

USANDO TRIGGERS EN UNA BASE DE DATOS

Trabajo de Alejandro Sainz Sainz

BD-
ACTIVIDAD 4.4

COMIENZO	3
TRIGGER 1. VUELVE AL INSTITUTO, CHAVAL.....	4
TRIGGER 2, NOMBRES DE MODA	6
TRIGGER 3. REGRESO AL FUTURO	7
TRIGGER 4. JEFATURA DEL SIGLO PASADO.....	8
TRIGGERS Y TABLAS.....	10
TRIGGER 5. RICACHONES	10
TRIGGER 6. LOS REYES DEL MAMBO.....	14
TRIGGER 7. ESTO LO TIENE QUE APROBAR DIRECCIÓN	17
DETALLES ADICIONALES	20

Alejandro Sainz Sainz

1 Trigger edad empleado	4
2 Intentamos contratar un niño de 6 años.....	4
3 Resultado de insert incorrecto	5
4 Insert no a la moda	6
5 Resultado del segundo trigger.....	6
6 El delorean no funciona	7
7 Creo que no me pagan de la forma correcta	7
8 Resultado deseado.....	7
9 Antigüedad de los jefes.....	8
10 Insert employees, insert dept_manager.....	9
11 Comprobando el resultado.....	9
12 Tabla millonetis	10
13 Nueva rica?	12
14 Subiendo el sueldo	13
15 Veamos quien inaugura la tabla de nuevos ricos	13
16 Resultado de SELECT	13
17 El fin de mi turno	14
18 Me ha quedado un poco largo	14
19 Bucando candidatos a darse de baja como jefes	15
20 Contenido de la tabla jefes	15
21 Dando de baja.....	15
22 Baja confirmada.....	16
23 Cambio confirmado	16
24 Comprobando mi tabla	16
25 Ejecución del select	16
26 Contenido Tabla finDeGuardia	16
27 Tabla cambioDeLook.....	17
28 Cambio de nombre	17
29 Información de departements.....	18
30 Contenido de departments	18
31 Nuevo nombre	18
32 Cambio realizado.....	19
33 Comprobando cambios.....	19
34 Operación correcta.....	19

COMIENZO

Para mantener una base de datos y automatizar algunas de sus operaciones o validar la inserción, modificación o borrado de algunos de sus datos se utilizan los triggers, que son una suerte de funciones que nos permiten automatizar procesos sin tener que evaluar registro por registro de forma manual.

En esta actividad vamos a realizar una serie de tareas con estos elementos y vamos a ver los resultados obtenidos con los mismos. Para mostrar de forma clara cada apartado intentaré dar una explicación de los pasos que he llevado a cabo basándome en las capturas que voy a ir adjuntando en cada sección. Comenzamos.

TRIGGER 1. VUELVE AL INSTITUTO, CHAVAL

En este apartado vamos a crear un trigger que evalúe la edad de un nuevo trabajador cuyos datos tratamos de introducir en nuestro sistema. En el caso de no tener una edad mínima de 16 años lanzamos un mensaje que indica el camino de vuelta a instalaciones educativas.

```
06 delimiter $$
07 • create trigger chaval
08 before insert on employees
09 for each row
10 begin
11     if timestampdiff(year, new.birth_date, current_date()) < 16
12     then signal sqlstate '50001' set message_text = 'Vuelve al colegio pollo, que estás muy verde';
13     end if;
14 end$$
```

1 Trigger edad empleado

Este trigger toma como objetivo la inserción de datos en la tabla employees, y la evaluación se ejecuta antes de insertar estos datos, de ahí la cláusula before insert.

Una vez definido el cuerpo del trigger, con un if, evaluamos si se cumple la condición que nos interesa, que el nuevo trabajador sea menor de 16 años. Para este caso, dentro del if, mediante la función timestampdiff, calculamos la edad del nuevo sujeto y comprobamos si es menor de 16. En caso de cumplirse dicha condición devolvemos un mensaje de error indicando que la nueva adquisición está muy verde como para empezar a trabajar.

Una vez completados estos pasos cerramos el condicional y el cuerpo del trigger.

Para comprobar la funcionalidad creada realizamos un insert de prueba que cumpla la condición de evaluación y comprobamos el resultado.

```
insert into employees values (55555, '2018-01-01', 'Juan', 'Lopez', 'M', '2020-01-01');
```

2 Intentamos contratar un niño de 6 años

Para que sea más exagerado, la fecha de nacimiento coincide con la de un niño de 6 años. Todavía estoy esperando la llamada de Inspección de Trabajo por querer poner a trabajar a un infante.

