UNIDAD 4 Consultas simples de SQL y Sentencias de Manipulación

Los sistemas de base de datos relacionales más comúnmente usados consultan y modifican su base de datos a través de un lenguaje llamado SQL (Structured Query Language – Lenguaje de Consulta Estructurado). Hay muchos diferentes dialectos de SQL. Los dos mayores estándares son ANSI (American National Standards Institute) y SQL2 o SQL-92. Hay un estándar emergiendo, el SQL3, que extiende SQL2 con nuevas características como triggers (disparadores), objetos, etc.
Existen versiones del SQL producidos por los principales vendedores de sistema de administración de base de datos.

4.1 Consultas simples

La forma más simple de consulta en SQL pregunta por aquellas tuplas de alguna tabla (utilizaremos indistintamente la palabra tabla o relación) que satisfacen una condición. Esta consulta simple como muchas de las consultas SQL, usa tres palabras reservadas: SELECT, FROM, y WHERE.

4.1.1 La cláusula SELECT-FROM-WHERE

Ejemplo:
La base de datos BIBLIOTECA, consiste de:

LIBROS (ISBN, Nombre, Genero, Estado, Sig_Top, Nro_I,
Editorial, Año_de_Ed, Cant_pag)
LIBROS_DONADOS (ISBN, Fecha_de_donación)
LIBROS_COMPRADOS (ISBN, Fecha_de_compra, Precio)
AUTORES (Nombre_Autor, Apellido_Autor, Fecha_de_Nacimiento)
EDITORIAL (Nombre, Dirección, Teléfono, Fax)
SOCIOS (DNI, Nombre, Apellido, Nro_Socio, Profesión, Nombre_Calle,
Nro_Calle)
TELEFONOS_SOCIOS (Nro_Socio, teléfono)
ESCRITO_POR (Libro_Nombre, Nombre_Autor, Apellido_Autor)
PRESTAMO (Nro_Socio, ISBN, fecha_de_prestamo)
DEVOLUCIÓN (Nro_Socio, ISBN, fecha_de_devolución, estado)

Para preguntar acerca de los libros cuya editorial sea Atlántida y el año de Edición sea 2001, en SQL, lo expresamos de la siguiente forma:

```
SELECT  *
FROM Libros
WHERE Editorial = “Planeta” AND Año_de_Edición = 2001;
```

La cláusula FROM especifica la relación o las relaciones a la(s) cual(es) se refiere la consulta. En nuestro ejemplo consultamos la tabla o relación LIBROS.

La cláusula WHERE especifica la condición que deben cumplir las tuplas para ser parte del resultado de nuestra consulta. En el ejemplo la condición es que los libros resultantes deben tener en el campo editorial el valor “Atlántida” y además, su año de edición debe ser 2001.

La cláusula SELECT indica qué atributos de las tuplas que igualan la condición serán producidas como parte del resultado de la consulta. En el ejemplo * indica que la tupla entera es producida como resultado.

NOTA: Siempre escribiremos las palabras reservadas de SQL en mayúscula, por ejemplo: SELECT, FROM, WHERE, LIKE, etc.

Si la relación LIBROS
LIBROS (ISBN, Nombre, Genero, Estado, Sig_Top, Nro_I, Editorial, Año_de_Ed, cant_pag)
contiene los siguientes valores:

ISBN	Nombre	Genero	Estado	Sig_Top	Nro_I	Editorial	Año_de_Ed	Cant_pag
0-13-861337-0	A First Course in Database System	Técnico	Bueno	A-12-456	827	Prentice Hall	1997	470
950-731-260-9	La Resistencia	Novela	Regular	C-45-896	956	Planeta	2008	148
950-9122-55-6	La Maquina Cultural	Novela	Bueno	C-78-965	124	Planeta	1998	292
0-415-11966-9	Wholeness and the Implicate Order	Novela	Bueno	D-45-655	456	Routledge	1980	224
950-13-6183-7	Apuntes y Aportes para la Gestión Curricular	Ciencias	Bueno	D-7-65	345	Kapelusz	1999	171

entonces el resultado de

```
SELECT  *
FROM Libros
WHERE Editorial = “Planeta” AND Año_de_Ed > 1997;
```

es:

ISBN	Nombre	Genero	Estado	Sig_Top	Nro_I	Editorial	Año_de_Ed	Cant_pag
950-731-260-9	La Resistencia	Novela	Regular	C-45-896	956	Planeta	2008	148
950-9122-55-6	La Maquina Cultural	Novela	Bueno	C-78-965	124	Planeta	1998	292

Note que el resultado es una relación en sí.

Ejemplo: Si la consulta es:

SELECT ISBN, Nombre, Sig_Top
FROM Libros
WHERE Editorial = “Planeta” AND Año_de_Ed > 1997;

el resultado sería:

ISBN	Nombre	Sig_Top
950-731-260-9	La Resistencia	C-45-896
950-9122-55-6	La Maquina Cultural	C-78-965

A veces, deseamos producir una relación con cabeceras de columnas diferentes a los atributos de la relación mencionados en la cláusula FROM. Podemos escribir luego del nombre del atributo en la cláusula SELECT por la palabra reservada AS y un alias, el cual representará el nombre en la relación resultante. AS es opcional.

Ejemplo:

SELECT ISBN AS isbn, Nombre AS nombre, Sig_Top as signatura
FROM Libros
WHERE Editorial = “Planeta” AND Año_de_Ed > 1997;

El resultado sería:

Isbn	Nombre	Signatura
950-731-260-9	La Resistencia	C-45-896
950-9122-55-6	La Maquina Cultural	C-78-965

Ejemplo:

También podemos permitir constantes como un item de la cláusula SELECT, lo cual puede resultar en una consulta más legible.

SELECT Nombre AS nombre_libro, cant_pag AS cantidad_de, ‘paginas’ AS paginas
FROM Libros
WHERE Editorial = “Planeta” AND Año_de_Ed > 1997;

Nombre_Libro	Cantidad_de	Paginas
La Resistencia	148	Páginas
La Maquina Cultural	292	Páginas

4. 1.2 Más sobre la cláusula WHERE

Es posible incluir expresiones en la cláusula WHERE comparando valores usando los seis operadores de comparación: = (igual), <> (distinto), < (menor), > (mayor), <= (menor igual) , y >= (mayor igual).

Los valores que pueden ser comparados incluyen constantes y atributos de la relación (o relaciones) mencionadas luego de la cláusula FROM. También es posible aplicar los operadores aritméticos +, *, /, -, y otros a los valores numéricos antes de compararlos.

Ejemplo:
Si queremos listar aquellos libros que fueron editados en los últimos 3 años, podemos consultar la relación libro de la siguiente forma.

```
SELECT  Nombre, ISBN, Año_de_Ed, Cant_pag
FROM Libros
WHERE (2008- Año_de_Ed) <= 3
```

El resultado es:

Nombre	ISBN	Año_de_Ed	Cant_pag
La Resistencia	950-731-260-9	2000	148

El resultado de cada comparación es un valor booleano: TRUE ó FALSE. Los valores boléanos pueden ser combinados por los operadores lógicos AND, OR y NOT. Por ejemplo en la primera consulta que realizamos:

```
SELECT  ISBN, Nombre, Sig_Top
FROM Libros
WHERE Editorial = "Planeta" AND Año_de_Edición > 1997;
```

La cláusula WHERE se evalua a true si y solo si ambas comparaciones son satisfechas.

