
Tema 2. ELEMENTOS DEL LENGUAJE

Introducción.

Hay dos cuestiones a tener en cuenta a la hora de abordar esta unidad:

1º Se trata de una guía para que sirva de referencia o de consulta cuando se necesite a lo largo del curso.

2º En esta unidad, al igual que en la última, se abordan cuestiones que, aunque están definidas por el estándar ANSI/ISO SQL, no están asumidas al 100% por todos los fabricantes. Por tanto, pueden existir ligeras diferencias de algunos productos con algunas de las especificaciones que se aquí se exponen.

Tipos de datos.

Las columnas de la base de datos almacenan valores que pueden ser de diversos tipos: numérico, carácter, fecha, etcétera. A continuación se indican algunos de los tipos más utilizados.

- CHAR (*longitud*) se utiliza para guardar cadenas de caracteres de longitud fija especificada entre paréntesis. El espacio no utilizado se rellena con blancos.
- VARCHAR (*longitud*) almacena cadenas de caracteres de longitud variable cuyo límite máximo es especifica como *longitud*.
- NUMBER(*escala, precisión*) se utiliza para guardar datos numéricos. La escala indica el número total de dígitos y la precisión el número de posiciones decimales.

En Access no se indica ni escala ni precisión. Por defecto crea un tipo Numérico Doble. Podremos indicar INTEGER, REAL, DOUBLE, BYTE.

- DATE puede almacenar fechas. En algunos SGDBR también se puede almacena la hora en este tipo de datos.

La mayoría de los productos incluyen tipos de datos extendidos e incluso algunos productos ofrecen la posibilidad de que el usuario defina sus propios tipos. Todos estos tipos y posibilidades aparecen documentados en las especificaciones del producto, y escapan del objetivo de este curso

Identificadores.

Son nombres que sirven para identificar objetos de la base de datos: usuarios, tablas, columnas. El estándar define que pueden tener hasta 18 caracteres empezando con un carácter alfabético y continuando con caracteres numéricos y alfabéticos.

En la práctica, algunos productos no permiten nombres de usuario de más de ocho caracteres, pudiendo incluir hasta 30 ó más en los nombres de tablas y columnas.

Los ejemplos que aparecen en este curso se corresponden a la notación utilizada por ORACLE y se ajustan a las especificaciones del estándar ANSI/ISO SQL.

Operadores y expresiones.

Las sentencias SQL pueden incluir expresiones constituidas por nombres de columnas, constantes, funciones y operadores.

Por ejemplo la siguiente sentencia visualizará el apellido, la fecha de alta, el salario y la suma del salario con un complemento o gratificación de 100000 Ptas. de todos los empleados.

```
SQL> SELECT apellido, fecha_alta, salario, salario + 100000 FROM empleados;
```

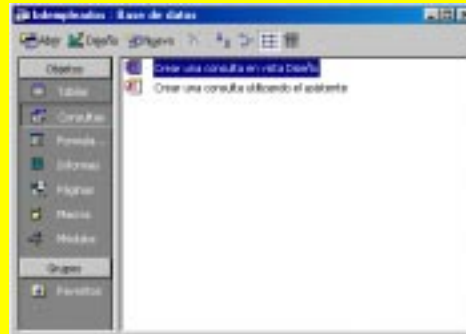
APELLIDO	FECHA_AL	SALARIO	SALARIO+100000
REY	17/11/81	600000	700000
GARRIDO	01/05/81	385000	485000
MARTINEZ	09/06/81	245000	345000
ALONSO	20/02/81	140000	240000
LOPEZ	08/05/81	135000	235000
MARTIN	28/09/81	150000	250000
CALVO	08/09/81	180000	280000
GIL	06/05/82	335000	435000
JIMENEZ	24/03/83	140000	240000

9 filas seleccionadas.

Para probar las consultas en Access seguiremos los siguientes pasos:

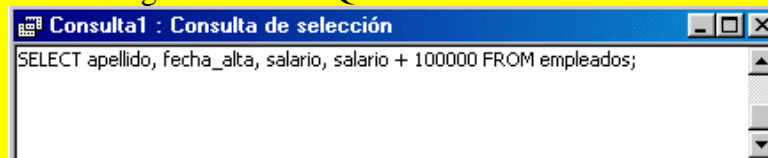
Abrir la base de datos haciendo doble clic sobre ella. Aparece la ventana de la Base de datos, a continuación elegimos el Objeto *Consultas* y doble clic en *Crear una consulta en vista Diseño*. Ver Figura1.

Figura1. Vista de la base de datos. Crear una consulta.



