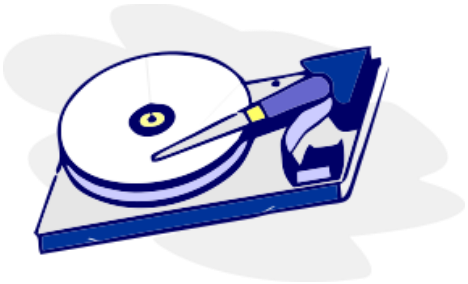
Existen diferentes recursos lingüísticos que son utilizados en la sintaxis de SQL:

- **Lenguaje de flujo de control**: es un lenguaje de consulta orientado a usuarios de conocimiento avanzado, para obtener información inmediata por la vía de consultas en bases de datos.
- **Procedimientos almacenados**: los procedimientos almacenados son programas SQL compilados en su primera ejecución almacenada y lista para su uso posterior. Tienen la ventaja de ejecutarse de forma rápida y permiten recibir y retornar parámetros variables usados para pasar datos almacenados.
- **Disparadores**: dentro de SQL a un disparador se le nombra *tigger,* el cual es un programa que se ejecuta automáticamente cuando se intenta hacer una actualización determinada sobre una tabla específica. Un *tigger* puede definir tres tipos de disparadores: inserción, actualización y eliminación.

Hay diferentes formas de entrar en la información, y no sólo en persona, sino también por procesos propios del sistema manejador de bases de datos.

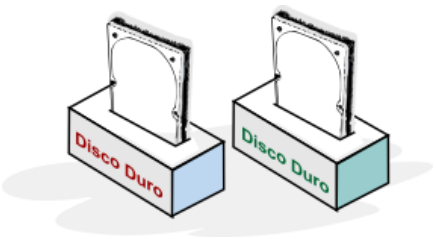
3.5. Organización física de los sistemas de bases de datos.

La organización física de una base de datos dependerá de su tamaño y requerimientos de espacio, ya que el almacenamiento se da en recursos secundarios de gran capacidad de almacenamiento. Uno de los principales medios de almacenamiento secundario son los discos duros.



Los dispositivos de almacenamiento registran o almacenan la información de forma directa y permiten el acceso a sus registros en cualquier tipo de orden, ya sea lectura o escritura.

Los discos son dispositivos utilizados para el almacenamiento de la información, los cuales se presentan en unidades separadas. La estructura de la pila de discos multiplica la capacidad de almacenamiento, lo cual ayuda a mantener el rendimiento, ya que cada cara de un disco posee una cabeza de lectura y escritura propia.



Organización física de una base de datos

La imagen muestra la estructura interna de un disco duro donde físicamente se encuentra toda la información de la máquina que puede ser la misma base de datos.

3.5.1. Organización física de los sistemas de bases de datos

El especialista de desarrollo de sistemas es quien se interesa en la organización física para un sistema de base de datos. Sin embargo, el rendimiento general de un sistema de base de datos se determina en gran medida por las estructuras de datos físicas utilizadas. A pesar de que los usuarios finales no conocen los detalles del diseño físico de la base de datos, éstos afectan el rendimiento de la misma, lo cual es de suma importancia, pues resulta ser el principal factor de satisfacción para el usuario de un sistema de base de datos.

Tanto el formato como la información no son afectados directamente por la organización física de la base de datos; sin embargo, el tiempo de respuesta sí se ve afectado (tiempo de respuesta es el tiempo que transcurre entre la inicialización de una operación sobre la base de datos y la disponibilidad de los resultados).

