CONSULTAS SIMPLES

SQL SERVER 2005

Manual de Referencia para usuarios

Salomón Ccance
CCANCE WEBSITE

CONSULTAS SIMPLES

Vamos a empezar por la instrucción que más se utiliza en SQL, la sentencia SELECT. La sentencia SELECT es, con diferencia, la más compleja y potente de las sentencias SQL, con ella podemos recuperar datos de una o más tablas, seleccionar ciertos registros e incluso obtener resúmenes de los datos almacenados en la base de datos. Es tan compleja que la estudiaremos a lo largo de varias unidades didácticas incorporando poco a poco nuevas funcionalidades.

El resultado de una SELECT es una tabla lógica que alberga las filas resultantes de la ejecución de la sentencia.

La sintaxis completa es la siguiente:

```
SELECT sentencia::=[WITH  <expresion_tabla_comun> [,...n]]
          <expresion_consulta>
          [ORDER BY {expression_columna|posicion_columna [ASC|DESC] }
        [ ,...n ]]
          [COMPUTE
         {{AVG|COUNT|MAX|MIN|SUM} (expression)}[ ,...n ] [BY
expression[ ,...n ]]
          ]
          [<FOR clausula_for>]
          [OPTION (<query_hint>[ ,...n ])]
<expresion_consulta> ::=
          {<especificacion_consulta>  | ( < expresion_consulta > ) }
          [ {UNION  [ALL]|EXCEPT|INTERSECT}
              <especificacion_consulta> |  (<expresion_consulta>)
[...n ]
          ]
<especificacion_consulta> ::=
      SELECT [ALL|DISTINCT]
          [TOP expresion [PERCENT] [WITH TIES] ]
          <lista_seleccion>
          [INTO nueva_tabla]
          [FROM { <origen> } [ ,...n ] ]
          [WHERE <condicion_busqueda> ]
          [GROUP BY [ ALL ] expresion_agrupacion [ ,...n ]
          [WITH { CUBE | ROLLUP } ]
          ]
          [HAVING < condicion_busqueda > ]
```

Debido a la complejidad de la sentencia (en la sintaxis anterior no se han detallado algunos elementos), la iremos viendo poco a poco, empezaremos por ver consultas básicas para luego ir añadiendo más cláusulas.

Empezaremos por ver las consultas más simples, basadas en una sola tabla y nos limitaremos a la siguiente sintaxis:

```
SELECT [ALL|DISTINCT]
    [TOP expresion [PERCENT] [WITH TIES]]
    <lista_seleccion>
    FROM  <origen>
    [WHERE  <condicion_busqueda> ]
    [ORDER BY {expression_columna|posicion_columna [ASC|DESC]} [ ,...n
]]
```

ORIGEN DE DATOS FROM

De la sintaxis anterior, el elemento <origen> indica de dónde se va a extraer la información y se indica en la cláusula FROM, es la única cláusula obligatoria. En este tema veremos un origen de datos basado en una sola tabla.

La sintaxis será la siguiente:

```
<origen>::=
      nb_tabla | nb_vista [[ AS ] alias_tabla ]
```

- nb_tabla representa un nombre de tabla.
- nb_vista un nombre de vista.

Tanto para las tablas como para las vistas, podemos hacer referencia a tablas que están en otras bases de datos (siempre que tengamos los permisos adecuados), en este caso tenemos que cualificar el nombre de la tabla, indicando delante el nombre de la base de datos (Lógica) y el nombre del esquema al que pertenece la tabla dentro de la base de datos.

Por ejemplo: MiBase.dbo.MiTabla se refiere a la tabla MiTabla que se encuentra en el esquema dbo de la base de datos MiBase.

Cuando no se definen esquemas, SQL-Server crea uno por defecto en cada base de datos denominado dbo.

Opcionalmente podemos definir un nombre de alias.

Un nombre de alias (alias_tabla) es un nombre alternativo que se le da a la tabla dentro de la consulta.

Si se define un nombre de alias, dentro de la consulta, será el nombre a utilizar para referirnos a la tabla, el nombre original de la tabla ya no tendrá validez.

Se utilizan los nombres de alias para simplificar los nombres de tablas a veces largos y también cuando queremos combinar una tabla consigo misma; ya volveremos sobre los alias de tabla cuando veamos consultas multitabla.

La palabra AS no añade ninguna operatividad, está más por estética.