Al ejecutar el comando anterior recibimos el siguiente resultado:

Sábado 12 de abril de 2025

113	11:17:06	select * from cambioDeLook LIMIT 0, 200	1 row(s) returned
114	11:24:30	insert into employees values (55555,'2018-01-01','Juan', 'Lopez','M','2020-01-01')	Error Code: 1644. Vuelve al colegio pollo, que estás muy verde

3 Resultado de insert incorrecto

Como vemos en la imagen anterior el propio sistema nos indica la ilegalidad de nuestro acto mandándonos un mensaje de error.

TRIGGER 2, NOMBRES DE MODA

Para este segundo ejercicio deberemos de comprobar que todos los empleados que queramos insertar en la tabla employees tengan como nombres Martín o Lucía, dado que estadísticamente son los nombres mas populares en nuestro país.

Para ello comenzamos desarrollando el trigger:

```

1 delimiter $$
2 • create trigger nombre_moda
3   before insert on employees
4   for each row
5   begin
6     if new.first_name not in ('Lucia','Martin')
7     then signal sqlstate '50001' set message_text = 'Tu nombre no está a la moda. Vete al registro a cambiarlo';
8     end if;
9   end$$

```

De nuevo en este caso debemos de comprobar las condiciones del insert antes de completarlo, para ello usamos la cláusula before insert.

Después de completar el cuerpo del trigger debemos definir un condicional if en el que indicamos que, si los nombres no son Martín o Lucía, usando not in ('Lucia','Martín'), nos devuelva un mensaje de error indicando que nuestro nombre no cumple con los dictados de la moda.

Una vez completada esta parte cerramos el condicional y cerramos el trigger.

De la misma forma que en el ejercicio anterior vamos a crear un insert en el que podamos comprobar que nos devuelve el mensaje que necesitamos.

```
insert into employees values (55556,'2000-01-01','Juan','Lopez','M','2020-01-01');
```

4 Insert no a la moda

Con este insert, cualquiera cuyo nombre no sean Martín o Lucía nos habría servido, vamos a comprobar el funcionamiento de nuestro trigger.

```
109 18:38:50 insert into employees values (55556,'2000-01-01','Juan','Lopez','M','2020-01-01')
```

Error Code: 1644. Tu nombre no está a la moda. Vete al registro a cambiarlo

5 Resultado del segundo trigger

Ya se que se ve muy pequeño, pero quiero mostrar que estoy realizando el insert que he mostrado anteriormente y que, en efecto, nos devuelve el mensaje que hemos creado en el trigger al no cumplir con los requisitos que hemos marcado. Si se van a llamar todos igual, vamos a tener que empezar a llamarlos por los apellidos.

TRIGGER 3. REGRESO AL FUTURO

Para el siguiente trigger debemos de validar campos de tipo date asegurándonos de que la fecha desde no es mayor que la fecha hasta, sea cual sea la tabla que elijamos. Como dijo Doc a Marty, podríamos crear una paradoja temporal con consecuencias dramáticas.

De nuevo el primer paso es crear nuestro trigger, el cual nos va a servir para evaluar la condición necesaria con respecto a las fechas.

En mi caso he elegido la tabla salaries, eso de intentar cobrar un sueldo en periodos de tiempo negativos me parece bastante paradójico.

```
delimiter $$
• create trigger fechas
  before insert on salaries
  for each row
  begin
    if new.to_date < new.from_date
      then signal sqlstate '50001' set message_text = 'La fecha de hasta de un salario no puede ser menor a la fecha desde';
    end if;
  end$$
```

6 El delorean no funciona

En este caso yo he optado simplemente por comparar una fecha con la otra dentro de un if. Me imagino que también se podría plantear mediante el uso de datediff y evaluar si el resultado es negativo, pero eso ya me parece liar mucho la madeja, y que podría darse el caso de que colocar así los atributos en el datediff probablemente ya resultaría en un warning del programa.

Otra vez evaluamos los datos antes de realizar el insert, así que otra vez usamos before insert y en el if usamos el indicador new, ya que es un insert y sólo podemos referirnos a estos atributos como los nuevos atributos.

