

Lenguaje SQL

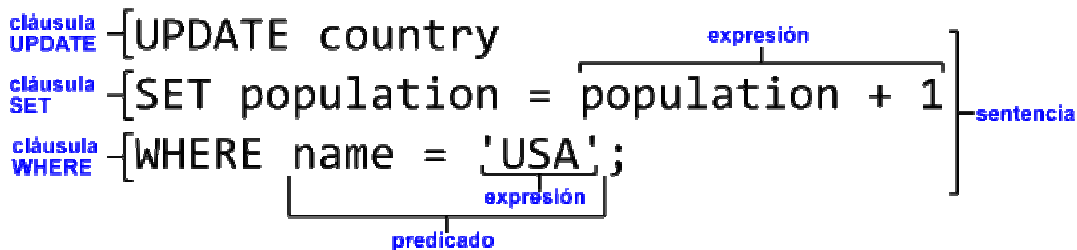
1. Introducción al lenguaje SQL y la herramienta SQL*PLUS

I. ¿Qué es SQL?

SQL (*Structured Query Language*, es decir, Lenguaje de consulta estructurado) es un lenguaje de comunicación con una base de datos diseñado para recuperar y gestionar los datos en sistemas de bases de datos relacionales (**RDBMS**), además de permitir crear y modificar elementos y objetos de una base de datos, así como establecer un sistema de control de acceso.

Es un lenguaje ANSI, es decir standard, pero la mayoría de fabricantes incluyen extensiones y funciones para gestionar sus sistemas de bases de datos.

II. Elementos de SQL



SQL se compone de varios elementos:

- **Sentencias** - Seleccionan o modifican datos u objetos del esquema, controlan el flujo de los programas, etc.
- **Consultas** - Recuperan datos en función de unos criterios.
- **Expresiones** - Generan valores escalares.
- **Predicados** - Especifican las condiciones que serán evaluadas.
- **Cláusulas** - Componen las sentencias. Opcionales en algunos casos.
- **Punto y coma:** No es requerido en todos los RDBMS pero es el estándar para indicar fin de la sentencia.

III.Sentencias SQL

SELECT	Selección de datos
INSERT UPDATE DELETE MERGE	DML: Lenguaje de manipulación de datos
CREATE ALTER DROP RENAME TRUNCATE	DDL: Lenguaje de definición de datos
COMMIT ROLLBACK SAVEPOINT	Control de transacciones
GRANT REVOKE	DCL: Lenguaje de control de datos

SELECT - Recupera información de la base de datos.

DML (Lenguaje de manipulación de datos)

INSERT - Inserta nuevos registros en la base de datos.

UPDATE - Actualiza registros ya existentes.

DELETE - Borra registros.

MERGE - Realiza una combinación de datos según unas condiciones

DDL (Lenguaje de definición de datos)

CREATE - Crea estructuras de datos (tablas, vistas, etc).

ALTER - Modifica estructuras de datos

DROP - Borra estructuras de datos.

RENAME - Renombra estructuras de datos.

TRUNCATE - Sirve para vaciar tablas.

Control de Transacciones

COMMIT - Confirma las operaciones DML.

ROLLBACK - Invalida las operaciones DML

SAVEPOINT - Introduce un punto de guardado para realizar COMMIT y ROLLBACK localizados.

