LENGUAJE SQL

ÍNDICE

1. LENGUAJE SQL

Lo que podemos hacer con el lenguaje SQL es:

- Consultar datos de la Base de Datos.
- Insertar, modificar y borrar datos.
- Crear, modificar y borrar objetos de la Base de Datos.
- Controlar el acceso a la información.
- Garantizar la consistencia de los datos.

Se distinguen tres tipos de sentencias SQL:

- Sentencias de definición de datos. (Lenguaje de Definición de Datos **DDL**), Se utilizan para:
  - Crear objetos de base de datos ---------- **SENTENCIA CREATE**
  - Eliminar objetos de base de datos ---------- **SENTENCIA DROP**
  - Modificar objetos de base de datos ---------- **SENTENCIA ALTER**
- Sentencias de manipulación de datos. (Lenguaje de Manipulación de Datos **DML**), Se utilizan para:
  - Recuperar información ---------- **SENTENCIA SELECT**
  - Actualizar la información:
    - Añadir filas ---------- **SENTENCIA INSERT**
    - Eliminar filas ---------- **SENTENCIA DELETE**
    - Modificar filas ---------- **SENTENCIA UPDATE**

1. LENGUAJE SQL

- Sentencias de control de datos. (Lenguaje de Control de datos DCL), se utilizan para:
  - Crear privilegios de acceso a los datos ------- **SENTENCIA GRANT**
  - Quitar privilegios de acceso a los datos ------- **SENTENCIA REVOKE**

1.1.- TIPOS DE SENTENCIAS SQL

Realizaremos algunas consideraciones sobre las notaciones y formatos utilizados.

- **a) Formatos de las instrucciones:** Se escriben utilizando la siguiente notación:
  - Las palabras reservadas de SQL suelen ir en mayúsculas.
  - Los nombres de objetos (tablas, columnas, etcétera) aparecen en el formato TipoTítulo (las iniciales de las palabras en mayúsculas)
  - Las llaves **{ }** indican la elección obligatoria entre varios elementos.
  - La barra vertical **|** separa los elementos en una elección.
  - Los corchetes **[ ]** encierran un elemento opcional.
  - El punto y coma **;** que aparece al final de cada comando es el separador de instrucciones y en realidad no forma parte de la sintaxis del lenguaje SQL, pero suele ser un elemento requerido por las herramientas de cliente para determinar el final del comando SQL y enviar la orden (sin él ;) al servidor.

1.1.- TIPOS DE SENTENCIAS SQL

- **b) Consulta de los datos.:** Realizar una consulta en SQL consiste en recuperar u obtener aquellos datos que, almacenados en filas y columnas de una o varias tablas de una base de datos, cumplen unas determinadas especificaciones. Para realizar cualquier consulta se utiliza la sentencia SELECT. La sentencia Select la veremos más adelante, por ejemplo:
  - SELECT { * | [*NombreColumna]*, [*NombreColumna*],[....] }
  - FROM *NombreTabla*
  - [WHERE *Condicion];*
- ***Explicación**: hay que escoger obligatoriamente una de las opciones, entre indicar los nombres de las columnas y el asterisco * (por eso aparecen las posibles opciones entre llaves y separadas por una barra). En caso de escoger la segunda opción se pueden indicar una o varias columnas (por eso aparece entre corchetes y seguido de puntos suspensivos). La cláusula WHERE es opcional (por eso aparece entre corchetes).*

1.2.- SENTENCIAS SQL

Otro ejemplo: Supongamos que escribimos:

- select apellido,(salario+ifnull(comision,0)) “salario total”, dept_no from empleados where oficio = ‘analista’ order by dept_no;

O bien que escribimos:

- SELECT apellido, salario+IFNULL(comision,0) “Salario total”, dept_no
- FROM empleados
- WHERE oficio = ’ANALISTA’
- ORDER BY dept_no;

Este segundo formato es más claro y visual. Cada cláusula, comenzando con una palabra reservada, aparece en una línea y se han introducido algunos espacios en blancos y saltos de línea para separar.