Ahora crearemos un nuevo insert para la tabla salaries creado expresamente para que incumpla con las leyes temporales.

```
insert into salaries values (55556, 21000, '2024-01-01', '2023-01-01');
```

7 Creo que no me pagan de la forma correcta

Aquí muestro el insert que se va a probar. Ahora sólo queda ejecutarlo y veremos cual es el resultado.

110 18:49:30 insert into salaries values (55556, 21000, '2024-01-01', '2023-01-01')

Error Code: 1644. La fecha de hasta de un salario no puede ser menor a la fecha desde

8 Resultado deseado

Alejandro Sainz Sainz

Siento que se vea tan pequeño otra vez, pero así se puede ver la ejecución y el error. El trigger no nos permite viajar en el tiempo.

TRIGGER 4. JEFATURA DEL SIGLO PASADO

En este caso vamos a comprobar que un candidato a jefe de departamento debe de tener como mínimo 10 años de experiencia en la empresa. Que se note que apuestan por la juventud.

Para resolver este trigger tuve que moverme un poco por la documentación de mysql y por internet, ya que, aunque habíamos hablado de hacer select dentro de un trigger y dentro de la función datediff me daba error directamente en el select. Puede que lo escribiese mal o que lo ejecutase de la forma que no era, pero, de todas formas, me ha servido para conocer formas de resolver problemas que me han venido bien para los ejercicios posteriores.

Muestro ahora el trigger y comento paso a paso lo que he hecho.

```

32 delimiter $$
33 • create trigger antigüedadJefes
34 before insert on dept_manager
35 for each row
36 begin
37     declare t int;
38     select timestampdiff(year, e.hire_date, current_date()) into t
39     from employees e
40     where e.emp_no = new.emp_no;
41
42     if t < 10 then
43         signal sqlstate '50001' set message_text = 'Para ser jefe de departamento la antigüedad mínima son 10 años';
44     end if;
45 end$$

```

9 Antigüedad de los jefes

Vamos a ir poco a poco.

De nuevo debemos de validar los datos antes de hacer el insert, de ahí el before insert.

Justo después de BEGIN lo que hago es declarar una variable de tipo int que me servirá después.

Ahora realizo una consulta en la que calculo la diferencia en años, que es un valor de tipo int, entre la fecha actual y la fecha de contratación e indico que el resultado lo almacene en la variable t mediante el fragmento de comando into t antes del from de la consulta.

Finalmente evalúo el valor de la variable t y, si esta es menor de 10, devuelvo un mensaje de error indicando que el candidato no cumple con los requisitos de antigüedad.

Ahora debo de crear un nuevo insert para comprobarlo. Para seguir con la tónica de los ejercicios y no tener que buscar entre todos los datos de employees declaro un nuevo insert que, además, cumpla con todos los requisitos de los trigger anteriores.

```
insert into employees values (55550, '2000-01-01', 'Martin', 'Gonzales', 'M', '2018-01-01');  
insert into dept_manager values (55550, 'd001', '2020-01-01', '9999-01-01');
```

10 Insert employees, insert dept_manager

Creo un nuevo empleado que sea mayor de 16 años, se llame Martín y lleve menos de 10 años en la empresa.

Después de esto, voy a intentar nombrar a este nuevo trabajador jefe de uno de los departamentos, en este caso da igual cual, pues si todo va como tiene que ir el programa nos cantará un fallo y no realizará el insert.

111	19/06/06	insert into employees values (55550, '2000-01-01', 'Martin', 'Gonzales', 'M', '2018-01-01')	1 row(s) affected
112	19/06/06	insert into dept_manager values (55550, 'd001', '2020-01-01', '9999-01-01')	Error Code: 1644. Para ser jefe de departamento la antigüedad mínima son 10 años

11 Comprobando el resultado

Como vemos en la imagen anterior podemos insertar al nuevo trabajador de forma correcta pero no podemos nombrarlo jefe de departamento y, es en ese momento, cuando el programa nos devuelve un mensaje de error.

TRIGGERS Y TABLAS

A partir de ahora todos los ejercicios involucran la creación de nuevas tablas que van a almacenar diferentes registros dependiendo de las condiciones que nos marque el ejercicio y que validemos dentro de nuestros triggers. Es a partir de aquí cuando he tenido que usar lo aprendido antes sobre variables para resolverlos. No sé si es la forma correcta o la más sencilla, pero a mí me han dado resultado. Eso sí, que no lo comente antes, en algunos sitios leí que no se recomienda el uso de queries dentro de los triggers, no entendí muy bien porque, pero sé que no es recomendable.

