SQL

Structured Query Language

SQL

"Lenguaje de Consulta Estructurada"

- Es un LENGUAJE
- Tabla
  - Campo (Field)
  - Registro (row/record)
  - Columna

ID	NOMBRE	EDAD
1	Ernesto	38
2	Edwin	26
3	Alejandro	49

SQL

- No es “case-sensitive”
- Generalmente las palabras reservadas son escritos en mayúsculas
- Pueden ser escritos en diferentes líneas
- Inician con un comando SQL
- Terminan con punto y coma “;”

SQL

```
SELECT “nombre_de_columna”
FROM “nombre_de_tabla”;
```

SQL - Tipos de Datos

Categorías:

- String
- Numbers
- Date/Time

Tipos de datos STRING en MySQL

CHAR(Size)	It is used to specify a fixed length string that can contain numbers, letters, and special characters. Its size can be 0 to 255 characters. Default is 1.
VARCHAR(Size)	It is used to specify a variable length string that can contain numbers, letters, and special characters. Its size can be from 0 to 65535 characters.
BINARY(Size)	It is equal to CHAR() but stores binary byte strings. Its size parameter specifies the column length in the bytes. Default is 1.
VARBINARY(Size)	It is equal to VARCHAR() but stores binary byte strings. Its size parameter specifies the maximum column length in bytes.
TEXT(Size)	It holds a string that can contain a maximum length of 255 characters.
TINYTEXT	It holds a string with a maximum length of 255 characters.
MEDIUMTEXT	It holds a string with a maximum length of 16,777,215.
LONGTEXT	It holds a string with a maximum length of 4,294,967,295 characters.
ENUM(val1, val2, val3,...)	It is used when a string object having only one value, chosen from a list of possible values. It contains 65535 values in an ENUM list. If you insert a value that is not in the list, a blank value will be inserted.
SET(val1,val2,val3,....)	It is used to specify a string that can have 0 or more values, chosen from a list of possible values. You can list up to 64 values at one time in a SET list.
BLOB(size)	It is used for BLOBs (Binary Large Objects). It can hold up to 65,535 bytes.

Tipos de datos NUMERIC en MySQL

BIT(Size)	It is used for a bit-value type. The number of bits per value is specified in size. Its size can be 1 to 64. The default value is 1.
INT(size)	It is used for the integer value. Its signed range varies from -2147483648 to 2147483647 and unsigned range varies from 0 to 4294967295. The size parameter specifies the max display width that is 255.
INTEGER(size)	It is equal to INT(size).
FLOAT(size, d)	It is used to specify a floating point number. Its size parameter specifies the total number of digits. The number of digits after the decimal point is specified by **d** parameter.
FLOAT(p)	It is used to specify a floating point number. MySQL used p parameter to determine whether to use FLOAT or DOUBLE. If p is between 0 to24, the data type becomes FLOAT (). If p is from 25 to 53, the data type becomes DOUBLE().
DOUBLE(size, d)	It is a normal size floating point number. Its size parameter specifies the total number of digits. The number of digits after the decimal is specified by d parameter.
DECIMAL(size, d)	It is used to specify a fixed point number. Its size parameter specifies the total number of digits. The number of digits after the decimal parameter is specified by **d** parameter. The maximum value for the size is 65, and the default value is 10. The maximum value for **d** is 30, and the default value is 0.
DEC(size, d)	It is equal to DECIMAL(size, d).
BOOL	It is used to specify Boolean values true and false. Zero is considered as false, and nonzero values are considered as true.

Tipos de datos DATE/TIME en MySQL

DATE	It is used to specify date format YYYY-MM-DD. Its supported range is from '1000-01-01' to '9999-12-31'.
DATETIME(fsp)	It is used to specify date and time combination. Its format is YYYY-MM-DD hh:mm:ss. Its supported range is from '1000-01-01 00:00:00' to 9999-12-31 23:59:59'.
TIMESTAMP(fsp)	It is used to specify the timestamp. Its value is stored as the number of seconds since the Unix epoch('1970-01-01 00:00:00' UTC). Its format is YYYY-MM-DD hh:mm:ss. Its supported range is from '1970-01-01 00:00:01' UTC to '2038-01-09 03:14:07' UTC.
TIME(fsp)	It is used to specify the time format. Its format is hh:mm:ss. Its supported range is from '-838:59:59' to '838:59:59'
YEAR	It is used to specify a year in four-digit format. Values allowed in four digit format from 1901 to 2155, and 0000.

