



Clase 12. SQL

SUBLENGUAJE DML 2

***RECUERDA PONER A GRABAR LA
CLASE***





OBJETIVOS DE LA CLASE

- Implementar subconsultas para complementar a las sentencias DML

CRONOGRAMA DEL CURSO

Clase 11



Sublenguaje DML 1



CONCEPTOS GENERALES,
SENTENCIAS Y
SUBLENGUAJES.



INSERCIÓN Y ACTUALIZACIÓN
DE TABLAS

Clase 12



Sublenguaje DML 2



SUBLENGUAJES CON DML
CON SUBCONSULTAS



INSERCIÓN Y ACTUALIZACIÓN
DE TABLAS II

Clase 13



Insertión con importación



CONCEPTOS GENERALES,
HERRAMIENTAS DEL SGBD
PARA IMPORTAR DATOS



PRÁCTICA SQL

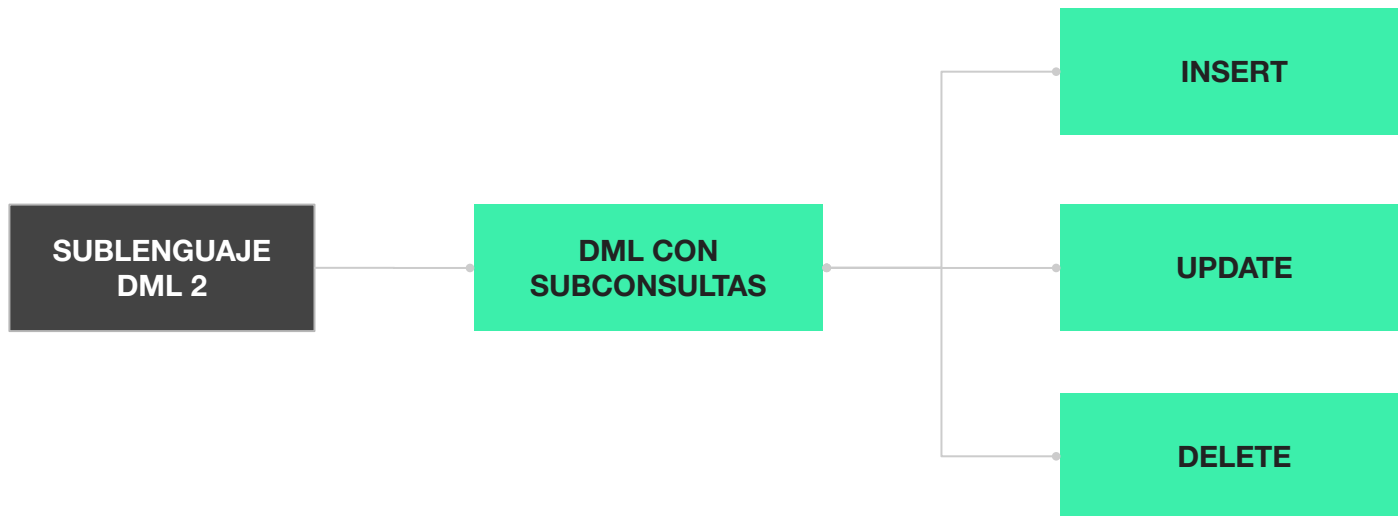


SCRIPT DE INSERCIÓN DE DATOS

MAPA DE CONCEPTOS

MAPA DE CONCEPTOS CLASE 12

¡Para
recordar!



SENTENCIAS INSERT, UPDATE, DELETE, COMPLEMENTADAS CON SUBCONSULTAS

CONCEPTO GENERAL

SUBLENGUAJE DML 2

Veamos cómo aprovechar las diferentes operaciones **DML** que nos permiten **manipular los datos** de una tabla, interviniendo dicha operación a través de una **subconsulta SQL** que puede traer datos importantes desde otras tablas.



LENGUAJE DML

La clase pasada aprendimos cómo el DML - **DATA MANIPULATION LANGUAGE** permite realizar diferentes tipos de operaciones básicas sobre los datos almacenados en tablas SQL, y cómo combinar estas operaciones contemplando diferentes condiciones.



