AGENCIA DE CASTINGS

Rodrigo Roca Román

Javier Alonso Lledó

Grado en Ingeniería Informática

12:00-14:00

29-10-2020

Contenido

[Fase 1 3](#_Toc58268473)

[Diccionario de datos 3](#_Toc58268474)

[Modelo Entidad-Relación Extendido 6](#_Toc58268475)

[Fase 2 7](#_Toc58268476)

[Transformación del modelo ER al modelo relacional 7](#_Toc58268477)

[Modelo relacional en Pgmodeler 8](#_Toc58268478)

[Consultas SQL 9](#_Toc58268479)

[Consultas Álgebra Relacional 16](#_Toc58268480)

[Fase 3 16](#_Toc58268481)

[Normalización 16](#_Toc58268482)

[Creación de roles y usuarios 16](#_Toc58268483)

[Implementación de disparadores 17](#_Toc58268484)

[Acceso desde Java 17](#_Toc58268485)

# Fase 1

## Diccionario de datos

|  |  |  |  |
| --- | --- | --- | --- |
| Entidad | Atributo | Dominio | Restricción(Pk, FK,…) |
| CLIENTE | codigo\_cliente | char[10] | PK, \* |
| CLIENTE | nombre | char[40] | \*\*\* |
| CLIENTE | direccion | dirección | \*\* |
| CLIENTE | telefono | int[9] |  |
| CLIENTE | persona\_de\_contacto | char[40] | \* |
| CLIENTE | actividad | char[1] | {M,P}(P= Publicidad y cine , M= moda) |
| CASTING | codigo\_casting | char[10] | PK, \* |
| CASTING | nombre | char[40] | \*\*\* |
| CASTING | descripcion | char[1000] |  |
| CASTING | fecha\_de\_contratacion | date |  |
| CASTING | coste | int | >0 |
| ONLINE | numero\_de\_personas | int | >0 |
| ONLINE | fecha | date |  |
| ONLINE | plataforma\_web | URL | Debe empezar por https://www y terminar en .com o .es . |
| PRESENCIAL | numero\_de\_personas | int | >0 |
| AGENTE | DNI | char[9] | \*\*\*\* |
| AGENTE | nombre | char[40] | \*\*\* |
| AGENTE | direccion | dirección | \*\* |
| FASE | codigo\_fase | int | PK,>0 |
| FASE | fecha\_de\_inicio | date |  |
| PRUEBA INVIDIDUAL | numero | int | PK,>0 |
| PRUEBA INVIDIDUAL | fecha | date |  |
| PRUEBA INVIDIDUAL | sala\_de\_celebracion | dirección | \*\* |
| PRUEBA INVIDIDUAL | descripcion | char[1000] |  |
| PRUEBA INVIDIDUAL | coste | int | >0 |
| CANDIDATO | codigo\_candidato | char[10] | PK,\* |
| CANDIDATO | nombre | char[40] | \*\*\* |
| CANDIDATO | direccion | dirección | \*\* |
| CANDIDATO | telefono | int[9] |  |
| CANDIDATO | fecha\_de­­­\_nacimiento | date |  |
| CANDIDATO | resultado\_prueba | char[1] | {Y,N},(Y=Aprobado ,N=Suspenso) |
| REPRESENTANTE | NIF | char[9] | PK, \*\*\*\* |
| REPRESENTANTE | nombre | char[40] | \*\*\* |
| REPRESENTANTE | telefono | int[8] |  |
| REPRESENTANTE | direccion | dirección |  |
| PERFIL | codigo\_de\_perfil | char[10] | PK , \* |
| PERFIL | provincia | provincia |  |
| PERFIL | sexo | char[1] | {M,F}(M=Masculino, F= Femenino) |
| PERFIL | altura | Int | >0 |
| PERFIL | edad | Int | >0 |
| PERFIL | color\_del\_pelo | char[20] |  |
| PERFIL | color\_de\_ojos | char[20] |  |
| PERFIL | especialidad | char[1] | {M,A}(M=modelo, A= Actor) |
| PERFIL | experiencia | char[1] | {Y,N}(Y= Si , N= No) |
| ADULTO | DNI | char[9] | PK, \*\*\*\* |
| NIÑO | nombre\_tutor | char[40] | PK, \*\*\* |

\*🡪 Un código contiene tanto números como letras pudiendo tener una longitud fija de 10 caracteres.

\*\*🡪 Una dirección contiene el siguiente formato: Nº/ Calle

\*\*\*🡪 Solo letras y espacio entre palabras, primera de cada palabra en mayúsculas

\*\*\*\*🡪 http://www.interior.gob.es/web/servicios-al-ciudadano/dni/calculo-del-digito-de-control-del-nif-nie .

Los 8 primeros numéricos, el último alfabético, resto de dividir el número por 23

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Relación | Entidades | Cardinalidad | Atributos | Dominio |
| Contrata | CLIENTE  CASTING | (0,1)  (0,n) | coste | int |
| Es | CASTING(MADRE)  ONLINE(HIJA)  PRESENCIAL(HIJA) | (1,1)  (0,1)  (0,1) |  |  |
| Dirige | PRESENCIAL  AGENTE | (1,1)  (1,1) |  |  |
| Necesita | CASTING  PERFIL | (1,n)  (1,n) |  |  |
| Consta De | PRESENCIAL  FASE | (1,1)  (1,n) |  |  |
| Se divide en | FASE  PRUEBA INDIVIDUAL | (1,1)  (1,n) |  |  |
| Realiza | PRUEBA INDIVIDUAL  CANDIDATO | (0,n)  (1,n) | resultado\_prueba | char[1]{Y,N}  (Y=Aprobado, N=) |
| Corresponde | CANDIDATO  PERFIL | (0,1)  (1,1) |  |  |
| Gestiona | CANDIDATO  REPRESENTANTE | (1,n)  (0,1) |  |  |
| Es | PERFIL(MADRE)  ADULTO(HIJA)  NIÑO(HIJA) | (1,1)  (0,1)  (0,1) |  |  |

Documentación detallada de entidades y relaciones

La entidad **CLIENTE** representa a un cliente de la compañía de catering con sus datos básicos como un **nombre, dirección, teléfono, etc…** . El **CLIENTE** al utilizar la relación **Contrata** debe proporcionar al **CASTING**  el **coste** de producción.

El **CASTING**  tiene que ser o **PRESENCIAL** u **ONLINE** y se encarga mediante la relación **Se corresponde**  de mostrar todos los perfiles (**PERFIL**)de los candidatos(**CANDIDATO)** para que sean seleccionados por el **CLIENTE** ,permitir que los **CANDIDATOS** usen la relación **realiza** para participar las pruebas individuales (**PRUEBA INDIVIDUAL**) al ser seleccionados por un **CASTING.**

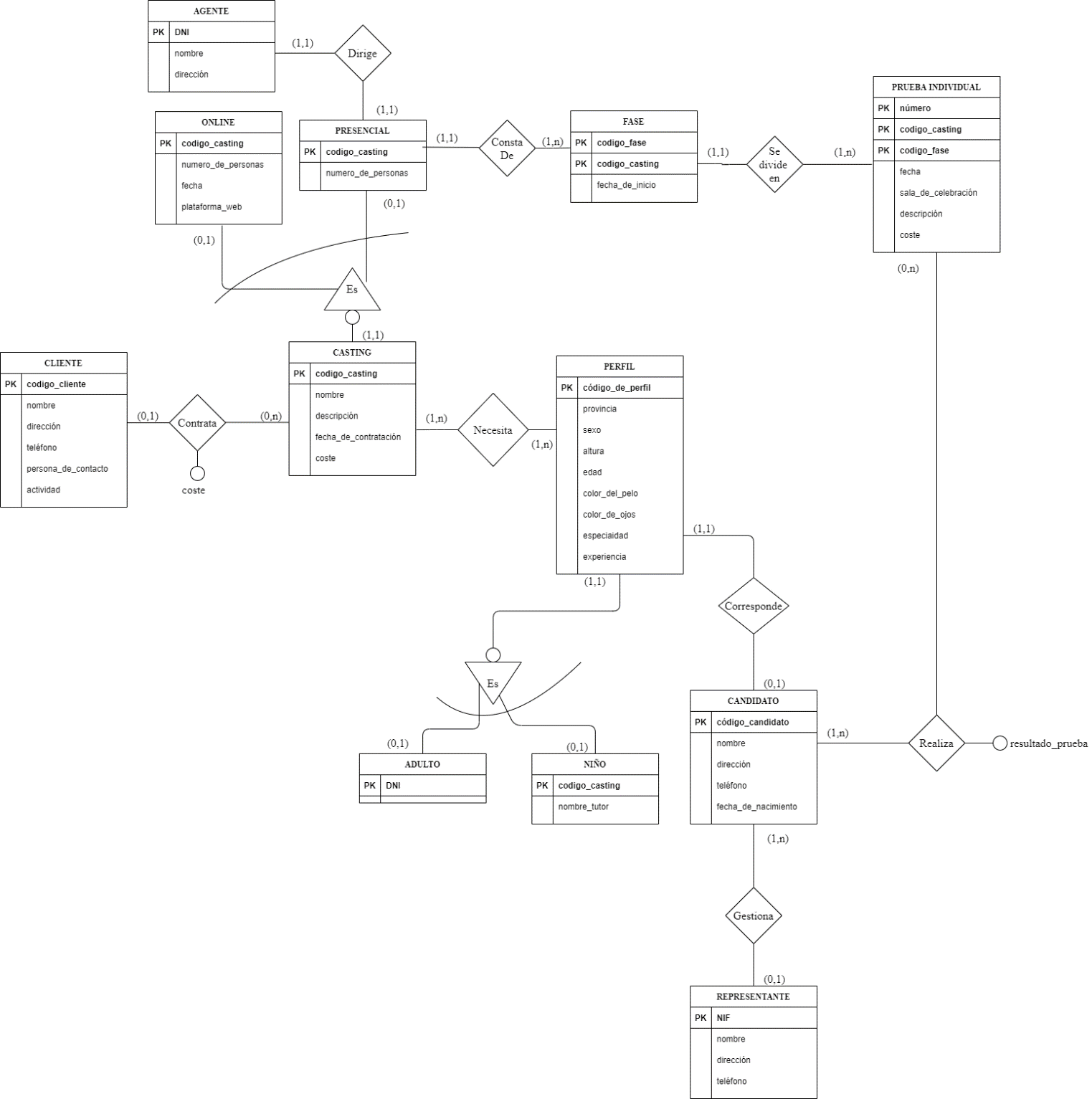
Los castings presenciales (**CASTING PRESENCIAL**)son dirigidos (relación **Dirige**)por un **AGENTE** y **consta de** una o varias fases (**FASE**)**.**

