

# Base de datos desde cero

Clase 11 - 23/11/2022

# Introducción Básica a la Programación

A probar algunas funciones:

```
select
    dni,
    nombreyapellido,
    fun_devEdad(fechanacimiento) as fechaNacimiento
from alumno;
```

- Funciones incorporadas
- Funciones de usuario

```
CREATE DEFINER=`root`@`localhost`
FUNCTION `fun_devEdad`(pFecha date) RETURNS int
BEGIN
    declare vResultado INTEGER;
    set vResultado = TIMESTAMPDIFF('YEAR', pfecha, curdate());
    RETURN vResultado;
END
```

# Introducción Básica a la Programación

Asignación del resultado de un Select a una variable

```
Select curdate() into vFecha;
```

Asignamos a la variable vFecha la fecha actual tomada desde el sistema.

La sintaxis del Select Into es la siguiente:

```
Select campo1, ... campo N into vVariable1, ..., vVariableN from OrigenDeDatos;
```

Muy utilizado en funciones y procedimientos.

# Introducción Básica a la Programación

```
select
    dni,
    nombreyapellido,
    fun_devEdad(fechanacimiento) as fechaNacimiento,
    fun_promedioAlumno(dni) as promedio
from alumno;
```

```
CREATE FUNCTION fun_promedioAlumno (pDni Integer)
RETURNS decimal(5,2)
BEGIN
declare vPromedio Decimal(5,2);
    select
        avg(if(exa.notapromedio is null,0,exa.notapromedio)) into vPromedio
    from exaxmatxalu as exa
    where exa.dni = pDni
        and exa.idtipoexamen = 3;
RETURN vPromedio;
END
```

# Introducción Básica a la Programación

Para facilitar la generación de funciones  
`SET GLOBAL log_bin_trust_function_creators = 1;`

Hagamos algunas funciones!!!

# Funciones

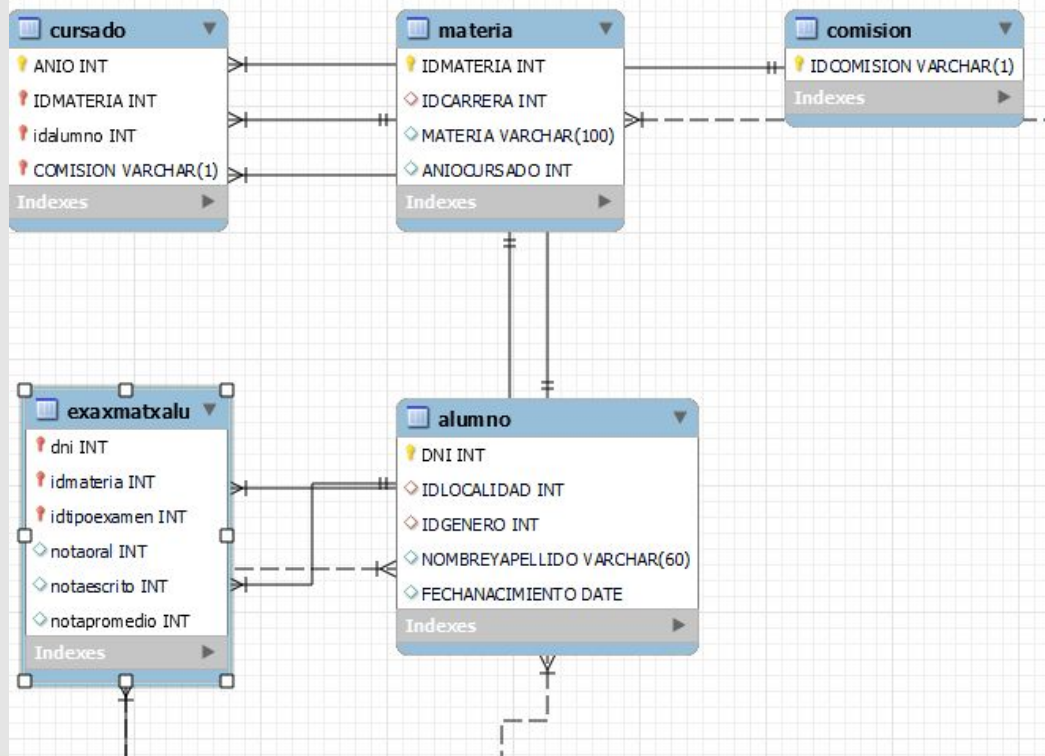
## Ejercicio Práctico

Consigna: Determinar la cantidad de materias aprobadas de un alumno.