TRIGGER 5. RICACHONES

En este supuesto debemos de crear una tabla en la que se almacene toda la información de los trabajadores que cobran más de 100.000 €. Para este ejercicio he supuesto que debían ser los datos completos, no sólo el emp_no y luego el salario.

Mostramos el código y lo explicamos paso a paso.

```
create table millonetis(  
    emp_no int,  
    nombre varchar(50),  
    apellidos varchar(50),  
    f_nacimiento date,  
    genero enum('M','F'),  
    f_contratacion date,  
    salario int,  
    primary key (emp_no, salario),  
    foreign key (emp_no) references employees(emp_no)  
);
```

12 Tabla millonetis

Lo primero es crear la tabla millonetis, la cual necesitaremos para almacenar los datos que requiere el ejercicio. Me aseguro de completar todos los pasos, que los campos sean de un tipo correcto y de definir una primary key y una foreign key en caso de necesitarla o crearla conveniente.


```

085 • create trigger mejorPagados
086   after insert on salaries
087   for each row
088   begin
089     declare id, sueldo int;
090     declare nombre, apellido varchar(50);
091     declare f_nac, f_cont date;
092     declare gen char(1);
093
094     select e.emp_no, e.birth_date, e.first_name, e.last_name, e.gender, e.hire_date
095     into id, f_nac, nombre, apellido, gen, f_cont
096     from employees e where e.emp_no = new.emp_no;
097
098     if new.salary >= 100000 then
099       insert into millonetis values(id, nombre, apellido, f_nac, gen, f_cont, new.salary);
100     end if;
101   end$$

```

Ahora definimos nuestro trigger.

Para resolver el ejercicio lo que he hecho es indicarle que el trigger se debe de ejecutar después de hacer un insert en la tabla salaries, en la cual almacenamos los salarios de los trabajadores.

Luego declaro todas las variables que necesito para almacenar el resultado de una query. Estas variables deben de coincidir en tipo con los resultados de la consulta que voy a ejecutar a continuación para obtener los datos del trabajador que va a cobrar ese sueldo.

De nuevo uso el comando into en la query, y usando el mismo orden que utilicé después del select indico en que variable se almacena cada atributo correspondiente. Tras el where, me aseguro de que emp_no sea igual a new.emp_no para obtener el trabajador que se corresponde con este salario.

Una vez los datos resultantes están almacenados debo de evaluar el valor del salario introducido con if new.salary > 100000.

De ser correcta la condición procedo a realizar un insert en la tabla millonetis usando las variables para indicar los valores de los diferentes atributos y con new.salary el valor del salario del trabajador. Como dato adicional, saber que podría usar new.emp_no en vez del valor de la variable id, cualquiera de las dos habría sido correcta.

Ahora toca proceder a la comprobación de que todo funciona de la forma correcta.

```

insert into employees values (60000, '1988-01-01', 'Lucia', 'Perez', 'F', '2010-01-01');
insert into salaries values (60000, 150000, '2010-01-01', '9999-01-01');

```

13 Nueva rica?

De nuevo creo un trabajador, da igual el que sea, pero tiene que cumplir con las condiciones de los inserts anteriores.

Alejandro Sainz Sainz

Posteriormente voy a registrar el valor del salario de esta trabajadora en la tabla correspondiente. Vamos a ejecutar para ver qué pasa.

Quiero comentar que a veces me da un error y me dice que la tabla no existe, supongo que quizá es por el lugar del script en el que estoy creando las tablas.

✓ 116 19:25:18 insert into salaries values (60000, 150000, '2010-01-01','9999-01-01') 1 row(s) affected

14 Subiendo el sueldo

Los inserts han ido bien, incluido el de la tabla salarios, ahora debemos de comprobar que en la tabla millonetis se haya guardado un registro.

```
select * from millonetis;
```

15 Veamos quien inaugura la tabla de nuevos ricos

Este comando no necesita explicación, vamos directos a ver el contenido.

	emp_no	nombre	apellidos	f_nacimiento	genero	f_contratacion	salario
▶	60000	Lucia	Perez	1988-01-01	F	2010-01-01	150000

16 Resultado de SELECT

Como podemos ver, el registro está completo. Queda inaugurada nuestra tabla de vividores.

TRIGGER 6. LOS REYES DEL MAMBO

Para este ejercicio debemos de crear una tabla en la que guardaremos los datos de los jefes de departamento que dejan de serlo.