Podemos escribir:

```
SELECT ...
FROM tabla1          Sacamos los datos de la tabla tabla1

SELECT ...
FROM tabla1 t1       Sacamos  los datos de la tabla tabla1 y le
asignamos un alias de tabla: t1

SELECT ...
FROM tabla1 AS t1    Es  equivalente a la sentencia anterior.
```

Si la tabla o la vista están en otra base de datos del mismo equipo que está ejecutando la instancia de SQL Server, se utiliza el nombre cualificado con el formato nbBaseDatos.nbEsquema.nbTabla.

Si la tabla o la vista están fuera del servidor local en un servidor vinculado, se utiliza un nombre de cuatro partes con el formato nbservidor.catalogo.nbEsquema.nbTabla. Volveremos más adelante sobre

las conexiones remotas.

LA LISTA DE SELECCIÓN

En la lista de selección <lista_seleccion> indicamos las columnas que se tienen que visualizar en el resultado de la consulta.

```
<lista_seleccion> ::=
       {   *
          | {nombre_tabla|nombre_vista|alias_tabla}.*
            | { [{nombre_tabla|nombre_vista|alias_tabla}.]
         {nb_columna|$IDENTITY|$ROWGUID}
         |<expresion>
      }[[AS] alias_columna]
          | alias_columna = <expresion>
       } [ ,...n ]
```

Separamos la definición de cada columna por una coma y las columnas del resultado aparecerán en el mismo orden que en la lista de selección.

Para cada columna del resultado su tipo de datos, tamaño, precisión y escala son los mismos que los de la expresión que da origen a esa columna.

Podemos definir las columnas del resultado de varias formas, mediante:

- Una expresión simple:
  - una referencia a una función.
  - una variable local
  - una constante
  - una columna del origen de datos.
- Una subconsulta escalar, que es otra instrucción SELECT que devuelve un único valor y se evalúa para cada fila del origen de datos (esto no lo veremos de momento).
- Una expresión compleja generada al usar operadores en una o más expresiones simples.
- La palabra clave *.
- La asignación de variables con el formato @variable_local = expresión.
- La palabra clave $IDENTITY.
- La palabra clave $ROWGUID.

COLUMNAS DEL ORIGEN DE DATOS

Cuando queremos indicar en la lista de selección una columna del origen de datos, la especificamos mediante su nombre simple o nombre cualificado. El nombre cualificado consiste en el nombre de la columna precedido del nombre de la tabla donde se encuentra la columna.

Si en el origen de datos hemos utilizado una vista o un nombre de alias, deberemos utilizar ese nombre. Es obligatorio utilizar el nombre cualificado cuando el nombre de la columna aparece en más de una tabla del origen de datos.

Ejemplos de consulta simple.

- Listar nombres, oficinas, y fechas de contrato de todos los empleados:

```
SELECT  nombre, oficina, contrato
FROM  empleados;
```

El resultado seria:

nombre	oficina	contrato
Antonio Viguer	12	1986-10-20
Alvaro Jaumes	21	1986-12-10
Juan Rovira	12	1987-03-01
José González	12	1987-05-19
Vicente Pantalla	13	1988-02-12
Luis Antonio	11	1988-06-14
Jorge Gutiérrez	22	1988-11-14
Ana Bustamante	21	1989-10-12
María Sunta	11	1999-10-12
Juan Victor	NULL	1990-01-13

- Listar una tarifa de productos

```
SELECT  idfab, idproducto, descripcion, productos.precio
FROM  productos;
```

Hemos cualificado la columna precio aunque no es necesario en este caso.

El resultado sería:

Idfab	idproducto	descripcion	precio
aci	41001	arandela	0,58
aci	41002	bisagra	0,80
aci	41003	art t3	1,12
aci	41004	art t4	1,23
aci	4100x	junta	0,26
aci	4100y	extractor	28,88
aci	4100z	mont	26,25
bic	41003	manivela	6,52
bic	41089	rodamiento	2,25

ALIAS DE COLUMNAS

Por defecto, en el encabezado de cada columna del resultado, aparece el nombre de la columna origen, pero esto se puede cambiar definiendo un alias de columna, el alias de columna es un nombre alternativo que se le da a esa columna.

El alias de columna se indica mediante la cláusula AS. Se escribe el nuevo texto tal cual sin comillas siguiendo las reglas de los identificadores.

