

NODE.JS

wawiwa

SQL

¿Qué es SQL?

Un lenguaje que permite el manejo de tablas de una base de datos tabular y la ejecución de cualquier operación en ellas (leer, actualizar y agregar).

El American Standards Institute ha definido a SQL como un lenguaje estándar para trabajar con tablas de bases de datos.

¿Qué es una tabla?

muestra de una tabla que contiene 5 columnas - personId, fname, lname, salary, birthday
La tabla contiene 4 filas. Cada fila representa una persona y muestra los detalles de la misma.

personId	fname	lname	salary	birthday
1	oren	uziel	2000	12/10/1987
2	david	azulay	3000	15/6/1990
3	oren	kabuli	1500	15/4/1984
4	shula	afula	5000	10//3/1995

Una fila en una tabla

La tabla contiene 4 filas. Cada fila representa una persona y muestra los detalles de la misma.

personId	fname	lname	salary	birthday
1	oren	uziel	2000	12/10/1987
2	david	azulay	3000	15/6/1990
3	oren	kabuli	1500	15/4/1984
4	shula	afula	5000	10//3/1995

muestra de fila en
la tabla

Una columna en una tabla

muestra de columna

La tabla contiene 5 columnas

muestra de columna en la
tabla



personId	fname	lname	salary	birthday
1	oren	uziel	2000	12/10/1987
2	david	azulay	3000	15/6/1990
3	oren	kabuli	1500	15/4/1984
4	shula	afula	5000	10//3/1995

Célula en una tabla

Muestra de célula

La tabla contiene 20 células

Ejemplo de una
célula

personId	fname	lname	salary	birthday
1	oren	uziel	2000	12/10/1987
2	david	azulay	3000	15/6/1990
3	oren	kabuli	1500	15/4/1984
4	shula	afula	5000	10//3/1995

Comando SQL para crear una tabla

```
CREATE TABLE tblCities(cityId INTEGER PRIMARY KEY, name varchar(50));
```

cityId	name
1	oren
2	david
3	oren
4	shula

Sql command to create a table

```
create Table if not exists tblCities(cityId INTEGER PRIMARY KEY autoincrement, name varchar(50));
```



comando	explicación
create Table tblCities	Crea una tabla llamada tblCities
cityId INTEGER PRIMARY KEY	La primera columna de la tabla es la llave maestra. Esto es, un valor que no puede ser repetido. La palabra "integer" un campo de tipo número
name varchar(50));	La segunda columna en la tabla indica que es un campo de tipo string de máximo 50 caracteres

INSERT a tblCities

Usando el comando INSERT metemos información en la tabla
INSERT INTO tblCities (cityId, name) VALUES (1,'Tel-Aviv');

Va a insertar otra fila en la tabla.

cityId	name
1	Tel-Aviv

Esto también va a funcionar: **INSERT INTO tblCities (name) VALUES ('Bucharest');**
El cityId se autoincrementa entonces es 2.

cityId	name
1	Tel-Aviv
2	'Bucharest'

INSERT a tblCities

Si borramos la línea 2

cityId	name
1	Tel-Aviv

Y agrega una línea : **INSERT INTO tblCities (name) VALUES ('Moscow');**

El cityId se autoincrementa entonces es 3.

El sistema SQL maneja el id e incluso si borramos el id, el sistema recuerda el cityId borrado.

Deja que SQL maneje los id

cityId	name
1	Tel-Aviv
3	Moscow

Hazlo tú mismo

1. Crea una tabla llamada `tblPerson` con los campos `-personId`, `fname`, `lname`, `age`
2. Inserta una línea en la tabla
3. Inserta líneas adicionales en la tabla

Estructura del comando Select

select	Defines the columns, fixed them and removable. Required command
from	Los nombres de las tablas de las cuales la información va a ser extraídas - mandatorio en el decreto
where	Las condiciones de consulta que va a determinar qué filas van a ser consultadas
group by	Permite agrupar listas
Having	Define condiciones lógicas en las filas agrupadas
Order By	Muestra el orden en el que los resultados consultados van a ser mostrados