De nuevo, paso a paso.

```

010 create table finDeGuardia(
011     emp_no int,
012     nombre varchar(50),
013     apellido varchar(50),
014     departamento varchar(50),
015     f_fin date,
016     primary key (emp_no, f_fin),
017     foreign key (emp_no) references employees (emp_no)
018 );
019

```

17 El fin de mi turno

Creamos la tabla requerida, con los datos que creamos oportunos, incluidas la primary y la foreign key. He decidió también incluir el nombre del departamento por lo que vamos a necesitar una consulta más larga dentro del trigger.

A continuación, veremos el trigger que hemos creado para la ocasión.

```

020 delimiter //
021 • create trigger finalDelDeber
022 before update on dept_manager
023 for each row
024 begin
025
026     declare id int;
027     declare nombre, apellido, departamento varchar(50);
028
029     select e.emp_no, e.first_name, e.last_name, d.dept_name
030     into id, nombre, apellido, departamento
031     from employees e
032     join dept_manager dem on e.emp_no = dem.emp_no
033     join dept_emp de on e.emp_no = de.emp_no
034     join departments d on de.dept_no = d.dept_no
035     where e.emp_no = old.emp_no;
036
037     if new.to_date < '9999-01-01' then
038         insert into finDeGuardia values (id, nombre, apellido, departamento, new.to_date);
039     end if;
040 end//

```

18 Me ha quedado un poco largo

Alejandro Sainz Sainz

Explicamos ahora el trigger anterior.

En este caso, como la acción se produce después de modificar la fecha de fin una de las condiciones debe de ser before update.

Como en el ejercicio anterior declaro las variables que considero oportunas para almacenar los datos que necesito.

Después creamos una query, una con unos cuantos join para obtener la información correcta y, tras la cláusula where, indicamos que el emp_no de la consulta debe de ser igual al old.emp_no. En este caso da igual new que old, pues en principio van a ser iguales, aunque no estoy seguro de si podría dar algún resultado inesperado.

Acto seguido y mediante un if evaluamos que el valor de new.to_date sea menor que '9999-01-01' para comprobar que ya se ha puesto fecha de fin a la jefatura del sujeto.

En caso de ser así insertamos un nuevo registro en la tabla creada incluyendo como último atributo new.to_date para que quede registrada también la fecha de fin de jefatura.

Ahora lo que corresponde es verificar los resultados de este ejercicio.

```
1
2 select * from dept_manager;
```

19 Bucando candidatos a darse de baja como jefes

Lo primero que hago es un select * en la tabla dept_manager para buscar un sujeto al que dar de baja.

10002	d007	1985-01-01	1991-10-01
10003	d004	1991-10-01	9999-01-01
10006	d005	1985-01-01	1989-12-17

20 Contenido de la tabla jefes

Aparecen más, pero entre estos tres ya me aparece un candidato perfecto para realizar la prueba.

```
3
4 update dept_manager set to_date = '2025-04-11' where emp_no = 10003;
5
```

21 Dando de baja

Lo que hago es actualizar la fecha hasta del jefe con el emp_no 10003 indicando como fecha la del día que creé el script.

Lo que tengo que hacer ahora es ejecutar este fragmento y ver que pasa.

116 09:50:08 update dept_manager set to_date = '2025-04-11' where emp_no = 10003 1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0

22 Baja confirmada

Lo que vemos en esta imagen es el resultado por consola de ejecutar el comando anterior. Vemos que nos indica que una fila ha sido afectada. Vamos a comprobar todos los cambios.

▶	10002	d007	1985-01-01	1991-10-01
	10003	d004	1991-10-01	2025-04-11
	10006	d005	1985-01-01	1989-12-17

23 Cambio confirmado

Si vuelvo a usar un `select * from dept_manager` puedo ver en la tabla que, efectivamente, se ha guardado la modificación de la fecha hasta del jefe con emp_no 10003.

Ahora lo que tengo que comprobar es si se ha insertado en la tabla que yo he creado la información que he indicado en el trigger.

```
18 select * from finDeGuardia;
```

24 Comprobando mi tabla

Al ejecutar este comando, si todo ha ido bien, debería de mostrarme los datos del jefe al que he actualizado la información.

118 09:54:12 select * from finDeGuardia LIMIT 0, 200 1 row(s) returned

25 Ejecución del select

El resultado por consola es correcto, vamos a ver lo que muestra la tabla.

	emp_no	nombre	apellido	departamento	f_fin
▶	10003	Parto	Bamford	Production	2025-04-11

26 Contenido Tabla finDeGuardia

Aquí vemos como la información que yo indiqué en el trigger se ha añadido correctamente.