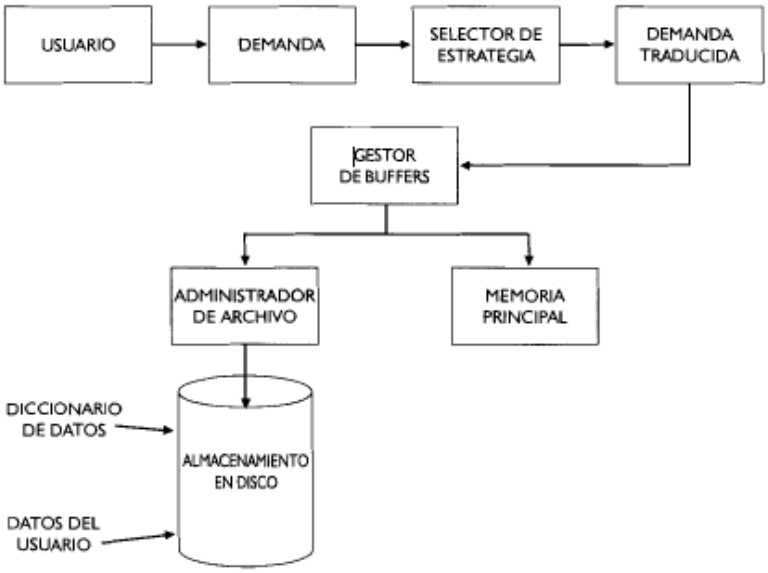
De esta forma, el poco conocimiento en el manejo de ciertas aplicaciones procesadoras de información, afectan en muchas de las ocasiones el rendimiento del programa para procesar datos, debido a que por el mal manejo se crean conflictos que aumentan el

tiempo de repuesta que puede reflejarse incluso, en la pérdida de la información solicitada.

3.5.2. Acceso físico a la base de datos

Durante el proceso de acceso físico a la base de datos el usuario interactúa con el sistema de base de datos al momento de iniciar una consulta. El modo selector de estrategia es el que se encarga de interpretar la orden del usuario en una forma más eficiente para la ejecución.

La orden que es traducida activa al administrador de *buffer,* que controla el movimiento de datos entre la memoria principal y el almacenamiento en disco. El administrador de archivos brinda el soporte al administrador de *buffer,* controlando la reserva. Los datos de los usuarios son almacenados en una base de datos física de registros de eventos. La siguiente imagen muestra un ejemplo de formas para el sistema de acceso físico a la base de datos.



Formas de almacenamiento físico a la base de datos
Tomada de Hansen y Hansen (1997).

Con esta información te podrás dar cuenta cuál es el recorrido de la información hasta llegar al almacenamiento físico de los datos; hay que destacar que es el mismo recorrido que se efectúa cuando se hace una consulta a la base de datos.

3.5.3. Formas de almacenamiento secundario

Existen ciertas técnicas o formas de almacenar la información de una base de datos, las cuales son ajenas al disco duro de una computadora. Es necesario conocerlas para realizar los diferentes respaldos y tener adecuadas formas de poder restaurar la información en caso de pérdida de datos en una BD.

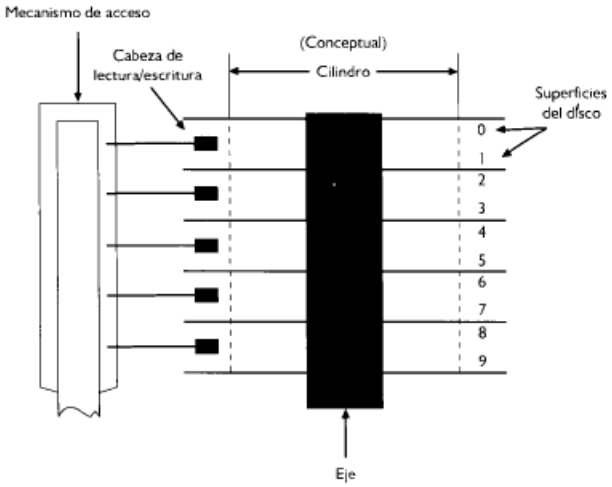
El almacenamiento secundario, dentro de un sistema de base de datos se compone comúnmente por:

- Almacenamiento en disco: almacenamiento de forma principal
- Almacenamientos en cinta: almacenamiento en forma secuencial y limitada

Por lo general la base de datos se almacena en disco de forma completa, y algunas porciones son transferidas desde el disco a la memoria primaria, a medida que se utilizan. El almacenamiento en disco es de forma principal con acceso directo, por tanto los registros individuales pueden ser accesibles de forma directa.