Utilizaremos este formato

1.2.- SENTENCIAS SQL

Un objeto tendrá un nombre (*NombreObjeto*) que lo identifique de forma única dentro de su base de datos.

El estándar define que pueden tener hasta 18 caracteres empezando con un carácter alfabético y continuando con caracteres numéricos y/o alfabéticos.

En la práctica este estándar se ha ampliado y MySQL permite nombres de identificadores de hasta 64 caracteres sin espacios en blanco. Para los nombres de las bases de datos y de las tablas no están permitidos '/', '\' ni '.'

3.1- IDENTIFICADORES

Existen palabras que tiene un significado especial para el gestor de la base de datos y no pueden ser utilizadas como identificadores.

Serán todas las palabras que irán apareciendo en los formatos de las instrucciones

3.1- PALABRAS RESERVADAS

Datos constantes o literales: En SQL podemos utilizar los siguientes tipos de constantes:

- **Constantes numéricas.:** Construidas mediante una cadena de dígitos que puede llevar un punto decimal, y que pueden ir precedidos por un signo + ó -. (Ej. : -2454.67) También se pueden expresar constantes numéricas empleado el formato de coma flotante, notación científica. (Ej. : 34.345E-8).
- **Constantes de cadena.:** Consisten en una cadena de caracteres encerrada entre comillas simples. (Ej.: 'Hola Mundo')
- **Constantes de fecha.:** En realidad las constantes de fecha, en MySQL, se escriben como constantes de cadena sobre las cuales se aplicarán las correspondientes funciones de conversión. Existe una gran cantidad de formatos americano, europeo, japonés, etcétera. La mayoría de los productos pueden trabajar también con FECHA Y HORA en distintos formatos.

3.3- DATOS

Datos variables: una parte importante de la definición de un dato es la especificación de su tipo. Al indicar el tipo de datos se suele indicar también el tamaño.

- a) Datos numéricos:
  - **INT[(*num*)] o INTEGER [(*num*)]:** Se utiliza para guardar datos numéricos enteros, siendo *num* el número de dígitos
  - **FLOAT(*escala*, *precision*):** Se utiliza para guardar datos numéricos en coma flotante. La escala indica el número total de dígitos. La precisión el número de posiciones decimales
  - **NUMERIC(*escala, precisión*):** Se utiliza para guardar datos numéricos. La escala indica el número total de dígitos. La precisión el número de posiciones decimales, y si no se especifica se supone 0

3.3- DATOS

- b) Datos alfanuméricos o cadenas de caracteres:
  - **CHAR *(long):*** Se utiliza para guardar cadenas de caracteres de longitud fija especificada entre paréntesis. La longitud, *long,* puede ser un número entre 0 y 255
  - **VARCHAR *(long):*** Se utiliza igualmente para almacena cadenas de caracteres de longitud variable cuyo límite máximo es el valor especificado como *long.* La longitud puede ser un número entre 0 y 255
  - **TEXT:** Un texto de longitud máxima 65.535 caracteres. (2 ^ 16 – 1). Se almacena como un VARCHAR
  - **LONGTEXT :** Un texto de longitud máxima 4 Gigas caracteres. (2 ^ 32 – 1) Se almacena como un VARCHAR

3.3- DATOS

- **b) Datos alfanuméricos o cadenas de caracteres:**
  - *NOTA: La diferencia entre ambos tipos, CHAR y VARCHAR, está en la forma de almacenarlos. Teniendo en cuenta que un carácter necesita un byte para su almacenamiento, un dato definido como CHAR(5) se almacena reservando espacio para los 5 caracteres, aunque el datos almacenado tenga menos de 5. Sin embargo, un dato definido como VARCHAR(5) en el que se almacene un datos solo reserva espacio para el número de caracteres que tenga ese dato más uno para indicar el final de la cadena.*

VALOR	CHAR(5)	TAMAÑO RESERVADO	VARCHAR(5)	TAMAÑO RESERVADO
' '	' '	5 BYTES	' '	1 BYTE
'AB'	'AB '	5 BYTES	'AB'	3 BYTES
'ABCDE'	'ABCDE '	5 BYTES	'ABCDE'	6 BYTES