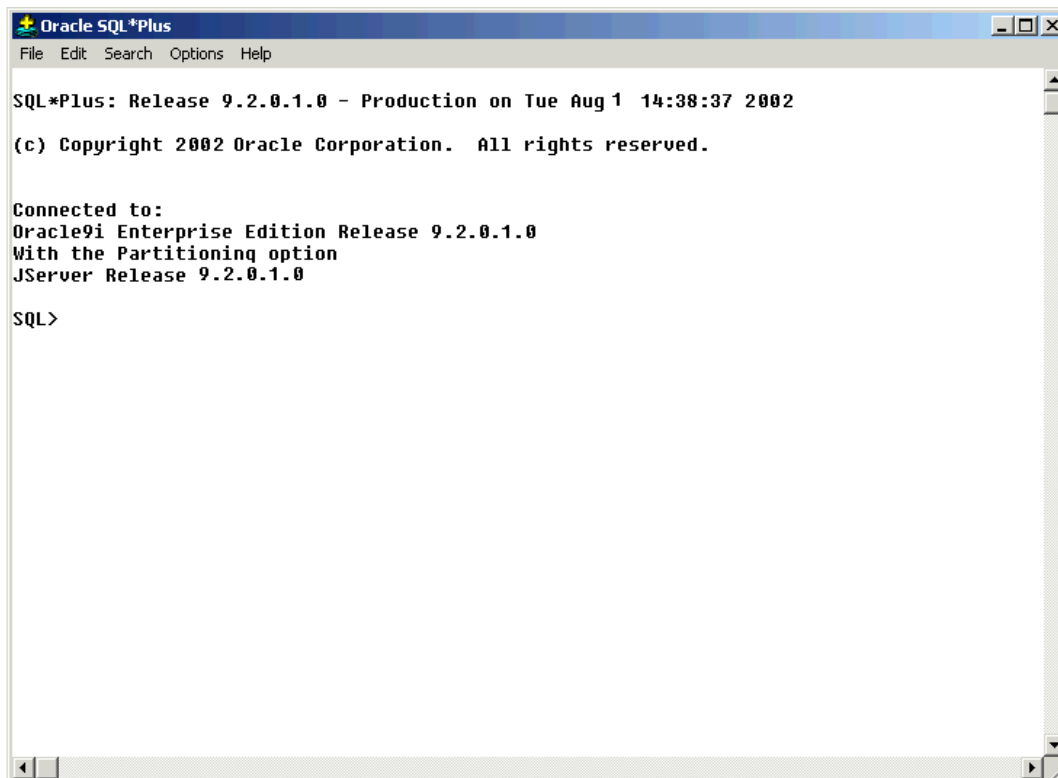
DCL (Lenguaje de control de datos)

GRANT - Otorga permisos a usuarios.

REVOKE - Quita permisos de los usuarios.