Supón que configuramos una tabla con 4 filas

tblPersons(personId, fname, lname, salary, birthday)

personId	fname	lname	salary	birthday
1	oren	uziel	2000	12/10/1987
2	david	azulay	3000	15/6/1990
3	oren	kabuli	1500	15/4/1984
4	shula	afula	5000	10/13/1995

Consultando las columnas personId, fname, lname

Select personId, fname, lname from Persons

personId	fname	lname
1	oren	uziel
2	david	azulay
3	oren	kabuli
4	shula	afula

Consulta todas las columnas: Un asterisco indica todas las columnas

Select * from tblPersons

personId	fname	lname
1	oren	uziel
2	david	azulay
3	oren	kabuli
4	shula	afula

Operadores

=	igual
\neq , $<>$, \neq	No igual
\wedge	Mayor que
\geq	Mayor o igual
\vee	Menor que
\leq	Menor o igual

Obtén todos los registros donde el “salary” es mayor a 3000

```
Select personId, fname, lname, salary, birthday  
from tblPersons  
Where salary > 3000
```

Consulta de salario superior a 3000

personId	fname	lname	salary	birthday
4	shulia	afula	5000	10/3/1995

Extrae todas las filas y las columnas donde el “salary” es mayor a 3000

Select *

from tblPersons

Where salary > 3000

Extrae todas las personas que su salario es mayor a 3000



personId	fname	lname	salary	birthday
4	shula	afula	5000	10/3/1995

Consultando registros en la tabla de todas las personas nacidas después del 14/05

```
select personId, fname, lname, salary, birthday  
from tblPersons
```

```
where birthday > '14.5.1999'
```

personId	fname	lname	salary	birthday
3	oren	kabuli	1500	15/6/1999
4	shula	afula	5000	10//3/1995

Between

```
select personId, fname, lname, salary, birthday  
from tblPersons  
where birthday BETWEEN '1.1.90 AND '31.12.96
```

personId	fname	lname	salary	birthday
2	david	azulay	3000	15/6/1990
4	shula	afula	5000	10/3/1995

Not Between

```
select personId, fname, lname, salary, birthday  
from tblPersons  
where birthday NOT BETWEEN '1.1.90 AND '31.12.96
```

personId	fname	lname	salary	birthday
1	oren	uziel	2000	12/10/1987
3	oren	kabuli	1500	15/4/1984

Distinct

Su trabajo es eliminar filas duplicadas

```
select distinct fname  
from tblPerson
```

fname
oren
david
shula

Update

```
UPDATE TABLE_NAME
```

```
SET COLUMN1 = 'VALUE1',COLUMN2 = 'VALUE2'
```

```
WHERE SOME_COLUMN='SOME_VALUE'
```

```
UPDATE tblPerson
```

```
SET fname='shoosha' lname='gusha'
```

```
WHERE fname='uzi';
```

Va a todas las columnas donde el “fname” es “uzi” y cambia el fname a “shoosha” y el “lname” a “gusha”

Update

```
UPDATE tblPerson  
SET fname='shoosha' lname='gusha'  
WHERE fname='uzi';
```

Va a todas las líneas donde el “fname” es “uzi” y cambia el fname a “shoosha” y el “lname” a “gusha”

Delete

```
DELETE FROM TABLE NAME  
WHERE SOME_COLUMN=SOME_VALUE
```

```
DELETE FROM tblPerson
```

```
WHERE salary>2000
```

Va a borrar todas las líneas con el salario mayor a 2000.