Ejemplo:
Supongamos que deseamos obtener los nombres de los libros editados por la Editorial "Kapelusz" o por la editorial "Prentice Hall", la consulta sería:

```
SELECT  Nombre
FROM Libros
WHERE Editorial = "Prentice Hall" or Editorial = "Kapelusz"
```

4.1.3 Comparación de Strings.

Dos strings son iguales si tienen la misma secuencia de caracteres.
Cuando comparamos strings por uno de los operadores "menor que", tal como < ó <=, preguntamos si uno precede a otro en orden alfabético. Por ejemplo: 'biblia' < 'biblioteca', y 'biblioteca' < 'bibliotecario'.

En general, sean dos strings $a_1\ a_2\ ...\ a_n$ y $b_1\ b_2\ ...b_m$, decimos que $a_1\ a_2\ ...\ a_n$ "es menor" que $b_1\ b_2\ ...b_m$ si $a_1 < b_1$, ó si $a_1 = b_1$ y $a_2 < b_2$, ó si $a_1 = b_1$ y $a_2 = b_2$ y $a_3 < b_3$, etc.

Expresiones que usan LIKE.

Una forma alternativa para la comparación de expresiones es *s* LIKE *p*. Siendo s un string y p un patrón (un string con el uso opcional de dos caracteres especiales % y _).

% en *p* puede igualar a una secuencia de 0 o más caracteres en *s*.
_ en *p* se iguala a cualquier carácter en *s* (sólo un carácter).

El valor de esta expresión es verdadera si y solo sí el string s se iguala al patrón p. Similarmente *s* NOT LIKE *p* tiene valor verdadero si y solo sí *s* no se igual a *p*.

Ejemplo.
No recordamos el titulo de un libro pero sabemos que empieza con "La ...". Cual puede ser este libro?

```
SELECT *
FROM Libros
WHERE Nombre LIKE 'La%'
```

Resultado de la consulta:

ISBN	Nombre	Genero	Estado	Sig_Top	Nro_I	Editorial	Año_de_Ed	Cant_pag
950-731-260-9	La Resistencia	Novela	Regular	C-45-896	956	Planeta	2000	148
950-9122-55-6	La Maquina Cultural	Novela	Bueno	C-78-965	124	Planeta	1998	292

Ejemplo:
No recordamos el titulo de un libro pero sabemos que tiene la palabra "Maquina" como parte de su nombre.

```
SELECT *
FROM Libros
WHERE Nombre LIKE '%Maquina%'
```

Resultado de la consulta:

ISBN	Nombre	Genero	Estado	Sig_Top	Nro_I	Editorial	Año_de_Ed	Cant_pag
950-9122-55-6	La Maquina Cultural	Novela	Bueno	C-78-965	124	Planeta	1998	292

4.1.4 Comparando tipos de Datos TIME y DATE.

Implementaciones de SQL generalmente suportan los tipos de datos TIME y DATE como tipos de datos especiales (Una explicación de los tipos de Datos provistos por MYSQL se incluye en la sección 2 de este Apunte). Los valores son a veces representables en una variedad de formatos tales como 14/5/1948 ó 14 May 1948.
Podemos comparar dias y horas usando los mismos operadores de comparación que usamos para números y strings. Esto es, < aplicado a dos atributos de tipo DATE, significa que el primer atributo tiene una fecha menor que un segundo atributo.

4.1.5 Ordenando la salida.

Podemos requerir que la salida resultado de una consulta sea producida en un orden especificado. El orden esta basado en el valor de cualquier atributo. Para obtener la salida en un orden dado, agregamos a la sentencia SELECT-FROM-WHERE la cláusula: ORDER BY <lista de atributos>

El orden por defecto es ascendente, pero podríamos especificar además la palabra reservada DESC (para "descendente"). Similarmente, podríamos especificar el orden ascendente, con ASC, pero esta especificación es innecesaria.