Las fases (**FASE**)de los castings presenciales (**CASTING PRESENCIAL**) están numeradas y ordenadas dependiendo del número de fases que tenga el casting. Cada **FASE**  se divide en una o varias pruebas individuales (**PRUEBA INDIVIDUAL**)con el objetivo de calificar a los diversos candidatos de una audición. Para ello se utiliza la relación **Realiza** y devuelve al **CANDIDATO** el **resultado\_prueba** que puede ser positivo o negativo indicando que el **CANDIDATO** ha superado o no la prueba.

Cada **CANDIDATO** puede tener o no un **REPRESENTANTE** que **Gestiona** las acciones y datos de este. Además a cada **CANDIDATO** le **Corresponde**  un único **PERFIL** con todas las características físicas y laborales de este.

Los perfiles (**PERFIL**) tienen que ser o un **ADULTO** o de **NIÑO** cada uno con sus datos únicos y con los que en total serán utilizados para poder participar en un **CASTING.**

## Modelo Entidad-Relación Extendido



# Fase 2

## Transformación del modelo ER al modelo relacional

Cliente (codigo\_cliente, nombre, direccion, telefono, persona\_de\_contacto, actividad)

Casting (codigo\_casting, nombre, descripcion, fecha\_de\_contratacion, **codigo\_cliente**, coste)

Online (**codigo\_casting**, numero\_de\_personas, fecha, plataforma\_web)

Presencial (numero\_de\_personas, **codigo\_casting**, **DNI**)

Agente (DNI, nombre, direccion)

Fase (codigo\_fase, fecha\_de\_inicio, **codigo\_casting**)

Prueba individual (numero, fecha, sala\_de\_celebración, descripcion, coste, **codigo\_fase**, **codigo\_casting**)

Candidatos (codigo\_candidato, nombre, direccion, telefono, fecha\_de\_nacimiento, **nif\_representante**, **codigo\_de\_perfil**)

Niño (nombre\_tutor, **codigo\_candidato**)

Adulto (DNI, **codigo\_candidato**)

Representante (NIF, nombre, telefono, direccion)

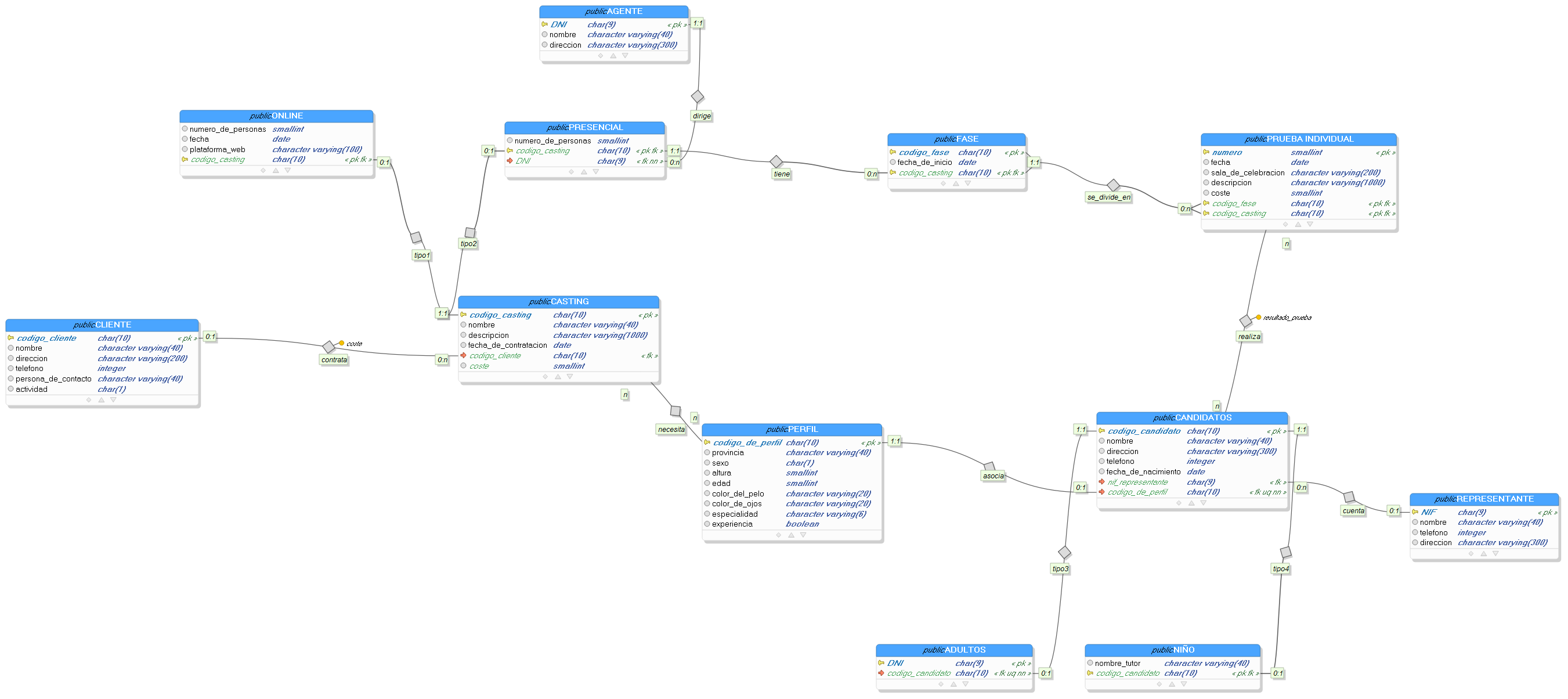
Perfil (codigo\_de\_perfil, provincia, sexo, altura, edad, color\_del\_pelo, color\_de\_ojos, especialidad, experiencia)

Casting\_necesita\_perfil (**codigo\_casting**, **codigo\_de\_perfil**)

Candidato\_realiza\_prueba (**codigo\_candidato**, **numero**, **codigo\_fase**, **codigo\_casting**, resultado\_prueba)

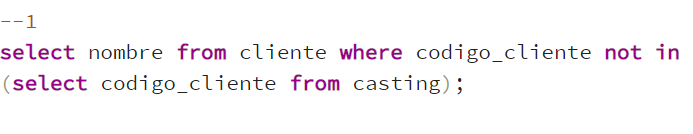
## Modelo relacional en Pgmodeler

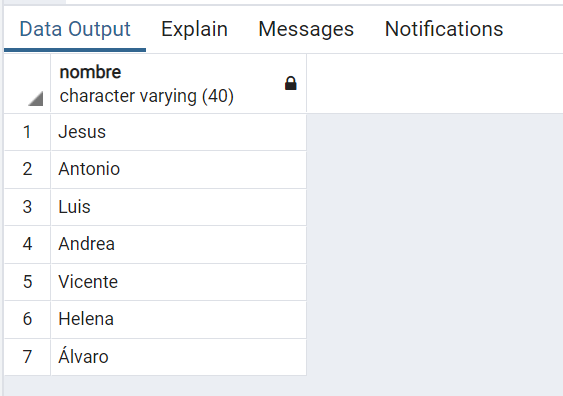
Se incluirá una captura de pantalla del diagrama generado. Se incluirán en la entrega el fichero .dbm obtenido y el fichero .sql generado.