Delete

```
DELETE FROM tblPersons  
WHERE salary>2000
```

.Va a borrar todas las líneas donde el salario es mayor que 2000

Práctica

1. Crea una tabla llamada tblProducts con los campos productId, name, description, price, category y quantity
2. tblProducts(productId,name,description,price,category,quantity)
3. **create Table** tblProducts(productId **INTEGER PRIMARY KEY**, name varchar(50),description **varchar**(200), price **INTEGER**,category varchar(50),quantity **INTEGER**);
4. Inserta una fila en la tabla
5. **INSERT INTO** tblProducts (productId,name,description,price,category,quantity)
VALUES (1,'Ball','Red ball',10,'Sport',123);
6. Inserta filas adicionales a la tabla

Práctica

1. Despues de configurar la tabla en mysql. Construye las siguientes solicitudes
2. tblProducts (productId, name, description, price, category, dateProduction, quantity)
3. Inserta 4 “products” en la tabla usando el comando “insert”
4. Ve todos los “products” en la tabla.
5. Ver todas las “categories”
6. Muestra solamente los cambios productId y price.
7. Ve todos los “products” cuales su category sea 8
8. Muestra todos los “products” que su código sea 10 o 15 y su “price” sea entre 100 y 200
9. Ve todos los “name” de los que contengan la palabra “game”
10. Ve todos los “products” ordenados por el año de “dateProduction”.
11. Borra todos los “products” que tengan una “quantity” igual a 0
12. Sube todos los “products” con “price” 100 con una solicitud “update”
13. Agrega una tabla “city”. Muestra name, price y city name (ayuda en unión interna)

Más reglas para SQL

(List of values) IN	Igual a una de las entradas de la lista
not in	No igual a ninguna de las entradas de la lista
LIKE string pattern	Comparado con un patrón de string
IS NULL	El valor es “null”
IS NOT NULL	El valor no es “null”

Ejemplo de - like

```
select personId, fname  
from tblPerson  
where fname like 'a%'
```

Muestra todas las personas que su nombre empieza con "a"

```
select personId, fname  
from tblPerson  
where fname like '%a'
```

Muestra todas las personas que su nombre termina con "a"

Ejemplo de null

Un valor NULL en el campo, significa que ese valor es desconocido, que falta.
Lista todas las personas que su “lname” fale en la tabla

```
select personId, lname  
from tblPerson  
where lname is null
```

Ejemplo de no null

Muestra todas las personas que tienen un “fname” en la tabla

```
select personId, lname  
from tblPerson  
where lname is not null
```

Un ejemplo de una condición compleja

Un ejemplo de una condición compleja

Muestra todas las personas que se llamen Uzi con el salario mayor a 2000

```
select personId, fName  
from tblPerson  
where salary > 2000 AND fName = 'uzi'
```

Un ejemplo de una condición compleja

Un ejemplo de una condición compleja

Muestra todas las personas que se llamen Uzi o que tengan el salario mayor a 2000

```
select personId, fName  
from tblPerson  
where salary > 2000 OR fName = 'uzi'
```

Muestra las filas en cierto orden

Va a mostrar todos los empleados con el salario mayor a 2000 en orden descendente de “salary”:

```
select lname , fname  
from tblPerson  
where salary>2000  
ORDER BY salary desc
```

Normalizar tablas

Supongamos que ya tienes la siguiente tabla:

Nos gustaría normalizar la tabla.

Esto es, cambiar el campo cityname
a cityId por dos razones:

1. Ahorro de memoria - podemos mantener un número de "city" en la memoria
2. Prevención de errores futuros

tblPersons

personId
name
age
cityname

personId	Name	Age	Cityname
1	Lewis	30	Bucharest
2	Adam	32	Moscow
3	Or	26	Bucharest

Normalizar tablas - Espacio para errores

Supongamos que queremos que todas las personas que viven en Bucharest, pero en "personId" 3 hay un error en el "Cityname". Por lo tanto, no lo vamos a recibir en el resultado

personId	Name	Age	Cityname
1	Lewis	30	Bucharest
2	Adam	32	Moscow
3	Or	26	Buchrest