Ejemplo 5.1.5.1:
SELECT *
FROM libros
WHERE Año_de_Ed > 1997
ORDER BY Nombre

Da como resultado cuatro tuplas, ordenadas alfabéticamente según el atributo Nombre:

ISBN	Nombre	Genero	Estado	Sig_Top	Nro_I	Editorial	Año_de_Ed	Cant_pag
950-13-6183-7	Apuntes y Aportes para la Gestión Curricular	Ciencias	Bueno	D-7-65	345	Kapelusz	1999	171
950-9122-55-6	La Maquina Cultural	Novela	Bueno	C-78-965	124	Planeta	1998	292
950-731-260-9	La Resistencia	Novela	Regular	C-45-896	956	Planeta	2008	148
0-415-11966-9	Wholeness and the Implicate Order	Novela	Bueno	D-45-655	456	Routledge	1980	224

4.1.6 Algunos Operadores de Agregación.

SQL provee cinco operadores que se aplican a una columna de una relación y producen alguna agregación de tal columna. Estos operados son:

1. **SUM**, la suma de los valores en esta columna.

2. **AVG**, el valor promedio de los valores en esta columna.
3. **MIN**, el valor más pequeño en la columna.
4. **MAX**, el valor más grande en la columna.
5. **COUNT**, el número de valores (incluyendo duplicados a menos que se los elimine explícitamente con la palabra **DISTINCT**);

Ejemplo:
Considerando la instancia de la relación LIBROS podemos preguntar obtener el libro que tiene la mayor cantidad de paginas.

SELECT **MAX**(cant_pag)
FROM Libros;

El resultado es:

Max(cant_pag)
470

Debido a que la siguiente tupla contiene el mayor valor en el columna cant_pag:

ISBN	Nombre	Genero	Estado	Sig_Top	Nro_I	Editorial	Año_de_Ed	Cant_pag
0-13-861337-0	A First Course in Database System	Técnico	Bueno	A-12-456	827	Prentice Hall	1997	470

Si hubiéramos utilizado MIN en lugar de MAX en la esta consulta el resultado sería:

El resultado es:

Min(cant_pag)
148

Debido a que:

ISBN	Nombre	Genero	Estado	Sig_Top	Nro_I	Editorial	Año_de_Ed	Cant_pag
950-731-260-9	La Resistencia	Novela	Regular	C-45-896	956	Planeta	2000	148

Ejemplo:
Si deseamos saber cuántos libros tiene la biblioteca, podríamos consultar:
SELECT COUNT(*)
FROM Libros;

El resultado utilizando la misma tabla que el ejemplo anterior, es:

Count
5

4.2 Tipos de Datos

Los tipos de datos que puede haber en un campo, se pueden agrupar en tres grandes grupos:

4.2.1 Tipos numéricos

Existen tipos de datos numéricos, que se pueden dividir en dos grandes grupos, los que están en coma flotante (con decimales) y los que no.

TinyInt: es un número entero con o sin signo. Con signo el rango de valores válidos va desde -128 a 127. Sin signo, el rango de valores es de 0 a 255.

Bit ó Bool: un número entero que puede ser 0 ó 1.

SmallInt: número entero con o sin signo. Con signo el rango de valores va desde -32768 a 32767. Sin signo, el rango de valores es de 0 a 65535.

MediumInt: número entero con o sin signo. Con signo el rango de valores va desde -8.388.608 a 8.388.607. Sin signo el rango va desde 0 a16777215.

Integer, Int: número entero con o sin signo. Con signo el rango de valores va desde -2147483648 a 2147483647. Sin signo el rango va desde 0 a 429.4967.295

BigInt: número entero con o sin signo. Con signo el rango de valores va desde -9.223.372.036.854.775.808 a 9.223.372.036.854.775.807. Sin signo el rango va desde 0 a 18.446.744.073.709.551.615.

Float: número pequeño en coma flotante de precisión simple. Los valores válidos van desde -3.402823466E+38 a -1.175494351E-38, 0 y desde 1.175494351E-38 a 3.402823466E+38.

xReal, Double: número en coma flotante de precisión doble. Los valores permitidos van desde -1.7976931348623157E+308 a -2.2250738585072014E-308, 0 y desde 2.2250738585072014E-308 a 1.7976931348623157E+308.