Ejemplo:

```
SELECT numclie,nombre AS  nombrecliente
FROM clientes;
```

El resultado será :

Numclie	nombrecliente
2101	Luis García Antón
2102	Alvaro Rodríguez
2103	Jaime Llorens

En vez de:

Numclie	nombre
2101	Luis García Antón
2102	Alvaro Rodríguez
2103	Jaime Llorens

La palabra AS es opcional.

```
SELECT numclie,nombre nombrecliente
FROM clientes;
```

Sería equivalente a la consulta anterior

Si queremos incluir espacios en blanco en el nombre lo debemos encerrar entre corchetes.

```
SELECT numclie,nombre AS [nombre cliente]
FROM clientes;
```

Nota importante: Este nombre de alias se podrá utilizar en la lista de selección y en la cláusula ORDER BY pero no en la cláusula WHERE.

FUNCIONES

Existen funciones que podemos utilizar en la lista de selección, e incluso en otras cláusulas que veremos más adelante, como el WHERE. Las principales funciones son las siguientes:

Funciones de fecha:

Función	Descripción
GETDATE	Devuelve la fecha actual.
GETUTCDATE	Devuelve la hora UTC.
DATEPART	Devuelve un entero que corresponde a la parte de la fecha solicitada.
DAY	Devuelve el día de la fecha indicada.
MONTH	Devuelve el mes de la fecha indicada.
YEAR	Devuelve el año de la fecha indicada.
DATENAME	Devuelve una cadena de caracteres que representa el valor de la unidad especificada de una fecha especificada.
DATEADD	Devuelve un valor datetime nuevo que resulta de sumar un intervalo de tiempo a una fecha especificada.>
DATEDIFF	Devuelve el nº de intervalos que hay entre dos fechas.
@@DATEFIRST	Devuelve el primer día de la semana establecido con SET DATEFIRST.
SET DATEFIRST	Establece el primer día de la semana en un número del 1 al 7.

Funciones de cadena:

Función	Descripción
ASCII	Devuelve el valor de código ASCII del carácter situado más a la izquierda de una expresión de caracteres.
CHAR	Devuelve el carácter ASCII del entero indicado.
NCHAR	Devuelve el carácter Unicode del entero indicado.
UNICODE	Devuelve el entero que se corresponde al carácter Unicode indicado.
LEN	Devuelve el total de caracteres de una cadena, excluidos los espacios en blanco finales.
LTRIM	Devuelve una cadena tras quitarle los espacios en blanco iniciales.
RTRIM	Devuelve una cadena tras quitarle los espacios en blanco finales.
LEFT	Devuelve los N últimos caracteres de una cadena.
RIGHT	Devuelve los N primeros caracteres de una cadena.
SUBSTRING	Devuelve parte de una expresión.
LOWER	Devuelve la cadena convertida a minúsculas.
UPPER	Devuelve la cadena convertida a mayúsculas.
REPLACE	Reemplaza una determinada cadena.
STUFF	Elimina el número de caracteres especificado e inserta otro conjunto de caracteres en el punto de inicio indicado.
QUOTENAME	Devuelve una cadena Unicode con los delimitadores agregados para convertirla en un identificador delimitado válido de Microsoft SQL Server 2005.
SPACE	Devuelve una cadena de espacios repetidos.
STR	Devuelve una cadena de caracteres a partir de datos numéricos.
REPLICATE	Repite una cadena N veces.
REVERSE	Devuelve una cadena invertida.
CHARINDEX	Devuelve la posición inicial de la expresión especificada en una cadena de caracteres.
PATINDEX	Devuelve la posición inicial de la primera repetición de un patrón en la expresión especificada, o ceros si el patrón no se encuentra, en todos los tipos de datos de texto y caracteres.

Otras funciones:

Función	Descripción
ROUND	Redondea un valor a la longitud y precisión indicadas.
CAST y CONVERT	Convierten de un tipo de datos a otro de forma explícita.
CASE	Evalúa una lista de condiciones.
ISNULL	Reemplaza el valor NULL por otro especificado.
COALESCE	Devuelve la primera expresión distinta de NULL entre sus argumentos.

COLUMNAS CALCULADAS

Además de las columnas que provienen directamente de la tabla origen, una consulta SQL puede incluir columnas calculadas cuyos valores se evalúan a partir de una expresión.

La expresión puede contener cualquier operador válido (+, -, *, /, &…), cualquier función válida, nombres de columnas del origen de datos, nombres de parámetros o constantes y para combinar varias operaciones se pueden utilizar los paréntesis.