3.3- DATOS

- **c) Fechas: s**e tratan como cadenas de caracteres por lo que estar entre comillas para su utilización.
  - **DATE:** permite almacenar fechas, incluyendo en esa información: año, mes y día con la forma 'YYYY-MM-DD'.
  - **DATETIME:** permite almacenar fechas y horas, incluyendo en esa información: año, mes, día, horas, minutos y segundos con la forma 'YYYY-MM-DD HH:MM:SS'.
  - **TIME:** permite almacenar horas, incluyendo en esa información: horas, minutos y segundos con la forma 'HH:MM:SS'.
- **d) Binarios.**
  - **BOOLEAN:** Almacena valores binarios formados por combinaciones de los valores 1(verdadero) y 0 (falso).

3.3- DATOS

Operadores aritméticos: Operan entre valores numéricos y devuelven un valor numérico como resultado de realizar los cálculos indicados, algunos de ellos se pueden utilizar también con fechas. Los operadores aritméticos son:

- + Suma
- - Resta
- * Multiplicación
- / División
- Div División entera (parte entera de la división, sin decimales)

4- OPERADORES

Operadores de comparación: los operadores de comparación dan como resultado un valor de tipo *Verdadero/Falso/Nulo (True/False/Null).* Los operadores de comparación son:

- = Igual
- != Distinto
- <> Distinto
- < Menor
- <= Menor o igual
- > Mayor
- >= Mayor o igual
- BETWEEN / NOT BETWEEN
- IN / NOT IN
- IS NULL / IS NOT NULL
- LIKE

4- OPERADORES

Operadores de comparación:

- **BETWEEN valor1 AND valor2:** Da como resultado VERDADERO si el valor comparado es mayor o igual que valor1 y menor o igual que valor2 y FALSO en el caso contrario
- **IN (lista de valores separados por comas):** Da como resultado VERADADERO si el valor comparado está dentro de la lista de valores especificado y FALSO en el caso contrario
- **IS NULL:** Da como resultado VERDADERO si el valor del dato comparado es nulo (NULL) y FALSO en el caso contrario
- **LIKE:** Permite comparar dos cadenas de caracteres con la peculiaridad de que admite caracteres comodines. Los caracteres comodines son '%' y '_'. Estos caracteres permiten utilizar patrones en la comparación.

4- OPERADORES

Ejemplos:

- La expresión: APELLIDO = 'JIMENEZ' será verdadera *(*true*)* en el caso de que el valor de la columna APELLIDO (suponemos que se trata de una columna) sea 'JIMENEZ' y falsa *(*false*)* en caso contrario.
- La expresión: SALARIO > 300000 será verdadera (true*)* en el caso de que SALARIO tenga un valor superior a 300000 y falsa (false*)* en caso contrario.
- La expresión APELLIDO LIKE 'A%' será verdadera (true) si el apellido empieza por A y después tiene cualquier grupo de caracteres y falsa (false) en caso contrario. La expresión APELLIDO LIKE 'A_B_C' será verdadera (true) si el primer carácter es una A, el segundo cualquiera, el tercero una B, el cuarto cualquiera y el quinto una C y falsa (false) en caso contrario.

4- OPERADORES

Estos operadores de comparación se utilizan fundamentalmente para construir condiciones de búsqueda en la base de datos. De esta forma se seleccionarán aquellas filas que cumplan la condición especificada (aquellas filas para las que el valor de la expresión sea *true).*

Por ejemplo, el siguiente comando seleccionará todas las filas de la tabla empleados que en la columna OFICIO aparezca el valor 'VENDEDOR'.

```
mysql> SELECT emp_no,
apellido,oficio,fecha_alta,comision,salario
-> FROM empleados
-> WHERE oficio = 'VENDEDOR';
```

4- OPERADORES

4- OPERADORES

EMP_Nº	APELLIDO	OFICIO	FECHA_ALTA	COMISIÓN	SALARIO
7499	ALONSO	VENDEDOR	1981-02-23	400.00	1400.00
7654	MARTÍN	VENDEDOR	1981-09-28	1600.00	1500.00
1844	CALVO	VENDEDOR	1991-09-08	0.00	1800.00

Operadores lógicos: Operan entre datos con valores lógicos *Verdadero/Falso/Nulo* y devuelven el valor lógico *Verdadero/Falso/Nulo.* Los operadores lógicos son:

- ! NOT
- && AND
- || OR
- XOR

NOT	VERDADERO	FALSO	NULO
	FALSO	VERDADERO	NULO

- El operador NOT devuelve VERDADERO cuando el operando es falso, y FALSO cuando el operando es verdadero y NULO cuando el operando es nulo.