Decimal, Dec, Numeric: Número en coma flotante desempaquetado. El número se almacena como una cadena

Tipo de Campo	Tamaño de Almacenamiento
TINYINT	1 byte
SMALLINT	2 bytes
MEDIUMINT	3 bytes
INT	4 bytes
INTEGER	4 bytes

BIGINT	8 bytes
FLOAT(X)	4 ú 8 bytes
FLOAT	4 bytes
DOUBLE	8 bytes
DOUBLE PRECISION	8 bytes
REAL	8 bytes
DECIMAL(M,D	M+2 bytes sí D > 0, M+1 bytes sí D = 0
NUMERIC(M,D)	M+2 bytes if D > 0, M+1 bytes if D = 0

4.2.2 Tipos Fecha

A la hora de almacenar fechas, hay que tener en cuenta que Mysql no comprueba de una manera estricta si una fecha es válida o no. Simplemente comprueba que el mes esta comprendido entre 0 y 12 y que el día esta comprendido entre 0 y 31.

Date: tipo fecha, almacena una fecha. El rango de valores va desde el 1 de enero del 1001 al 31 de diciembre de 9999. El formato de almacenamiento es de año-mes-dia

DateTime: Combinación de fecha y hora. El rango de valores va desde el 1 de enero del 1001 a las 0 horas, 0 minutos y 0 segundos al 31 de diciembre del 9999 a las 23 horas, 59 minutos y 59 segundos. El formato de almacenamiento es de año-mes-dia horas:minutos:segundos

TimeStamp: Combinación de fecha y hora. El rango va desde el 1 de enero de 1970 al año 2037. El formato de almacenamiento depende del tamaño del campo:

Tamaño	**Formato**
14	AñoMesDiaHoraMinutoSegundo aaaammddhhmmss
12	AñoMesDiaHoraMinutoSegundo aammddhhmmss
8	ñoMesDia aaaammdd
6	AñoMesDia aammdd
4	AñoMes aamm
2	Año aa

Time: almacena una hora. El rango de horas va desde -838 horas, 59 minutos y 59 segundos a 838, 59 minutos y 59 segundos. El formato de almacenamiento es de 'HH:MM:SS'

Year: almacena un año. El rango de valores permitidos va desde el año 1901 al año 2155. El campo puede tener tamaño dos o tamaño 4 dependiendo de si queremos almacenar el año con dos o cuatro dígitos.

Tipo de Campo	Tamaño de Almacenamiento
DATE	3 bytes
DATETIME	8 bytes
TIMESTAMP	4 bytes
TIME	3 bytes
YEAR	1 byte

4.2.3 Tipos de cadena

Char(n): almacena una cadena de longitud fija. La cadena podrá contener desde 0 a 255 caracteres.

VarChar(n): almacena una cadena de longitud variable. La cadena podrá contener desde 0 a 255 caracteres.

Dentro de los tipos de cadena se pueden distinguir otros dos subtipos, los tipo Test y los tipo BLOB (Binary large Object)

La diferencia entre un tipo y otro es el tratamiento que reciben a la hora de realizar ordenamientos y comparaciones. Mientras que el tipo test se ordena sin tener en cuenta las Mayúsculas y las minúsculas, el tipo BLOB se ordena teniéndolas en cuenta.

Los tipos BLOB se utilizan para almacenar datos binarios como pueden ser ficheros.

TinyText y TinyBlob: Columna con una longitud máxima de 255 caracteres.

Blob y Text: un texto con un máximo de 65535 caracteres.

MediumBlob y MediumText: un texto con un máximo de 16.777.215 caracteres.

LongBlob y LongText: un texto con un máximo de caracteres 4.294.967.295. Hay que tener en cuenta que debido a los protocolos de comunicación los paquetes pueden tener un máximo de 16 Mb.

Enum: campo que puede tener un único valor de una lista que se especifica. El tipo Enum acepta hasta 65535 valores distintos.

