SQL

(Structured Query Language)

INDICE

¿Qué es SQL?

- Es un lenguaje de programación utilizado para gestionar y manipular bases de datos relacionales. Específicamente, SQL permite a los usuarios realizar consultas, insertar, actualizar y eliminar datos, así como crear, alterar y eliminar tablas y estructuras relacionadas.

- Por otro lado, MySQL es un sistema de gestión de bases de datos relacionales de código abierto. Es conocido por su rápida velocidad de procesamiento y su fiabilidad, así como por ser una solución asequible para gestionar bases de datos.

INSTALACION DE MYSQL

Buscamos en internet:

mysql community server installer

Download Mysql Installer

Windows (x86, 32-bit), MSI Installer 425.7 M Download

No thanks, just start my download

Se inicia la descarga de MYSQL

INSTALACION DE MYSQL

Busco en mis ficheros y en descargas y ejecuto el
mysql installer-community-8.0.28.0

Se abre el MYSQL Installer

Choosing a Setup Type
Elegimos Custom
Next
Select products
MYSQL Servers

Elegimos MYSQL Server 8.0.28 – x64 y presionamos la flecha verde para que aparezca en Products to be installed

INSTALACION DE MYSQL

Tambien elegimos MYSQL Shell 8.0.28 – x64 y con la flecha verde la pasamos a la ventana de la derecha (Products to be installed)

También elegimos MYSQL Workbench 8.0.28 – x64 y con la flecha verde también la pasamos a la ventana de la derecha

NEXT

Se abre la ventana Installation y veremos los 3 productos elegidos anteriormente

MYSQL Server 8.0.28 – X64

MYSQL Shell 8.0.28 – X64

MYSQL Workbench 8.0.28 – X64

EXECUTE

INSTALACION DE MYSQL

Cuando se acaba la instalación presionamos NEXT

Otra vez NEXT

Otra vez NEXT

En Authenticaton Method lo dejamos como esta y presionamos NEXT

Sale la ventana Accouts and Roles

Hay que indicarle la contraseña que debemos memorizar para poder entrar en el MYSQL Workbench la próxima vez

MYSQL Root Password

Repeat Password

INSTALACION DE MYSQL

Despues de meter la contraseña presionamos NEXT

Aparece la ventana Windows Service

No tocamos nada y presionamos NEXT

Aparece la ventana Apply Configuration

No tocamos nada y presionamos Execute

Presionamos FINISH

Presionamos NEXT

INSTALACION DE MYSQL

Sae abre la ventana Installation Complete

Dejamos en blanco el cuadrado a la izquierda de

Start Mysql Shell after setup

Solo dejamos seleccionada la opción Start Mysql Workbench after setup

Presionamos FINISH

Se debe abrir ventana grande que dice

Welcome to Mysql Workbench

INSTALACION DE MYSQL

Debe aparecer un cuadrado en color gris que se llama:

Local instance Mysql 80

root

localhost :3306

Presionamos ese cuadrado gris y se abre ventana donde nos pide la contraseña

INSTRUCCIONES DE SQL

- A continuación vamos a ver las instrucciones más básicas en MySQL divididas en secciones. Pero antes de empezar te enseñaré 2 auxiliares que te vendrán bien para comprobar los resultados de lo que vayas haciendo;

SELECT * FROM nombretabla;

se usa para ver el resultado de una tabla con todos sus registros.

DESCRIBE nombretabla;

se usa para ver la estructura de una tabla.

Para abrir un nuevo proyecto presionamos a la izquierda debajo de la casita en SQL con el signo +. Para ejecutar código hay dos maneras: señalar todo el código y presionar el rayo amarillo que esta debajo de los nombres de los proyectos, la segunda manera es colocar el cursor encima del código y clickar y presionar el rayo amarillo con la I.