Normalizando tablas

tblPersons

personId
name
age
cityname

tblCities

cityId
cityname

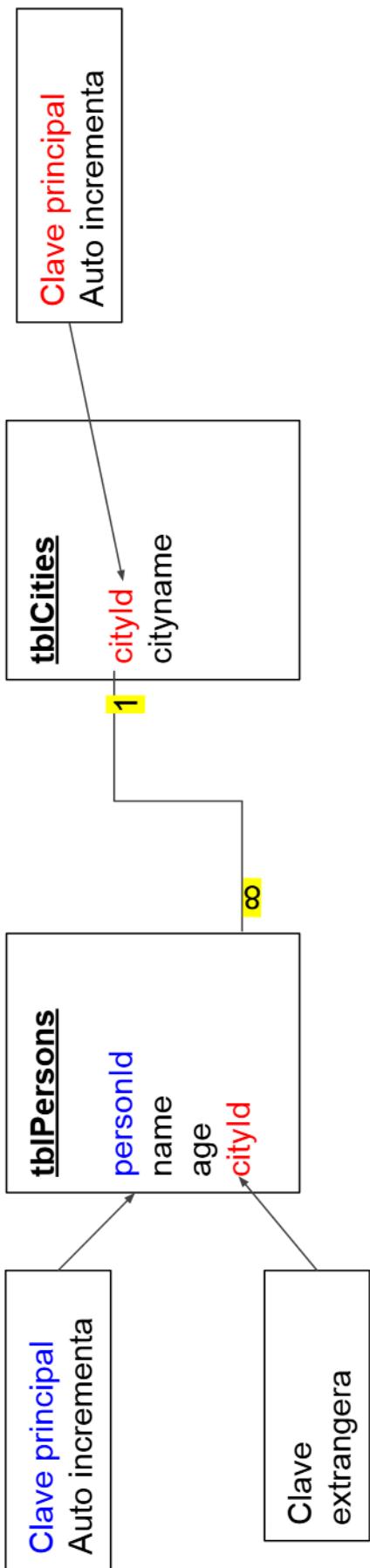
Vamos a cambiar el campo "cityname" a
"cityId" y crear una tabla "city

personId	Name	Age	cityId
1	Lewis	30	1
2	Adam	32	2
3	Or	26	1

Relación - uno a muchas

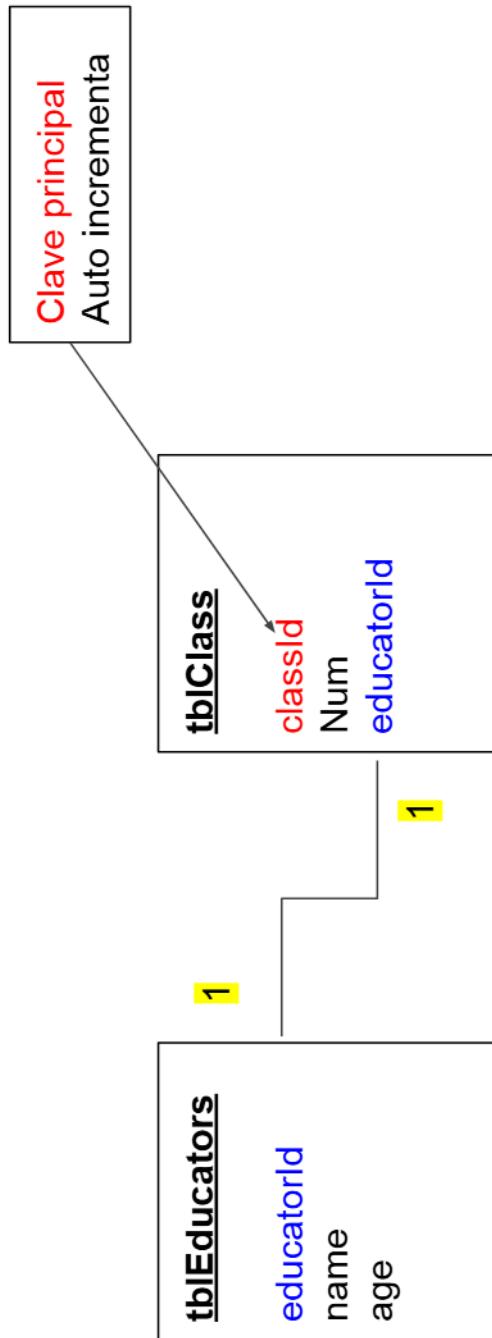
Una conexión a muchas: Cada “person” vive en una “city”, pero en una “city” viven muchas “persons”