TRIGGER 7. ESTO LO TIENE QUE APROBAR DIRECCIÓN

Vamos ahora con el último apartado de este ejercicio. Tenemos que almacenar el nuevo nombre de departamento y su código siempre que alguien intente modifica dicho nombre.

Este trigger le he resuelto siguiendo pasos muy similares a los usados en apartados anteriores, la única diferencia es que yo he añadido algo de información adicional en la nueva tabla que almacena estos datos.

Al lio.

```
52 create table cambioDeLook(  
53     dept_no char(4),  
54     dept_name_viejo varchar(40),  
55     dept_name_nuevo varchar(40),  
56     f_cambio date,  
57     primary key (dept_no, dept_name_nuevo, f_cambio),  
58     foreign key (dept_no) references departments (dept_no)  
59 );
```

27 Tabla cambioDeLook

Para esta tabla, he decidido almacenar en ella el número de departamento, el nombre viejo, el nuevo nombre que se le va a dar y la fecha en que se cambió de nombre. Le he dado también la Primary Key y la Foreign Key que he considerado convenientes.

Ahora vamos a ver como he ido desarrollando el trigger.

```
363 delimiter //  
364 create trigger cambioDeNombre  
365 after update on departments  
366 for each row  
367 begin  
368     insert into cambioDeLook values (old.dept_no, old.dept_name, new.dept_name, current_date());  
369 end//  
370
```

28 Cambio de nombre

Este trigger es bastante más sencillito que los dos anteriores.

Como indicaciones, este trigger se debe de ejecutar después de realizar una actualización en la tabla departemnte, así que usamos after update.

Cuando se realiza el update indico en el trigger que en la tabla cambioDeLook debe de insertar el dept_no viejo (old.dept_no), el nombre de

Alejandro Sainz Sainz

departamento que vamos a sustituir (old.dept_name), el nuevo nombre del departamento (new.dept_name) y, por último, la fecha en la que se realiza el cambio de nombre (current_date());

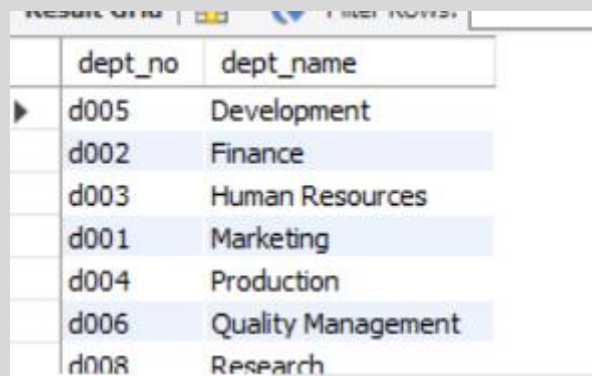
El motivo de añadir información adicional es para que quede una tabla bien diferenciada de la tabla departments.

Vamos a comprobar con un ejemplo que todo funciona como debería.

```
70  
71 • select * from departments;  
72
```

29 Información de departements.

Con este comando vamos a ver la información que tenemos en departments y vamos a elegir un departamento aleatorio.



	dept_no	dept_name
▶	d005	Development
	d002	Finance
	d003	Human Resources
	d001	Marketing
	d004	Production
	d006	Quality Management
	d008	Research

30 Contenido de departments

Esto es lo que hay almacenado hasta el momento.

```
3  
4 update departments set dept_name = 'Servicio de Quejas' where dept_no = 'd009';
```

31 Nuevo nombre

Vamos a cambiar el nombre de atención al cliente por otro distinto.



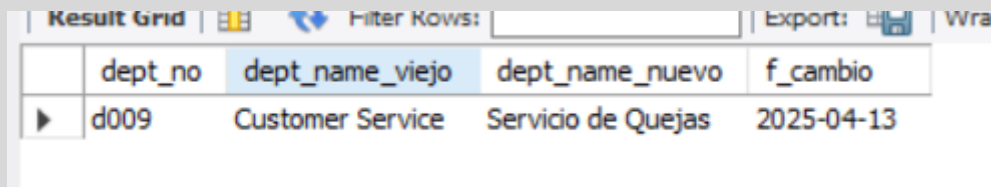
	dept_no	dept_name
	d001	Marketing
	d004	Production
	d006	Quality Management
	d008	Research
	d007	Sales
	d009	Servicio de Quejas

32 Cambio realizado

Una vez ejecutado el comando podemos realizar otro select y ver el cambio de nombre.