Ejemplos de columnas calculadas:

Listar la ciudad, región y el superávit de cada oficina. Consideraremos el superávit como el volumen de ventas que se encuentran por encima o por debajo del objetivo de la oficina.

```
SELECT  ciudad, region, (ventas-objetivo) AS superavit
FROM  oficinas;
```

El resultado será:

ciudad	region	superavit
Valencia	este	11800,00
Alicante	este	-6500,00
Castellon	este	1800,00
Badajoz	oeste	11100,00
A Coruña	oeste	-11400,00
Madrid	centro	NULL
Madrid	centro	-10000,00
Pamplona	norte	NULL
Valencia	este	-90000,00

De cada producto queremos saber el id de fabricante, id de producto, su descripción y el valor de sus existencias.

```
SELECT  idfab,idproducto,descripcion,(existencias*precio) AS
valoracion
FROM productos;
```

El resultado sería:

Idfab	idproducto	descripcion	valoracion
aci	41001	arandela	160,66
aci	41002	bisagra	133,60
aci	41003	art t3	231,84
aci	41004	art t4	170,97
aci	4100x	junta	9,62
aci	4100y	extractor	722,00
aci	4100z	mont	735,00
bic	41003	manivela	19,56
bic	41089	rodamiento	175,50

Listar el nombre, mes y año del contrato de cada vendedor.

```
SELECT nombre, MONTH(contrato) AS  [Mes de contrato], YEAR(contrato)
AS [Año de contrato]
FROM   empleados;
```

El resultado será:

Nombre	Mes de contrato	Año de contrato
Antonio Viguer	10	1986
Alvaro Jaumes	12	1986
Juan Rovira	3	1987

Listar las ventas en cada oficina con el formato: 22 tiene ventas de 186,042.00 €

```
SELECT  oficina, 'tiene ventas de ' AS [ ], ventas
FROM oficinas;
```

El resultado sería:

oficina		ventas
11	tiene ventas de	69300,00
12	tiene ventas de	73500,00
13	tiene ventas de	36800,00
21	tiene ventas de	83600,00
22	tiene ventas de	18600,00
23	tiene ventas de	NULL
24	tiene ventas de	15000,00
26	tiene ventas de	NULL
28	tiene ventas de	0,00

El incluir una constante como columna en la lista de selección puede parecer inútil (se repetirá el mismo valor en todas las filas) pero veremos más adelante que tiene utilidad en ciertos casos.

También podemos utilizar la sintaxis:

```
alias_columna = <expresion>
```

Ejemplo:

```
SELECT oficina, superavit =  ventas-objetivo
```

Esto tiene el mismo efecto que

```
SELECT oficina, ventas-objetivo AS  superavit
```

UTILIZACIÓN DEL ASTERISCO

Si queremos visualizar todas las columnas del origen de datos, en lugar de indicar todas las columnas una a una se puede utilizar el carácter de sustitución *.

Mostrar todos los datos de la tabla oficinas.

```
SELECT  *
FROM  oficinas;
```

Obtener todos los datos y el superávit de cada oficina.

```
SELECT  *, (ventas-objetivo) AS superavit
FROM  oficinas;
```

También podemos cualificar el * con un nombre de tabla, de vista o un alias de tabla:

```
SELECT oficinas.*
FROM oficinas;
```

oficinas.* se interpreta como: todas las columnas de la tabla oficinas.

Esta forma se utiliza normalmente cuando el origen está basado en varias tablas y queremos indicar todas las columnas no del origen completo sino de una tabla concreta.

LAS PALABRAS CLAVE $IDENTITY Y $ROWGUID

- La palabra clave $IDENTITY se interpreta como la columna de la tabla que tiene la propiedad IDENTITY (la columna de identidad que vimos en un tema anterior).

  Por ejemplo, si en la columna codigo de la tabla usuarios (BD Biblio) se ha definido la propiedad IDENTITY.

  ```
  SELECT $IDENTITY, nombre, apellidos
  FROM usuarios;
  ```

  Es equivalente a:

  ```
  SELECT codigo, nombre, apellidos
  FROM usuarios;
  ```

- La palabra clave $ROWGUID se interpreta como la columna de la tabla que tiene la propiedad ROWGUIDCOL y se puede utilizar de la misma forma que $IDENTITY.