LENGUAJE DML

Aprendimos a utilizar la sentencia:

INSERT para agregar uno o más registros en una tabla.

UPDATE para modificar registros existentes en una tabla.

DELETE para eliminar uno o más registros en una tabla.

```
INSERT INTO nombre_de_la_table VALUES (dato1, dato2, dato3, ...);
```

```
UPDATE nombre_de_la_tabla SET campo2 = 'dato2';
```

```
DELETE FROM nombre_de_la_table WHERE (campo = 'dato');
```

LENGUAJE DML

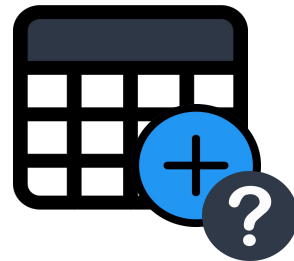
En esta clase, comenzaremos a complejizar las situaciones del **DML - DATA MANIPULATION LANGUAGE**, integrando subconsultas a las diferentes operaciones, y aprendiendo algunas variantes más que enriquecen aún más a **SQL**.

INSERT CON SUBCONSULTAS

INTERACTUAR CON SUBCONSULTAS DE UNA OPERACIÓN DE INSERCIÓN

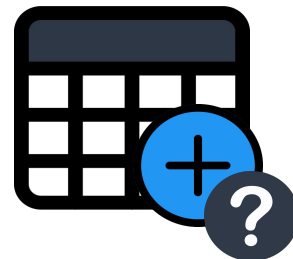
INSERT + SUBCONSULTA: definición

Una subconsulta es básicamente una consulta **SELECT** realizada con el propósito de devolver un dato importante; que será utilizado por **INSERT** para cumplir con su objetivo.



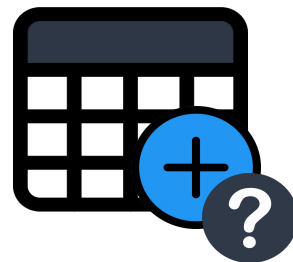
INSERT + SUBCONSULTA

Un caso requerido es, por ejemplo, cuando tenemos dos tablas (`tabla1` y `tabla2`) relacionadas entre sí por una clave foránea, y necesitamos resolver que ambas tablas posean sus datos normalizados.



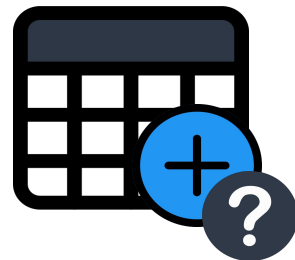
INSERT + SUBCONSULTA

Por ejemplo vamos a agregar nuevas clases de juegos, en una tabla **CLASS_NEW** para luego pasarlos a nuestra tabla **CLASS** de la DB GAMERS. Debemos tener en cuenta que los id_level deben hacer referencia a la tabla **LEVEL_GAME**, entonces los nuevos id_level que agreguemos luego deberán estar en dicha tabla.



INSERT + SUBCONSULTA

... no sabemos cuales **id_level** ya están en la
tabla **LEVEL_GAME** y cuáles no, entonces
deberemos encontrarlos para poder agregarlos,
crearemos también una tabla de
NEW_LEVEL_GAME.



```
INSERT INTO tabla2 (campo1, campo2)
VALUES ((subc SELECT), campo2);
```

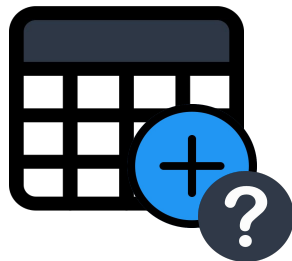
SINTAXIS

CODER HOUSE

Sintaxis de INSERT + SUBCONSULTA

Volvamos a nuestro ejemplo de la clase anterior, donde tenemos nuestra tabla **PAY**, que debió estar relacionada con las tablas **GAME** y **SYSTEM_USER**.

Insertamos registros en **PAY**, desatendiendo la integridad de las otras tablas.