4- OPERADORES

AND	VERDADERO	FALSO	NULO
VERDADERO	VERDADERO	FALSO	NULO
FALSO	FALSO	FALSO	FALSO
NULO	NULO	FALSO	NULO

- El operador AND devolverá VERDADERO cuando los dos operandos sean verdaderos, FALSO cuando alguno de los dos operandos sea falso y NULO en los demás casos.

4- OPERADORES

OR	VERDADERO	FALSO	NULO
VERDADERO	VERDADERO	VERDADERO	VERDADERO
FALSO	VERDADERO	FALSO	NULO
NULO	VERDADERO	NULO	NULO

- El operador OR devolverá VERDADERO cuando alguno de los operandos sea verdadero, FALSO cuando los dos operandos sean falsos; y NULO en los demás casos.

4- OPERADORES

4- OPERADORES

XOR	VERDADERO	FALSO	NULO
VERDADERO	FALSO	VERDADERO	NULO
FALSO	VERDADERO	FALSO	NULO
NULO	NULO	NULO	NULO

- El operador XOR devolverá VERDADERO si uno de los operandos es verdadero y el otro falso, FALSO cuando ambos sean verdaderos o ambos falsos y NULO si alguno de ellos es nulo.

Ejemplos:

- Supongamos que queremos consultar los empleados cuyo OFICIO = 'VENDEDOR' y que además su SALARIO >1500. En este caso emplearemos el operador lógico AND. Este operador devolverá el valor *true* cuando los dos operandos o expresiones son verdaderas. Podemos decir que se utiliza el operador AND cuando queremos que se cumplan las dos condiciones.

```
mysql> SELECT apellido, salario, oficio
    -> FROM empleados
    -> WHERE oficio = 'VENDEDOR' AND salario > 1500;
```

APELLIDO	SALARIO	OFICIO
CALVO	1800.00	VENDEDOR

- Cuando lo que queremos es buscar filas que cumplan alguna de las condiciones que se indican emplearemos el operador OR. Este operador devolverá el valor VERDADERO cuando alguno de los dos operandos o expresiones es verdadero. Podemos decir que se utiliza el operador OR cuando queremos que se cumpla la primera condición, o la segunda o ambas.

```
mysql> SELECT apellido, salario, oficio
    -> FROM empleados
    -> WHERE oficio = 'VENDEDOR' OR salario > 1300;
```

APELLIDO	SALARIO	OFICIO
ALONSO	1400.00	VENDEDOR
MARTIN	1500.00	VENDEDOR
GARRIDO	3650.00	DIRECTOR
REY	6000.00	PRESIDENTE
LÓPEZ	1350.00	EMPLEADO

4- OPERADORES

- El operador NOT se utiliza para cambiar el valor devuelto por una expresión lógica o de comparación, tal como se ilustra en el siguiente ejemplo:

```
mysql> SELECT apellido, salario, oficio
    -> FROM empleados
    -> WHERE NOT (oficio = 'VENDEDOR');
```

APELLIDO	SALARIO	OFICIO
LÓPEZ	1350.50	EMPLEADO
GARRIDO	3850.00	DIRECTOR
REY	6000.00	PRESIDENTE

4- OPERADORES

- Podemos formar expresiones lógicas en las que intervengan varios operadores lógicos de manera similar a como se haría con expresiones aritméticas en las que intervienen varios operadores aritméticos.

```
mysql> SELECT apellido, salario, oficio
    -> FROM empleados
    -> WHERE NOT (oficio = 'VENDEDOR' AND salario > 1500);
```