ORDENACIÓN DE LAS FILAS DEL RESULTADO DE ORDER BY

Si queremos que las filas del resultado de la consulta aparezcan ordenadas, lo podemos indicar mediante la cláusula ORDER BY.

```
ORDER BY {expression_columna|posicion_columna [ASC|DESC]}[ ,...n ]
```

Podemos indicar una columna o varias separadas por una coma, la columna de ordenación se especifica mediante el nombre de columna en el origen de datos o su posición dentro de la lista de selección. Si utilizamos el nombre de columna, no hace falta que la columna aparezca en la lista de selección. Si utilizamos la posición es la posición de la columna dentro de la lista de selección empezando en 1.

Por defecto la filas se ordenarán en modo ascendente (ASC), de menor a mayor; si queremos invertir ese orden añadimos detrás de la columna la palabra DESC.

- Si la columna de ordenación es alfanumérica, las filas se ordenarán por orden alfabético.
- Si la columna de ordenación es numérica, las filas se ordenarán de menor a mayor.
- Si la columna de ordenación es de tipo fecha, las filas se ordenarán de más antigua a más reciente o futura.

Ejemplos:

Mostrar las ventas de cada oficina, ordenadas por orden alfabético de región y dentro de cada región por ciudad.

```
SELECT  oficina, region, ciudad, ventas
FROM  oficinas
ORDER  BY region, ciudad;
```

Da como resultado:

Oficina	region	ciudad	ventas
24	centro	Aranjuez	15000,00
23	centro	Madrid	NULL
12	este	Alicante	73500,00
13	este	Castellón	36800,00
11	este	Valencia	69300,00
28	este	Valencia	0,00
26	norte	Pamplona	NULL
22	oeste	A Coruña	18600,00
21	oeste	Badajoz	83600,00

Listar las oficinas de manera que las oficinas de mayores ventas aparezcan en primer lugar.

```
SELECT  ciudad, region, ventas
FROM  oficinas
ORDER  BY ventas DESC;
```

ciudad	region	ventas
Badajoz	oeste	83600,00
Alicante	este	73500,00
Valencia	este	69300,00
Castellon	este	36800,00
A Coruña	oeste	18600,00
Aranjuez	centro	15000,00
Valencia	este	0,00
Pamplona	norte	NULL
Madrid	centro	NULL

Listar las oficinas clasificadas en orden descendente de rendimiento de ventas, de modo que las de mayor rendimiento aparezcan las primeras.

```
SELECT  ciudad, region, ventas-objetivo
FROM  oficinas
ORDER  BY 3 DESC;
```

Lo mismo que el anterior pero agrupadas por región.

```
SELECT  region, ciudad, (ventas-objetivo) AS superavit
FROM oficinas
ORDER BY region, superavit DESC;
```

Resultado:

Region	ciudad	superavit
centro	Aranjuez	-10000,00
centro	Madrid	NULL
este	Valencia	11800,00
este	Castellón	1800,00
este	Alicante	-6500,00
este	Valencia	-90000,00
norte	Pamplona	NULL
oeste	Badajoz	11100,00
oeste	A Coruña	-11400,00

En este caso hemos utilizado el alias de columna para hacer referencia a la columna calculada y también se puede observar que las filas aparecen ordenadas por región ascendente (no hemos incluido nada después del nombre de la columna) y dentro de cada región por superávit y descendente.

ELIMINAR FILAS DUPLICADAS DISTINCT/ALL

SQL no elimina las filas duplicadas en el resultado de la consulta, si nosotros no queremos que se repitan las filas, tenemos la cláusula DISTINCT.

Al incluir la cláusula DISTINCT en la SELECT, se eliminará del resultado las repeticiones de filas de resultado. Si por el contrario queremos que aparezcan todas las filas seleccionadas podemos incluir la cláusula ALL o nada, ya que ALL es el valor por defecto.

Listar los nº de empleado de los directores de las oficinas.

```
SELECT  dir
FROM oficinas;
```

dir
106
104
105
108
108
108
108
NULL
NULL

Si un mismo empleado dirige varias oficinas (por ejemplo el 108), su código aparece repetido en el resultado. Para evitarlo modificamos la consulta:

```
SELECT DISTINCT dir
FROM oficinas;
```

dir
NULL
104
105
106
108

Han desaparecido los valores duplicados.

Los que se eliminan son valores duplicados de filas del resultado, por ejemplo:

```
SELECT DISTINCT dir, region
FROM oficinas;
```

dir	region
NULL	este
NULL	norte
104	este
105	este
106	este
108	centro
108	oeste

Ahora el 108 aparece dos veces porque las dos filas donde aparece no son iguales (porque tienen distinta región).