- 1) Lo primero es saber cómo puedo llegar a determinar eso,
  - lo tengo que entender mirando el modelo físico de la BD
  - ¿Cómo sé que un alumno aprobó una materia?
    - cuando el alumno aprueba un final
    - los finales son un tipo de examen, ellos están en la tabla tipoexamen
  - ¿Con cuánto debe aprobar?  $\geq 4$
  - ¿Dónde encuentro los exámenes, los tengo? si en la tabla exaxmatxalu
- Tengo todos los datos ahí o tengo que buscar en otras tablas?
  - Si tengo todos ahí - Entonces: manos a la obra

# Funciones

## Ejercicio Práctico





## Funciones

### Ejercicio Práctico

Miremos la tabla, que datos contiene, como los tiene almacenados

- select \* from exaxmatxalu;

Solucionemos por partes para que sea más fácil.

- select \* from exaxmatxalu where notapromedio >=4;

\*\* Es importante reconocer que con esta consulta listamos todos los campos de la tabla exaxmatxalu, donde la nota promedio es mayor o igual a cuatro, por ende todos los exámenes aprobados.

Agreguemos el filtro de que sean solo finales

- select \* from exaxmatxalu where notapromedio >=4 and idtipoexamen=3;

\*\* Ahora listamos los mismos datos, pero filtramos solo finales, o sea finales y aprobados. esto me da la información, de todos los alumnos que hayan aprobado uno o más fianles.

Filtremos entonces solo él alumno que nos interesa.

- select \* from exaxmatxalu where notapromedio >=4 and idtipoexamen=3 and dni = 5;



# Funciones

## Ejercicio Práctico

Listo:

dni	idmateria	idtipoexamen	notaoral	notaescrito	notapromedio
5	4	3	NULL	8	8
5	5	3	NULL	8	8
NULL	NULL	NULL	NULL	NULL	NULL

Observando podemos ver que el alumno aprobó 2 finales. Que es lo que se pedía.  
¿Cómo podemos hacer para tenerla cada vez que la necesite?

1. Grabemos el script
  2. Creemos una consulta
  3. Hagamos una función
- 1) La primera opción, ya la tenemos hecha, es simplemente necesario que guardemos la consulta (archivo .sql) en algún lugar donde nos acordemos por si nos piden nuevamente la consulta. Lo único es que habría que cambiar el dni.

# Funciones

## Ejercicio Práctico

Listo:

2) La segunda opción, es crear una vista, esto último tiene la ventaja de que ya la dejamos guardada en el servidor y la podemos utilizar cuando queremos sin tener que acordar donde guardamos el script.

Ojo, para guardar la vista, es recomendable quitar el filtro de dni, así podemos filtrar la vista solo por el dni que nos piden, caso contrario tendríamos que modificar la vista cada vez que nos hagan el pedido.

-- la idea es ver solo de un alumno - filtramos uno de ejemplo

```
select * from exaxmatxalu where notapromedio >=4 and idtipoexamen=3 and dni=5 ;
```



Lo único que tenemos que hacer ahora es agregar adelante:

Create View Nombre de la Vista as (el texto de la consulta)

Ejemplo: si llamamos a la consulta vw\_mataprobxalumno la consulta quedaría así:

```
Select * from vw_mataprobxalumno where dni = dniQueQuiero;
```

# Funciones

## Ejercicio Práctico

De esta manera, con solo filtrar el dni sobre la vista, obtenemos la información correspondiente.

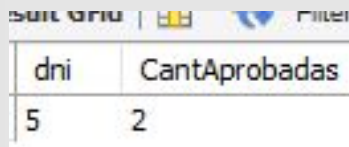
Ahora, no es muy práctico tener que contar todas las veces la cantidad de registros devueltos, así que modifiquemos la consulta.