APELLIDO	SALARIO	OFICIO
ALONSO	1400.00	VENDEDOR
LÓPEZ	1350.50	EMPLEADO
MARTÍN	1500.00	VENDEDOR
GARRIDO	3850.00	DIRECTOR
REY	6000.00	PRESIDENTE

4- OPERADORES

Prioridad de los operadores:

PRIORIDAD	OPERADOR	OPERACIÓN
1º	*,/,DIV	Multiplicación, División
2º	+,-	Suma, resta
3º	=.!=, <>,<=,>= IS, LIKE, BETWEEN, IN	Comparación
4º	NOT	Negación
5º	AND	Conjunción
6º	OR,XOR	Inclusión, exclusión

4- OPERADORES

Las funciones no modifican los valores del argumento, indicado entre paréntesis, sino que devuelven un valor creado a partir de los argumentos que se le pasan en la llamada, y ese valor puede ser utilizado en cualquier parte de una sentencia SQL.

Existen muchas funciones predefinidas para operar con todo tipo de datos. A continuación se indican las funciones predefinidas más utilizadas. Estas funciones se tratarán con más detalle y se realizarán ejemplos más adelante

Vamos a ver estas funciones agrupadas según el tipo de datos con los que operan y/o el tipo de datos que devuelven.

Nota: estas funciones pueden variar según el sistema gestor que se utilice.:

5.- FUNCIONES

Funciones numéricas o aritméticas

Función	Nombre Función	Valor que devuelve
ABS(num)	Valor absoluto	Valor absoluto de num
CEIL(num)	Función "Techo"	Devuelve el entero más pequeño mayor que num
FLOOR(num)	Función "Suelo"	Devuelve el entero más grande menor que num
EXP(num)	Potencia del número e	Devuelve el número e elevado a num
LN(num)	Logaritmo neperiano	Devuelve el logaritmo en base e de num
LOG(num)	Logaritmo	Devuelve el logaritmo en base 10 de num
MOD(num1, num2)	Módulo	Resto de la división entera de num1 por num2
Pi()	Pi	Devuelve el valor de la constante PI
POWER(num1, num2).	Potencia	Devuelve num1 elevado a num2.
RAND()	Número aleatorio	Genera un número aleatorio entre 0 y 1
ROUND(num1,num2)	Redondeo	Devuelve num1 redondeado a num2 decimales. Si se omite num2 redondea a 0 decimales

5.- FUNCIONES

Funciones numéricas o aritméticas

Función	Nombre Función	Valor que devuelve
SIGN(num)	Signo	Si num<0 devuelve –1 Si num=0 devuelve 0 Si num >0 devuelve 1
SQRT(num)	Raíz cuadrada	Devuelve la raíz cuadrada de num
TRUNCATE(num1,num2)	Truncado	Devuelve num1 truncado a num2 decimales. Si se omite num2 truca a 0 decimales

Funciones de caracteres

Función	Nombre Función	Valor que devuelve
ASCII(cad2)	ASCII	Código ascci del carácter cad1
CHAR(num)	Carácter ASCII	Devuelve el carácter cuyo código ASCII es num
CONCAT(cad1,cad2,..)	Concatenar	Concatena cad1 con cad2. Si existieses más cadenas, cad3, .., las concatenaría a continuación
INSERT(cad1,pos,len,cad2)	Insertar	Devuelve cad1 con len caracteres desde pos en adelante sustituidos en cad2
LENGTH(cad1)	Longitud	Devuelve la longitud de cad1
LOCATE(cad1,cad2,pos)	Localizar	Devuelve la posición de la primera ocurrencia de cad1 en cad2 empezando desde pos
LOWER(cad2)	Minúsculas	La cadena cad1 en minúsculas
LPAD(*cad1, n, cad2)*	Rellenar (Izquierda)	Añade a *cad1* por la izquierda *cad2*, hasta que tenga n caracteres. Si se omite *cad2,* añade blancos.
LTRIM(*cad1*)	Suprimir (Izquierda)	Suprime blancos a la izquierda de *cad1.*