NOTA: La cláusula DISTINCT hace que la consulta tarde algo más en ejecutarse debido al proceso adicional de buscar y eliminar las repeticiones, por lo que se aconseja utilizarla únicamente cuando sea

imprescindible.

SELECCIÓN DE FILAS – WHERE

La cláusula WHERE se emplea para especificar las filas que se desean recuperar del origen de datos.

```
WHERE <condicion_búsqueda>
  <condicion_búsqueda> ::=
        { [NOT]<predicado>
          |(<condicion_búsqueda>)
         }
         [{AND|OR} [NOT] {<predicado>|(<condicion_búsqueda>)}]
    [ ...n ]
```

En el resultado de la consulta sólo aparecerán las filas que cumplan que la condición de búsqueda sea TRUE, los valores NULL no se incluyen, por lo tanto, en las filas del resultado. La condición de búsqueda puede ser una condición simple o una condición compuesta por varias condiciones (predicados) unidas por operadores AND y OR, no hay límite en cuanto al número de predicados que se pueden incluir. En las condiciones compuestas se pueden utilizar paréntesis para delimitar predicados y se aconseja su uso cuando se incluyen operadores AND y OR en la misma condición de búsqueda.

PREDISCADOS

En SQL tenemos 7 tipos de predicados, condiciones básicas de búsqueda:

- Comparación estándar
- Pertenencia a un intervalo (BETWEEN)
- Pertenencia a un conjunto (IN)
- Test de valor nulo (IS NULL).
- Coincidencia con patrón (LIKE)
- Si contiene (CONTAINS)
- FREETEXT

Comparación estándar.

Compara el valor de una expresión con el valor de otra. Para la comparación se pueden emplear = , <> , !=, < , <= , !<, > , >= ,!>

Sintaxis:

```
<expresion>  {=|<>|!=|>|>=|!>|<|<=|!<} <expresion>
```

<expresion> Puede ser:

- Un nombre de columna,
- una constante,
- una función (inclusive la función CASE),
- una variable,
- una subconsulta escalar o
- cualquier combinación de nombres de columna, constantes y funciones conectados mediante uno o varios operadores o una subconsulta.

Ejemplo:

Listar los "buenos" vendedores (los que han rebasado su cuota).

```
SELECT numemp, nombre,  ventas, cuota
FROM empleados
WHERE ventas > cuota
```

numemp	nombre	ventas	cuota
101	Antonio Viguer	30500,00	30000,00
102	Alvaro Jaumes	47400,00	35000,00
103	Juan Rovira	28600,00	27500,00
105	Vicente Pantalla	36800,00	35000,00
106	Luis Antonio	29900,00	27500,00
108	Ana Bustamante	36100,00	35000,00
109	María Sunta	39200,00	3000,00

Las columnas que aparecen en el WHERE no tienen por qué aparecer en la lista de selección, esta instrucción es igual de válida:

```
SELECT numemp, nombre
FROM empleados
WHERE ventas > cuota;
```

Hallar vendedores contratados antes de 1988.

```
SELECT numemp, nombre, contrato
FROM empleados
WHERE contrato < '01/01/1988';
```

numemp	nombre	contrato
101	Antonio Viguer	1986-10-20
102	Alvaro Jaumes	1986-12-10
103	Juan Rovira	1987-03-01
104	José González	1987-05-19

También podemos utilizar funciones, ésta es equivalente a la anterior:

```
SELECT numemp, nombre
FROM empleados
WHERE YEAR(contrato) < 1988;
```

La función YEAR(fecha) devuelve el año de una fecha.

Hallar oficinas cuyas ventas estén por debajo del 80% de su objetivo:

```
SELECT  oficina
FROM  oficinas
WHERE  ventas < (.8 * objetivo);
```

Hallar las oficinas dirigidas por el empleado 108:

```
SELECT  oficina
    FROM  oficinas
    WHERE  dir = 108;
```

Pertenencia a un intervalo. BETWEEN

```
<expresion> [NOT] BETWEEN <expresion2> AND <expresion3>
```

Examina si el valor de la expresión de test está en el rango delimitado por los valores resultantes de expresion1 y expresion2, estos valores no tienen porqué estar ordenados en ANSI/ISO; expresion1 debe ser menor o igual a expresion2.