Tipos de datos STRING en SQLServer

char(n)	It is a fixed width character string data type. Its size can be up to 8000 characters.
varchar(n)	It is a variable width character string data type. Its size can be up to 8000 characters.
varchar(max)	It is a variable width character string data types. Its size can be up to 1,073,741,824 characters.
text	It is a variable width character string data type. Its size can be up to 2GB of text data.
nchar	It is a fixed width Unicode string data type. Its size can be up to 4000 characters.
nvarchar	It is a variable width Unicode string data type. Its size can be up to 4000 characters.
ntext	It is a variable width Unicode string data type. Its size can be up to 2GB of text data.
binary(n)	It is a fixed width Binary string data type. Its size can be up to 8000 bytes.
varbinary	It is a variable width Binary string data type. Its size can be up to 8000 bytes.
image	It is also a variable width Binary string data type. Its size can be up to 2GB.

Tipos de datos NUMERIC en SQLServer

bit	It is an integer that can be 0, 1 or null.
tinyint	It allows whole numbers from 0 to 255.
Smallint	It allows whole numbers between -32,768 and 32,767.
Int	It allows whole numbers between -2,147,483,648 and 2,147,483,647.
bigint	It allows whole numbers between -9,223,372,036,854,775,808 and 9,223,372,036,854,775,807.
float(n)	It is used to specify floating precision number data from -1.79E+308 to 1.79E+308. The n parameter indicates whether the field should hold the 4 or 8 bytes. Default value of n is 53.
real	It is a floating precision number data from -3.40E+38 to 3.40E+38.
money	It is used to specify monetary data from -922,337,233,685,477.5808 to 922,337,203,685,477.5807.

Tipos de datos DATE/TIME en SQLServer

datetime	It is used to specify date and time combination. It supports range from January 1, 1753, to December 31, 9999 with an accuracy of 3.33 milliseconds.
datetime2	It is used to specify date and time combination. It supports range from January 1, 0001 to December 31, 9999 with an accuracy of 100 nanoseconds
date	It is used to store date only. It supports range from January 1, 0001 to December 31, 9999
time	It stores time only to an accuracy of 100 nanoseconds
timestamp	It stores a unique number when a new row gets created or modified. The time stamp value is based upon an internal clock and does not correspond to real time. Each table may contain only one-time stamp variable.

Operadores

- Operadores Aritméticos
- Operadores de Comparación
- Operadores Lógicos

Operadores Aritméticos

+
-
*
/
%

Operadores de Comparación

=
!=
< >
>
<
>=
<=
!<
!>

Operadores Lógicos

ALL
AND
ANY
BETWEEN
IN
NOT
OR
EXISTS
LIKE

Conectarse a Base de Datos

MYSQL

mysql -u root -p;
El password por defecto en mysql es vacío

Conectarse a Base de Datos

SQLSERVER

Utilizando autenticación windows

SQLCMD -S SERVERNAME -E;

SQLCMD -S SERVERNAME\INSTANCENAME

Utilizando autenticación de SQLSERVER

SQLCMD -S SERVERNAME -U sa -P pa55w0rd

Ubicación de SQLCMD por defecto C:\Program Files\Microsoft SQL Server\100\Tools\Binn\

Tipos de comandos SQL

- Lenguaje de Definición de Datos (DDL)
  - CREATE
  - DROP

- Lenguaje de Manipulación de Datos (DML)
  - INSERT
  - SELECT

Crear Base de Datos

```
CREATE DATABASE nombre_base_de_datos;
```

Borrar Base de Datos

```
DROP DATABASE nombre_base_de_datos;
```

Seleccionar Base de Datos

```
USE nombre_base_de_datos;
```

Crear Tabla

```
CREATE TABLE nombre_de_tabla
(“columna1” “tipo_de_dato”,
“columna2” “tipo_de_dato”,
...
“columnaN” “tipo_de_dato”);
```

Crear Tabla

```
CREATE TABLE carrera (nombre VARCHAR(30),
semestres INT);
```

Describir Tabla

```
DESCRIBE nombre_de_tabla;
```

Equivalente en MS SQLServer:
exec sp_columns doughnut_list

Renombrar Tabla

```
ALTER TABLE nombre_de_tabla RENAME TO
nuevo_nombre_de_tabla;
```