Para hacer comentarios al margen que no afecten al programa usamos dos maneras:

--

/* */

CREAR UNA BASE DE DATOS

- Desde el punto de vista de SQL, una base de datos es sólo un conjunto de relaciones (o tablas). A nivel de sistema operativo, cada base de datos se guarda en un directorio diferente.
- Por tanto, crear una base de datos es una tarea muy simple. Claro que, en el momento de crearla, la base de datos estará vacía, es decir, no contendrá ninguna tabla.
- Para crear una base de datos por ejemplo de una empresa, se usa

```
CREATE DATABASE empresa;

USE empresa;
```

esto es para no tener que repetir el nombre de la base de datos en consultas y seleccionar esa base de datos por defecto

CREAR UNA TABLA

Para crear una tabla por ejemplo de clientes, dentro de una base de datos ya creada usamos la sintaxis siguiente:

```
CREATE TABLE clientes(
    código int,
    nombre varchar(30)
    );
```

Hemos creado una tabla llamada clientes con solo 2 columnas: 1.-código, que usara como valores un numero entero (int), 2.-nombre, que usara una cadena de caracteres (varchar) con un numero máximo de caracteres dentro del paréntesis, el varchar(49) aceptaría hasta 49 letras, números, signos, espacios en blanco

TIPOS DE DATOS PARA CREAR UNA TABLA

Algunos de los tipos de datos mas usados son:

1.- Numéricos

INT Numero entero

FLOAT(m, d) Numero decimal donde m es el total de números y d la cantidad de numero después de la coma

2.- Fecha

DATE AAAA-MM-DD

TIMESTAMP AAAA-MM-DD 00:00:01

YEAR 2023

TIPOS DE DATOS PARA CREAR UNA TABLA

3.- Cadena de caracteres (string)

VARCHAR ( ) Ocupacion variable cuya longitud comprende de 1 y 255 caracteres

CHAR( ) Ocupación fija cuya longitud comprende de 1 a 255 caracteres.

Si queremos que no este permitido valores nulos usaremos NOT NULL

```
CREATE TABLE ciudades (
        nombre VARCHAR (20) NOT NULL,
         población INT NOT NULL
         );
```

CLAVES PRIMARIAS

- Sólo puede existir una clave primaria en cada tabla, y el atributo sobre el que se define una clave primaria no puede tener valores NULL. Sirven para identificar a cada elemento y no puede haber 2 claves primarias iguales en la tabla

```
CREATE TABLE productos (
        ID INT NOT NULL PRIMARY KEY,
         nombre VARCHAR(20),
         precio INT
         );
```

CLAVES PRIMARIAS

Columnas autoincrementadas

Se suele usar con las claves primarias para no tener que darle a cada fila un numero de ID, el programa le dará automáticamente un numero creciente a cada fila por ejemplo: 1, 2, 3, etc

```
CREATE TABLE facturas (
        ID INT AUTO_INCREMENT PRIMARY KEY,
         importe INT,
          fecha DATE
          );
```

BORRAR UNA TABLA

Para borrar una tabla o una base de datos usamos

DROP TABLE clientes;

A veces resulta la manera mas rápida y sencilla de modificicar varios elementos de una tabla que acabamos de crear

MODIFICAR ESTRUCTURA DE UNA TABLA

Para modificar una tabla usamos ALTER TABLE

1.- Si queremos cambiar el nombre de la tabla

```
ALTER TABLE clientes RENAME misclientes;
```

Si queremos cambiar el nombre de una columna

```
ALTER TABLE clientes RENAME COLUMN edad TO años;
```

2.- Para eliminar una columna de una tabla

```
ALTER TABLE clientes DROP COLUMN ciudad;
```

3.- Para insertar una nueva columna al final de la tabla

```
ALTER TABLE clientes ADD ciudad VARCHAR(20);
```

MODIFICAR ESTRUCTURA DE UNA TABLA

4.- Para añadir una nueva columna después de otra columna

```
ALTER TABLE clientes ADD facturación INT AFTER nombre;
```

5.- Para añadir una nueva columna en la primera posición de la tabla

```
ALTER TABLE clientes ADD id INT FIRST;
```

6.- Si queremos modificar alguna de las columnas, por ejemplo que la columna codigopostal sea también NOT NULL

```
ALTER TABLE clientes MODIFY COLUMN codigopostal INT NOT NULL;
```

INSERCION DE FILAS

Una vez creada la tabla y sus columnas tenemos que crear las filas con sus valores

La forma mas común de insertar una fila es:

```
INSERT INTO clientes (id, nombre, facturación)
VALUES (1, ‘Rodarsa’, 15000), (2, ‘Miresa’, 20000), (3, ‘Patesa’, 18000);
```

Se puede volver a usar el INSERT INTO más adelante si nos damos cuenta que necesitamos añadir más filas a la tabla.