Sintaxis de ***INSERT + SUBCONSULTA***

Habremos creado varios registros en nuestra tabla **NEW_CLASS**.

```
INSERT INTO new_class (id_level,  
id_class, description) VALUES  
(17, 10, 'Adventure Other'),  
(15, 1, 'Spy Other'),  
(17, 20, 'British Comedy'),  
(17, 30, 'Adventure '),  
(14, 1, ''),  
(18, 1, '');
```

Sintaxis de INSERT + SUBCONSULTA

.. vamos a buscar los
id_level que agregamos que
no están en la tabla
LEVEL_GAME para
insertarlos en nuestra nueva
tabla **NEW_LEVEL_GAME**.

```
INSERT INTO new_level_game (id_level, description)
(
SELECT DISTINCT id_level, 'New level'
FROM new_class
WHERE id_level NOT IN (
SELECT id_level
FROM level_game)
);
```



EJEMPLO EN VIVO

Llevemos el anterior ejemplo a un script de Mysql Workbench.

INSERT + SUBCONSULTA: ejemplo

Ejemplo
en vivo



Abrimos una nueva *pestaña de Script* y agregamos las tablas, los insert y las consultas de las diapositivas anteriores:

- 1** Creamos las tablas nuevas
- 2** Hacemos los insert en **NEW_CLASS**
- 3** Hacemos los insert en **NEW_LEVEL_GAME**

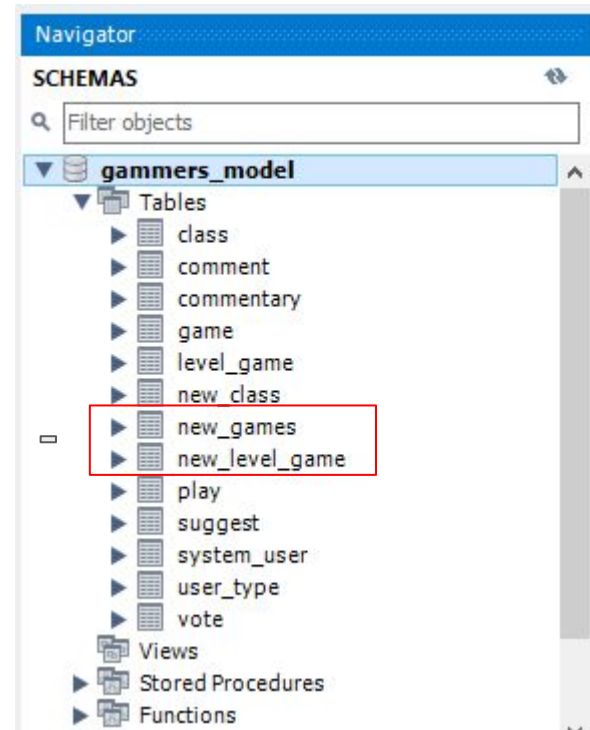


INSERT + SUBCONSULTA: ejemplo

1

Creamos las tablas nuevas.

```
1 • CREATE TABLE new_class (  
2     id_level int NOT NULL,  
3     id_class int NOT NULL,  
4     description varchar(200) NOT NULL,  
5     PRIMARY KEY (id_class, id_level)  
6 );  
7  
8 • CREATE TABLE new_level_game (  
9     id_level int NOT NULL,  
10    description varchar(200) NOT NULL,  
11    PRIMARY KEY (id_level)  
12 );
```



INSERT + SUBCONSULTA: ejemplo

Ejemplo
en vivo



2 Hacemos los insert en **NEW_CLASS**.

```
1 • INSERT INTO new_class (id_level, id_class, description) VALUES (17, 10, 'Adventure Other'),  
2                                                                    (15, 1, 'Spy Other'),  
3                                                                    (17, 20, 'British Comedy'),  
4                                                                    (17, 30, 'Adventure'),  
5                                                                    (14, 1, ''),  
6                                                                    (18, 1, '');
```

INSERT + SUBCONSULTA: ejemplo

Ejemplo
en vivo



3 Hacemos los insert en **NEW_LEVEL_GAME**

| | id_level | description |
|---|----------|-------------|
| ▶ | 17 | New level |
| | 18 | New level |
| * | NULL | NULL |

```
INSERT INTO new_level_game (id_level, description) (SELECT DISTINCT id_level, 'New level'
                                                    FROM new_class
                                                    WHERE id_level NOT IN (
                                                        SELECT id_level
                                                        FROM level_game));
```

Finalmente llegamos al resultado esperado, integrando una subconsulta dentro de una sentencia SQL **INSERT**.