Clave extranjera - La clave principal en otra tabla



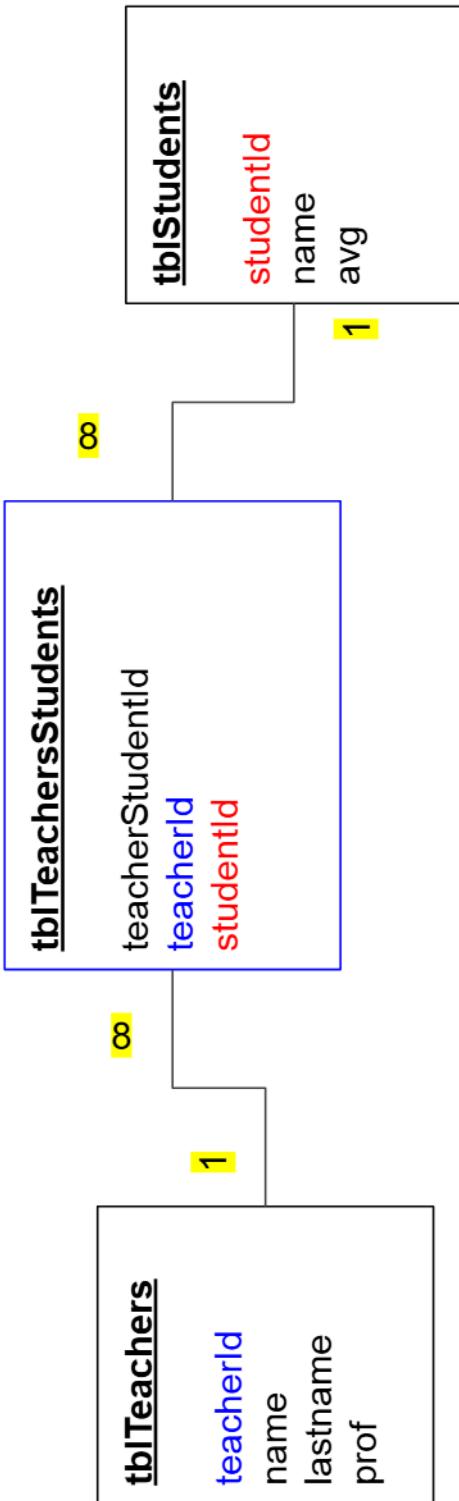
Relación - uno a uno

Relación Individual-a-individual: "educator" y "class". Cada "class" tiene un "educator" y cada "educator" tiene una "class"



Relación - muchos a muchos

Muchos a muchos: "teacher" y "students". Cada "teacher" tiene muchos "students" y cada "student" tiene muchos "teachers"



Products - muchos a muchos

tblStore(storeId, name, address)

tblProducts(productId, name, price)

tblProductStore(ProdutStoreId,storeId, productId)

storeId	name	address
1	a	telaviv
2	b	hadera

ProductStoreId	storeId	productId
1	1	1
2	1	2
3	2	1

productId	name	price
1	ball	5
2	TS	10

Unión interna

Combina resultados de dos tablas cuando hay valores que coinciden en un campo común.

Sintaxis :

```
FROM table1 INNER JOIN table2 ON table1.field1composertable2.field2
```

El operador INNER JOIN tiene estas partes:

Part	Descripción
<i>table1, table2</i>	Los nombres de las tablas de donde estos resultados se combinan.
<i>field1, field2</i>	Los nombres de los campos unidos. Si no son numéricos, los campos deben tener el mismo tipo de información, pero no tienen que tener el mismo nombre.
<i>comopr</i>	Cualquier operador de comparación relacional: "≤", "<", ">", "<=", ">=", "or "<>"

Inner join

Puedes utilizar una operación INNER JOIN en cualquier cláusula FROM. Este es el tipo de unión más común. Las uniones internas combinan registros de dos tablas siempre que haya valores coincidentes en un campo común a ambas tablas.