¿Cómo lo hacemos? - - - Vamos al cliente

-

-

`select * from vw_mataprobxalumno where dni = 5;` – me debería devolver la cantidad de finales que aprobó el dni = 5. Por ejemplo así:



dni	CantAprobadas
5	2

Bueno ahora, hagamos lo mismo pero por medio de una función.

# Funciones

## Ejercicio Práctico

Creando la función:

```
CREATE FUNCTION fun_devCantMatAproXAlu (pAlumno Integer)
RETURNS integer
BEGIN
    declare vResultado Integer;
    select count(*) into vResultado from
        exaxmatxalu
    where notapromedio >=4 and idtipoexamen=3 and dni = pAlumno;
RETURN vResultado;
END
```

```
select fun_devCantMatAproXAlu(5);
```

fun_devCantMatAproXAlu(5)
2

## Funciones

### Ejercicio Práctico

Podemos decir que tanto la consulta como la función pueden tener un resultado óptimo y fácil de ser usado.  
Va a depender del problema a resolver, la estrategia que vamos a usar para darle solución.

## Funciones

### Triggers / Disparadores

Los triggers o disparadores son un bloque de código Sql, que se ejecuta ante las operaciones de Insert, Update o Delete sobre alguna tabla.

Por ejemplo, si el motor detecta que se quiere hacer un insert sobre una tabla, nosotros podemos definir un bloque de código que ejecute determinadas operaciones antes o después de la ejecución de dicho Insert.

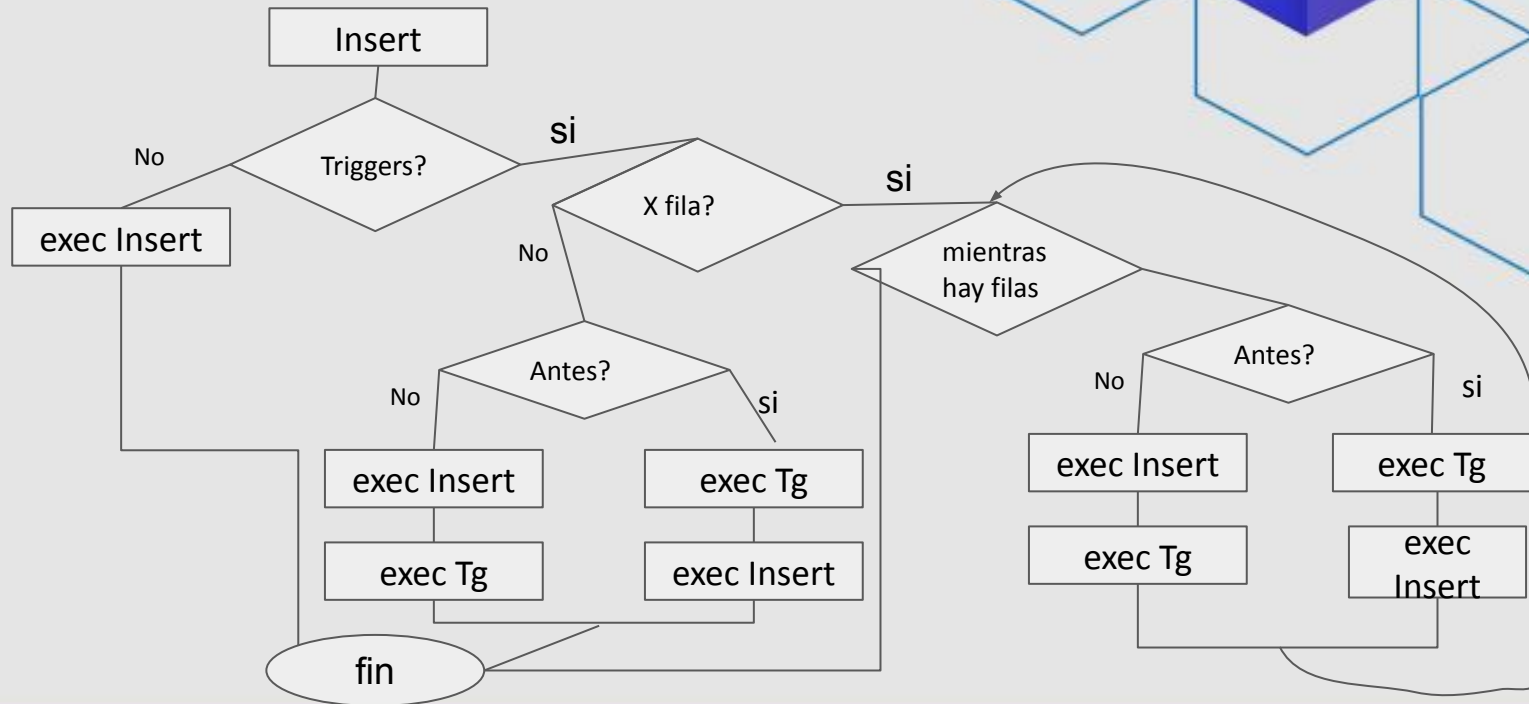
A su vez, podemos definir que dicho bloque se ejecute una vez por cada fila afectada o bien se ejecute por cada sentencia Insert, sin importar la cantidad de filas afectadas.