IV. SQL*PLUS

SQL*Plus es el cliente básico de acceso a una base de datos Oracle. Permite realizar consultas y sentencias SQL que serán ejecutadas por la base de datos en el servidor. Es posible ejecutar cualquier consulta o programa PL/SQL que el servidor de Oracle pueda entender.



```
Oracle SQL*Plus
File Edit Search Options Help

SQL*Plus: Release 9.2.0.1.0 - Production on Tue Aug 1 14:38:37 2002

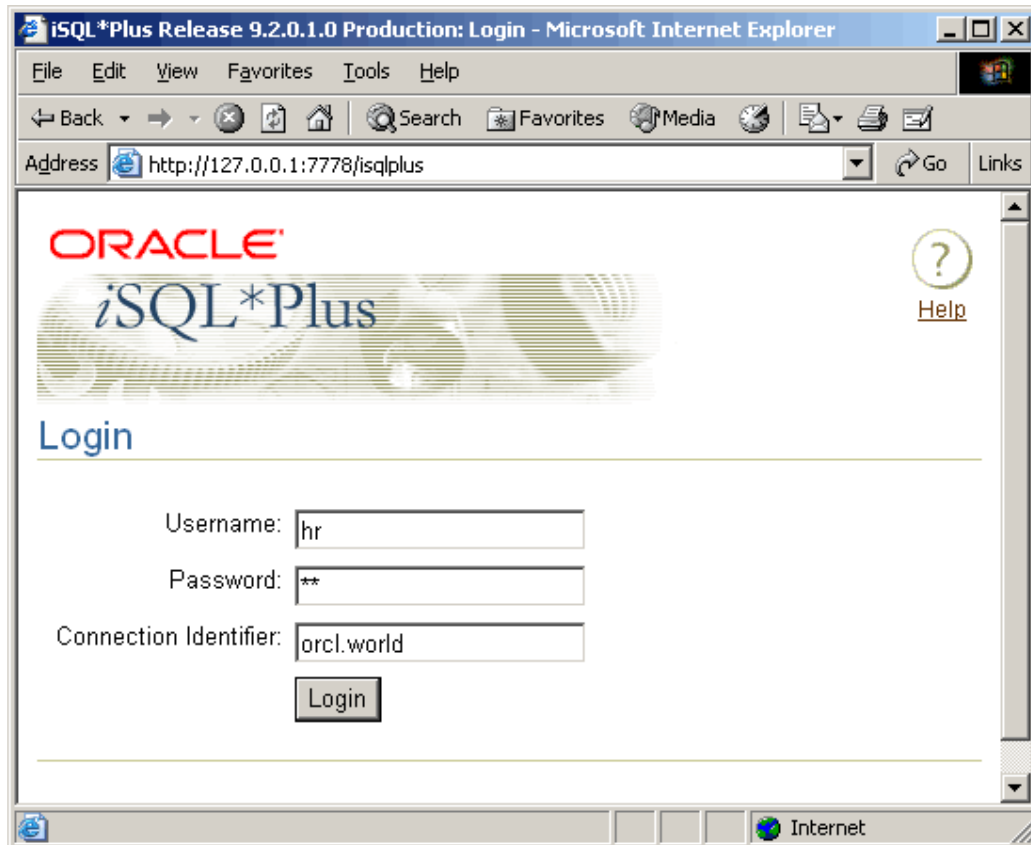
(c) Copyright 2002 Oracle Corporation. All rights reserved.

Connected to:
Oracle9i Enterprise Edition Release 9.2.0.1.0
With the Partitioning option
JServer Release 9.2.0.1.0

SQL>
```

V.iSQL*PLUS

Es una versión que aparece a partir de la versión 9i de Oracle, es muy similar al SQL*Plus, pero no requiere instalación en el cliente, sino que se ejecuta desde un navegador web. Es necesario que el servidor de la base de datos acepte peticiones para este programa, normalmente se accede a través de la url *http://ip-del-servidor:5560*



2. Consultas SQL: Sentencia SELECT

La sentencia **SELECT** se utiliza para recuperar información de la base de datos, y puede **proyectar** las columnas seleccionadas, es decir, realizar un filtro sobre la tabla o tablas originales y recuperar solamente datos de las columnas filtradas. También puede **seleccionar**, es decir, filtrar los registros según varios criterios, o realizar **uniones**, que recopilan datos de diferentes tablas a través de una relación entre ellas.

todas las columnas
cláusula de selección | suprime los duplicados | asigna el nombre de las cabeceras
SELECT * | { **[DISTINCT]** column | expression **[alias]**, ... }
especifica la tabla seleccionada **[FROM** table
[WHERE condition(s)];
restringe las filas que cumplen la condición

Las cláusulas **SELECT**, que especifica las columnas o expresiones que vamos a recuperar, y **FROM**, que indica la tabla de la que se recuperan, son **obligatorias**.

SELECT - Lista las columnas a recuperar.
***** - Selecciona todas las columnas
DISTINCT - Elimina duplicados
column | expression - selecciona la columna y/o expresión (pueden ser varias) que se van a recuperar
alias - Indica la cabecera de cada una de las columnas o expresiones
FROM - La tabla que contiene las columnas
WHERE - Filtro de los registros a visualizar
condition(s) - Lista de condiciones de filtro.

Las sentencias SQL no son sensibles a mayúsculas o minúsculas pero se recomienda escribir las palabras clave en mayúsculas y el resto en minúsculas.