Funciones de caracteres

Función	Nombre Función	Valor que devuelve
REPLACE(cad1,cad2,cad3)	Reemplazar	Devuelve cad1 con todas las ocurrencias de cad2 reemplazadas por cad3
RPAD(cad1, n, cad2)	Rellenar (Derecha)	Igual que LPAD pero por la derecha.
RTRIM(c1)	Suprimir (Derecha)	Suprime blancos a la derecha de c1. Igual que LTRIM pero por la izquierda.
SUBSTR(c1, n, m)	Subcadena	Devuelve una subcadena a partir de c1 comenzando en la posición n tomando m caracteres
UPPER(cad1)	Mayúsculas	La cadena cad1 en mayúsculas.

Funciones de fecha

Función	Nombre Función	Valor que devuelve
ADDDATE(Fecha, Num)	Incremento de días	Devuelve Fecha incrementada en Num días
SUBDATE(Fecha, Num)	Decremento de días	Devuelve Fecha decrementada en Num días
DATE_ADD(Fecha, INTERVAL Num Formato)	Incremento	Devuelve Fecha incrementada en Num veces lo indicado en Formato El Formato puede ser entre otros: DAY, WEEK, MONTH, YEAR, HOUR, MINUTE, SECOND
DATE_SUB(Fecha, INTERVAL num Formato)	Decremento	Devuelve Fecha decrementada en Num veces lo indicado en Formato El Formato puede ser entre otros: DAY, WEEK, MONTH, YEAR, HOUR, MINUTE, SECOND
DATEDIFF(Fecha1,Fecha2)	Diferencia de fechas	Devuelve el número de días entre Fecha1 y Fecha2
DAYNAME(Fecha)	Nombre del día de la semana	Devuelve el nombre del día de la semana de Fecha
DAYOFMONTH(Fecha)	Día del mes	Devuelve el número del día del mes de Fecha

Funciones de fecha

Función	Nombre Función	Valor que devuelve
DAYOFWEEK(Fecha)	Día de la semana	Devuelve el número del día de la semana de Fecha (1.Domingo, 2:Lunes.....7:Sábado)
DAYOFYEAR(Fecha)	Día del año	Devuelve el número de día del año de Fecha (de 1 a 366)
WEEKOFYEAR(Fecha)	Semana	Devuelve el número de semana de Fecha (de 1 a 53)
MONTH(Fecha)	Mes	Devuelve el número de mes de Fecha (de 1 a 12)
YEAR(Fecha)	Año	Devuelve el número de año con 4 dígitos de Fecha (de 0000 a 9999)
HOUR(Tiempo)	Hora	Devuelve la hora de Tiempo (de 0 a 23)
MINUTE(Tiempo)	Minutos	Devuelve los minutos de Tiempo (de 0 a 59)
SECOND(Tiempo)	Segundos	Devuelve los segundos de Tiempo (de 0 a 59)
CURDATE()		Devuelve la fecha actual con el formato 'YYYYMM- DD'
CURTIME()		Devuelve la hora actual con el formato 'HH:MM:SS'
SYSDATE()		Devuelve la fecha y la hora actual YYYY-MM-DD H:MM:SS'

Conversión de fechas a otro tipo de datos

Función	Valor que devuelve
DATE_FORMAT(*Fecha*,*Formato*)	Devuelve una cadena de caracteres con la Fecha con el Formato especificado. El formato es una cadena de caracteres que incluye las siguientes máscaras: **Mascara Descripción** %a Abreviatura (3 letras) del nombre del día de la semana %b Abreviatura (3 letras) del nombre mes %c Número del mes (1 a 12) %e Número del día del mes (0 a 31) %H Número de la hora en formato 24 horas (00 a 23) %h Número de la hora en formato 12 horas (01 a 12) %i Número de minutos (00 a 59) %j Número del día del año (001 a 366) %M Nombre del mes %m Número de mes (01 a 12)