Cerramos la ventana *Mostrar Tabla*, a continuación abrimos la *vista SQL* pulsando al botón **SQL** de la barra de herramientas. Aparece la ventana para consultas SQL, en esa ventana escribimos la orden SELECT. Ver Figura 2:

Figura 2. Vista SQL. Consulta de selección.



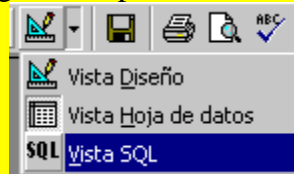
Para ver el resultado de la consulta pulsamos el botón *Ejecutar consulta* (icono de un signo de exclamación) de la barra de herramientas, o al botón *Vista hoja de datos* (icono de una hoja), aparece el resultado de la SELECT, en la vista de hoja de datos. Ver Figura 3:

Figura 3. Vista de hoja de datos. Resultado de la consulta.

	APELLIDO	FECHA_ALTA	SALARIO	Expr1
▶	ALONSO	20/02/81	140000	240000
	LOPEZ	08/05/81	135000	235000
	MARTIN	28/09/81	150000	250000
	GARRIDO	01/05/81	385000	485000
	MARTINEZ	09/06/81	245000	345000
	REY	17/11/81	600000	700000
	CALVO	08/09/81	180000	280000
	GIL	06/05/82	335000	435000
	JIMENEZ	24/03/83	140000	240000
*				

Para volver a la vista SQL desplegamos el botón *Vista* de la barra de herramientas y elegimos *Vista SQL*. Ver figura 4.

Figura 4. Opciones de vistas.




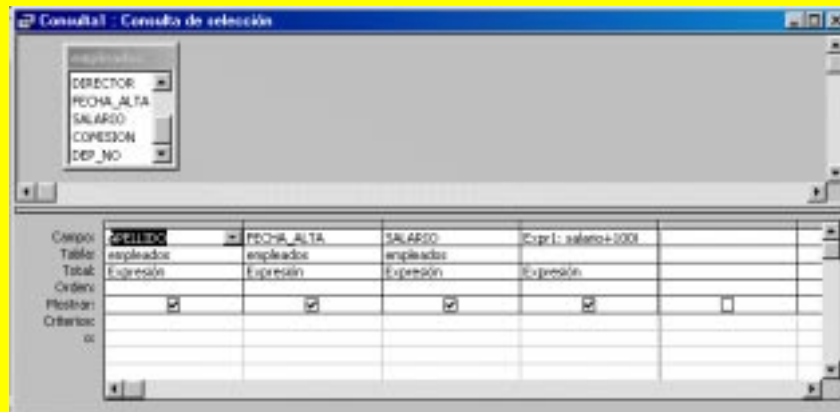
Si pulsamos el botón *Vista diseño* , sin desplegar la lista, aparece la vista de diseño de la consulta, ver Figura 5. Esta es otra forma de realizar consultas sobre una tabla, sólo basta con seleccionar el campo y arrastrarlo a las columnas inferiores.

Figura 5. Vista diseño de consultas.




En este curso todas las consultas las realizaremos en modo *vista SQL*.

Bueno pues si hemos entrado en la *vista de diseño* y luego volvemos a la *vista SQL* observamos que la SELECT ha cambiado automáticamente y aparece:

```
SELECT empleados.APELLIDO, empleados.FECHA_ALTA,  
empleados.SALARIO, salario+100000 AS Expr1  
FROM empleados;
```

Vemos que cada columna va acompañada del nombre de la tabla y Access además asigna un alias a las expresiones (un nombre asociado para identificarlas). Estos cambios los hace de forma automática, solo si cambiamos de la *Vista diseño* a la *Vista SQL*, o viceversa.

Si pulsamos al botón *Guardar*  Access guarda la consulta para una posterior utilización. Al guardarla la damos un nombre.

Constantes.

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. (Ej.: 34.345E-8).

Constantes de cadena.

Consisten en una cadena de caracteres encerrada entre comillas simples. (Ej.: 'Hola Mundo').

En Access las constantes de cadena se pueden definir indistintamente utilizando la comilla simple o la doble. .

Constantes de fecha.

En realidad las constantes de fecha, en Oracle y otros productos que soportan este tipo, se escriben como constantes de cadena sobre las cuales se aplicarán las correspondientes funciones de conversión (ver TO_DATE en el epígrafe *de funciones de conversión* de este mismo capítulo) o bien, el gestor de la base de datos realizará una conversión automática de tipo. (Ej.: '27-SEP-1997').

En Access las constantes de fecha se definen entre el carácter #fecha#. Por ejemplo #08-SEP-81#.

Existe una gran cantidad de formatos aplicables a estas constantes (americano, europeo, japonés, etcétera) . Algunos productos como Oracle pueden trabajar también con FECHA Y HORA en distintos formatos.

Operadores aritméticos.