I. Seleccionar las columnas de una tabla

```
SELECT *  
FROM viajero;
```

ID_VIAJERO	NOMBRE	APELLIDOS	DIRECCION	TELEFONO	FECHA_NACIMIENTO
1	Alberto	Gómez	Calle de la Amapola 1	93000001	01/01/50
2	Sofia	Martínez	Calle de la Violeta 32	93000071	01/08/75
3	Juan	López	Calle de la Margarita 21	93000039	31/03/89
4	Marcos	Fernández	Calle de la Rosa 10	93000068	31/08/77
5	Marta	Pérez	Calle de la Amapola 12	93000003	09/03/85
6	Lorena	Hernández	Calle de los Girasoles 12	93000088	25/08/80
7	Alba	Álvarez	Calle de la Violeta 22	93000079	20/10/82
8	Jose Luís	Suárez	Calle de la Margarita 1	93000033	30/05/58

Para seleccionar todas las columnas de una tabla se utiliza el * tras el SELECT. Otra forma de hacerlo, sería enumerar todas las columnas, separadas por comas, de la tabla.

```
SELECT id_viajero, nombre, apellidos, direccion, telefono,  
fecha_nacimiento  
FROM viajero;
```

Para filtrar por columnas, tan solo hay que especificarlas separadas por comas.

```
SELECT nombre, apellidos  
FROM viajero;
```

NOMBRE	APELLIDOS
Alberto	Gómez
Sofia	Martínez
Juan	López
Marcos	Fernández
Marta	Pérez
Lorena	Hernández
Alba	Álvarez
Jose Luís	Suárez
Damián	Rodríguez
Julián	Rodríguez

II. Expresiones aritméticas

Sirven para realizar cálculos sobre la información recuperada y mostrar los resultados de los mismos, se pueden usar los operadores aritméticos en cualquier cláusula de una sentencia SQL, excepto en la cláusula FROM.

Los operadores aritméticos (ordenados por prioridad) son

*** / + -**

```
SELECT nombre, continente,  
       num_habitantes, num_habitantes + 10000  
FROM pais;
```

NOMBRE	CONTINENTE	NUM_HABITANTES	NUM_HABITANTES+10000
España	Europa	46080737	46090737
Portugal	Europa	10605870	10615870
Italia	Europa	59337888	59347888
Francia	Europa	60876136	60886136
Reino Unido	Europa	60609153	60619153
Alemania	Europa	82210000	82220000
Estados Unidos	América del Norte	302791280	302801280
Canadá	América del Norte	33187800	33197800
Méjico	América del Norte	107449525	107459525
Brasil	América del Sur	188078227	188088227

Los operadores tienen prioridad y se debe utilizar paréntesis para forzar operaciones que se realizarían de forma diferente según su prioridad.

```
SELECT nombre, num_habitantes, 2 *num_habitantes + 1000000,
      2*(num_habitantes + 1000000)
FROM pais;
```

NOMBRE	NUM_HABITANTES	2*NUM_HABITANTES +1000000	2*(NUM_HABITANTES +1000000)
España	46080737	93161474	94161474
Portugal	10605870	22211740	23211740
Italia	59337888	119675776	120675776
Francia	60876136	122752272	123752272
Reino Unido	60609153	122218306	123218306
Alemania	82210000	165420000	166420000
Estados Unidos	302791280	606582560	607582560
Canadá	33187800	67375600	68375600
Méjico	107449525	215899050	216899050
Brasil	188078227	377156454	378156454

III. Valores nulos

El valor nulo se representa con null e implica la ausencia de valor para una columna de una fila determinada. Es diferente del número 0 y del caracter espacio.

```
SELECT nombre, habitantes, clima
FROM lugar;
```

NOMBRE	HABITANTES	CLIMA
Budapest	1696128	
Osaka	2629252	
Kyoto	1464900	
Tibet	2740000	
Shangai	18403769	Subtropical
Rajastán		
Himalaya		
Bombay	13691836	Tropical
Casablanca	2949805	
Rabat		

