**Clase 12 - Sublenguaje DML 2**

SENTENCIAS INSERT, UPDATE, DELETE, COMPLEMENTADAS CON SUBCONSULTAS

## INTERACTUAR CON SUBCONSULTAS DE UNA OPERACIÓN DE INSERCIÓN

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.

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.

### Sintaxis de INSERT + SUBCONSULTA

**INSERT INTO** **tabla2 (campo1, campo2)**

**VALUES ((subc SELECT), campo2);**

Ejemplo:

**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, ‘’);**

**CREATE TABLE** **NEW\_CLASS (**

**id\_level int NOT NULL,**

**id\_class int NOT NULL,**

**description varchar(200) NOT NULL,**

**CONSTRAINT PK\_CLASS PRIMARY KEY (id\_class,id\_level)**

**);**

**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)**

**);**

## SELECT INTO

### PROPUESTA DE MYSQL PARA LA CLÁUSULA ‘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.

Ejemplo:

**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.

Ejemplo:

**CREATE TABLE** **PLAY\_INCOMPLETED**  
**(SELECT** **\*** **FROM** **PLAY** **WHERE** **completed = ‘FALSE’);**

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.

Ejemplo:

**CREATE TABLE** **PLAY\_INCOMPLETED\_W**  
**(SELECT** **id\_game, id\_system\_user FROM** **PLAY  
WHERE** **completed = ‘FALSE’);**

## UPDATE CON SUBCONSULTAS

#### UPDATE + subconsulta: sintaxis

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.

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

Ejemplo:

**UPDATE tabla**

**SET unCampo = valor**

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

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.

## DELETE CON SUBCONSULTAS

#### DELETE + SUBCONSULTA: definición

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.

Ejemplo:

**DELETE FROM tabla**

**WHERE** **campo =** **(SELECT** **\*** **FROM otraTabla WHERE condición);**

Mientras practicas 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.

**BUENAS PRÁCTICAS PERFORMANCE TUNING**

**1. Columnas clave de índice**

Asegurarnos de indexar las columnas que se utilizan en las cláusulas JOIN para que la consulta devuelva el resultado rápidamente.

Si se usa la declaración UPDATE, que involucra más de una tabla, asegurarnos de que todas las columnas que se usan para unir las tablas estén indexadas.

**2. Utilizar la sentencia LIMIT(1) cuando se utilice DELETE/UPDATE**

Cada actualización/eliminación individual se realiza mediante el valor único y está limitada a exactamente una fila (usando LIMIT(1))

Esto nos asegura que cada comando impacte exactamente en una fila. Además, para las actualizaciones, esta es a veces la única opción, porque puede esperar que tenga diferentes valores que desee establecer para diferentes filas.