Ahora veremos si todo ha quedado almacenado en la tabla que hemos creado.

```
5 select * from cambioDeLook;
```

33 Comprobando cambios

	dept_no	dept_name_viejo	dept_name_nuevo	f_cambio
▶	d009	Customer Service	Servicio de Quejas	2025-04-13

34 Operación correcta

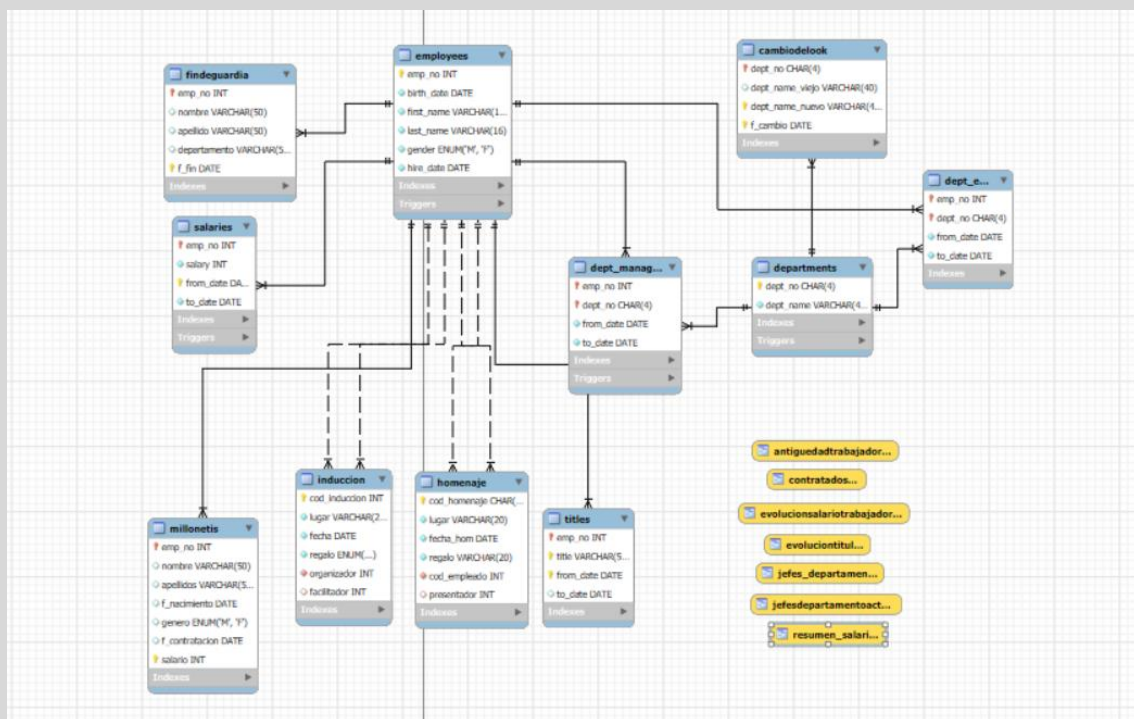
Ahora vemos como el cambio de nombre ha quedado almacenado en la nueva tabla que hemos creado.

DETALLES ADICIONALES

Lo primero a comentar es que, por alguna razón, puede que porque en este esquema ya tengo demasiados ejercicios acumulados no me muestra bien la ingeniería inversa y con el comando show tables no aparecen todas las tablas que he creado para este ejercicio.

Mirándolo durante un rato observe que a la hora de hacer la ingeniería inversa me logeaba como uno de los usuarios creados para otro ejercicio sin todos los privilegios.