Hallar vendedores cuyas ventas estén entre 20.000 euros y 50.000.

```
SELECT numemp, nombre, ventas
FROM empleados
WHERE  ventas BETWEEN 20000 AND  100000;
```

numemp	nombre	ventas
101	Antonio Viguer	30500,00
102	Alvaro Jaumes	47400,00
103	Juan Rovira	28600,00
105	Vicente Pantalla	36800,00
106	Luis Antonio	29900,00
108	Ana Bustamante	36100,00
109	María Sunta	39200,00

La instrucción anterior es equivalente a:

```
SELECT numemp, nombre, ventas
FROM empleados
WHERE  ventas >= 20000 AND ventas <=100000;
```

Parece que con BETWEEN se lee mejor.

Observa que no hemos utilizado separadores de millares (100.000), porque se habría interpretado por una coma decimal.

Test de pertenencia a conjunto IN

```
<expresion> IN (  <exp_valor> [ ,...n ] )
```

Examina si el valor de la expresion es uno de los valores incluidos en la lista de valores indicados entre paréntesis. Se pueden expresar los valores mediante cualquier expresión, la única condición es que todas las exp_valor devuelvan el mismo tipo de datos.

Ejemplo:

Obtener los empleados que trabajan en las oficinas 11, 20 o 22:

```
SELECT oficina, numemp, nombre
FROM empleados
WHERE oficina IN (11,20,22);
```

oficina	numemp	nombre
11	106	Luis Antonio
22	107	Jorge Gutiérrez
11	109	María Sunta

Test de valor nulo IS NULL

```
<expression> IS [NOT] NULL
```

Una condición de búsqueda puede ser TRUE, FALSE o NULL/UNKNOW, este último caso se produce cuando algún campo que interviene en la condición tiene valor NULL.
A veces es útil comprobar explícitamente los valores NULL en una condición de búsqueda ya que estas filas puede que queramos darles un tratamiento especial, para ello tenemos el predicado IS NULL.

Este test produce un valor TRUE o FALSE, por lo que se podrá combinar con otras condiciones. El valor NULL no es en sí un valor por eso no lo podemos utilizar en una igualdad.

```
SELECT numemp,nombre
FROM empleados
WHERE oficina = NULL;
```

Esta instrucción no da error pero no obtiene lo que en principio parece que quiere obtener. No obtenemos los empleados cuya oficina sea un valor nulo (es decir los empleados que no tienen oficina), no obtenemos nada, en cambio los obtendremos utilizando el test de valor nulo:

```
SELECT numemp,nombre, oficina
FROM empleados
WHERE oficina IS NULL;
```

Resultado:

numemp	nombre	oficina
110	Juan Victor	NULL

Juan Victor es el único empleado que no tiene oficina asignada.

Listar los vendedores asignados a alguna oficina.

```
SELECT numemp, nombre, oficina
FROM empleados
WHERE oficina IS NOT NULL;
```

numemp	nombre	oficina
101	Antonio Viguer	12
102	Alvaro Jaumes	21
103	Juan Rovira	12
104	José González	12
105	Vicente Pantalla	13
106	Luis Antonio	11
107	Jorge Gutiérrez	22
108	Ana Bustamante	21
109	María Sunta	11

Test de correspondencia con patrón LIKE

Se utiliza cuando queremos comparar el valor de una columna con un patrón en el que se utilice caracteres comodines.

```
<expression>  [NOT] LIKE  <patron>  [ESCAPE 'car_escape']
```

Con expresión indicamos el valor a comparar (normalmente será el nombre de una columna) y patrón es la cadena que se busca. El patrón es de tipo texto y tiene que escribirse entre comillas simples. Dentro del patrón podemos utilizar los siguientes comodines:

% representa cualquier cadena de cero o más caracteres.

```
SELECT numemp,nombre
FROM empleados
WHERE nombre LIKE 'An%';
```

numemp	nombre
101	Antonio Viguer
108	Ana Bustamante

Obtiene todos los nombres que empiecen por An.

```
SELECT numemp,nombre
FROM empleados
WHERE nombre LIKE '%z';
```

numemp	nombre
104	José González
107	José González

Obtiene los nombres que acaban en z.

```
SELECT numemp,nombre
FROM empleados
WHERE nombre LIKE '%on%';
```

numemp	nombre
101	Antonio Viguer
104	José González
106	Luis Antonio

Obtiene los nombres que contienen on.