REEMPLAZAR FILAS

Si queremos modificar un registro anterior cuando hicimos el INSERT INTO

REPLACE INTO clientes (id, nombre, facturación) VALUES (3, ‘Patesa’, 40000);

Con que el ID no varie podemos variar el resto de los encabezados de las columnas

ACTUALIZAR FILAS

Con la sentencia UPDATE podemos actualizar los valores de las filas, los cambios se aplicaran a las filas y columnas que especifiquemos

UPDATE clientes SET facturación =25000;

En este caso nos cambiara la facturación de las 3 filas a la cantidad de 25000 euros.

UPDATE clientes SET facturación = 25000 WHERE id=1;

Esta vez solo se modificara la facturación en la fila del cliente cuyo id sea igual a 1.

Para DELETE y UPDATE necesitamos usar el id por cuestión de seguridad del sistema para evitar eliminar mucha información necesaria. Nos arrojaría ERROR 1175 si usaramos otra columna en vez de la de id

ELIMINAR FILAS

Para eliminar filas usamos la sentencia DELETE FROM

Si queremos eliminar todas las filas de la tabla clientes

DELETE FROM clientes;

Si solo queremos eliminar determinadas filas usamos la clausula WHERE

DELETE FROM clientes WHERE id=2;

Esta vez solo eliminamos la fila cuyo id es 2

VACIAR UNA TABLA

Una manera rápida de eliminar todas las filas de una tabla es usando la sentencia TRUNCATE, sucede algo parecido a cuando usamos DELETE para eliminar todas las filas también, pero en el caso de TRUNCATE es mejor ya que borra la tabla y la vuelve a crear vacia lo que es mejor

```
TRUNCATE clientes;
```

CONSULTAR UNA TABLA

Si queremos mostrar todos los registros de una tabla usamos

```
SELECT * FROM clientes;
```

en este caso el asterisco quiere decir ‘todas las columnas’

Si queremos que solo nos muestre una fila usamos

```
SELECT*FROM clientes LIMIT 1;
```

Si queremos que la consulta solo nos muestre unas columnas de la tabla usaremos:

```
SELECT id, nombre  FROM clientes;
```

CONSULTAR UNA TABLA

Tambien podemos mostrar la misma información pero cambiando el encabezado de una columna usando el comando AS

SELECT nombre, facturación AS ventas FROM clientes;

En una consulta podría haber registros repetidos si queremos evitar las filas repetidas usamos el comando DISTINCT

SELECT DISTINCT nombre, facturación FROM clientes;

CONSULTAR UNA TABLA

Si queremos filtrar las filas que aparecene n la consulta podemos usar la sentencia WHERE por ejemplo en el caso de la tabla de clientes

```
SELECT * FROM clientes WHERE facturación >= 15000;
```

Podemos filtrar aun mas usando el comando AND

```
SELECT * FROM clientes  WHERE facturación >15000 AND id<2;
SELECT *FROM clientes WHERE facturación >15000 OR id<2;
SELECT *FROM clientes WHERE facturación != 15000;
SELECT*FROM clientes WHERE nombre like ‘%sa%’;
SELECT * FROM clientes ORDER BY facturación DESC;
SELECT  MAX (facturación) AS MAYOR from clientes;
```

CONSULTAR UNA TABLA

Si usamos la clausula LIMIT podemos usar 2 parámetros, donde el primer parámetro muestra la fila a partir de la que debemos empezar a mostrar y el segundo parámetro muestra el numero de filas a mostrar

```
SELECT * FROM clientes LIMIT 1,3;
```