SELECT INTO

PROPUESTA DE MYSQL PARA LA CLÁUSULA 'SELECT INTO'

DDL para SELECT INTO

Es posible crear nuevas tablas a partir de una existente, insertando registros en la nueva tabla, de acuerdo a una o más condiciones específicas mediante la sentencia **CREATE TABLE**.

```
CREATE TABLE nuevatabla  
  
(SELECT * FROM viejatabla  
  
WHERE condiciones);
```

CREATE TABLE + (subconsulta)

De esta forma **creamos una nueva tabla** a partir de la estructura y los datos de una tabla existente. Y hasta decidimos qué registros copiar, a través de uno o más condicionales.

```
CREATE TABLE PLAY_INCOMPLETED
```

```
(SELECT * FROM PLAY WHERE completed = 'FALSE');
```

CREATE TABLE + (subconsulta)

También contamos con la posibilidad de agregar solo algunos campos **en la nueva tabla**; si es que no necesitamos llevarnos todos los campos existentes de la tabla original.

```
CREATE TABLE PLAY_INCOMPLETED_W  
  
(SELECT id_game, id_system_user FROM PLAY  
  
WHERE completed = 'FALSE');
```




BREAK

¡5/10 MINUTOS Y VOLVEMOS!

UPDATE CON SUBCONSULTAS

DEFINICIÓN

UPDATE + SUBCONSULTA: definición

De igual forma que con la cláusula **INSERT**, también podemos aplicar una actualización de información en tablas utilizando la sentencia **UPDATE** combinada con una subconsulta SQL.



UPDATE + subconsulta: sintaxis

La sentencia SQL para llevar a cabo este propósito, tendría una estructura similar a la representada aquí.

En la cláusula WHERE podremos reemplazar el operador de comparación por el cual creamos conveniente.

```
UPDATE tabla
```

```
SET unCampo = valor
```

```
WHERE otroCampo = (SELECT  
  
campo FROM tabla WHERE  
  
condiciones);
```

UPDATE + SUBCONSULTA: Sintaxis

Alternando el operador de comparación entre las diferentes opciones que pone a disposición SQL, podremos controlar que la actualización de datos sea masiva (*aplicada a varios registros a la vez*), o de forma individual.