Set: un campo que puede contener ninguno, uno ó varios valores de una lista. La lista puede tener un máximo de 64 valores.

Tipo de campo	Tamaño de Almacenamiento
CHAR(n)	n bytes
VARCHAR(n)	n +1 bytes
TINYBLOB, TINYTEXT	Longitud+1 bytes
BLOB, TEXT	Longitud +2 bytes
MEDIUMBLOB, MEDIUMTEXT	Longitud +3 bytes
LONGBLOB, LONGTEXT	Longitud +4 bytes
ENUM('value1','value2',...)	1 ó dos bytes dependiendo del número de valores
SET('value1','value2',...)	1, 2, 3, 4 ó 8 bytes, dependiendo del número de valores

Nota:

Al realizar comparaciones de campos contra valores literales, es importante notar que todos los campos que puedan contener valores numéricos pueden compararse directamente contra el valor literal; mientras que si se comparan campos de tipo cadena o tipo fechas, los valores literales deben ir entre comillas simples.

4.3 Modificaciones a la Base de Datos

Existen otras sentencias SQL que no retornan un resultado, pero en su lugar cambian el estado de la base de datos. En esta sección nos enfocaremos en tres tipos de sentencias que nos permiten:

1. Insertar tuplas en la relacion.
2. Borrar ciertas tuplas en la relación
3. Actualizar ciertas componentes (atributos) de ciertas tuplas existentes.

Nos referiremos a estos tres tipos de operaciones colectivamente como *modificaciones*. Estas instrucciones son parte del Lenguaje de Manipulación de Datos (LMD).

4.3.1 Inserción

La forma básica de una sentencia consiste de:

1. La palabra INSERT INTO,
2. El nombre de una relación, supongamos R,
3. Una lista de atributos entre paréntesis de R,
4. La palabra VALUES, y
5. Una expresión de tupla, que es, una lista entre paréntesis de *valores* concretos, uno para cada atributo en la lista (3).

Es decir, la forma de insertar, básicamente consiste de:
INSERT INTO R(A_1, A_2, ..., A_n) VALUES (v_1, v_2,…., v_n);

Una tupla es creada usando el valor v_i para el atributo A_i para i = 1, 2, … , n.
Si la lista de atributos no incluye todos los atributos de la relación R, entonces, la tupla es creada con valores por defecto para todos los atributos faltantes. Un valor común por defecto es el valor NULL.

Ejercicio 6.1.1.
Supongamos que queremos agregar una tupla a la tabla de Libros, la relación es:

> LIBROS (ISBN, Nombre, Genero, Estado_de_cons, Sig_Top, Nro_Inventario, Editorial, Año_de_Ed, cant_de_pag)
>
> INSERT INTO libros (ISBN, Nombre, Genero, Estado_de_cons, Sig_Top, Nro_Inventario, Editorial, Año_de_Ed, cant_de_pag)
> VALUES (‘950-07-0428-5’, ´Crónica de una muerte Anunciada´, ‘Novela’, ‘Bueno’, ‘D-456’, 9897, ‘ Sudamericana’, 1993, 193)
>
> Y el efecto es agregar una tupla con los compontes enumerados en la última linea.

Ahora, la tablas de LIBROS tiene las siguientes tuplas.

ISBN	Nombre	Genero	Estado	Sig_Top	Nro_I	Editorial	Año_de_Ed	Cant_pag
0-13-861337-0	A First Course in Database System	Técnico	Bueno	A-12-456	827	Prentice Hall	1997	470
950-731-260-9	La Resistencia	Novela	Regular	C-45-896	956	Planeta	2008	148
950-9122-55-6	La Maquina Cultural	Novela	Bueno	C-78-965	124	Planeta	1998	292
0-415-11966-9	Wholeness and the Implicate Order	Novela	Bueno	D-45-655	456	Routledge	1980	224
950-13-6183-7	Apuntes y Aportes para la Gestión Curricular	Ciencias	Regular	D-7-65	345	Kapelusz	1999	171
950-07-0428-5	Crónica de una muerte Anunciada	Novela	Bueno'	D-456'	9897	Sudamericana'	1992	193