Se emplean para realizar cálculos. Son los ya conocidos: (+ , - , * , /).

Devuelven un valor numérico como resultado de realizar los cálculos indicados.

Algunos de ellos se pueden utilizar **también con fechas**:

f1 - f2 Devuelve el número de días que hay entre las fechas *f1* y *f2*.

f + n Devuelve una fecha que es el resultado de sumar *n* días a la fecha *f*.

f – n Devuelve una fecha que es el resultado de restar n días a la fecha f .

Operadores de concatenación:

Para unir dos o más cadenas se utiliza el operador de concatenación ||

Ej.: 'buenos' || 'días' daría como resultado 'buenosdías'

En Access para concatenar cadenas se utiliza el signo +, no reconoce ||

Operadores de comparación:

Igual	=
Distinto	!= En Access se utiliza <>
Menor que	<
Mayor que	>
Menor o igual	<=
Mayor o igual	>=
Otros operadores	IS NULL, BETWEEN, LIKE, IN, etcétera

Las expresiones formadas con operadores de comparación dan como resultado un valor de tipo *verdadero/falso* (*true/false*).

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.

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'.

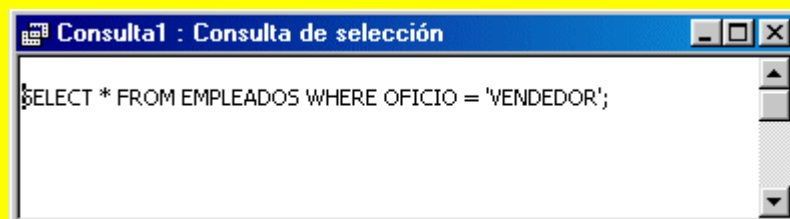
Lenguaje SQL

```
SQL> SELECT * FROM EMPLEADOS WHERE OFICIO = 'VENDEDOR';
```

EMP_NO	APELLIDO	OFICIO	DIRECTOR	FECHA_AL	SALARIO	COMISION	DEP_NO
7499	ALONSO	VENDEDOR	7698	20/02/81	140000	40000	30
7654	MARTIN	VENDEDOR	7698	28/09/81	150000	160000	30
7844	CALVO	VENDEDOR	7698	08/09/81	180000	0	30

Para probarlo en Access, desde la Vista SQL, borrar la sentencia anterior y copiar la nueva sentencia SELECT, ver figura 6:

Figura 6. Vista SQL. Nueva consulta.





Pulsar el botón *Ejecutar consulta*  o al botón *Vista hoja de datos*  para ver el resultado. Ver Figura 7.

Figura 7. Vista hoja de datos. Nueva consulta.

EMP_NO	APELLIDO	OFICIO	DIRECTOR	FECHA_ALTA	SALARIO	COMISION	DEP_NO
7499	ALONSO	VENDEDOR	7698	20/02/81	140000	40000	30
7654	MARTIN	VENDEDOR	7698	28/09/81	150000	160000	30
7844	CALVO	VENDEDOR	7698	08/09/81	180000	0	30

Nota sobre la utilización de valores nulos

En SQL la ausencia de valor se expresa como valor nulo (NULL). Esta ausencia de valor o valor nulo **no equivale** en modo alguno al valor 0.

Cualquier expresión aritmética que contenga algún valor nulo retornará 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:

```
SQL> SELECT APELLIDO, SALARIO, COMISION, SALARIO + COMISION
      FROM EMPLEADOS;
```

APELLIDO	SALARIO	COMISION	SALARIO+COMISION
REY	600000		
GARRIDO	385000		
MARTINEZ	245000		

Lenguaje SQL

ALONSO	140000	40000	180000
LOPEZ	135000		
MARTIN	150000	160000	310000
CALVO	180000	0	180000
GIL	335000		
JIMENEZ	140000		

9 filas seleccionadas.

Para probarlo en Access hacer lo mismo que en el caso anterior. Desde la Vista SQL, borrar la sentencia anterior y copiar la nueva sentencia SELECT.

En el ejemplo anterior 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.

En SQL un valor nulo ni siquiera es igual a otro valor nulo tal como podemos apreciar en el siguiente ejemplo:

```
SQL> SELECT * FROM EMPLEADOS WHERE COMISION = NULL;
```

ninguna fila seleccionada

Probar la SELECT en Access desde la vista SQL.