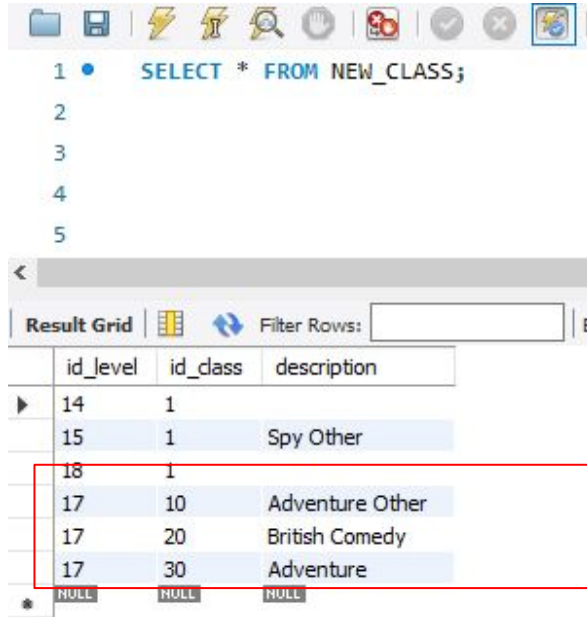


EJEMPLO EN VIVO

*Actualizaremos de forma masiva registros almacenados en la
tabla NEW_CLASS.*

UPDATE + SUBCONSULTA: ejemplo

Ejemplo
en vivo



1 • `SELECT * FROM NEW_CLASS;`

2

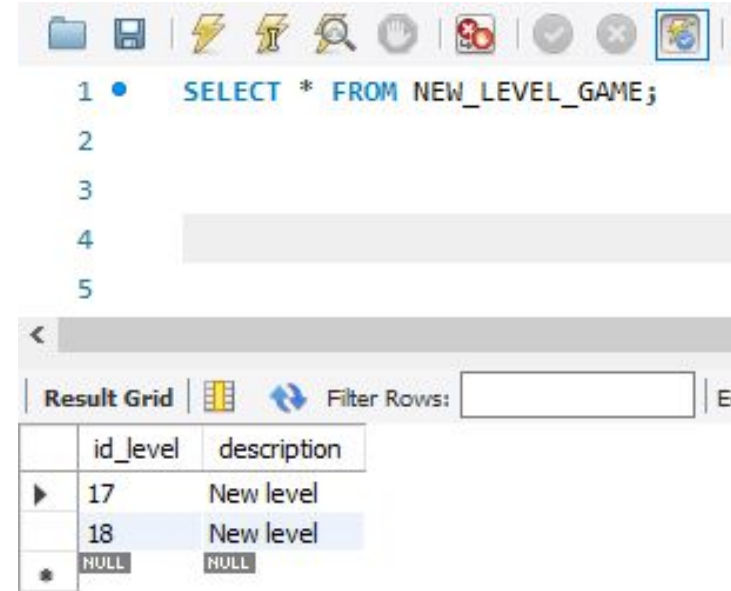
3

4

5

Result Grid

| | id_level | id_class | description |
|---|----------|----------|-----------------|
| ▶ | 14 | 1 | |
| | 15 | 1 | Spy Other |
| | 18 | 1 | |
| | 17 | 10 | Adventure Other |
| | 17 | 20 | British Comedy |
| | 17 | 30 | Adventure |
| * | NULL | NULL | NULL |



1 • `SELECT * FROM NEW_LEVEL_GAME;`

2

3

4

5

Result Grid

| | id_level | description |
|---|----------|-------------|
| ▶ | 17 | New level |
| | 18 | New level |
| * | NULL | NULL |

Debemos actualizar al nivel 20 todos aquellos registros de la tabla

NEW_CLASS que su identificador de `id_level` se encuentre dentro
de la tabla **NEW_LEVEL_GAME**.

CODER HOUSE

UPDATE + SUBCONSULTA: ejemplo

Ejemplo
en vivo



Limit to 5000 rows

```
1 • UPDATE NEW_CLASS SET id_level = 20 WHERE id_level IN (SELECT id_level FROM NEW_LEVEL_GAME);
2
3
4 • SELECT * FROM NEW_CLASS;
5
```

Result Grid

| | id_level | id_class | description |
|---|----------|----------|-----------------|
| ▶ | 14 | 1 | |
| | 15 | 1 | Spy Other |
| | 20 | 1 | |
| | 20 | 10 | Adventure Other |
| | 20 | 20 | British Comedy |
| | 20 | 30 | Adventure |
| * | NULL | NULL | NULL |

DELETE CON SUBCONSULTAS

DELETE + SUBCONSULTA: definición

También podemos planificar la eliminación de uno o varios registros utilizando la cláusula **DELETE**, combinándola con una subconsulta SQL.



DELETE + subconsulta: sintaxis

La estructura **DELETE** se establece definiendo una condición que se deba cumplir para eliminar registros.