Operar valores nulos con expresiones aritméticas devolverá valores nulos:

```
SELECT nombre, habitantes + 100000, clima  
FROM lugar;
```

NOMBRE	HABITANTES+100000	CLIMA
Budapest	1796128	
Osaka	2729252	
Kyoto	1564900	
Tibet	2840000	
Shangai	18503769	Subtropical
Rajastán		
Himalaya		
Bombay	13791836	Tropical
Casablanca	3049805	
Rabat		

IV. Alias de columna

Los alias para las columnas sirven para renombrar la cabecera de una columna devuelta por una consulta SQL.

Para utilizarlos, se sitúa el alias después de la columna o expresión separado por un espacio, o se puede utilizar la palabra **AS** entre la columna y el alias.

Si el nombre del alias contiene espacios, tiene caracteres especiales (#, \$, ...), o es sensible a mayúsculas/minúsculas se debe entrecomillar con comillas dobles ("")

```
SELECT nombre AS "Nombre del Lugar", 2*(habitantes + 5000)
"Nueva Población" FROM lugar;
```

Nombre Del Lugar	Nueva Población
Budapest	3402256
Osaka	5268504
Kyoto	2939800
Tibet	5490000
Shangai	36817538
Rajastán	
Himalaya	
Bombay	27393672
Casablanca	5909610
Rabat	

V.Operador de concatenación

El operador de concatenación || sirve para unir columnas o expresiones entre sí y mostrarlas como una sola.

Es posible unir las columnas con cadenas de literales (entre comillas simples) para formar frases o literales.

```
SELECT apellidos || ', ' || nombre AS "Nombre Completo",
nombre || ' vive en ' || direccion || ' y nació el '
|| fecha_nacimiento AS "Información"
FROM viajero;
```

Nombre Completo	Información
Gómez,Alberto	Alberto vive en Calle de la Amapola 1 y nació el 01/01/50
Martínez,Sofia	Sofia vive en Calle de la Violeta 32 y nació el 01/08/75
López,Juan	Juan vive en Calle de la Margarita 21 y nació el 31/03/89
Fernández,Marcos	Marcos vive en Calle de la Rosa 10 y nació el 31/08/77
Pérez,Marta	Marta vive en Calle de la Amapola 12 y nació el 09/03/85
Hernández,Lorena	Lorena vive en Calle de los Girasoles 12 y nació el 25/08/80
Álvarez,Alba	Alba vive en Calle de la Violeta 22 y nació el 20/10/82
Suárez,Jose Luís	Jose Luís vive en Calle de la Margarita 1 y nació el 30/05/58
Rodríguez,Damián	Damián vive en Calle de los Tulipanes 10 y nació el 10/11/60
Rodríguez,Julián	Julián vive en Calle de los Claveles 3 y nació el 01/12/58

VI. Uso de ***DISTINCT*** para eliminar duplicados

En ocasiones, habrá consultas que devuelvan filas duplicadas, utilizando la palabra clave **DISTINCT**, podemos eliminar estos duplicados del resultado de una sentencia SELECT.

```
SELECT continente  
FROM pais;
```

CONTINENTE
Europa
Europa
Europa
Europa
Europa
Europa
América del Norte
América del Norte
América del Norte
América del Sur

```
SELECT DISTINCT continente  
FROM pais;
```

CONTINENTE
Oceanía
África
América del Norte
Europa
América del Sur
Asia

3. Restricción de datos: Cláusula WHERE

Para limitar los registros recuperados por una consulta SQL, se usa la cláusula WHERE, justo después de la cláusula FROM, seguida de las condiciones de la comparación.

La cláusula WHERE puede comparar valores de columnas, expresiones, funciones, listas de valores, constantes...