Conversión de fechas a otro tipo de datos

Función	Valor que devuelve
DATE_FORMAT(*Fecha*,*Formato*)	%p Am o PM %r Hora en formato 12 horas (hh:mm seguido de AM o PM) %s Número de segundos (00 a 59) %T Hora en formato 24 horas (hh:mm:ss) %u Número de semana en el año (00 a 53) %W Nombre del día de la semana %w Número del día de la semana (0:domingo a 6:Sábado) %Y Número de año con cuatro dígitos %y Número de año con dos dígitos

Funciones de comparación

Función	Nombre Función	Valor que devuelve
GREATEST(lista de valores)	Mayor de la lista	Devuelve el valor más grande de una lista de columnas o expresiones de columna
LEAST(lista de valores)	Menor de la lista	Devuelve el valor más pequeño de una lista de columnas o expresiones de columna
IFNULL(exp1, exp2)	Conversión de nulos	Si exp1 es nulo devuelve exp2, sino devuelve exp1
ISNULL(exp)	Comprobación de nulo	Devuelve 1(True) si exp es NULL y 0 (False) en caso contrario
STRCMP(cad1,cad2)	Comparación de cadenas	Devuelve 0(True) si cad1 y cad2 son iguales, -1 (False) si la primera es menor y 1 si es mayor la segunda. Si alguna de ellas es nula

5.- FUNCIONES

Otras funciones

Función	Nombre Función	Valor que devuelve
DATABASE()	Base de datos	Nombre de la base de datos actual
USER()	Usuario	Devuelve el usuario y el host de la sesión usuario@host
VERSION()	versión	Devuelve una cadena indicando la versión que estamos Utilizando

En SQL la ausencia de valor se expresa como valor nulo (NULL). Es importante ser conscientes de que esta ausencia de valor o valor nulo no equivale en modo alguno al valor 0 o a una cadena de caracteres vacía.

Hay que tener cuidado al operar con columnas que pueden contener valores nulos, pues la lógica cambia. Vamos a ver como se trabaja con estos valores nulos.

- Si realizamos operaciones aritméticas hay que tener en cuenta que cualquier expresión aritmética que contenga algún valor nulo dará como resultado un valor nulo. Así, por ejemplo, si intentamos visualizar la expresión formada por las columnas SALARIO + COMISION de la tabla empleados la salida será similar a la siguiente:
  - mysql> SELECT apellido, salario, comision, salario + comision
    -> FROM empleados;

6- VALORES NULOS (NULL)

APELLIDO	SALARIO	COMISIÓN	SALARIO + COMISIÓN
ALONSO	1400.00	400.00	1800.00
MARTÍN	1500.00	1600.00	3100.00
CALVO	1800.00	0.00	1800.00
LÓPEZ	1350.00	NULL	NULL
GARRIDO	3850.00	NULL	NULL

Observamos que la expresión SALARIO + COMISION retornará un valor nulo siempre que alguno de los valores sea nulo incluso aunque el otro no lo sea. También podemos observar que el valor 0 en la comisión retorna el valor calculado de la expresión.

6- VALORES NULOS (NULL)

Si comparamos expresiones que contienen el valor nulo con otro valor nulo el resultado no es ni mayor ni menor ni igual. En SQL un valor nulo ni siquiera es igual a otro valor nulo tal y como podemos ver en el siguiente ejemplo:

- mysql> SELECT emp_no, apellido,oficio,fecha_alta,comision,salario
  -> FROM empleados
  -> WHERE comision = NULL;
- No nos mostrará nada

La explicación es que un valor nulo es indeterminado, y por tanto, no es igual ni distinto de otro valor nulo. Cuando queremos comprobar si un valor es nulo emplearemos el operador IS NULL (o IS NOT NULL para comprobar que es distinto de nulo):

- mysql> SELECT emp_no, apellido,oficio,fecha_alta,comision,salario
  -> FROM empleados
  -> WHERE comision IS NULL;

6- VALORES NULOS (NULL)

EMP_Nº	APELLIDO	OFICIO	FECHA_ALTA	COMISIÓN	SALARIO
7521	LÓPEZ	EMPLEADO	1981-05-08	NULL	1350.00
7698	GARRIDO	DIRECTOR	1981-06-09	NULL	3850.00

Los valores nulos en muchas ocasiones pueden representar un problema, especialmente en columnas que contienen valores numéricos. Para evitar estos problemas y asegurarnos de la ausencia de valores nulos en una columna ya vimos que existe una restricción NOT NULL (es una orden de definición de datos) que impide que se incluyan valores nulos en una columna.