Añadir Columnas a una Tabla

```
ALTER TABLE nombre_de_tabla ADD
nombre_de_columna definición_de_columna;
```

Copiar Tabla

```
CREATE TABLE tabla_nueva LIKE tabla_original;

INSERT INTO tabla_nueva SELECT * FROM
tabla_original;

CREATE TABLE tabla_nueva SELECT * FROM
tabla_original;
```

Truncate Tabla

```
TRUNCATE TABLE nombre_de_tabla;
```

Remueve todos los registros de una tabla, es una operación DDL, no permite borrado selectivo, no se activan los triggers

Borrar Filas de una Tabla

```
DELETE FROM nombre_de_tabla [WHERE condición];
```

Remueve registros de una tabla, es una operación DML, permite borrado selectivo, pueden activarse los triggers

Borrar Tabla

```
DROP TABLE “nombre_de_tabla”;
```

Clave Primaria

```
CREATE TABLE  estudiantes
(S_Id int NOT NULL,
Apellido varchar (255) NOT NULL,
Nombre varchar (255),
Direccion varchar (255),
Ciudad varchar (255),
PRIMARY KEY (S_Id));

-- SQLServer Access, Oracle
S_Id int NOT NULL PRIMARY KEY,
```

Clave Foránea (MySQL)

```
CREATE TABLE pedidos
( P_Id int NOT NULL,
Pedido_No int NOT NULL,
S_Id int, PRIMARY KEY (P_Id),
FOREIGN KEY (S_Id) REFERENCES Personas (S_Id));
```

Clave Foranea (SQLServer, Access, Oracle)

```
CREATE TABLE pedidos
( P_Id int NOT NULL PRIMARY KEY,
Pedido_No int NOT NULL,
S_Id int FOREIGN KEY REFERENCES Personas
(S_Id));
```

Composite Key

```
CREATE TABLE TABLA_EJEMPLO
(COL1 integer,
COL2 varchar(30),
COL3 varchar(50),
PRIMARY KEY (COL1, COL2));
```

Insert Statement

```
INSERT INTO nombre_de_tabla
[(col1, col2, col3,... colN,)]
VALUES (valor1, valor2, valor 3, ...valor N);
```

Update Statement

```
UPDATE nombre_de_tabla SET
[nombre_de_columna1 = valor1,...
nombre_de_columnaN = valorN] [WHERE
condicion];
```

Delete Statement

```
DELETE FROM nombre_de_tabla [WHERE condición];
```

Select

```
SELECT expresiones
FROM tablas
WHERE condiciones;
```

Select Distinct

```
SELECT DISTINCT nombre_de_columna
FROM nombre_de_tabla;
```

Se utiliza para recuperar un elemento único o distinto de la tabla.

Nota: UNIQUE es una sintaxis antigua que se usó en la descripción de Oracle, pero luego el estándar ANSI define DISTINCT como la palabra clave oficial.

Order By con orden ascendente

```
SELECT ciudad_del_proveerdor
FROM proveedores
ORDER BY ciudad_del_proveedor;
```

Order By con orden Descendente

```
SELECT ciudad_del_proveerdor
FROM proveedores
ORDER BY ciudad_del_proveedor DESC;
```

Select Top

```
SELECT TOP (expresión);

SELECT * FROM nombre_tabla LIMIT valor_limite;
```

Select First

SELECT FIRST (nombre_de_columna) FROM nombre_de_tabla;
Solo soportado en algunos DBMS

SELECT (nombre_de_columna) FROM nombre_de_tabla LIMIT 1;

Select - Operaciones Matemáticas

- Sumar (+)
- Restar (-)
- Multiplicar (*)
- Dividir (/ , div)
- Modulo (mod)

Select - Operaciones Matemáticas

```
SELECT 5+2, 5-2, 5*2;

SELECT 5/2, 5 div 2, 5 mod 2;
```

Select - Funciones de Agregación

- count()
- sum()
- min()
- max()
- avg()

Select - Función Count

```
SELECT COUNT (expresión)
FROM tablas
WHERE condiciones;
```

Select - Función Sumar

```
SELECT SUM (expression)
FROM tablas
WHERE condiciones;
```

Group by

```
SELECT COUNT(nombre_columna)
FROM tabla
GROUP BY nombre_columna;
```