El *almacenamiento en cinta magnética* es menos costoso que el almacenamiento en disco; sin embargo, se puede tener acceso a los registros de forma secuencial y más lenta, además de que su función en el sistema de base de datos está limitada para archivar datos. El uso de este dispositivo puede mencionarse solo como un antecedente de los tipos de dispositivos actuales, aunque se están realizando estudios para reutilizarse nuevamente por la gran cantidad de información que permite almacenar

Una unidad física donde se encuentra integrado el medio de grabación del disco se llama *controlador de disco*, el cual contiene un paquete de disco. En la siguiente imagen se pueden observar los componentes básicos de dicho paquete y el mecanismo de lectura y escritura que es la transmisión de los datos.



Almacenamiento secundario
Tomada de Hansen y Hansen (1997).

Los datos pueden ser respaldados y colocados en diferentes medios para simplemente recuperar ese respaldo y poder entrar en él, esto puede servir ante situaciones como los casos en los que se pierde información que se encuentra en la unidad principal, pero tendría como desventaja el proceso que implica la restauración de la base de datos para ser consultada; este almacenamiento secundario es recomendado sólo para pocos y esporádicos accesos.

3.5.4. Factores de rendimiento del disco

Ahora conocerás los factores que determinan el buen o mal funcionamiento en cuanto al tiempo se refiere, para dar respuestas a instrucciones requeridas por los usuarios de los sistemas manejadores de bases de datos. A continuación se mencionarán y definirán cada una de ellos.

Existen cuatro factores que pueden afectar el rendimiento o la velocidad del disco con relación a los datos que son transferidos al medio de almacenamiento, los cuales son:

- **Tiempo de posicionamiento**: este factor también es conocido como tiempo de búsqueda; es considerado como el tiempo necesario para mover los cabezales de lectura/escritura, iniciando en una posición determinada y posteriormente a una nueva dirección.
- **Tiempo de activación de la cabeza**: éste es el tiempo utilizado para activar de forma electrónica el cabezal que está ubicado sobre la superficie cuando se realiza una transferencia de datos.
- **Retraso de rotación**: la latencia es otro de los factores de tiempo en un medio de almacenamiento, éste representa la cantidad de tiempo que es necesario para el

bloqueo seleccionado para rotar la cabeza, de tal forma que la transferencia de datos pueda comenzar.

- **Velocidad de transferencia de datos**: es la cantidad del tiempo necesario para transferir los datos desde el disco hasta la memoria principal, esto dependerá de la velocidad de rotación y de la cantidad de datos que son almacenados.

Es importante conocer cada uno de los factores mencionados en este tema, ya que el pleno conocimientos de ellos ayudará a contar con información rápida y adecuada al momento de ser solicitada o guardada, debido a que si algo llega a suceder, se tomará en cuenta que algunos de estos factores se encuentran actuando o interfiriendo en la toma de decisiones para obtener o guardar información.

3.5.5. Formatos de almacenamiento de datos en disco

Existen diferentes medios físicos para la manipulación de datos sobre los discos:

- **Formatos de pista**: los registros se almacenan en formatos de cuenta clave, éste incluye una que es externa al registro de datos; también existe el formato de cuenta dato, el cual incluye una clave interna al registro de datos.
- **Formatos de registros**: los formatos de registros son almacenados sobre las pistas en cualquiera de los formatos fijo y variable. Los registros de longitud fija son de la misma longitud y los registros de longitud variable, permiten que la longitud de cada registro sea variado.
- **Gestión de entrada/salida**: están basados en los formatos de datos de las operaciones de entrada y salida de un *SGDB*-

El formato en cuestión dependerá del punto en el que se encuentre la base de datos o justo lo que se encuentre realizando el gestor con los mismos.