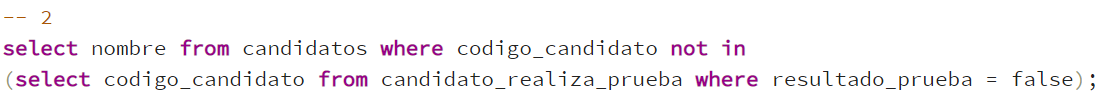
## Consultas SQL

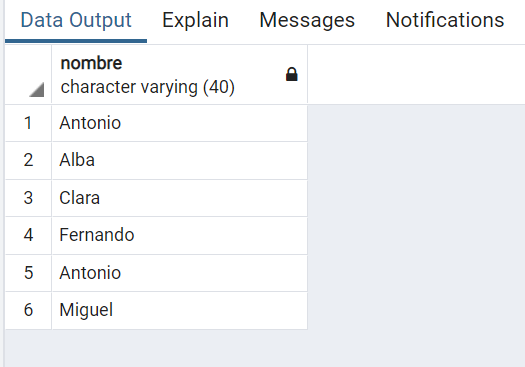
**1)** **Mostrar el nombre de los clientes que no han contratado ningún casting.**

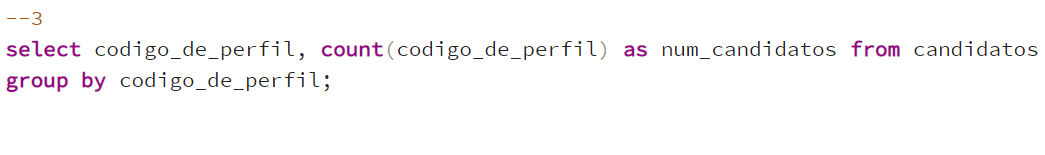


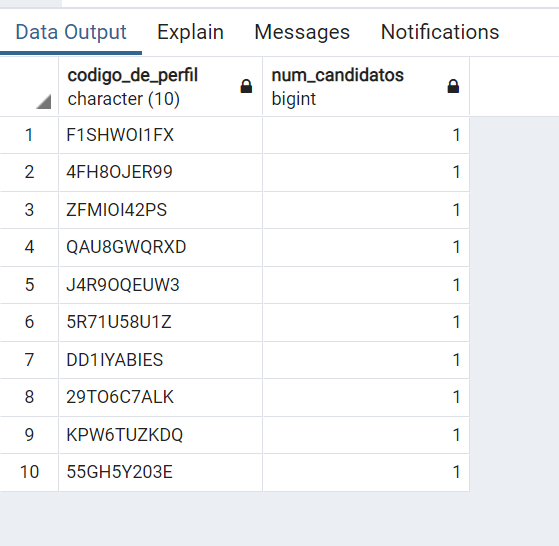


**2)** **Mostrar el nombre de los candidatos que han superado todas las pruebas.**

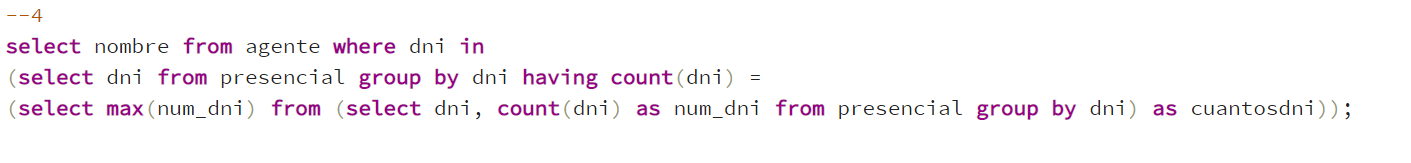


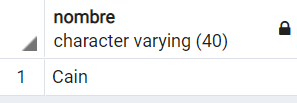


**3) Mostrar el número de candidatos que hay asociados a cada perfil.**

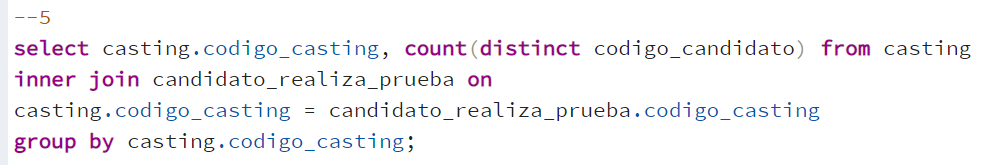


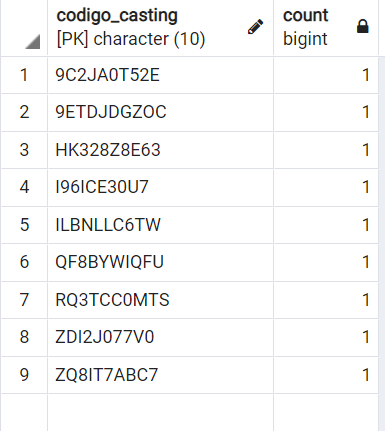
**4)** **Mostrar el nombre del empleado que más castings ha dirigido.**



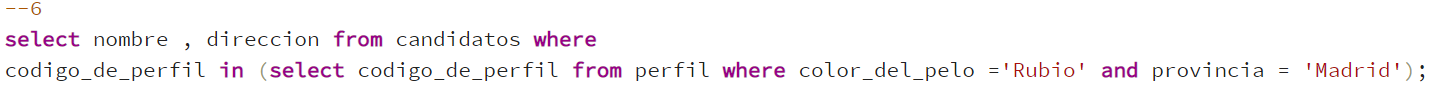


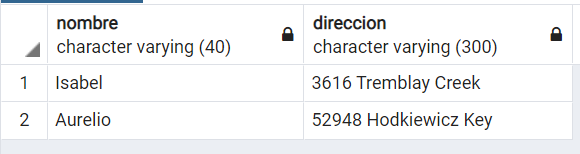
**5)** **Mostrar el número de candidatos que se han presentado a cada casting (que al menos hayan realizado una prueba).**



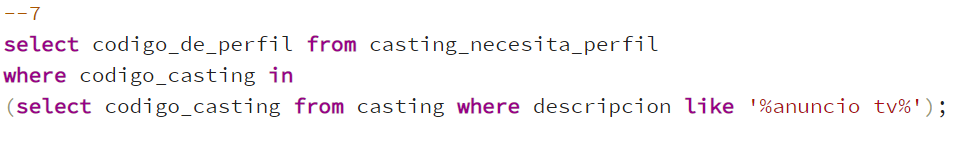


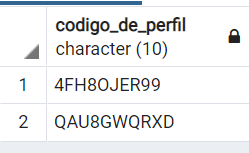
**6)** **Mostrar el nombre y la dirección de las candidatas que tengan el pelo rubio y sean de Madrid.**



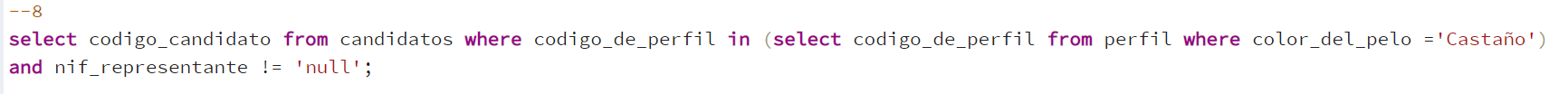


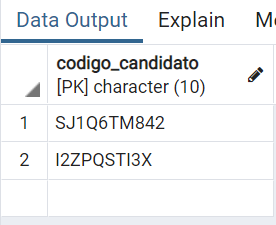
**7)** **Mostrar el código de perfil de los perfiles requeridos en los castings que incluyen la subcadena “anuncio tv” en su descripción.**



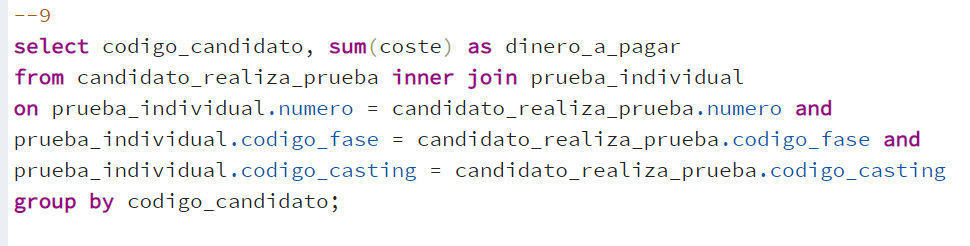


**8)** **Mostrar el código de los candidatos que tienen representante y tienen el pelo castaño.**



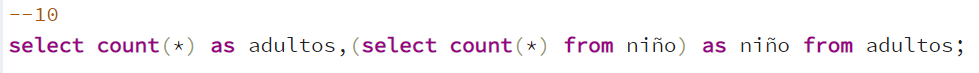


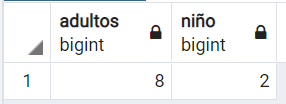
**9)** **Mostrar el precio total que ha de pagar cada candidato.**