Los triggers una vez definidos se ejecutan automáticamente, no requieren ser ejecutados por un usuario.

# Funciones

## Triggers / Disparadores

A la hora de intentar ejecutar un Insert, de 10 registros.





## Funciones

### Triggers / Disparadores

Los triggers funcionan con una estructura de tipo cursor, que permite tener acceso a los datos antes y después de las operaciones detectadas.

El ejemplo donde podemos observar bien esto, es en los updates, ya que el motor cuenta con los datos de los registros antes de ser actualizados y también tiene acceso a los datos del registro con los valores a posterior de ser actualizados.

Podemos decir que cuenta con dos estructuras, con los datos viejos y nuevos ante cada transacción.

**:old** y **:new** en oracle, **old** y **new** en mySql.

Por ejemplo, para acceder al valor de un campo nuevo dentro de la estructura haríamos referencia de esta manera:

**new.nombreDelCampo**

## Funciones

### Triggers / Disparadores

Las estructuras old y new, se generan según sean necesarias, la necesidad está marcada por la operación que se ejecute.

Ante un Update, se generarán las estructuras new y old

Ante un Delete, solo la estructura old

Ante un Insert, solo la estructura new

**Update**, como es una actualización dentro de un registro, dichos campos tendrán un valor anterior y un valor posterior a dicho update.

**Delete**, sólo se puede completar la estructura del old, ya que no hay valores nuevos.

**Insert**, solo se puede completar la estructura del new.

# Funciones

## Triggers / Disparadores

CREATE

```
[DEFINER = user]
TRIGGER [IF NOT EXISTS] trigger_name
trigger_time trigger_event
ON tbl_name FOR EACH ROW
[trigger_order]
trigger_body
```

*trigger\_time*: { BEFORE | AFTER }

*trigger\_event*: { INSERT | UPDATE | DELETE }

*trigger\_order*: { FOLLOWS | PRECEDES } *other\_trigger\_name*

**Hagamos un ejercicio!!!**

**fuentes:**

<https://dev.mysql.com/doc/refman/8.0/en/create-trigger.html>

## Funciones

### Triggers / Disparadores

- Trigger para guardar todas los movimientos que se hagan sobre la tabla
- de alumnos
- Sistema de logs
- Pasos
- 1) Crear la tabla de logs - nombre cualquiera, en el ejercicio
- log\_alumno
- Crear la tabla, con todos los campos de la tabla alumno, en el caso de una
- situación real, se guardarán los campos que nos interesan, para este caso
- además de dichos campos, agregaremos los campos fechamov, usuario, operación.

## Debería quedar algo así

Table Name:

Charset/Collation:

[illegible]

## Funciones

### Triggers / Disparadores

Create trigger tr\_alumnoinsertlog

before Insert on alumno for each row

insert into log\_alumno

(dni, nombreyapellido, idlocalidad,idgenero, fechanacimiento, usuario,operacion)

values

(new.dni, new.nombreyapellido, new.idlocalidad, new.idgenero,new.fechanacimiento, user(),'I')

# ¡Muchas gracias!

    /poloticmisiones