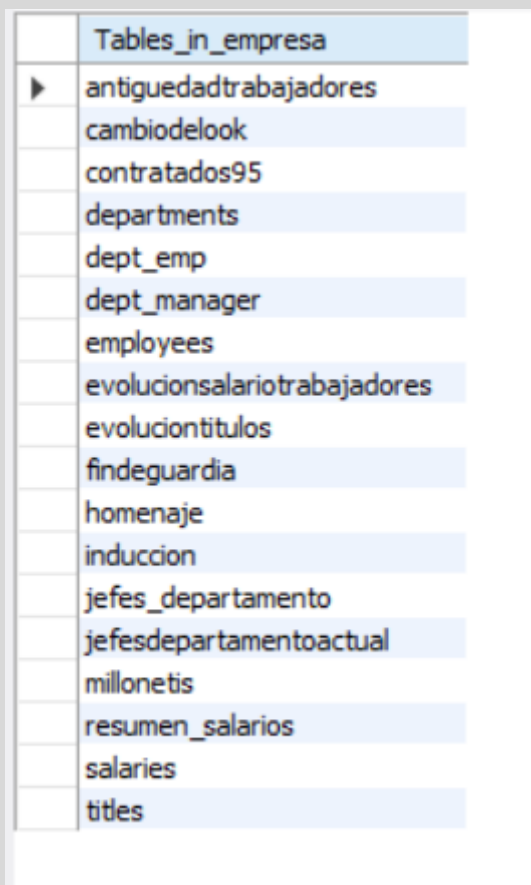
Una vez arreglado esto podemos ver la ingeniería



Es un poco caótica por contar con elementos de otros ejercicios, como las tablas del último examen, que tengo aquí los dos exámenes hechos, el A y el B.

En amarillo vemos las vistas creadas en los últimos ejercicios.

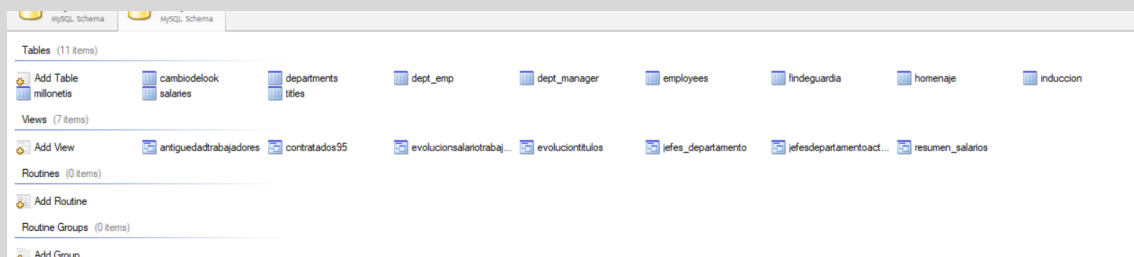
Si ejecuto también el comando show tables es otra forma de verlo, pero menos gráfica.

A screenshot of a database interface showing a list of tables. The title 'Tables_in_empresa' is at the top. Below it is a list of table names: 'antiguedadtrabajadores', 'cambiodelook', 'contratados95', 'departments', 'dept_emp', 'dept_manager', 'employees', 'evolucionsalariotrabajadores', 'evoluciontitulos', 'findeguardia', 'homenaje', 'induccion', 'jefes_departamento', 'jefesdepartamentoactual', 'millonetis', 'resumen_salarios', 'salaries', and 'titles'. Each table name is on a separate line with a small blue arrow icon to its left.

Tables_in_empresa
antiguedadtrabajadores
cambiodelook
contratados95
departments
dept_emp
dept_manager
employees
evolucionsalariotrabajadores
evoluciontitulos
findeguardia
homenaje
induccion
jefes_departamento
jefesdepartamentoactual
millonetis
resumen_salarios
salaries
titles

En esta imagen podemos ver todas las tablas, incluidas las vistas que he ido creando a lo largo de todos estos ejercicios.

Después de esto, si queremos ver los triggers de forma más gráfica también hay una forma que tiene que ver con la ingeniería inversa.



Dentro del esquema vemos las tablas. Si pulsamos con el botón derecho sobre una de ellas y seleccionamos editar vamos a ver la información de la tabla. Lo voy a hacer con employees.

Column Name	Datatype	PK	NN	UQ	B	UN	2F	AI	G	Default/Expression
emp_id	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
birth_date	DATE	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
first_name	VARCHAR(14)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
last_name	VARCHAR(16)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
gender	ENUM('M', 'F')	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
hire_date	DATE	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Esta es la pantalla que nos aparece con la información de la tabla. Vamos a fijarnos en la parte de debajo de esta ventana.

Aquí podemos apreciar la pestaña trigger. Pulsamos sobre ella y vemos su contenido.

Podemos ver, como muestra en la imagen, un listado de los triggers asociados a la tabla y de que tipo son estos triggers.

Con todo esto tenemos una buena forma de estar informados de como tenemos estructurada nuestra BD de una forma más gráfica que nos puede proporcionar mucha información y nos puede ser de gran utilidad.