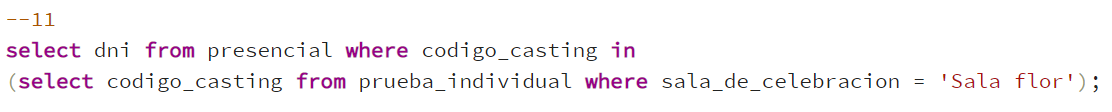


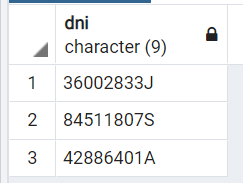
**10)** **Mostrar el número de candidatos adultos y el número de candidatos niños que hay en la base de datos.**



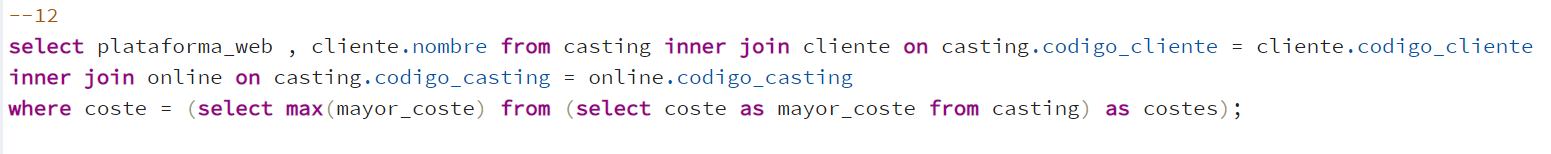


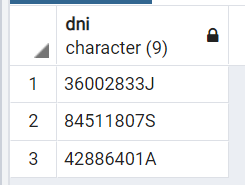
**11)** **Mostrar el dni del agente que ha dirigido el casting en el que alguna prueba individual se ha llevado a cabo en la sala “flor”.**



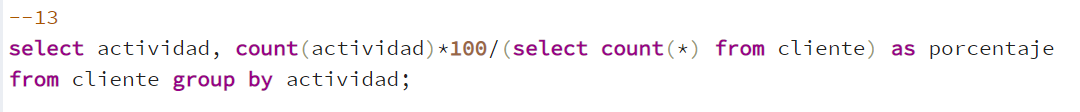


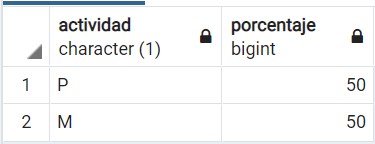
**12)** **Mostrar la plataforma web que se ha usado en el casting online más caro, así como el nombre del cliente que ha contratado dicho casting.**



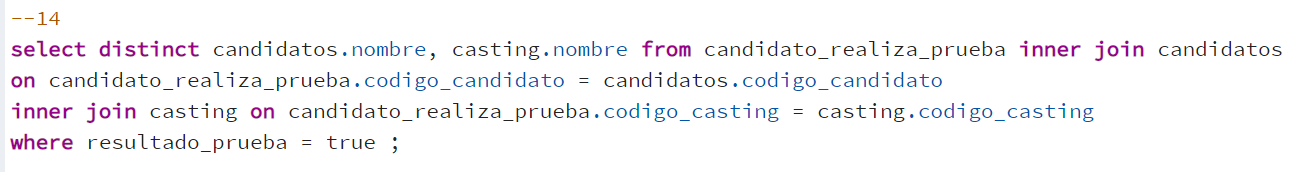


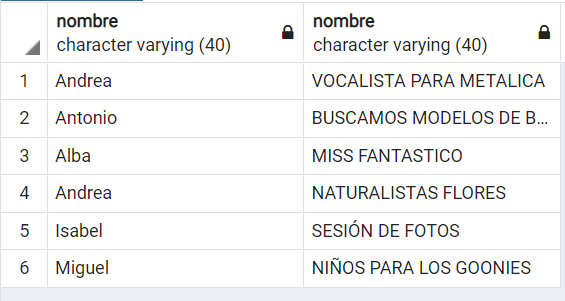
**13)** **Mostrar el porcentaje de clientes que hay de cada tipo**



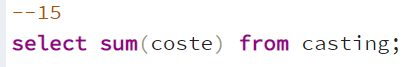


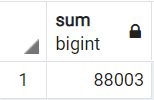
**14)** **Mostrar el nombre de los candidatos que han superado alguna prueba de algún casting, así como el nombre del casting.**



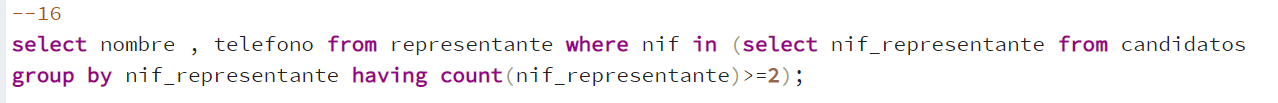


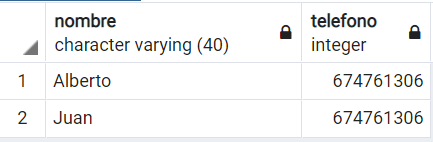
**15)** **Mostrar el dinero total recaudado por la empresa.**



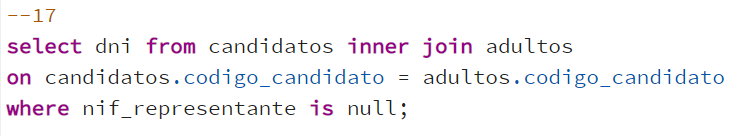


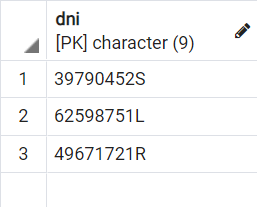
**16)** **Mostrar el nombre y el teléfono de los representantes que representen a 2 candidatos como mínimo.**



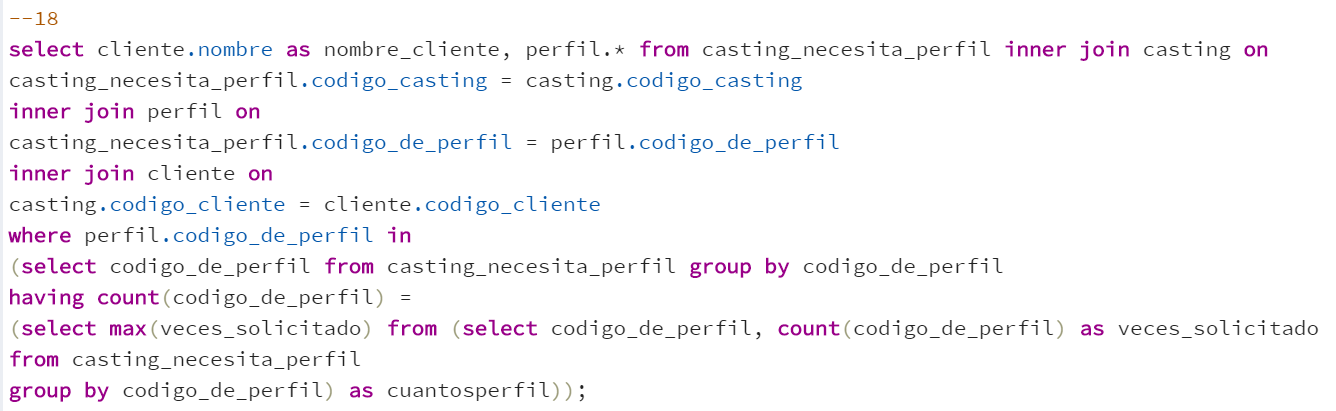


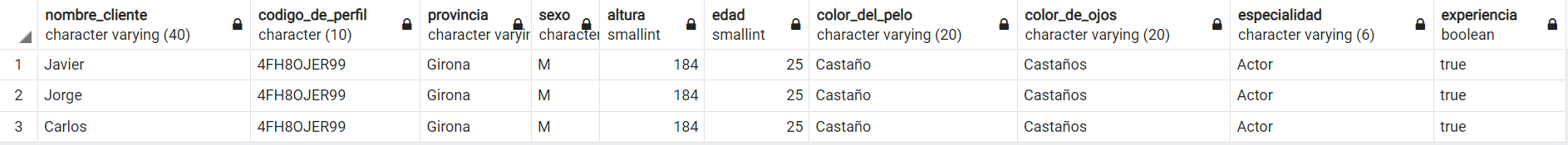
**17)** **Mostrar el dni de los adultos que no tengan representante**



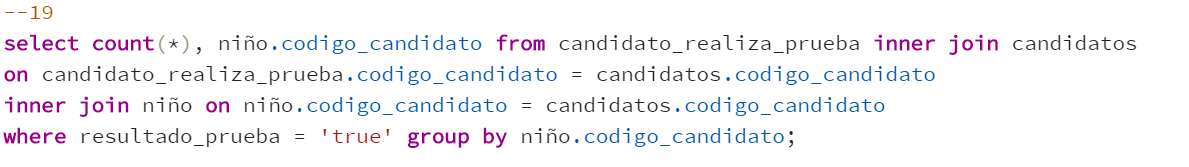


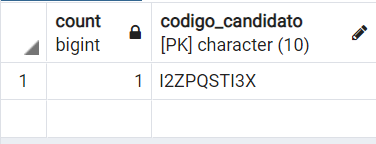
**18)** **Mostrar los datos del perfil más demandado así como el nombre del cliente que lo ha requerido para su casting.**



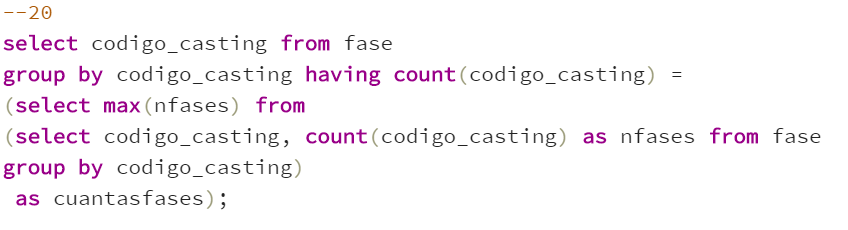


**19)** **Mostrar el número de pruebas superadas por cada niño.**





**20)** **Mostrar el código del casting que más fases tiene.**



# Fase 3

## Normalización

Análisis de dependencias funcionales y normalización de cada tabla debidamente justificada.