Ejemplo:
Dado el esquema Alumno(Legajo, DNI, Nombre, Edad) y la siguiente relación:

Legajo	DNI	Nombre	Edad
90489	30456782	López Raúl	25
89456	29452789	González María	24
90856	30478236	Ruiz Alejandro	24
90478	26741258	Sosa Mario	29

Supongamos que queremos insertar los datos del siguiente alumno:
Nombre: Pérez Gloria
D.N.I.: 25123478
Legajo: 88123
Edad: 30

Deberemos ejecutar:
INSERT INTO Alumno (Legajo, DNI, Nombre, Edad)
VALUES (88123, 25123478, Pérez Gloria, 30);

El resultado de la inserción se verá reflejado como:

Legajo	DNI	Nombre	Edad
90489	30456782	López Raúl	25
89456	29452789	González María	24
90856	30478236	Ruiz Alejandro	24
90478	26741258	Sosa Mario	29
88123	25123478	Pérez Gloria	30

Por otro lado, si lo que deseamos es insertar más de un alumno, la sintaxis sería:

INSERT INTO Alumno (Legajo, DNI, Nombre, Edad)
VALUES (88883, 25123478, Sánchez Susana, 30);
VALUES (89153, 30167532, Pérez Martín, 26);
VALUES (90236, 30127943, Ramírez Pablo, 25);

Y el resultado de la inserción de verá reflejado como:

Legajo	DNI	Nombre	Edad
90489	30456782	López Raúl	25
89456	29452789	González María	24
90856	30478236	Ruiz Alejandro	24
90478	26741258	Sosa Mario	29
88123	25123478	Pérez Gloria	30
88883	25123478	Sánchez Susana	30
89153	30167532	Pérez Martín	26
90236	30127943	Ramírez Pablo	25

4.3.2 Borrado

Una sentencia de borrado consiste de:

1. Las palabras claves DELETE FROM,
2. El nombre de una relación, digamos R,
3. la palabra WHERE, y
4. una condición.

Asi, la forma genérica de un borrado es:

DELETE FROM R **WHERE** <condicion>;

El efecto de ejecutar esta sentencia es que toda tupla que satisface la condición será borrada desde la relación R.

Ejemplo:
Podemos borrar desde la relación Libros aquel libro que tienen un estado ‘Regular’ y cuyo nombre comienza con la palabra “Apuntes”.

```
DELETE FROM Libros
WHERE nombre LIKE “Apuntes%” AND
        Estado = “Regular”;
```

Ejemplo:
Podemos también borrar varias tuplas al mismo tiempo, por ejemplo si queremos eliminar todos libros cuyo Año de Edición es menor que 1999.

```
DELETE FROM Libros
```

WHERE Año_de_Edición < 1999.

Ejemplo:

Se cuenta con la tabla “Alumno” de la siguiente manera:

Legajo	DNI	Nombre	Edad	Provincia
90489	30456782	López Raúl	25	Nqn.
89456	29452789	González María	24	Nqn.
90856	30478236	Ruiz Alejandro	24	Nqn.
90478	26741258	Sosa Mario	29	Nqn.

Si se desea eliminar el registro correspondiente al alumno cuyo legajo es: 90489

Deberemos ejecutar:

```
DELETE *
FROM   Alumno
WHERE Legajo = 90489
```

Y el resultado de la eliminación se verá reflejado como:

Legajo	DNI	Nombre	Edad	Provincia
89456	29452789	González María	24	Nqn.
90856	30478236	Ruiz Alejandro	24	Nqn.
90478	26741258	Sosa Mario	29	Nqn.