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):

```
SQL> SELECT * FROM EMPLEADOS WHERE COMISION IS NULL;
```

EMP_NO	APELLIDO	OFICIO	DIRECTOR	FECHA_AL	SALARIO	COMISION	DEP_NO
7839	REY	PRESIDENTE		17/11/81	600000		10
7698	GARRIDO	DIRECTOR	7839	01/05/81	385000		30
7782	MARTINEZ	DIRECTOR	7839	09/06/81	245000		10
7521	LOPEZ	EMPLEADO	7782	08/05/81	135000		10
7876	GIL	ANALISTA	7782	06/05/82	335000		20
7900	JIMENEZ	EMPLEADO	7782	24/03/83	140000		20

6 filas seleccionadas.

Probar la SELECT en Access desde la vista SQL.

Como acabamos de ver los valores nulos en muchas ocasiones pueden representar un problema, especialmente en columnas que contienen valores numéricos. Para evitar estos problemas se suele utilizar:

- La restricción NOT NULL (es una orden de definición de datos) que impide que se incluyan valores nulos en una columna.
- La función NVL (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. Por ejemplo NVL(COMISION, 0) retornará 0 cuando el valor de comisión sea nulo. (En Access se llama NZ)

Operadores logicos: AND, OR y NOT.

Ya hemos indicado que los operadores de comparación devuelven un valor lógicos de tipo verdadero/falso (*true/false*). En ocasiones se necesita trabajar con varias expresiones de comparación (por ejemplo cuando queremos formar una condición búsqueda que cumpla dos condiciones, etcétera) en estos casos debemos recurrir a los operadores lógicos AND, OR y NOT .

Supongamos que queremos consultar los empleados cuyo OFICIO = 'VENDEDOR' y que además su SALARIO > 150000. En este caso emplearemos el operador lógico **AND**. Este operador **devolverá el valor true cuando los dos operandos o expresiones son verdaderas**. Simplificando podemos decir que se utiliza cuando queremos que se cumplan las dos condiciones.

Ejemplo:

```
SQL> SELECT * FROM EMPLEADOS WHERE OFICIO = 'VENDEDOR' AND SALARIO > 150000;
```

EMP_NO	APELLIDO	OFICIO	DIRECTOR	FECHA_AL	SALARIO	COMISION	DEP_NO
7844	CALVO	VENDEDOR	7698	08/09/81	180000	0	30

Probar la SELECT en Access desde la vista SQL.

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 true cuando alguno de los dos operandos o expresiones es verdadero** (cuando se cumple la primera condición, o la segunda o ambas).

Ejemplo:

```
SQL> SELECT * FROM EMPLEADOS WHERE OFICIO = 'VENDEDOR' OR SALARIO > 150000;
```

EMP_NO	APELLIDO	OFICIO	DIRECTOR	FECHA_AL	SALARIO	COMISION	DEP_NO
7839	REY	PRESIDENTE		17/11/81	600000		10
7698	GARRIDO	DIRECTOR	7839	01/05/81	385000		30
7782	MARTINEZ	DIRECTOR	7839	09/06/81	245000		10
7499	ALONSO	VENDEDOR	7698	20/02/81	140000	40000	30
7654	MARTIN	VENDEDOR	7698	28/09/81	150000	160000	30
7844	CALVO	VENDEDOR	7698	08/09/81	180000	0	30
7876	GIL	ANALISTA	7782	06/05/82	335000		20

Probar la SELECT en Access desde la vista SQL.

Lenguaje SQL

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:

```
SQL> SELECT * FROM EMPLEADOS WHERE NOT OFICIO = 'VENDEDOR';
```

EMP_NO	APELLIDO	OFICIO	DIRECTOR	FECHA_AL	SALARIO	COMISION	DEP_NO
7839	REY	PRESIDENTE		17/11/81	600000		10
7698	GARRIDO	DIRECTOR	7839	01/05/81	385000		30
7782	MARTINEZ	DIRECTOR	7839	09/06/81	245000		10
7521	LOPEZ	EMPLEADO	7782	08/05/81	135000		10
7876	GIL	ANALISTA	7782	06/05/82	335000		20
7900	JIMENEZ	EMPLEADO	7782	24/03/83	140000		20

Probar la SELECT en Access desde la vista SQL.

Observamos en el ejemplo anterior que han sido seleccionadas aquellas filas en las que **no** se cumple la condición de que el oficio sea vendedor.

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.

Ejemplos:

```
SQL> SELECT * FROM EMPLEADOS WHERE NOT OFICIO = 'VENDEDOR' AND SALARIO > 150000;
```

EMP_NO	APELLIDO	OFICIO	DIRECTOR	FECHA_AL	SALARIO	COMISION	DEP_NO
7839	REY	PRESIDENTE		17/11/81	600000		10
7698	GARRIDO	DIRECTOR	7839	01/05/81	385000		30
7782	MARTINEZ	DIRECTOR	7839	09/06/81	245000		10
7876	GIL	ANALISTA	7782	06/05/82	335000		20

```
SQL> SELECT * FROM EMPLEADOS WHERE OFICIO = 'VENDEDOR' AND SALARIO > 150000 OR DEP_NO = 20;
```

EMP_NO	APELLIDO	OFICIO	DIRECTOR	FECHA_AL	SALARIO	COMISION	DEP_NO
7844	CALVO	VENDEDOR	7698	08/09/81	180000	0	30
7876	GIL	ANALISTA	7782	06/05/82	335000		20
7900	JIMENEZ	EMPLEADO	7782	24/03/83	140000		20

Probar la SELECT en Access desde la vista SQL.