## Creación de roles y usuarios

Para poder crear los roles administrador , gestor y recepcionista hemos utilizado la siguiente operación: **create role “nombre del rol a crear”;**

Después de crear los roles utilizamos la operación **grant “lista de privilegios” on “tablas” to “rol”;** para asignar los distintos privilegios sobre las tablas a cada rol.

El código siguiente es el utilizado en la práctica para crear los roles y asignarles los distintos privilegios:

*create role administrador;*

*grant all privileges on adultos, agente, candidato\_realiza\_prueba, candidatos, casting, casting\_necesita\_perfil, cliente, fase, niño, online, perfil,*

*presencial, prueba\_individual, representante to administrador;*

*GRANT ALL ON schema public TO administrador;*

*create role gestor;*

*grant select, update, delete, insert on adultos, agente, candidato\_realiza\_prueba, candidatos, casting, casting\_necesita\_perfil, cliente, fase, niño, online, perfil,*

*presencial, prueba\_individual, representante to gestor;*

*create role recepcionista;*

*grant select on adultos, agente, candidato\_realiza\_prueba, candidatos, casting, casting\_necesita\_perfil, cliente, fase, niño, online, perfil,*

*presencial, prueba\_individual, representante to recepcionista;*

Con los roles ya creados solo falta crear y configurar los usuarios. Para ello utilizamos la operación **create user “nombre de usuario” password ‘contraseña’;** para crearlos y la operación **grant “rol” to “usuario”;** para asignarle el rol que corresponda a cada usuario.

*create user javier password 'javier';*

*create user rodrigo password 'rodrigo';*

*create user jefe password 'jefe';*

*grant administrador to jefe;*

*grant gestor to rodrigo;*

*grant recepcionista to javier;*

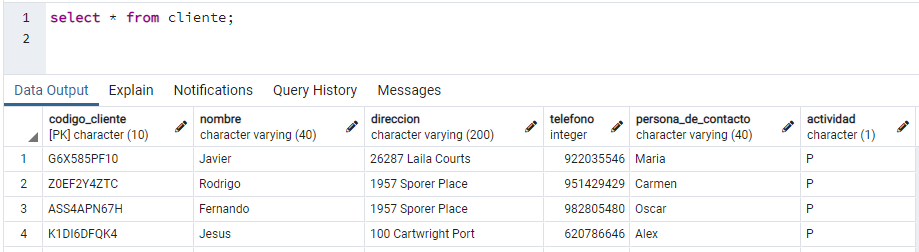
En el projecto se encontrarán el fichero SETUPBD-PL· donde se pueda obtener este código.

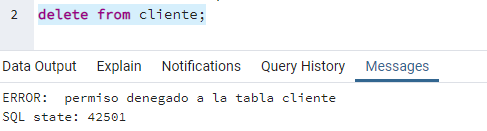
Los usuarios con sus contraseñas creados para esta práctica son los siguientes:

* Usuario: javier , Rol: Recepcionista , Contraseña: javier
* Usuario: rodrigo , Rol: Gestor , Contraseña: rodrigo
* Usuario: jefe , Rol: Administrador , Contraseña: jefe

Ejemplo de operación válida y operación no válida para el usuario javier con el rol de recepcionista (Privilegios: SELECT).

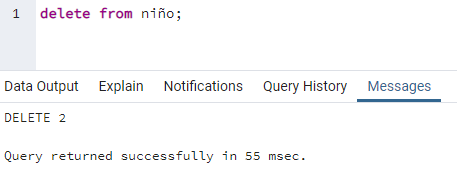
Operación permitida:

Operación no permitida:

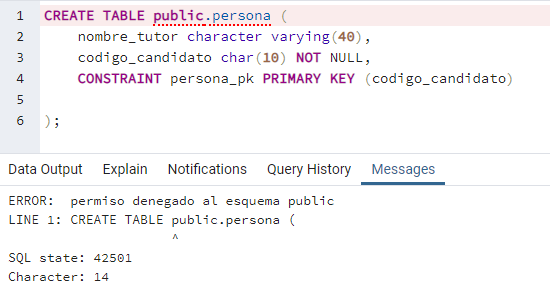


Ejemplo de operación válida y operación no válida para el usuario rodrigo con el rol de gestor (Privilegios: SELECT, INSERT, UPDATE, DELETE).

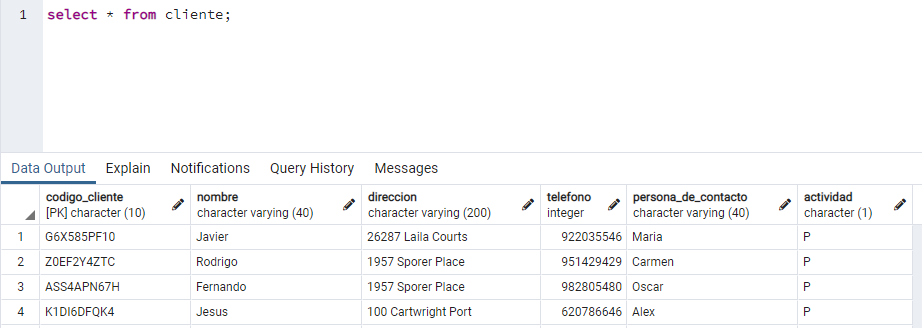
Operación permitida:



Operación no permitida:



Ejemplo de operación válida y operación no válida para el usuario rodrigo con el rol de gestor (Privilegios: ALL).

Operación permitida:

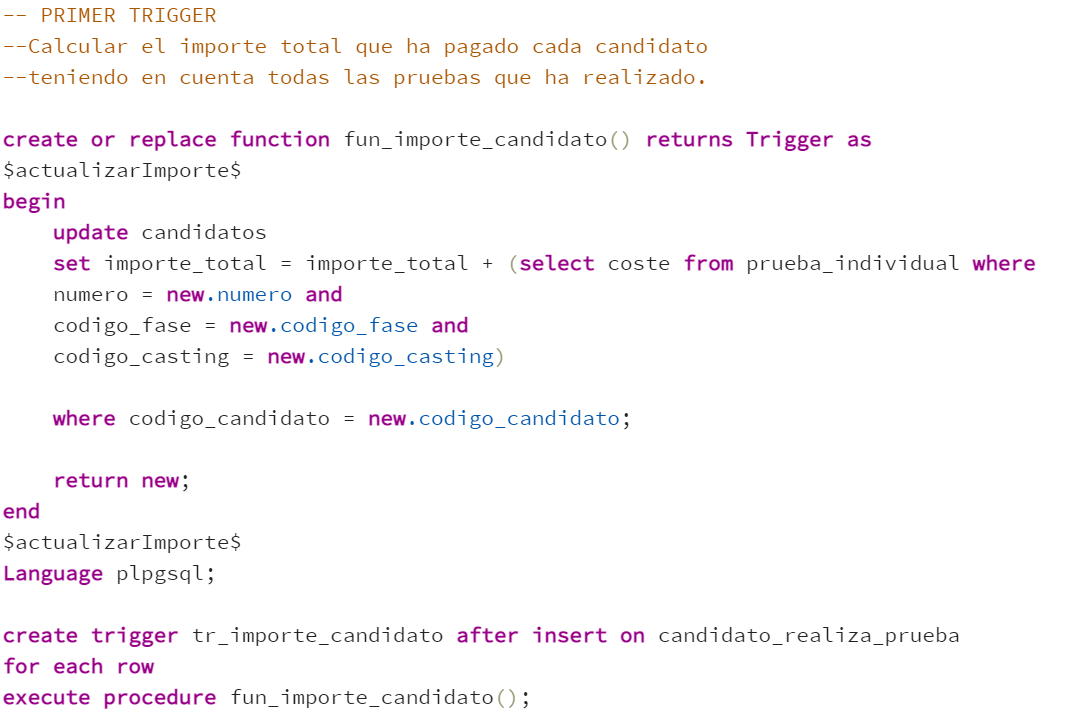
Operación no permitida:

No hay ninguna operación no permitida para el rol administrador.