Finalmente, el valor a especificar en la condición, tendrá como resultado lo que devuelva la subconsulta **SELECT**.

```
DELETE FROM tabla
```

```
WHERE campo = (SELECT *
```

```
FROM otraTabla WHERE
```

```
condición);
```



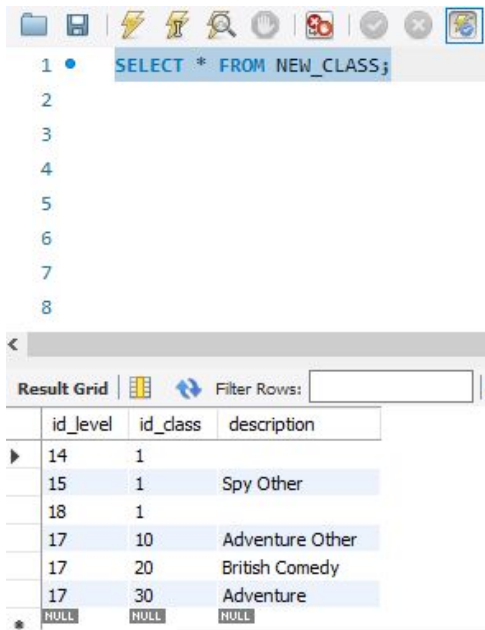
EJEMPLO EN VIVO

*Eliminemos aquellas nuevas clases que no van a poder estar
relacionados con ningún registro de la tabla
NEW_LEVEL_GAME*



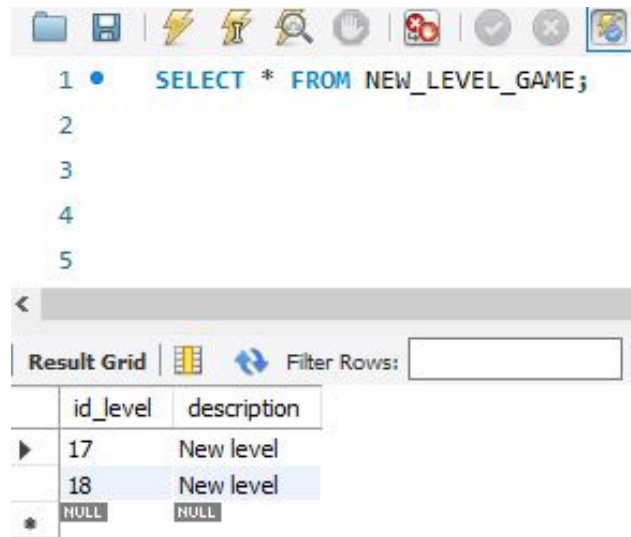
DELETE + SUBCONSULTA: ejemplo

Continuemos con el ejemplo de las nuevas clases y niveles trabajados hasta el momento:



```
1 • SELECT * FROM NEW_CLASS;
```

| id_level | id_class | description |
|----------|----------|-----------------|
| 14 | 1 | |
| 15 | 1 | Spy Other |
| 18 | 1 | |
| 17 | 10 | Adventure Other |
| 17 | 20 | British Comedy |
| 17 | 30 | Adventure |
| NULL | NULL | NULL |



```
1 • SELECT * FROM NEW_LEVEL_GAME;
```

| id_level | description |
|----------|-------------|
| 17 | New level |
| 18 | New level |
| NULL | NULL |



DELETE + SUBCONSULTA: ejemplo

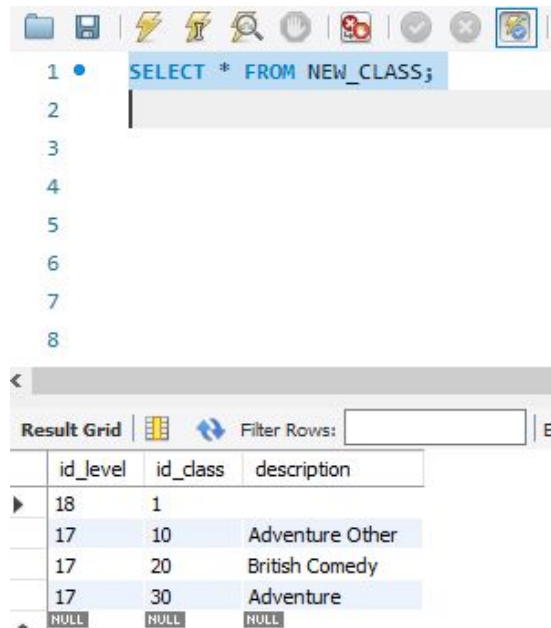
Generemos la sentencia para eliminar aquellas nuevas clases que no van a poder estar relacionados con ningún registro de la tabla