Diagrama de la sintaxis SQL con anotaciones:

```
SELECT * | { [DISTINCT] column | expression [alias], ... }
FROM table
WHERE condition(s);
```

Las anotaciones en azul indican:

- todas las columnas (sobre *)
- cláusula de selección (sobre el signo |)
- suprime los duplicados (sobre [DISTINCT])
- asigna el nombre de las cabeceras (sobre [alias])
- especifica la tabla seleccionada (sobre FROM table)
- restringe las filas que cumplen la condición (sobre WHERE condition(s))

```
SELECT nombre, continente
FROM pais
WHERE continente = 'Europa';
```

NOMBRE	CONTINENTE
España	Europa
Portugal	Europa
Italia	Europa
Francia	Europa
Reino Unido	Europa
Alemania	Europa
Rusia	Europa
República Checa	Europa
Hungría	Europa

Las columnas que intervienen en la comparación no tienen porque estar presentes en la cláusula SELECT para poder ser filtradas por el WHERE.

Las cadenas de texto deben estar entrecomilladas con comillas simples y son sensibles a mayúsculas / minúsculas, sin embargo los números se expresan sin comillas.

Para las fechas, se debe tener en cuenta el formato por defecto, que suele depender del idioma de instalación, puede ser DD/MM/YY y se deben expresar entre comillas simples.

Condiciones de Comparación

Las condiciones de comparación se usan para comparar una columna o expresión con otra expresión o valor.

WHERE expresión *operador* valor

Las condiciones de comparación son las siguientes:

Operador	Significado
=	Igualdad
>	Mayor que
>=	Mayor / Igual que
<	Menor que
<=	Menor / Igual que
<>	Distinto
BETWEEN ... AND ...	Entre dos valores (ambos incluidos)
IN (lista)	La expresión está incluida en la lista de valores
LIKE	Igualdad usando comodines
IS NULL	Valor nulo

```
SELECT nombre, habitantes
FROM lugar
WHERE habitantes <= 1000000;
```

NOMBRE	HABITANTES
Vladivostok	586829
Szeged	163173
Jersey	87186
Manchester	441200
Cumbria	496200
San Francisco	739426
Nueva Orleans	454863
Boston	596638
Washington	582049
Oviedo	223617

Condición BETWEEN

Se utiliza para filtrar filas incluidas en un rango de valores definido por un límite inferior y un límite superior.

```
SELECT nombre, habitantes
FROM lugar
WHERE habitantes BETWEEN 1000000 AND 2000000;
```

NOMBRE	HABITANTES
Budapest	1696128
Kyoto	1464900
Volgograd	1012800
Barcelona	1595110
San Diego	1255240
Valencia	1738690
Milán	1308735
Patagonia	1738251
Córdoba	1315540
Praga	1188126

Condición IN

Se utiliza para comprobar si una columna o expresión está contenida en una lista determinada de valores. Esta condición también se llama *condición de pertenencia*.

```
SELECT nombre, apellidos, fecha_nacimiento
FROM viajero
WHERE fecha_nacimiento IN
('01/01/50', '25/08/80', '31/03/89');
```

NOMBRE	APELLIDOS	FECHA_NACIMIENTO
Alberto	Gómez	01/01/50
Juan	López	31/03/89
Lorena	Hernández	25/08/80
Carlos	Vázquez	01/01/50

Se puede utilizar **NOT IN** para especificar que una columna o expresión no está en una lista determinada de valores.

Condición LIKE

Se utiliza para filtrar utilizando comodines de búsqueda en la cadena de comparación. Se pueden utilizar estos comodines:

% - Secuencia de 0 o más caracteres.

_ - Secuencia de un único carácter.

```
SELECT nombre, apellidos, fecha_nacimiento
FROM viajero
WHERE apellidos LIKE 'M%';
```

NOMBRE	APELLIDOS	FECHA_NACIMIENTO
Sofía	Martínez	01/08/75
Diana	Menéndez	10/11/79
Isabel	Méndez	25/08/85
María	Muñoz	22/10/80
Alejandra	Martínez	30/06/64
Mercedes	Martínez	05/06/77

Los símbolos % y _ se pueden combinar entre ellos y con cualquier otro carácter o cadena de caracteres.