## Implementación de disparadores

Se indicará el funcionamiento de cada disparador y cuándo se ejecuta. Para cada uno de ellos se incluirá el código de la función disparadora y una captura con la configuración del disparador. A continuación, se ejecutará la sentencia que dispare dicho código y se incluirán capturas de pantalla en las que se verifique el correcto funcionamiento del disparador.

Primer disparador:

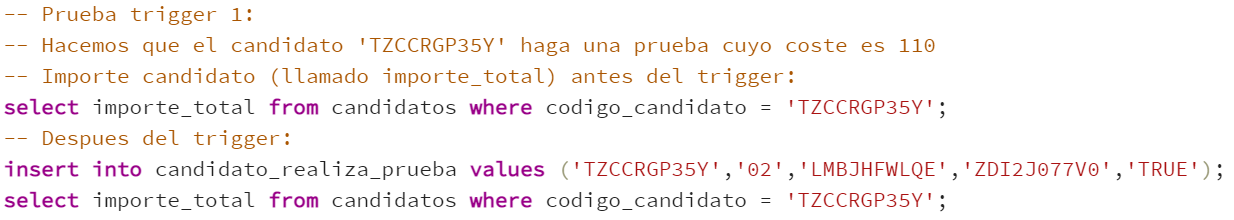


Cuando se inserta una nueva tupla en la tabla candidato\_realiza \_prueba, la cual contiene toda la información necesaria para saber qué candidato ha hecho qué prueba (y es la manera de indicar que un candidato ha realizado una prueba en nuestra base de datos), se activa este disparador.

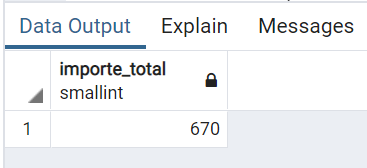
Lo que hace este disparador es obtener el coste de la prueba almacenado en la tabla prueba\_individual y añadirlo al total del importe que el candidato debe pagar por realizar las pruebas que realice. El campo que almacena este número es importe\_total y está en la tabla candidatos.

El cómo funciona se fundamenta en hacer un update en la tabla candidatos y actualizando el valor importe\_total sumando el valor que ya había más el de la prueba realizada por el candidato. El valor de importe\_total cuando se inserta un candidato nuevo es, por defecto, 0.

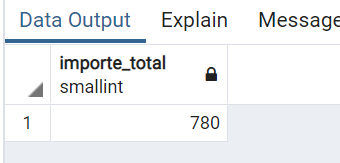
Comprobación:

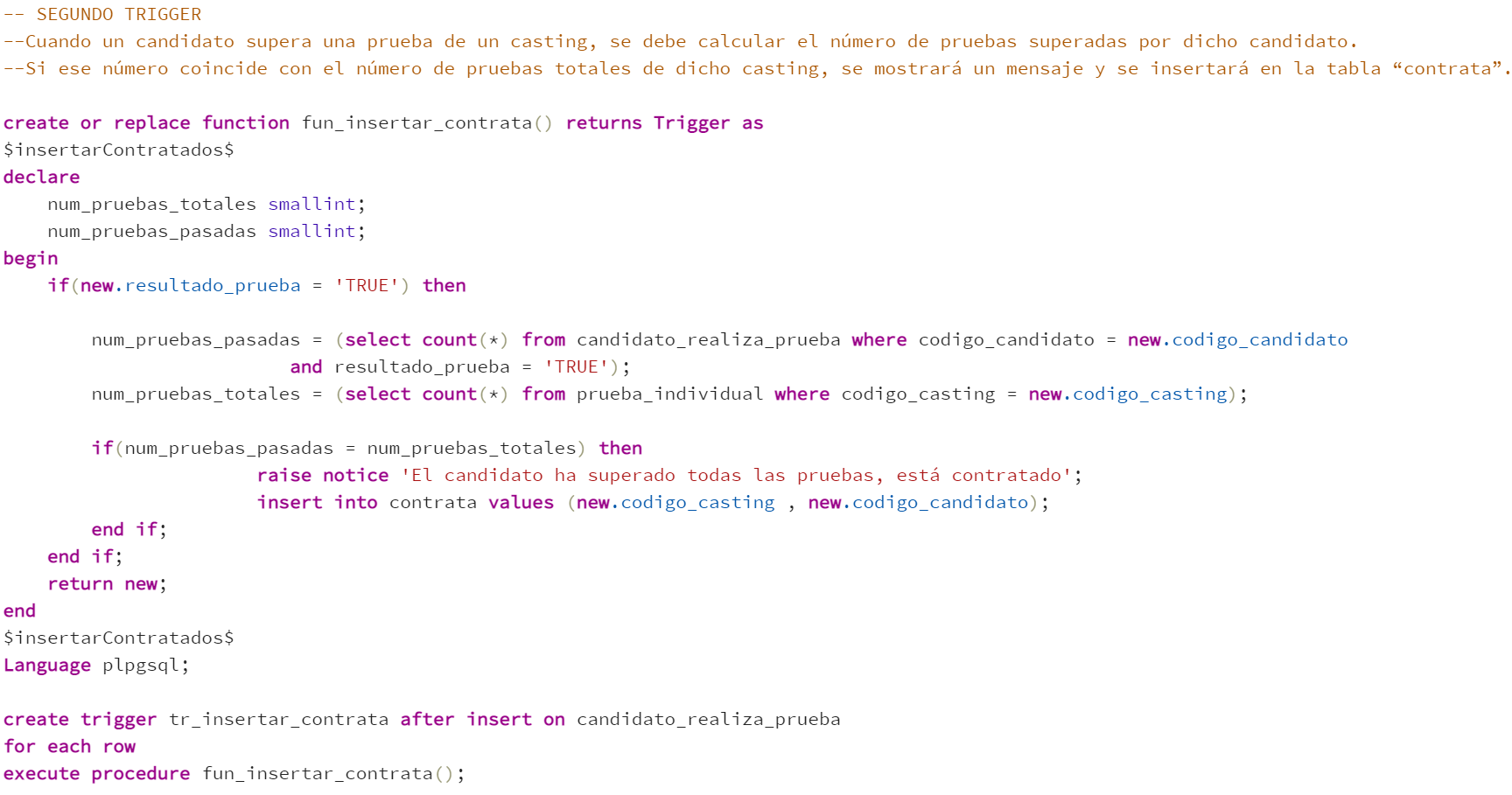


Antes del disparador:



Después del disparador:

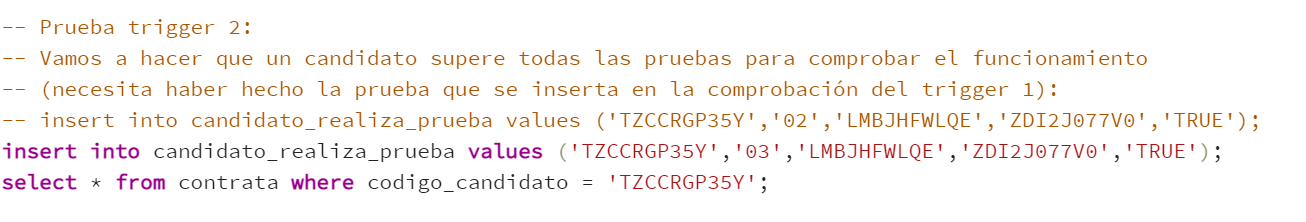


Segundo disparador:

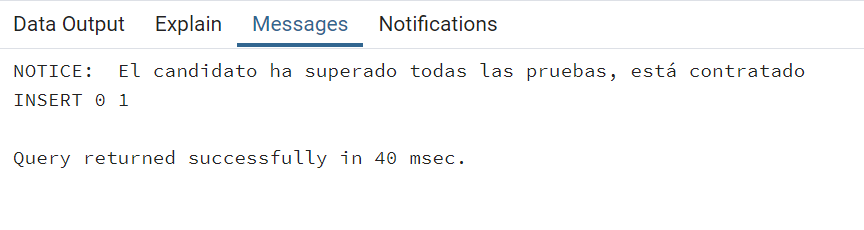
El evento que hace que este disparador se active es el mismo que en el caso anterior: una inserción en la tabla candidato\_realiza\_prueba.

Lo que hace este disparador es que si un candidato ha superado todas las pruebas (el campo resultado\_prueba tiene que ser TRUE en todas las pruebas) avisa con un raise notice de que se han superado todas las pruebas e inserta los valores del candidato que ha realizado la prueba y del casting en el que participa en la tabla *contrata*.

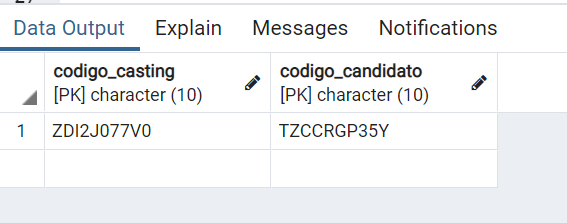
Si el resultado de la prueba que ha hecho que el disparador se active es FALSE, entonces el disparador no hace nada ya que es evidente que no ha superado todas las pruebas (y así se ahorra tiempo).