En todo caso deberemos tener en cuenta la **prioridad o precedencia del operador** ya que puede afectar al resultado de la operación.

A continuación se detallan las tablas de valores de los operadores lógicos NOT, AND y OR, teniendo en cuenta todos los posibles valores incluida la ausencia de valor (NULL).

AND	TRUE	FALSE	NULL
TRUE	TRUE	FALSE	NULL
FALSE	FALSE	FALSE	FALSE
NULL	NULL	FALSE	NULL

OR	TRUE	FALSE	NULL
TRUE	TRUE	TRUE	TRUE
FALSE	TRUE	FALSE	NULL
NULL	TRUE	NULL	NULL

NOT	TRUE	FALSE	NULL
	FALSE	TRUE	NULL

Podemos establecer:

- El operador AND devolverá *true* cuando los dos operandos sean verdaderos, y *false* cuando algún operando sea falso; en el caso de que ambos operandos tengan valor Null devolverá Null y cuando ningún operando es False y algún operando es Null también devolverá Null.
- El operador OR devolverá *true* cuando alguno de los operandos sea verdadero (con independencia de que el otro sea verdadero, falso o nulo); *false* cuando los dos operandos sean falsos; y *null* en los demás casos (cuando ningún operando sea verdadero y alguno sea nulo).
- El operador NOT devuelve *true* cuando el operando es falso, y *false* cuando el operando es *true*; cuando el operando es nulo devuelve *null*.

Precedencia o prioridad en los operadores.

El orden de precedencia o prioridad de los operadores determina el orden de evaluación de los operandos de una expresión.

Por ejemplo, la siguiente expresión: $12 + 24 / 6$

Dará como resultado 16

Ya que la división se realizará primero.

La tabla siguiente muestra los operadores disponibles agrupados y ordenados de mayor a menor por su orden de precedencia.

Lenguaje SQL

Prioridad	Operador	Operacion
1º	**, NOT	exponenciación, negación
2º	*, /	multiplicación, división
3º	+, -,	suma, resta, concatenación. Para concatenar Access se utiliza el signo +.
4º	=, !=, <, >, <=, >=, IS NULL, LIKE, BETWEEN, IN	Comparación. En Access la comparación != es <>
5º	AND	conjunción
6º	OR	inclusión

Esta es la prioridad establecida por defecto. Se puede cambiar utilizando paréntesis.

En la expresión anterior, si queremos que la suma se realice antes que la división, lo indicaremos:

$$(12 + 24) / 6$$

En este caso el resultado será: 6

Los operadores que se encuentran en el mismo grupo tienen la misma precedencia. En estos casos no se garantiza el orden de evaluación. Si queremos que se evalúen en algún orden concreto deberemos utilizar paréntesis.

NOTA: un error relativamente frecuente consiste en utilizar expresiones del tipo:

100000 >= SALARIO <= 200000

Este tipo de expresiones es ilegal y provocará un error ya que al evaluar la primera parte de la expresión se sustituirá por un valor lógico de tipo *true/false* y este resultado no puede compararse con un valor numérico. La expresión correcta sería:

SALARIO BETWEEN 100000 AND 200000

O bien:

SALARIO >= 100000 AND SALARIO <= 200000

Ejemplos de expresiones:

SALARIO + COMISION -> Devuelve un valor numérico como resultado de sumar al salario del empleado la comisión correspondiente.

Lenguaje SQL

EMPLEADOS.DPT_NO = DEPARTAMENTOS.DPT_NO -> Devuelve verdadero o falso dependiendo de que el número de departamento del empleado seleccionado coincida con el número de departamento del departamento seleccionado.

FECHA_AL BETWEEN '01/01/80' AND '01/10/82' -> 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.

Funciones predefinidas.

En SQL disponemos de funciones predefinidas que devuelven un valor en función de un argumento que se pasa en la llamada. Algunas de estas funciones no se pueden utilizar en Access o se llaman de otra manera.