AGRUPAR FILAS

Es posible agrupar filas en la salida de una sentencia SELECT según los distintos valores de una columna, usando la cláusula GROUP BY. Esto, en principio, puede parecer redundante, ya que podíamos hacer lo mismo usando la opción DISTINCT. Sin embargo, la cláusula GROUP BY es más potente ya que permite usar funciones

```
SELECT ciudad FROM clientes GROUP BY ciudad;
```

Con la sentencia GROUP BY la salida se ordena según los valores de la columna indicada, en este caso se ordenan alfabéticamente por el nombre de la ciudad

AGRUPAR FILAS

Como dijimos antes la clausula GROUP BY permite usar funciones como por ejemplo COUNT( ) que sirve para contar las cantidades

```
SELECT ciudad, COUNT(*) AS cuenta FROM clientes GROUP BY ciudad;
```

El asterisco dentro del paréntesis de COUNT significa todas las filas

Además de COUNT podemos usar otras funciones como AVG( ), MAX( ), MIN( ), etc.

OPERADORES DE COMPARACION

Operadores	Signos
Igualdad	=
SELECT *FROM clientes WHERE facturación=20000;	
Distinto	<>
SELECT*FROM clientes WHERE facturación <> 20000;	
Menor que, menor o igual que	< <= > >=
SELECT *FROM clientes WHERE facturación <=25000;	
Dentro de rango	BETWEEN …..AND ……..
SELECT*FROM clientes WHERE facturación BETWEEN 15000 and 25000;	

OPERADORES ARITMETICOS

Operador	Signo
Suma	+
Resta	-
Multiplicacion	*
Division	/
Resto	%

OTROS OPERADORES

Operador	Significado
AND	y
OR	o
NOT	no
IN	dentro de
NOT IN	no dentro de

CONSULTAS MULTITABLA

Hasta el momento hemos visto como hacer consultas en una sola tabla. También es posible realizar consultas en varias tablas. Ya hemos visto las claves primarias (PRIMARY KEYS) y ahora vamos a ver las claves foráneas (FOREIGN KEYS).

La clave foránea es una columna de una tabla que sirve para vincular dicha tabla con la columna de Clave Primaria de otra tabla. Vamos a ver un ejemplo

.- Crea tabla departamento

```
CREATE TABLE departamento(
id INT AUTO_INCREMENT PRIMARY KEY,
nombre VARCHAR(25) NOT NULL,
presupuesto INT NOT NULL,
gastos INT NOT NULL);
```

.- Crea tabla empleado

```
CREATE TABLE empleado(
id INT AUTO_INCREMENT PRIMARY KEY,
nif VARCHAR(9) NOT NULL,
nombre VARCHAR(25),
apellido1 VARCHAR(100) NOT NULL,
apellido2 VARCHAR(100),
id_departamento INT,
FOREIGN KEY (id_departamento) REFERENCES departamento(id));
```

- Para ello SQL dispone de diversas formas de hacer consultas multitabla que explicamos a continuación.