3.5.6. Organización de archivos y métodos de direccionamiento

Hasta hoy se conocen algunos métodos para la organización de archivos sobre los dispositivos de almacenamiento, los cuales se mencionan a continuación:

- **Organización secuencial**: la organización secuencial de un archivo se basa en que los registros son almacenados de forma adyacente unos con otros, con respecto a la clave de almacenamiento.
- **Organización secuencial indexada de un archivo**: este tipo de organización consiste en que los archivos se encuentran organizados de forma secuencial, mas es posible acceder directamente a los registros; tiene la ventaja de poder facilitar el acceso a los registros de ambas formas, ya sea secuencial o directamente.

- **Organización directa de un archivo**: los registros son almacenados de forma directa de manera simple, conocida como función *hash* la cual pueden ser estática y dinámica.

Los métodos a aplicar dependerán del sistema gestor de bases de datos.

3.5.7. Correspondencia entre estructuras de datos lógicas y estructuras de datos físicas

La aplicación de un sistema de base de datos se basa en las estructuras de datos lógicas y físicas, a continuación se muestran algunas de estas correspondencias entre ambas estructuras.

- La correspondencia en la base de datos relacional: se almacena cada relación como un archivo con un registro para cada fila, generalmente esto es correcto para bases de datos de tamaño muy pequeño; tras el aumento de tamaño de la base de datos es conveniente que sea almacenado en varias relaciones en un solo archivo.
- Correspondencia en la base de datos en red: dentro de una interrelación compleja de red “muchos a muchos”, es conveniente modificar una red simple mediante registros de intersección.
- Correspondencia en la base de datos jerárquicos: al igual que la correspondencia en red, en ésta también se pueden utilizar múltiples punteros en los registros padres en las relaciones de una jerarquía o árbol.

Acabas de conocer cómo se deben de estructurar las bases de datos según vayan cambiando o se vayan modificando; la correspondencia de datos no es igual, ya que en un principio una base de datos puede ser pequeña y ordenada, pero conforme va pasando el tiempo, puede aumentar, causando desorden en los datos internos.

Cierre de la Unidad

Has concluido la unidad 3 del curso. A lo largo de ésta se trataron conceptos básicos sobre álgebra relacional, cálculo relacional, SQL y dispositivos de almacenamiento, además de que aprendiste a realizar bases de datos utilizando comandos y realizando consultas de los datos almacenados en ellas, dependiendo de las necesidades que pudiera tener el usuario; asimismo, cuentas con el conocimiento sobre los diferentes dispositivos de almacenamiento para el guardado de las bases de datos.

Todo lo que aprendiste es de suma importancia, debido a que generarás de forma correcta bases de datos útiles y confiables, además de que tendrás la satisfacción de que

la información que se muestre al realizar una consulta es fiable y adecuada a lo que el cliente te pida.

Es aconsejable que revises nuevamente la unidad en caso de que los temas que se acaban de mencionar no te sean familiares o no los recuerdes.

Para saber más

- *Manual de instalación SQL Server 2000*: http://sabd15n2.wikispaces.com/file/view/manual+de+instalacion+de+sql+server.pdf
- *Manual de instalación SQL Server 2008*: http://es.scribd.com/doc/33502653/Manual-de-Instalacion-SQL-Server-2008

Fuentes de consulta

Date, C .J. (2001). *Introducción a los sistemas de bases de datos* (2ª ed.). México: Pearson Educación.

Hansen, G. W. y Hansen, J. V. (1997). *Diseño y administración de bases de datos* (2ª ed.). Madrid: Prentice Hall.

Ramez, E. y Shamkant, B. (2007). *Sistemas de bases de datos* (2ª ed.). Madrid: Navathe.

Silberschatz, A. (2002). *Fundamentos de bases de datos* (4ª ed.). Madrid: Mc Graw-Hill.