Si se necesita buscar exactamente los caracteres % y _, podemos utilizar la opción **ESCAPE**, que define un carácter de escape

```
SELECT nombre, apellidos, perfil_viajero
FROM viajero
WHERE perfil_viajero LIKE 'CLASE\_%' ESCAPE '\\';
```

NOMBRE	APELLIDOS	PERFIL_VIAJERO
Alberto	Gómez	CLASE_TURISTA
Marcos	Fernández	CLASE_TURISTA
Lorena	Hernández	CLASE_TURISTA
Beatriz	González	CLASE_TURISTA
Margarita	Sánchez	CLASE_TURISTA
Carmen	Ruiz	CLASE_TURISTA
Susana	Díaz	CLASE_PREMIER
Esteban	Jiménez	CLASE_BUSSINESS
Natalia	Fernández	CLASE_TURISTA
Oriol	Pérez	CLASE_TURISTA

Se puede usar **NOT LIKE** para especificar que una columna o expresión no coinciden con la cadena especificada.

Condiciones NULL

Para comprobar si un valor es NULL se utiliza la condición IS NULL. Un valor nulo significa la ausencia del mismo, un valor indeterminado, no disponible o no asignado. Si se comparara con =, el resultado sería siempre NULL, ya que un valor nulo no es comparable con ningún otro, ni siquiera con otro nulo.

```
SELECT nombre, clima
FROM lugar
WHERE clima IS NULL;
```

NOMBRE	CLIMA
Jersey	
Manchester	
Exeter	
Cumbria	
San Francisco	
Gran Cañón	
Maui	
Nueva Orleans	
Montañas Rocosas	
Boston	

Para comprobar los valores no nulos, se utiliza **IS NOT NULL**.

Condiciones Lógicas

Operador	Significado
AND	Devuelve TRUE si las dos condiciones son TRUE
OR	Devuelve TRUE si al menos una de las condiciones es TRUE
NOT	Devuelve TRUE si la condicion es FALSE

```
SELECT nombre, clima, habitantes
FROM lugar
WHERE clima = 'Continental' AND habitantes IS NOT NULL;
```

NOMBRE	CLIMA	HABITANTES
Nueva York	Continental	8214426
Moscú	Continental	12622000
Praga	Continental	1188126
Vladivostok	Continental	586829
Chicago	Continental	2842518

```
SELECT nombre, clima, habitantes
FROM lugar
WHERE clima = 'Continental' OR habitantes > 10000000;
```

NOMBRE	CLIMA	HABITANTES
Nueva York	Continental	8214426
Tokio	Templado	12527115
Pekin	Continental	-
Moscú	Continental	12622000
Praga	Continental	1188126
Shangai	Subtropical	18403769
Bombay	Tropical	13691836
Vladivostok	Continental	586829
Chicago	Continental	2842518
Sao Paulo	Tropical	10927985

```
SELECT nombre, continente
FROM pais
WHERE continente NOT IN ('Europa', 'Asia', 'África');
```

NOMBRE	CONTINENTE
Estados Unidos	América del Norte
Canadá	América del Norte
Méjico	América del Norte
Brasil	América del Sur
Argentina	América del Sur
Uruguay	América del Sur
Australia	Oceanía
Nueva Zelanda	Oceanía
Ecuador	América del Sur

Prioridad de Ejecución

Las condiciones lógicas junto con las aritméticas, tienen una prioridad de ejecución.