Puedes utilizar INNER JOIN con las tablas Departamentos y Empleados para seleccionar todos los empleados de cada departamento. Por el contrario, para seleccionar todos los departamentos (incluso si algunos no tienen empleados asignados) o todos los empleados (incluso si algunos no están asignados a un departamento), puedes utilizar una operación UNIÓN IZQUIERDA o UNIÓN DERECHA para crear una unión externa.

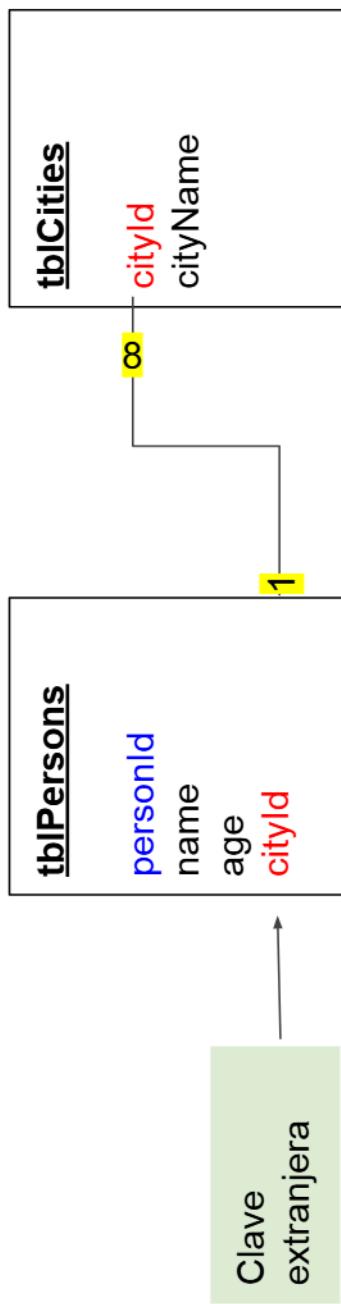
Si intenta unir campos que contienen datos de Memo u Objeto OLE, se produce un error.

Puede unir dos campos numéricos cualesquiera de tipos similares. Por ejemplo, puede unirse en campos Autonumérico y Largo porque son tipos similares. Sin embargo, no puede unir tipos de campos Simple y Doble.

Inner join

```
select tblPerson.name,tblPerson.age,tblCity.cityName  
FROM tblPersons  
INNER JOIN tblCities  
on tblCities.cityId = tblPerson.cityId
```

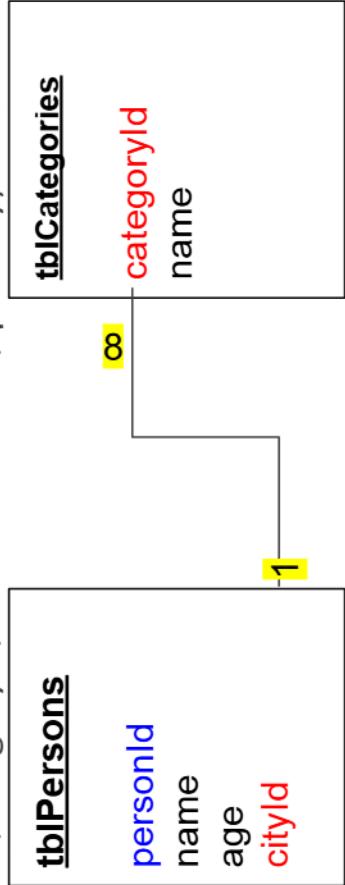
Va a mostrar las columnas first name, age, cityName



Hazlo tú mismo

Configura 2 tablas

tblProducts(productId, name, description, price, categoryId, dateProduction, quantity)
tblCategories(categoryId, name)



Add 3 line to **tblProduct**

1	ball	blue ball	50	1	12/12/1978	5
2	ball	t shirt	60	2	12/12/1978	2
3	ball	red ball	100	1	12/10/1980	4

Agrega 2 "categories"

- 1 | toys
- 2 | clothing

Consulta el product name, description, price y category del "product" (comando JOIN)

wawiwa

¿Preguntas?