Ejemplos:

Llamada a la función	Valor devuelto	Explicación.
ABS(-5)	5	el valor absoluto del número.
ROUND(8.66, 0)	9	el número redondeando a cero decimales
TRUC(8.66, 0). En Acces esta función no se encuentra.	8	el número truncando los decimales.
SIGN(8.88). En Acces se llama SGN()	1	ya que el número es mayor de 0
INITCAP('hola'). En Acces esta función no se encuentra.	'Hola'	cadena con el primer carácter en mayúsculas.
UPPER('hola'). En Acces se llama UCASE()	'HOLA'	cadena en mayúsculas
SUBSTR('hola', 2, 2). En Acces se llama MID()	'ol'	subcadena tomando dos posiciones desde la posición 2.
SYSDATE. En Acces se llama DATE()	01/02/00	la fecha del sistema
USER. En Acces se llama CURRENTUSER()	CURSOSQL	el nombre del usuario en nuestro caso el que se indica.

Lenguaje SQL

Cabe subrayar que la función no modifica el valor del argumento sino que devuelve un valor creado a partir del argumento que se le pasa en la llamada.

A continuación se indican las funciones predefinidas más utilizadas:

Funciones numéricas o aritméticas:

Función	Valor que devuelve.
ABS(<i>num</i>)	Valor absoluto de <i>num</i> .
MOD(<i>num1</i>, <i>num2</i>). En Access se pone: <i>num1 MOD num2</i>	Resto de la división entera.
POWER(<i>num1</i>, <i>num2</i>). En Access se pone: <i>num1 ^ num2</i> .	Devuelve <i>num1</i> elevado a <i>num2</i> .
ROUND(<i>num1</i>, <i>num2</i>)	Devuelve <i>num1</i> redondeado a <i>num2</i> decimales.
SIGN(<i>num</i>). En Access se llama SGN() .	Si <i>num</i> < 0 devuelve -1, si <i>num</i> = 0 devuelve 0, si <i>num</i> > 0 devuelve 1.
SQRT(<i>num</i>). En Access se llama SQR .	Raíz cuadrada de <i>num</i>
TRUNC(<i>num1</i>, <i>num2</i>). No se encuentra en Access	Devuelve <i>num1</i> truncado a <i>num2</i> decimales. Si se omite <i>num2</i> , a 0 decimales.

Funciones de caracteres:

Función	Valor que devuelve.
ASCII(<i>c1</i>). En Access se llama ASC .	Código ASCII del carácter <i>c1</i> .
CHR(<i>num</i>)	Carácter correspondiente al código indicado por <i>num</i> del juego de caracteres utilizado.
CONCAT(<i>c1</i>,<i>c2</i>). Para concatenar en Access utilizamos + o &.	Concatena <i>c1</i> con <i>c2</i> . Es equivalente al operador . En Access. Pondremos <i>c1&c2</i> o también <i>c1+c2</i> .
INITCAP(<i>c1</i>). No se encuentra en Access	Devuelve <i>c1</i> poniendo en mayúscula la primera letra de cada palabra de la cadena y el resto en minúsculas.

Lenguaje SQL

LENGTH(<i>c1</i>) En Access se llama LEN() .	Longitud de <i>c1</i> .
LOWER(<i>c1</i>) En Access se llama LCASE .	La cadena <i>c1</i> en minúsculas.
UPPER(<i>c1</i>) En Access se llama UCASE .	La cadena <i>c1</i> en mayúsculas.
LPAD(<i>c1</i>, <i>n</i>, <i>c2</i>) No se encuentra en Access	<i>c1</i> se visualiza con longitud <i>n</i> y justificado a la derecha. Si <i>c1</i> < <i>n</i> , <i>c2</i> es la cadena con la que se rellena por la izda.
RPAD(<i>c1</i>, <i>n</i>, <i>c2</i>) No se encuentra en Access	Igual que LPAD pero por la derecha.
LTRIM(<i>c1</i>)	Suprime blancos a la izquierda de <i>c1</i> .
RTRIM(<i>c1</i>)	Suprime blancos a la derecha de <i>c1</i> .
SUBSTR(<i>c1</i>, <i>n</i>, <i>m</i>) En Access se llama MID()	Devuelve una subcadena a partir de <i>c1</i> comenzando en la posición <i>n</i> tomando <i>m</i> caracteres.
TRANSLATE(<i>c1</i>, <i>c2</i>, <i>c3</i>) No se encuentra en Access	Devuelve la cadena <i>c1</i> con cada ocurrencia de <i>c2</i> que contenga reemplazada por <i>c3</i> .