Operador
Operadores aritméticos
Operadores de concatenación
Operadores de comparación
IS NULL, LIKE, IN (y sus variantes con NOT)
BETWEEN y NOT BETWEEN
NOT
AND
OR

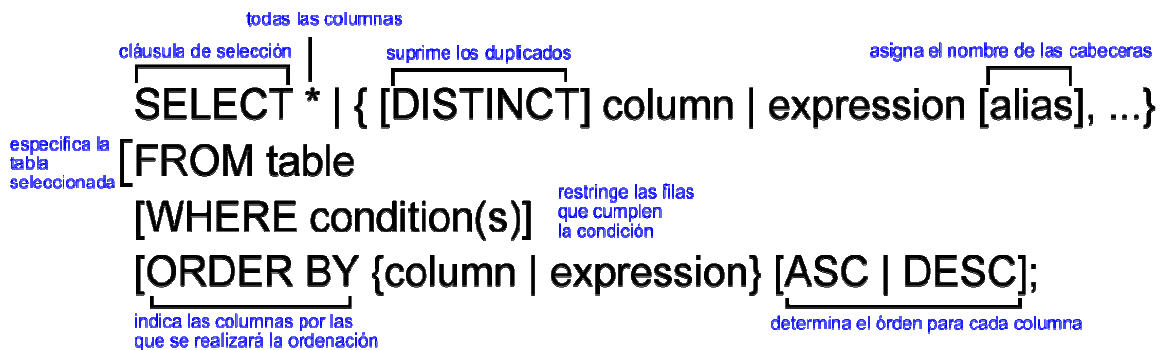
Para modificar esta prioridad de evaluación, se utilizan los paréntesis.

4. Ordenación de datos: Cláusula ORDER BY

El orden en el que una sentencia SELECT devuelve las filas no está predefinido, sino que depende del orden interno de la tabla, así que, podría pasar que ejecutemos la misma consulta sin ORDER BY y nos devuelva registros con diferente orden.

Se puede especificar la cláusula **ORDER BY** para modificar el orden inicial.

La cláusula **ORDER BY** aparece en el último lugar de la sentencia SELECT y se le puede especificar el tipo de orden para cada columna, los tipos de orden son **ASC** (ascendente, por defecto) y **DESC** (descendente)



```
SELECT nombre, apellidos
FROM viajero
ORDER BY apellidos DESC;
```

NOMBRE	APELLIDOS
Carlos	Vázquez
Jose Luís	Suárez
Margarita	Sánchez
Carmen	Ruiz
Damián	Rodríguez
Manuel	Rodríguez
Julián	Rodríguez
Francisco	Ramírez
Oriol	Pérez
Marta	Pérez

Si no se especifica ASC o DESC, el orden por defecto es ascendente, que implica:

- Para valores numéricos se muestran los valores más bajos primero.
- Para valores de fecha, se muestran los valores con fecha anterior.
- Para valores de caracteres, se usa el orden alfabético.
- Los valores nulos se muestran los últimos para orden ascendente, y los primeros para orden descendente

Es posible ordenar por múltiples columnas y también utilizar el alias de la columna para realizar ordenaciones. En el caso de ordenar por múltiples columnas, se debe especificar ASC o DESC en cada una, de lo contrario se ordenarán ascendentemente.

Se puede utilizar una columna o expresión que no esté presente en la SELECT para realizar el orden.

```
SELECT apellidos || ', ' || nombre as nombre_completo,  
       fecha_nacimiento  
FROM viajero  
ORDER BY fecha_nacimiento ASC,  
         nombre_completo DESC,  
         id_viajero;
```

NOMBRE_COMPLETO	FECHA_NACIMIENTO
Vázquez, Carlos	01/01/50
Gómez, Alberto	01/01/50
Fernández, Roberto	15/02/55
Suárez, Jose Luís	30/05/58
Rodríguez, Julián	01/12/58
González, Israel	31/03/59
Rodríguez, Damián	10/11/60
González, Beatriz	22/04/62
Rodríguez, Manuel	05/06/63
Martínez, Alejandra	30/06/64