- Composiciones internas

- Composiciones externas

-

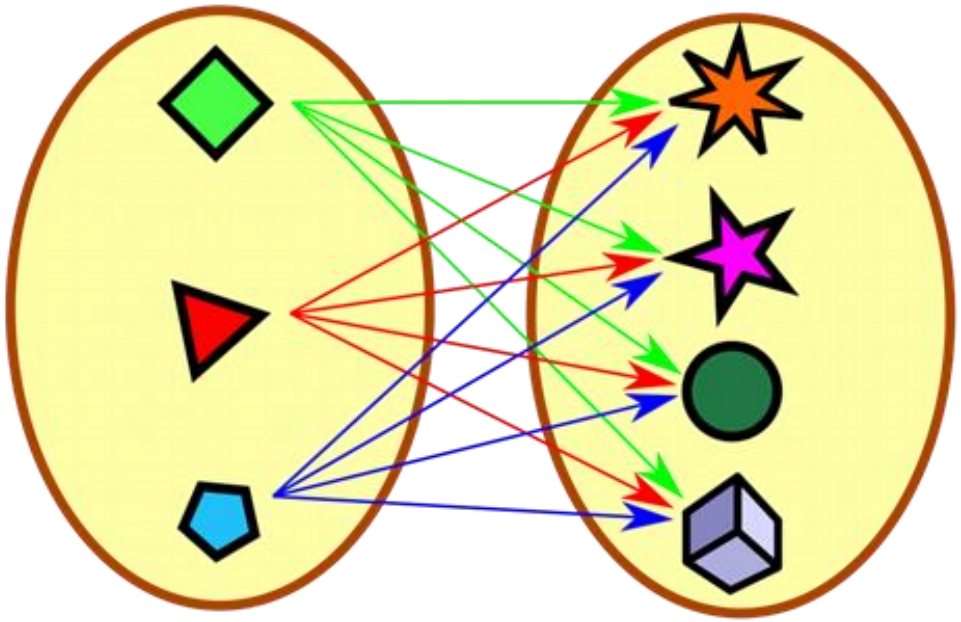
- Las **Composiciones internas** son aquellas que parten de un producto cartesiano entre dos tablas, es decir, la combinación de todas las filas de una tabla con las filas de la otra tabla. A partir de ese producto cartesiano de tablas, se pueden hacer filtros o restricciones al resultado.

- Las **Composiciones externas** no se originan de un producto cartesiano sino que se toma un registro de una tabla y se busca registros de otra tabla que coincidan en el atributo indicado.

CONSULTAS MULTITABLAS

En matemáticas un producto cartesiano sobre 2 conjuntos es el conjunto resultante de las combinación de todos los elementos del primer conjunto emparejados con cada uno de los elementos del segundo conjunto

Producto cartesiano de AxB
A=(3, 4)
B=(5,6,7)
AXB=(3,5), (3, 6), (3, 7), (4, 5), (4, 6), (4, 7)



CONSULTAS MULTITABLAS

COMPOSICIONES INTERNAS

```
SELECT* FROM tabla1 JOIN tabla2;
```

```
SELECT*FROM tabla JOIN tabla2 ON(tabla1.atributo=tabla2.atributo);
```

Para realizar el producto cartesiano de 2 tablas jugador y pichichi usamos el comando JOIN, por ejemplo:

Tabla jugadores

id	nombre
1	Cristiano
2	Messi
3	Suarez
4	Aspas
5	Benzema

Tabla goleadores

id	goles
1	35
3	20
5	15

Para hacer el producto cartesiano debemos realizar un JOIN entre ambas tablas. Como podemos observar van a aparecer los 2 atributos de cada tabla y cada registro de la tabla jugadores se ha combinado con cada registro de la tabla goleadores

SELECT*FROM jugadores JOIN goleadores;

id	nombre	id	goles
1	Cristiano	1	35
1	Cristiano	3	20
1	Cristiano	5	15
2	Messi	1	35
2	Messi	3	20
2	Messi	5	15
3	Suarez	1	35
3	Suarez	3	20
3	Suarez	5	15
4	Aspas	1	35
4	Aspas	3	20
4	Aspas	5	15
5	Benzema	1	35
5	Benzema	3	20
5	Benzema	5	15

Ahora queremos hacer una consulta diferente y solo deseamos mostrar las filas donde el ‘id’ de jugadores coincida con el ‘id’ de goles, para ello deberíamos usar la restricción ON, que funciona parecida a la restricción WHERE que ya hemos visto anteriormente

```
SELECT*FROM jugadores JOIN goleadores ON(jugadores.id=goleadores.id);
```

```
id | nombre      | id |  goles |

| 1 | Cristiano   | 1 |      35 |

| 3 | Suarez      | 3 |      20 |
| 5 | Benzema           | 5 |          15 |
```

Esta vez solo se muestran las filas cuyos ‘id’ coinciden

Podemos simplificar la expresión anterior usando el NATURAL JOIN el cual automáticamente busca los atributos de ambas tablas coincidentes.