Funciones de fecha:

Función	Valor que devuelve.
ADD_MONTHS(<i>f</i>, <i>n</i>) En Access se utiliza: DateAdd("y",<i>n</i>, <i>f</i>). Suma <i>n</i> años DateAdd("m",<i>n</i>,<i>f</i>). Suma <i>n</i> meses DateAdd("d",<i>n</i>,<i>f</i>). Suma <i>n</i> días	Incrementa <i>n</i> meses a la fecha <i>f</i> .. En el caso de Access podemos sumar años, días o meses a una fecha dada.
LAST_DAY(<i>f</i>) No se encuentra en Access	Último día del mes de la fecha.
MONTHS_BETWEEN (<i>f1</i>, <i>f2</i>) En Access se utiliza: DateDiff("m",<i>f1</i>,<i>f2</i>). Los meses. DateDiff("d",<i>f1</i>,<i>f2</i>). Los días. DateDiff("yyyy",<i>f1</i>,<i>f2</i>). Los años	Número de meses entre las dos fechas. El resultado puede contener decimales correspondientes a fracciones de mes. En el caso de Access podemos saber el número de días, de años y de meses.

Lenguaje SQL

SYSDATE. En Access se llama DATE().	Fecha actual. También la hora si se especifica en el formato. En Access es NOW() la función que visualiza la hora y la fecha.
--	--

Funciones de conversión:

Se utilizan para pasar datos de un tipo a otro: carácter a número, fecha a carácter, etcétera.

Función	Valor que devuelve.																																								
TO_NUMBER(c1) En Access se llama VAL().	Convierte una cadena a tipo numérico.																																								
TO_CHAR(n, formato)	<div>Devuelve un número en formato char según las siguientes especificaciones de formato:</div> <table><tr><th>Cod.</th><th>Ejemplo</th><th>Descripción</th></tr><tr><td>9</td><td>999</td><td>Cada 9 indica un dígito</td></tr><tr><td>0</td><td>099</td><td>Visualiza ceros a la izquierda</td></tr><tr><td>\$</td><td>\$99</td><td>Antepone el símbolo de \$</td></tr><tr><td>MI</td><td>99MI</td><td>Visualiza un '-' después de un número negativo</td></tr><tr><td>B</td><td>B99</td><td>En lugar de 0 visualiza blancos</td></tr><tr><td>,</td><td>999,999</td><td>Visualiza el separador de los miles</td></tr><tr><td>.</td><td>999,999.99</td><td>Visualiza el punto de separación de decimales</td></tr></table>	Cod.	Ejemplo	Descripción	9	999	Cada 9 indica un dígito	0	099	Visualiza ceros a la izquierda	\$	\$99	Antepone el símbolo de \$	MI	99MI	Visualiza un '-' después de un número negativo	B	B99	En lugar de 0 visualiza blancos	,	999,999	Visualiza el separador de los miles	.	999,999.99	Visualiza el punto de separación de decimales																
Cod.	Ejemplo	Descripción																																							
9	999	Cada 9 indica un dígito																																							
0	099	Visualiza ceros a la izquierda																																							
\$	\$99	Antepone el símbolo de \$																																							
MI	99MI	Visualiza un '-' después de un número negativo																																							
B	B99	En lugar de 0 visualiza blancos																																							
,	999,999	Visualiza el separador de los miles																																							
.	999,999.99	Visualiza el punto de separación de decimales																																							
TO_CHAR(fecha, formato)	<div>Devuelve cadena a partir de una fecha según las siguientes especificaciones de formato:</div> <table><tr><th>Formato</th><th>Descripción</th></tr><tr><td>YYYY</td><td>Cuatro dígitos en el año.</td></tr><tr><td>YY</td><td>Últimos dos dígitos del año (también con tres y uno)</td></tr><tr><td>YEAR</td><td>Año deletreado.</td></tr><tr><td>MM</td><td>Mes en número.</td></tr><tr><td>MONTH</td><td>Mes en letra.</td></tr><tr><td>MON</td><td>Abreviatura tres letras del mes</td></tr><tr><td>DD</td><td>Día del mes</td></tr><tr><td>DDD</td><td>Día del año</td></tr><tr><td>D</td><td>Día de la semana</td></tr><tr><td>DAY</td><td>Nombre del día de la semana</td></tr><tr><td>DY</td><td>Nombre del día con tres letras</td></tr><tr><td>WW</td><td>Semana del año</td></tr><tr><td>W</td><td>Semana del mes</td></tr><tr><td>Q</td><td>Trimestre del año</td></tr><tr><td>AM/PM</td><td>Formato 12 horas indicando AM o PM</td></tr><tr><td>HH24</td><td>Formato 24 horas</td></tr><tr><td>MI</td><td>Minutos</td></tr><tr><td>SS</td><td>Segundos</td></tr><tr><td>SSSS</td><td>Hora en segundos desde las 0 horas del día.</td></tr></table>	Formato	Descripción	YYYY	Cuatro dígitos en el año.	YY	Últimos dos dígitos del año (también con tres y uno)	YEAR	Año deletreado.	MM	Mes en número.	MONTH	Mes en letra.	MON	Abreviatura tres letras del mes	DD	Día del mes	DDD	Día del año	D	Día de la semana	DAY	Nombre del día de la semana	DY	Nombre del día con tres letras	WW	Semana del año	W	Semana del mes	Q	Trimestre del año	AM/PM	Formato 12 horas indicando AM o PM	HH24	Formato 24 horas	MI	Minutos	SS	Segundos	SSSS	Hora en segundos desde las 0 horas del día.
Formato	Descripción																																								
YYYY	Cuatro dígitos en el año.																																								
YY	Últimos dos dígitos del año (también con tres y uno)																																								
YEAR	Año deletreado.																																								
MM	Mes en número.																																								
MONTH	Mes en letra.																																								
MON	Abreviatura tres letras del mes																																								
DD	Día del mes																																								
DDD	Día del año																																								
D	Día de la semana																																								
DAY	Nombre del día de la semana																																								
DY	Nombre del día con tres letras																																								
WW	Semana del año																																								
W	Semana del mes																																								
Q	Trimestre del año																																								
AM/PM	Formato 12 horas indicando AM o PM																																								
HH24	Formato 24 horas																																								
MI	Minutos																																								
SS	Segundos																																								
SSSS	Hora en segundos desde las 0 horas del día.																																								