Si, por otro lado se desea eliminar más de un registro, por ejemplo, si quisiéramos eliminar todos los alumnos cuya Edad sea mayor o igual a 25 años de la tabla Alumno

Legajo	DNI	Nombre	Edad	Provincia
90489	30456782	López Raúl	25	Nqn.
89456	29452789	González María	24	Nqn.
90856	30478236	Ruiz Alejandro	24	Nqn.
90478	26741258	Sosa Mario	29	Nqn.

Deberemos ejecutar:

```
DELETE *
FROM   Alumno
WHERE Edad >= 25
```

Y el resultado de la eliminación se verá reflejado como:

Legajo	DNI	Nombre	Edad	Provincia
89456	29452789	González María	24	Nqn.
90856	30478236	Ruiz Alejandro	24	Nqn.

Por último, si se desea borrar todos los registros de la tabla Alumno, deberemos ejecutar:

Deberemos ejecutar:

DELETE *
FROM Alumno

Y el resultado de la eliminación se verá reflejado como:

Legajo	DNI	Nombre	Edad	Provincia

4.3.3 Modificación

Con una actualización cambiaremos el valor de los atributos de una o más tuplas que existen en la base de datos. La forma general de una declaración UPDATE es:

1. La palabra clave UPDATE.
2. El nombre de una relación, digamos R,
3. la palabra SET,
4. Una lista de formulas en la cual cada una modifica el valor contenido en un atributo de la relación R asignándole un nuevo valor. El valor asignado puede ser una expresión o un valor constante,
5. La palabra clave WHERE,
6. Una condición.

La forma de un **UPDATE** es:
UPDATE R **SET** <asignaciones de nuevos valores> **WHERE** <condicion>;

Cada asignación de nuevo valor de <asignaciones de nuevos valores> es un atributo, un signo igual, y una fórmula. Si existe más de una asignación, ellas son separadas por comas.
El efecto de esta sentencia es encontrar todas las tuplas de R, que satisfacen la condición especificada y cambiar sus valores de acuerdo a las formulas en las asignaciones.

Ejemplo:
Supongamos que hemos cometido un error al insertar la tupla del ejemplo anterior sobre el libro 'Cronica de una muerte Anunciada', la cantidad de páginas es 192, no 193. Y el año de edición es 1992, no 1993.

UPDATE Libros
SET cant_de_pag = 192, año_de_ed = 1992
WHERE ISBN = ‘950-07-0428-5’;

Ejemplo:

Se cuenta con la tabla “Alumno” de la siguiente manera:

Legajo	DNI	Nombre	Edad	Provincia
90489	30456782	López Raúl	25	Nqn.
89456	29452789	González María	24	Nqn.
90856	30478236	Ruiz Alejandro	24	Nqn.
90478	26741258	Sosa Mario	29	Nqn.

Se desea modificar los datos almacenados en todos los registros para el campo “Provincia”, cambiándolos por *Neuquén*.

Deberemos ejecutar:

UPDATE Alumno
SET Provincia = ‘Neuquén’

Y el resultado de la modificación se verá reflejado como:

Legajo	DNI	Nombre	Edad	Provincia
90489	30456782	López Raúl	25	Neuquén
89456	29452789	González María	24	Neuquén
90856	30478236	Ruiz Alejandro	24	Neuquén
90478	26741258	Sosa Mario	29	Neuquén

Si, por otro lado se desea modificar la Edad y el Nombre del alumno cuyo legajo es: Legajo: 89456 con los siguientes datos

Nombre: González Mariana
Edad: 27

Deberemos ejecutar:

UPDATE Alumno
SET Nombre = ‘Gonzáles Mariana’,
Edad = 27
WHERE Legajo=89456

Y el resultado de la modificación se verá reflejado como:

Legajo	DNI	Nombre	Edad	Provincia
90489	30456782	López Raúl	25	Neuquén
89456	29452789	González Mariana	24	Neuquén
90856	30478236	Ruiz Alejandro	24	Neuquén
90478	26741258	Sosa Mario	29	Neuquén