Select As

```
SELECT dia_del_pedido AS ”Fecha”
Cliente AS “Cliente”,
Producto,
Cantidad
FROM pedidos;
```

Select Null

No debemos comparar valor nulo con 0. Estos no son equivalentes

Where

```
SELECT columna1, columna2,...columna n
FROM nombre_de_tabla
WHERE [condiciones];
```

And

```
SELECT columnas
FROM tablas
WHERE condicion 1
AND condicion 2;
```

OR

```
SELECT columnas
FROM tablas
WHERE condicion 1
OR condición 2;
```

IN

Expresion IN (valor 1, valor 2...valor n);

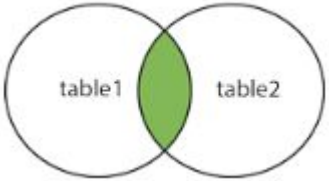
```
SELECT *
FROM estudiantes
WHERE nombre_de_estudiantes IN (Juan, Alex,
Elmer);
```

Join

Sirven para combinar registros de dos o más tablas

- INNER JOIN
- LEFT JOIN
- RIGHT JOIN
- OUTER JOIN

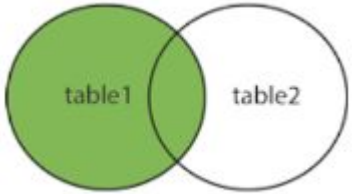
INNER JOIN



Devuelve todas las filas cuando hay al menos una coincidencia entre las columnas de ambas tablas

```
SELECT nombre_columna(s)
FROM tabla1
INNER JOIN tabla2
ON tabla1.nombre_colum = tabla2.nombre_colum
```

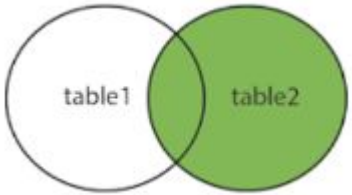
LEFT JOIN



Devuelve todas las filas de la tabla de la izquierda y las filas coincidentes de la tabla de la derecha

```
SELECT nombre_columna(s)
FROM tabla1
LEFT JOIN tabla2
ON tabla1.nombre_colum = tabla2.nombre_colum
```

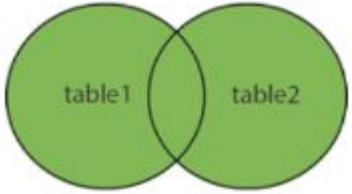
RIGHT JOIN



Devuelve todas las filas de la tabla de la derecha y las filas coincidentes de la tabla de la izquierda

```
SELECT nombre_columna(s)
FROM tabla1
RIGHT JOIN tabla2
ON tabla1.nombre_colum = tabla2.nombre_colum
```

OUTER JOIN



Devuelve todas las filas de las dos tablas, la izquierda y la derecha (FULL OUTER JOIN)

```
SELECT nombre_columna(s)
FROM tabla1
OUTER JOIN tabla2
ON tabla1.nombre_colum = tabla2.nombre_colum
```

Join - Ejemplo

Tabla: CLIENTE

idCliente	nombreCliente
1	Juan
2	Alex
3	Edwin
4	Claudia

Tabla: PEDIDO

idPedido	idCliente	factura
234	4	160
235	2	48
236	3	64
237	4	92

INNER JOIN - MySQL

```
SELECT Cliente.nombreCliente, Pedido.idPedido
FROM Cliente
INNER JOIN Pedido
ON Cliente.idCliente = Pedido.idCliente
```

LEFT JOIN & RIGHT JOIN - MySQL

```
SELECT Cliente.nombreCliente, Pedido.idPedido
FROM Cliente
LEFT JOIN Pedido
ON Cliente.idCliente = Pedido.idCliente

SELECT Cliente.nombreCliente, Pedido.idPedido
FROM Cliente
RIGHT JOIN Pedido
ON Cliente.idCliente = Pedido.idCliente;
```

OUTER JOIN - MySQL

```
SELECT Cliente.nombreCliente, Pedido.idPedido
FROM Cliente
LEFT JOIN Pedido
ON Cliente.idCliente = Pedido.idCliente
UNION
SELECT Cliente.nombreCliente, Pedido.idPedido
FROM Cliente
RIGHT JOIN Pedido
ON Cliente.idCliente = Pedido.idCliente;
```

FIN