Por ejemplo, para la anterior consulta podríamos hacer esta vez

SELECT*FROM jugadores NATURAL JOIN goleadores;

id | nombre | goles |

1	Cristiano	35
3	Suarez	20
5	Benzema	15

Ahora solo aparece un campo ‘id’ que unifica a los 2 ‘id’ de las tablas. Para poder realizar el NATURAL JOIN, los campos han de llamarse igual

CONSULTAS MULTITABLAS

- Composiciones externas

Para realizar las composiciones externas no partimos del producto cartesiano de las tablas, sino que por cada una de las filas de la primera tabla se buscará una fila en la segunda tabla cuyo atributo en común coincida.

Usaremos la siguiente sintaxis

SELECT … FROM *tabla1* LEFT JOIN *tabla2* ON (*tabla1.atributo =tabla2.atributo*) ;

SELECT … FROM *tabla1* RIGHT JOIN *tabla2* ON (*tabla1.atributo =tabla2.atributo*) ;

- Por ejemplo, imaginemos que al anterior ejemplo le añadimos 2 entradas en la tabla goleadores con “id” que no aparecen en la tabla jugadores.

Tabla goleadores

| id | goles |

1	35
3	20
5	15
11	6
12	3

Tabla jugadores

| id | nombre|

1	Cristiano
2	Messi
3	Suarez
4	Aspas
5	Benzema

Como se puede observar en la tabla jugadores hay algunos ‘id’ que no aparecen en la tabla goleadores y viceversa

- **LEFT JOIN** hace referencia a que la consulta se hará por cada uno de los registros de la primera tabla. Si un registro de la primera tabla no coincide con el campo común de cualquier registro de la segunda tabla, se usará NULL para los campos que no puedan ser rellenados.

- Por ejemplo, si se desea mostrar todos los registros de jugadores de la tabla “jugadores” junto con sus goles correspondientes de la tabla “goleadores”, y que INCLUSO los jugadores de la tabla “jugadores” que no tienen goles en la tabla “goleadores” también aparezcan, usaríamos un **LEFT JOIN**.

```
SELECT*FROM jugadores LEFT JOIN goleadores ON (jugadores.id=goleadores.id);
```

```
| id | nombre      | id     |      goles |

| 1 |Cristiano     |1       |         35|
| 2 | Messi        | NULL   |       NULL |

| 3 | Suarez       |3       |         20 |
| 4 | Aspas        | NULL   |       NULL |

| 5 | Benzema      |5       |         15 |
```

Como se puede observar, los jugadores con id=2 y id=4 no aparecen en la table goleadores, por tanto los campos pertenecientes a las columnas de la table goleadores se han rellenado con NULL

- Lo mismo ocurre con **RIGHT JOIN**, solo que en vez de crear el resultado recorriendo cada uno de los registros de la primera tabla, se usará con los registros de la segunda.

- Siguiendo el ejemplo, al usar RIGHT JOIN, por cada registro de la tabla "goleadores" se buscarán registros en la tabla "jugadores" que coincidan en el campo común (el "id").

SELECT*FROM jugadores RIGHT JOIN goleadores ON (jugadores.id=goleadores.id);

id	nombre	id	goles
1	Cristiano	1	35
3	Suarez	3	20
5	Benzema	5	15
NULL	NULL	11	6
NULL	NULL	12	3

- Al igual que en las composiciones internas, con NATURAL JOIN nos podemos ahorrar la necesidad de indicarle a MySQL qué campo en común tienen ambas tablas.