6- VALORES NULOS (NULL)

En caso de que permitamos la existencia de valores nulos hay que evitar estos problemas y utilizar la función IFNULL (que veremos en detalle más adelante) que se utiliza para devolver un valor determinado en el caso de que el valor del argumento sea nulo. Así nos aseguramos en las operaciones aritméticas que no hay nulos con los que operar. Por ejemplo, si queremos sumar el *salario* + *comision* debemos utilizar IFNULL (*comision,*0) retornará 0 cuando el valor de comisión sea nulo y sumaremos un 0 a *salario*.

- mysql> SELECT apellido, salario, comision,

  -> salario + IFNULL(comision,0) "SALARIO TOTAL“

  -> FROM empleados;

APELLIDO	SALARIO	COMISIÓN	SALARIO TOTAL
ALONSO	1400.00	400.00	1800.00
MARTÍN	1500.00	1600.00	3100.00
CALVO	1800.00	0.00	1800.00
LÓPEZ	1350.00	NULL	1350.00
GARRIDO	3850.00	NULL	3850.00

6- VALORES NULOS (NULL)

Una expresión es un conjunto de variables, constantes o literales, funciones, operadores y paréntesis. Los paréntesis sirven para variar el orden de prioridad y sólo son obligatorios en ese caso. Un caso especial de expresión es lo que llamamos condición. Condición es un expresión cuyo resultado es V*erdadero/Falso/Nulo (*True/False/Null*)*

Las sentencias SQL pueden incluir expresiones y condiciones con nombres de columnas y literales. Ejemplos de expresiones:

- SALARIO + COMISION -> Devuelve un valor numérico como resultado de sumar al salario del empleado la comisión correspondiente.
- EMPLEADOS.DEPT_NO = DEPARTAMENTOS.DEPT_NO -> Devuelve verdadero o falso dependiendo de que el número de departamento del empleado seleccionado coincida con el número de departamento del departamento seleccionado.

7- EXPRESIONES Y CONDICIONES

- FECHA_AL BETWEEN ‘1980-01-01’ AND ‘1982-10-01’-> Devuelve verdadero si la fecha de alta del empleado seleccionado se encuentra entre las dos fechas especificadas.
- COMISION IS NULL -> dará como resultado verdadero si la comisión del empleado seleccionado no tiene ningún valor.

Por ejemplo la siguiente sentencia visualizará el apellido, la fecha de alta, el salario y la suma del salario más una gratificación de 1.000 euros

- mysql> SELECT apellido, fecha_alta, salario,
  -> salario + 1000 "SALARIO CON GRATIFICACION“
  -> FROM EMPLEADOS;

7- EXPRESIONES Y CONDICIONES

Hay unas reglas de sintaxis que se deben respetar. Un error relativamente frecuente consiste en utilizar expresiones del tipo: 1000 >= SALARIO <= 2000

Este tipo de expresiones no es correcto y no producirá el resultado deseado, ya que al evaluar la primera parte de la expresión se sustituirá por un valor lógico de tipo true/false/null y este resultado no puede compararse con un valor numérico. La expresión correcta sería:

- SALARIO BETWEEN 1000 AND 2000,

O bien:

- SALARIO >= 1000 AND SALARIO <= 2000.

- mysql> SELECT apellido, fecha_alta, salario,
  -> salario + 1000 "SALARIO CON GRATIFICACION“
  -> FROM EMPLEADOS;

APELLIDO	FECHA_ALTA	SALARIO	SALARIO CON GRATIFICACIÓN
ALONSO	1981-02-23	1400.00	2400.00
MARTÍN	1981-09-28	1500.00	2500.00
CALVO	1981-09-08	1800.00	2800.00
LÓPEZ	1981-05-09	1350.00	2350.00
GARRIDO	1981-05-01	3850.00	4850.00