_ representa cualquier carácter (sólo uno).

```
SELECT numemp,nombre
FROM empleados
WHERE nombre LIKE '__a%';
```

numemp	nombre
103	Juan Rovira
108	Ana Bustamante
110	Juan Victor

Obtiene los nombres cuya tercera letra sea una a (en el patrón tenemos dos caracteres subrayado).

[ ] sirve para indicar un carácter cualquiera perteneciente al conjunto indicando.
El conjunto se indica enumerando los caracteres o indicando un intervalo.

```
SELECT numemp,nombre
FROM empleados
WHERE nombre LIKE '[a-d]%';
```

Obtiene los nombres que empiezan por cualquier letra de la a a la d.

Es equivalente a escribir:

```
SELECT numemp,nombre
FROM empleados
WHERE nombre LIKE '[abcd]%';
```

[^] significa cualquier carácter individual que no se encuentre en el conjunto.

```
SELECT numemp,nombre
FROM empleados
WHERE nombre LIKE '[^abcd]%';
```

Y

```
SELECT numemp,nombre
FROM empleados
WHERE nombre LIKE '[^a-d]%';
```

Obtienen los nombres que no empiecen por a, b, c ni d.

Es importante tener en cuenta que dentro del patrón el espacio en blanco es considerado como un carácter más, si colocamos dos espacios en el patrón, se buscarán dos espacios en el campo.

Si queremos incluir en el patrón uno de los caracteres comodines y que no sea interpretado como un comodín, sino como un carácter normal, lo tenemos que encerrar entre corchetes o utilizar un carácter de escape.

[ESCAPE 'car_escape']

La cláusula ESCAPE es opcional y permite definir un carácter de escape.

Un carácter de escape es un carácter que se coloca delante de un carácter comodín para indicar que el comodín no debe interpretarse como tal, sino como un carácter normal.

Por ejemplo queremos buscar los nombres compuestos que incluyen un subrayado. En este caso tenemos que poner el carácter _ como un carácter normal no como un comodín, así que lo escribiremos así:


```
SELECT numemp,nombre
FROM empleados
```

```
WHERE nombre LIKE '%[_]%';
```

O bien,

```
SELECT numemp,nombre
FROM empleados
WHERE nombre LIKE '%!_%' ESCAPE '!';
```

CONDICIONES DE BÚSQUEDA COMPUESTAS

En una cláusula WHERE podemos incluir una condición de búsqueda simple (formada por un solo predicado) o compuesta (formada por la combinación de predicados unidos por los operadores lógicos NOT, AND, OR).

Cuando la condición incluye varios operadores lógicos, el orden de prioridad de estos operadores es:

- NOT (el más alto),
- seguido de AND y OR (estos dos al mismo nivel).

Como siempre, se pueden utilizar paréntesis para alterar esta prioridad en una condición de búsqueda.

El orden de evaluación de los operadores lógicos puede variar dependiendo de las opciones elegidas por el optimizador de consultas.

Los operadores lógicos pueden devolver tres valores distintos: TRUE, FALSE, NULL (UNKNOWN).

Tablas de verdad de los operadores:

AND Combina dos condiciones y se evalúa como TRUE cuando ambas condiciones son TRUE.

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

OR Combina dos condiciones y se evalúa como TRUE cuando alguna de las condiciones es TRUE.

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

NOT Niega la expresión booleana que especifica el predicado

NOT	TRUE	FALSE	NULL
	FALSE	TRUE	NULL

Hallar los vendedores que están por debajo de su cuota y tienen ventas inferiores a 30.000.


```
SELECT nombre
FROM empleados
WHERE ventas < cuota AND ventas  < 30000;
```

Hallar los vendedores que están debajo de su cuota, pero cuyas ventas no sean inferiores a 150.000.

```
SELECT nombre
FROM empleados
WHERE ventas < cuota AND ventas  < 150000;
```

Hallar las oficinas no dirigidas por el empleado 108

```
SELECT oficina
FROM oficinas
WHERE NOT dir = 108;
```

O

```
SELECT oficina
FROM oficinas
WHERE  dir <> 108;
```

Devuelven:

oficina
11
12
13

Las oficinas sin director no aparecen, para que aparezcan deben añadir otro predicado:

```
SELECT oficina, dir
FROM oficinas
WHERE NOT dir = 108 or dir is  null;
```

oficina	dir
11	106
12	104
13	105
26	NULL
28	NULL