- Por tanto, en el ejemplo, la última sentencia de MySQL usada podría substituirse por la siguiente:

```
SELECT *FROM jugadores NATURAL RIGHT JOIN goleadores;
|id    | goles    | nombre |

| 1    |    35    | Cristiano |

| 3    |    20    | Suarez |
| 5    |    15    | Benzema |

| 11   |    6     | NULL |
| 12   |    3     | NULL |
```

Como se puede observer, no es necesario especificar con ON cual es el campo en comun, y ademas en la tabla resultante de la consulta se unifica el campo ‘id’ de ambas tablas

- Lo mismo aplica para NATURAL LEFT JOIN, solo que en vez de la segunda tabla(RIGHT) se formará la consulta a partir de la primera(LEFT)

SELECT *FROM jugadores NATURAL LEFT JOIN goleadores;

|id | nombre | goles |

| 1 | Cistiano | 35|
| 2 | Messi | NULL |

3	Suarez	20
4	Aspas	NULL
5	Benzema	15

SUBCONSULTAS

Una subconsulta es una consulta dentro de otra consulta (a esta segunda la llamamos la principal). Se suelen colocar en la cláusula WHERE de la consulta principal, pero también pueden añadirse en el SELECT o en el FROM.

- Para una subconsulta situada en la cláusula WHERE puede usarse los operadores de comparación (> , >=, <, <=, !=, =). Cuando esto ocurre se realiza una comparación entre el valor indicado de la consulta principal y el valor resultante de la subconsulta. Y como es normal en las comparaciones, ambos tipos de datos deben ser iguales.

- En el modo sencillo, cuando se usa operadores de comparación, la consulta interna (osea la subconsulta) devuelve 1 solo registro y 1 sola columna.

SUBCONSULTAS

- Si el resultado de la subconsulta devuelve más de un registro, se usan comandos especiales que se escriben entre el operador de comparación y la subconsulta.

- ANY : compara cada registro de la consulta principal con los de la subconsulta. Si la comparación entre un registro de la consulta principal con otro de la subconsulta se valida, el registro de la principal pasará el filtro.

- ALL : en este caso un registro debe validarse para todos los registros de la subconsulta.

- IN : En este caso no se usa el comparador. Se comprueba si cada registro de la consulta principal está o no contenido en la tabla resultante de la subconsulta.

- NOT IN : lo opuesto al IN.

Imaginemos que tenemos las siguientes tablas

Tabla materiales

codigo	nombre
1	polietileno
2	alumnio
3	cobre

Tabla proveedores

id	nombre
1001	Sipem
1002	Alcur
1003	Fendor
1004	Dragon

Tabla suministros

códigopieza	idproveedores	precio
1	1001	28
1	1002	30
1	1003	23
2	1001	20
2	1002	21
3	1002	44
3	1004	34

- ¿Cómo realizaríamos una consulta que nos devolviese el nombre de los proveedores quesuministran la pieza 1 ?

- Podemos hacerlo de 2 formas, con una subconsulta o con una consulta multitabla.

- Para hacerlo como una consulta multitabla bastaría con hacer un JOIN … ON… tal y comose explica en el apartado de consultas multitabla.

- Para hacerlo como una subconsulta realizariamos una consulta que muestre los nombresde los proveedores que suministran una pieza determinada:

SELECT nombre FROM proveedores WHERE id IN(SELECT idproveedores from suministros WHERE codigopieza=1);

|nombre|

|Sipem|

|Alcur|

|Fendor|

Como se puede observar, en este caso hemos realizado la subconsulta usando el IN, para ver que proveedores estaban en la lista devuelta por la subconsulta

Más subconsultas

SELECT….. FROM…WHERE NOT IN (SELECT….. FROM ….. WHERE…….);

NOT IN es lo opuesto a IN

SELECT……FROM…..WHERE >= ANY (SELECT…. FROM ……WHERE…….);

Compara cada registro de la consulta principal con los de la subconsulta. Si la comparación entre un registro de la consulta principal

con otro de la subconsulta se valida el registro de la principal pasara el filtro

SELECT……FROM……WHERE >= ALL(SELECT…..FROM…..WHERE……);

Con el ALL un registro debe validarse para todos los registros de la subconsulta

SELECT……FROM….WHERE >= (SELECT….FROM….WHERE……);

Funciones de grupo

COUNT Devuelve el numero de valores distintos de NULL

AVG Devuelve al valor medio

MIN Devuelve el valor minimo

MAX Devuelve el valor maximo

STD Devuelve la desviación estandar

SUM Devuelve la suma

VARIANCE Devuelve la varianza

DIAGRAMAS

En el MySQL Workbench debajo del delfin, arriba a la izquierda presionamos la casita. Se abren 3 opciones presionamos la del medio que son 3 tablas unidas por unas líneas.