NEW_LEVEL_GAME.

```
DELETE FROM NEW_CLASS  
  
WHERE id_level NOT IN (SELECT id_level  
  
                        FROM NEW_LEVEL_GAME);
```



DELETE + SUBCONSULTA: ejemplo



The screenshot shows a database management interface. At the top, there is a toolbar with various icons. Below it, a SQL query is entered in a text area:

```
1 • SELECT * FROM NEW_CLASS;  
2  
3  
4  
5  
6  
7  
8
```

Below the query editor, there is a section labeled "Result Grid". It includes a "Filter Rows:" input field and a table with the following data:

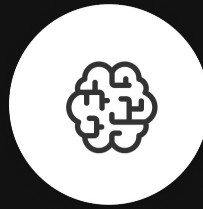
| | id_level | id_class | description |
|---|----------|----------|-----------------|
| ▶ | 18 | 1 | |
| | 17 | 10 | Adventure Other |
| | 17 | 20 | British Comedy |
| | 17 | 30 | Adventure |
| * | NULL | NULL | NULL |

Puedes validar luego, directamente sobre la tabla **NEW_CLASS**, que los registros indicados hayan sido eliminados correctamente.



Mientras practiques operaciones de actualización al igual que operaciones de eliminación de registros, te recomendamos **crear tablas de backup** con los datos a modificar/eliminar, para que puedas restaurar rápidamente los mismos en el caso que la cláusula especificada haya modificado o eliminado más o menos información de la esperada.





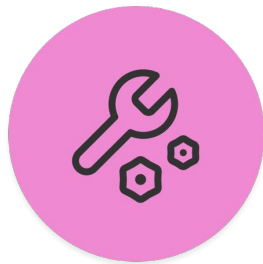
¡PARA PENSAR!

Hasta ahora vimos diferentes operaciones DML con subconsultas de un solo nivel. ¿Crees que se pueden establecer en SQL más de una subconsulta anidada?

¿SÍ o NO?

CONTESTA LA ENCUESTA DE ZOOM





INSERCIÓN Y ACTUALIZACIÓN DE TABLAS II

Realizaremos operaciones de inserción y actualización de tablas utilizando subconsultas.

Tiempo estimado: 20 minutos

INSERCIÓN Y ACTUALIZACIÓN DE TABLAS II

Desafío
generico

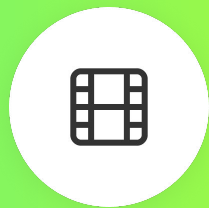


Trabajaremos sobre la BD GAMERS

- Crearemos una nueva tabla de juegos denominada '*ADVERGAME*' donde agregaremos juegos de propaganda de empresa.
- Crearemos a continuación 5 juegos nuevos en la tabla *ADVERGAME*.
- Finalmente, insertaremos los registros correspondientes en la tabla **ADVERCLASS**, obteniendo mediante una subconsulta los id de las clase y niveles nuevos insertados.

Tiempo estimado: 20 minutos

CODER HOUSE



***¿QUIERES SABER MÁS? TE DEJAMOS
MATERIAL AMPLIADO DE LA CLASE***



- [Subconsulta \(update - delete\)](#) | ***Diego Moisset de Espanes***

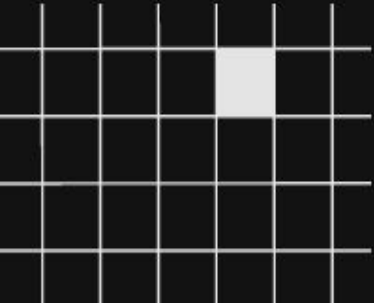
¿PREGUNTAS?





¡MUCHAS GRACIAS!

Resumen de lo visto en clase hoy:

- Sentencias insert, update, delete, complementadas con subconsultas.
 - Implementación del sublenguaje.
- 



OPINA Y VALORA ESTA CLASE

#DEMOCRATIZANDO LA EDUCACIÓN

CODER HOUSE