Para convertir una fecha o un número a cadena Access utiliza la función **FORMAT**.

Format(fecha, formato)
Format(número, formato)

Si no ponemos formato Access devuelve la cadena correspondiente a la fecha o al número. En algunos casos utiliza los mismos formatos que ORACLE. También tiene otros formatos predefinidos como "Long Time". Para representar dígitos Access utiliza el carácter #.

Ejemplos:

Format(Now(), "Long Time") Devuelve la hora actual. Hora larga. "09:00:33 p.m."
 Format(Date(), "Long Date") Devuelve la fecha actual completa: "Martes, 17 de Mayo de 2001"
 Format(#21:6:51#, "h:m:s"). Devuelve "21:6:51"
 Format(#21:6:51#, "hh:mm:ss AMPM"). Devuelve "09:06:51 p.m."
 Format(#12/11/99#, "dddd, d mmm yyyy"). Devuelve: "Sábado, 11 Dic 1999"
 Format(33459.46, "##,##0.000"). Devuelve: " 33,459.460".
 Format(334.9, "###0.00"). Devuelve: "334,90".
 Format(67, "0.00%"). Devuelve: "6700,00%".
 Format("MINÚSCULAS", "<"). Devuelve "minúsculas".
 Format("Ejemplito", ">") . Devuelve "EJEMPLITO".

TO_DATE(fecha, formato) En Acces se llama CDATE(fecha), y la convierte al tipodd/mm/aa.	Devuelve fecha a partir de una cadena según las especificaciones de formato indicadas antes. Ej.: TO_DATE('27-OCT-95', 'DD-MON-YY') Ej en Access: CDate("12 mayo 96"), devuelve 12/5/96
---	--

Otras funciones:

Función	Valor que devuelve.
GREATEST(lista de valores) No se encuentra en Access	Valor más grande de una lista.
LEAST(lista de valores) No se encuentra en Access	Valor más pequeño de una lista.
NVL(exp1, exp2) En Access se llama NZ(exp1, exp2)	Si exp1 es nulo devuelve exp2. Sino devuelve exp1.

Lenguaje SQL

USER En Acces se llama CURRENTUSER()	Devuelve el nombre del usuario actual.
UID No se encuentra en Access	Devuelve el número de identificación del usuario actual. Esta función, igual que la anterior, no tiene argumentos.

En los próximos temas se estudiarán todas estas funciones con diversos ejemplos y ejercicios.

Consideraciones sobre la sintaxis utilizada.

Para especificar la sintaxis el estándar SQL ANSI/ISO utiliza una notación muy precisa y completa pero presenta serios problemas didácticos especialmente para personas que se inician en este lenguaje. Por esta razón hemos utilizado para especificar los formatos una notación más sencilla y asequible primando el aspecto didáctico. A este respecto procede realizar las siguientes puntualizaciones:

- Las palabras reservadas de SQL aparecen en mayúsculas.
- Los nombres de objetos (tablas, columnas, etcétera) suelen aparecer en minúsculas.
- La notación *lista_de_elementos* especifica una lista de elementos separados por comas.
- La barra vertical (|) indica la elección entre dos elementos.
- Las llaves ({ }) indican la elección obligatoria entre varios elementos.
- Los corchetes ([]) encierran un elemento opcional.
- El punto y coma (;) que aparece al final de cada comando, 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 el ;) al servidor.