Presionamos el signo + junto a Models

Presionamos Add Diagram, se abre el cuadro donde crear el diagrama de cada una de las tablas

Presionamos 2 veces ‘user’ a la derecha del cuadro y se abre un diagrama por defecto para que le cambiemos las columnas que queramos, presionamos user cada vez que queramos crear una tabla nueva.

DIAGRAMAS

- Vemos que salen diagramas con columnas ya creadas, clickamos dos veces encima del nombre del diagrama y se abre una sección debajo donde podemos modificar los nombres de la tabla, los nombres de las columnas, los tipos de datos, etc.
- Presionamos sobre el nombre de la tabla y le ponemos el nuevo nombre que queramos
- Presionamos dos veces sobre el nombre de la columna que viene por defecto y nos deja ponerle el nuevo nombre a la columna y cambiarle el tipo de datos (INT, VARCHAR(20), etc.

- Con el botón izquierdo presionado sobre el nombre de la columna que viene por defecto podemos presionar CUT y desaparece esa columna

DIAGRAMAS

Para mover arriba o abajo el nombre de una columna click derecho y presionamos MOVE UP o MOVE DOWN

Una vez que hemos creado las tablas y hemos marcado los cuadrados de PK (clave primaria), NN (not null), UQ (unique), por ejemplo vamos a crear las FK (claves foráneas).

Abajo en la pantalla presionamos en Foreign Keys para crear la relación entre las diferentes tablas.

Nos aparecen dos cuadros en la parte de debajo de la pantalla, en la de la izquierda tenemos que llenar los datos de Foreign key y Referenced Table. Debajo de Foreign key escribiremos el nombre de la columna que sirve de clave foránea y debajo de Referenced Table escribimos la otra tabla con la que se relaciona la tabla donde esta le Foreign key.

DIAGRAMAS

En el cuadro de la derecha tenemos que rellenar 2 datos que son Column y Referenced columna.

En Colum escribimos de nuevo el nombre de la columna que sirve de clave foránea. Debajo de Referenced column escribimos el nombre de la clave primaria de la Referenced Table que escribimos en el cuadro de la izquierda

El borde del rombo a la izquierda de la Foreign key debe ponerse de color rojo. Si al principio, cuando creamos esa Foreign key le pusimos que fuera NOT NULL entonces el rombo entero a la izquierda de la Foreign key se pondrá de color rojo.

IMPORTAR BASE DE DATOS DE www.kaggle.com

Kaggle es una plataforma web que reúne la comunidad Data Science más grande del mundo, con más de **536 mil miembros activos en 194 países**, recibe más de 150 mil publicaciones por mes, que brindan todas las herramientas y recursos más importantes para progresar al máximo en data science

Creas una base de datos

CREATE DATABSE importKaggle;

USE importKaggle;

En la sección de Schema a la izquierda de la pantalla presionamos botón derecho sobre Tables debajo del nombre de la base de datos que acabamos de crear, se abre desplegable y presionamos en

Table Data Import Wizard

IMPORTAR BASE DE DATOS DE www.kaggle.com

Presionamos sobre

Table Data Import Wizard

Se abre ventana y buscamos en nuestros archivos una base de datos en formato .csv y despues presionamos

Browse

Después presionamos Next en cada ventana nueva que se abre hasta que al final presionamos Finish en la ultima ventana que se abre.

Refrescamos en Schemas y vemos que ya esta creada la Tabla con la base de datos que hemos importado de Kaggle.

No siempre va a ser tan fácil ya que algunas veces la tabla de Kaggle incluirá signos o símbolos que el MYSQL no acepte, entonces tendremos que limpiar esa base de datos para que MYSQL la pueda aceptar