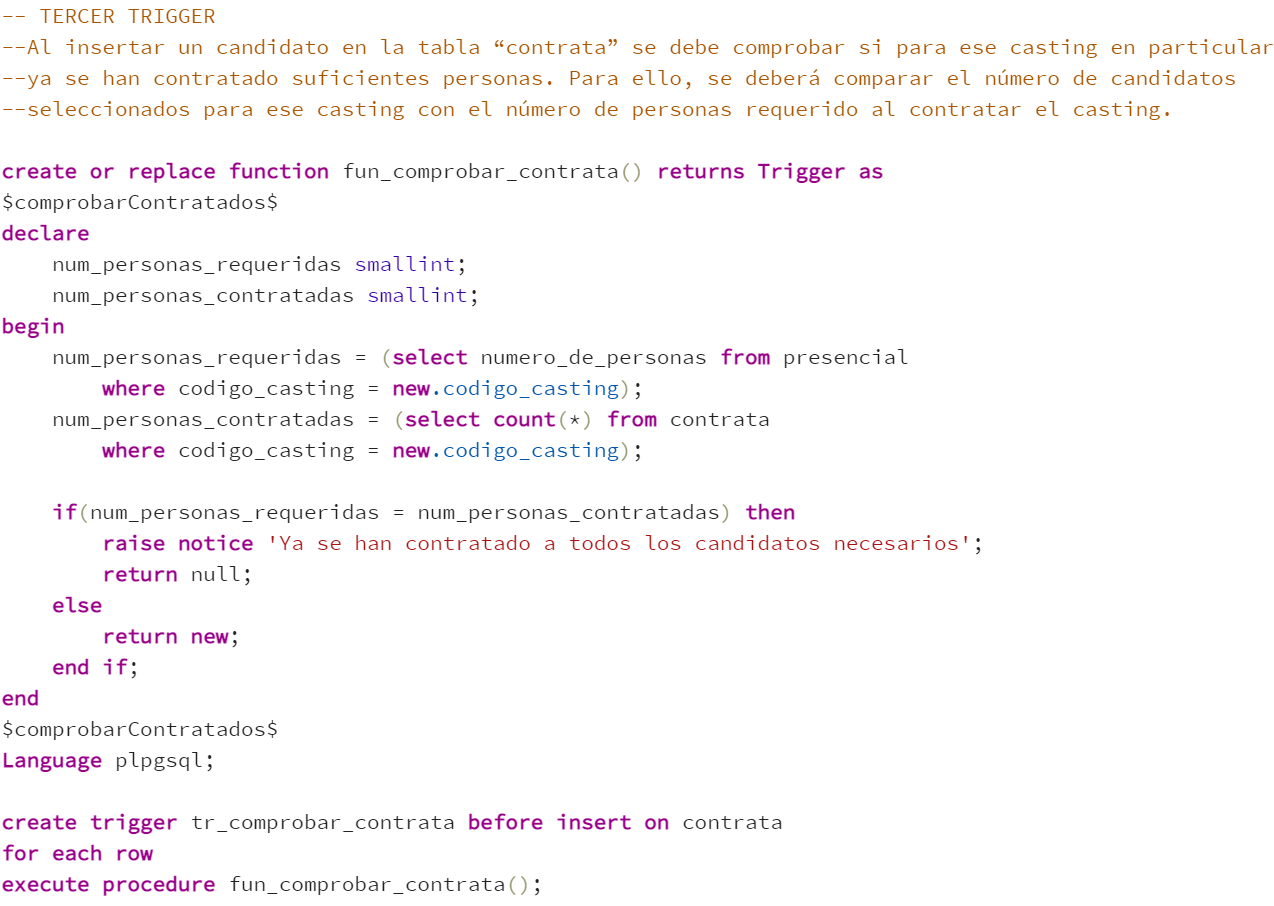
Comprobación:

Al hacer el insert se activa el disparador:



Se añade la entrada en la tabla contrata:



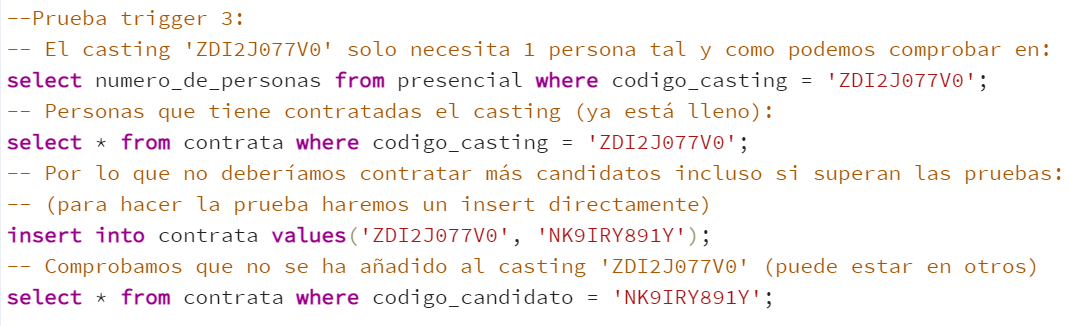
Tercer disparador:

En este caso, el evento que hace que este disparador se active es la inserción en la tabla contrata. Al tratarse de que queremos evaluar la validez de los datos, el disparador se activa antes de la inserción y no después.

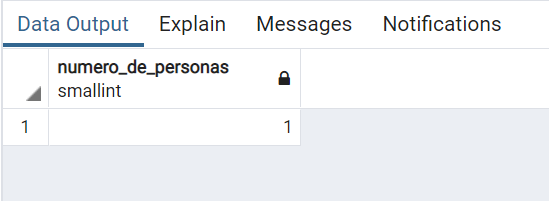
Al dispararse con la inserción de tuplas en la tabla contrata, el segundo disparador puede que active este.

Tenemos que comprobar que el número de personas que necesita el casting es superior al número actual de candidatos contratados (no queremos que el casting contrate personal de más). Por ello, en la función que se invoca cuando el disparador se activa, si ya no se necesitan más personas (primer if) devolvemos null y avisamos con un raise notice de tal manera que no se produzca la inserción en la tabla contrata. En caso contrario el casting necesita más candidatos (el else) y por lo tanto permitimos que se haga la inserción devolviendo new (la nueva tupla). Nunca se va a dar el caso de que tengamos más personas de las que necesitamos (debido al primer if) por ello basta con utilizar un if y un else y no se contempla otra posibilidad.

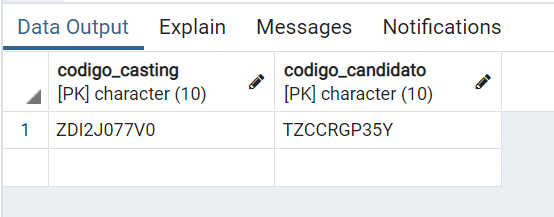
Comprobación:



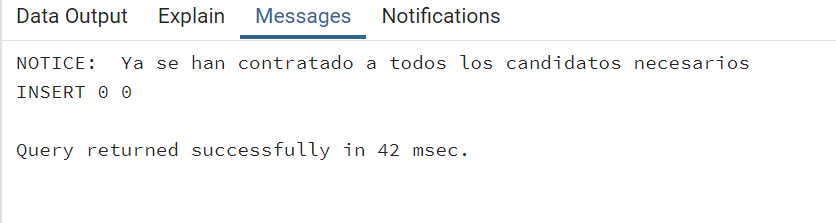
Numero de personas que necesita el casting:



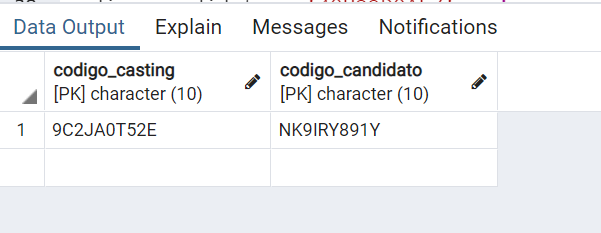
Personas que ya tiene contratadas:



Si intentamos hacer el insert:

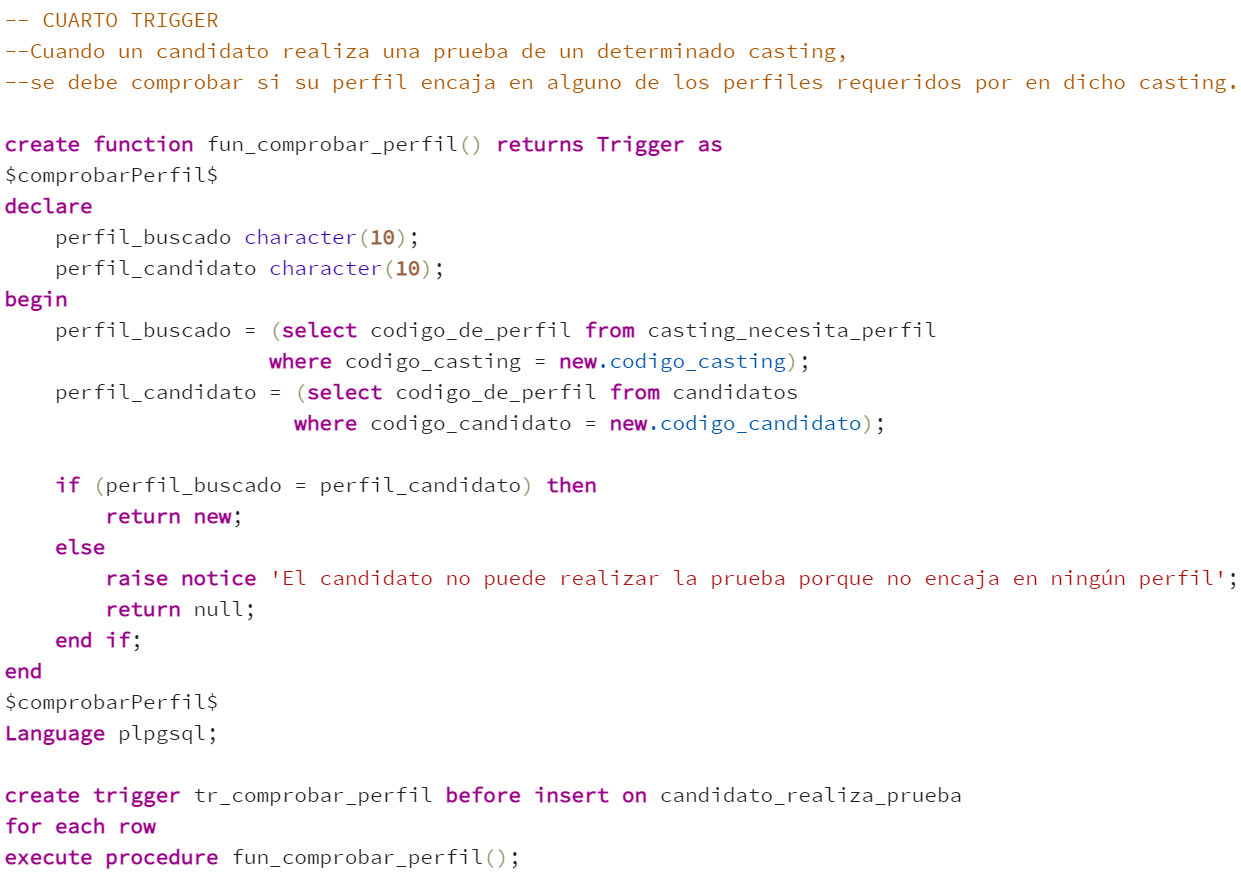


Comprobamos que la tupla que hemos intentado insertar no está en la tabla contrata: ('ZDI2J077V0', 'NK9IRY891Y')



(Está en otro casting, pero no en el que hemos intentado meterlo estando ya lleno).

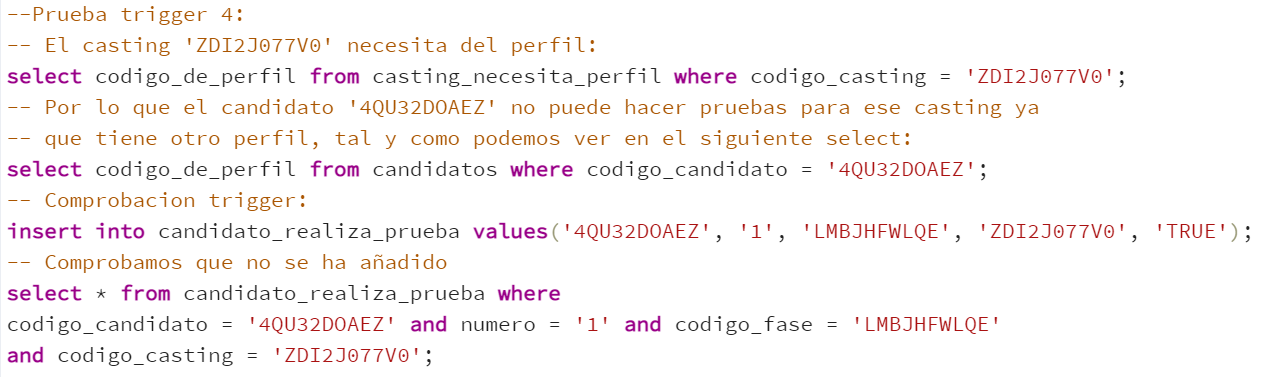
Cuarto disparador:



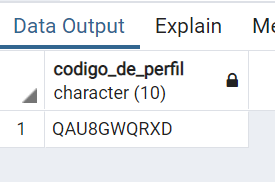
Este disparador, al igual que el primero y el segundo, se activa mediante la inserción en la tabla candidato\_realiza\_prueba. En este caso pretendemos validar los datos por lo que, a diferencia de los dos primeros disparadores, este se activa antes de la inserción.

La función que invoca el disparador se basa en comparar el perfil buscado por el casting (localizado en la tabla casting\_necesita\_perfil) y el perfil que posee el candidato que pretende hacer una prueba para ese casting. Si ambos coinciden quiere decir que el casting está buscando a gente que sean como el candidato por lo que permitimos que realice la prueba (caso else, devolvemos new y permitimos la insercion). Si no es el caso, no le permitimos hacer la prueba porque incluso si las supera el casting no está interesado en contratar a candidatos de su perfil (caso del primer if, notificamos con un raise notice y no hacemos la inserción devolviendo null).

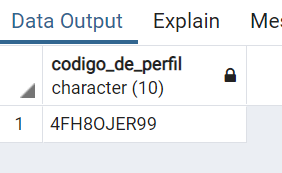
Comprobación:



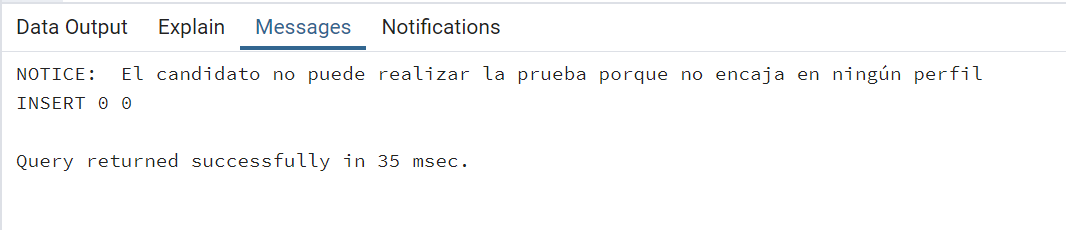
Perfil que busca el casting:



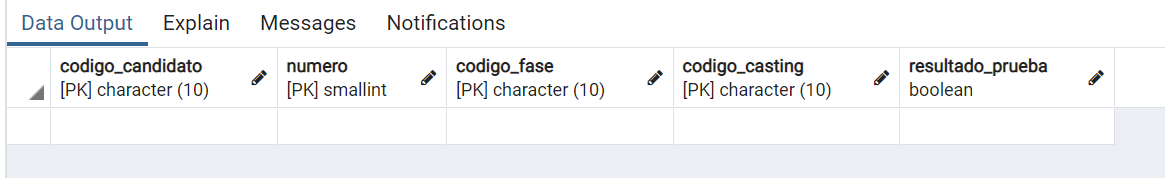
Perfil que tiene el candidato que pretendemos hacer que participe en el casting:



Dado que son diferentes no podremos hacer la inserción:

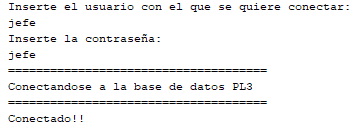


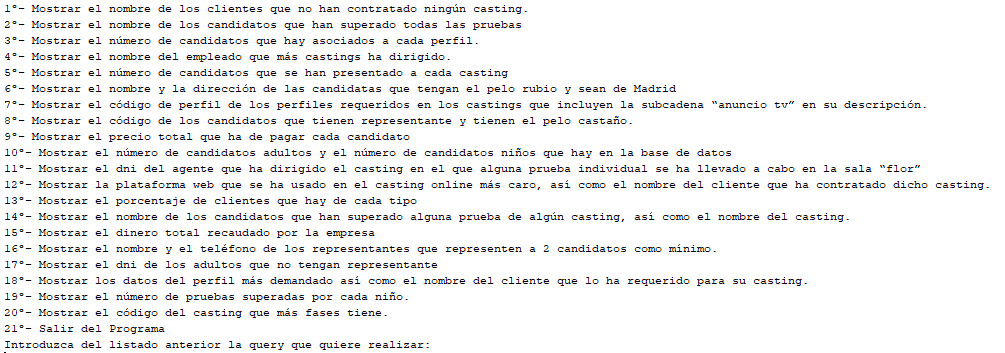
Comprobación de que no se ha añadido:

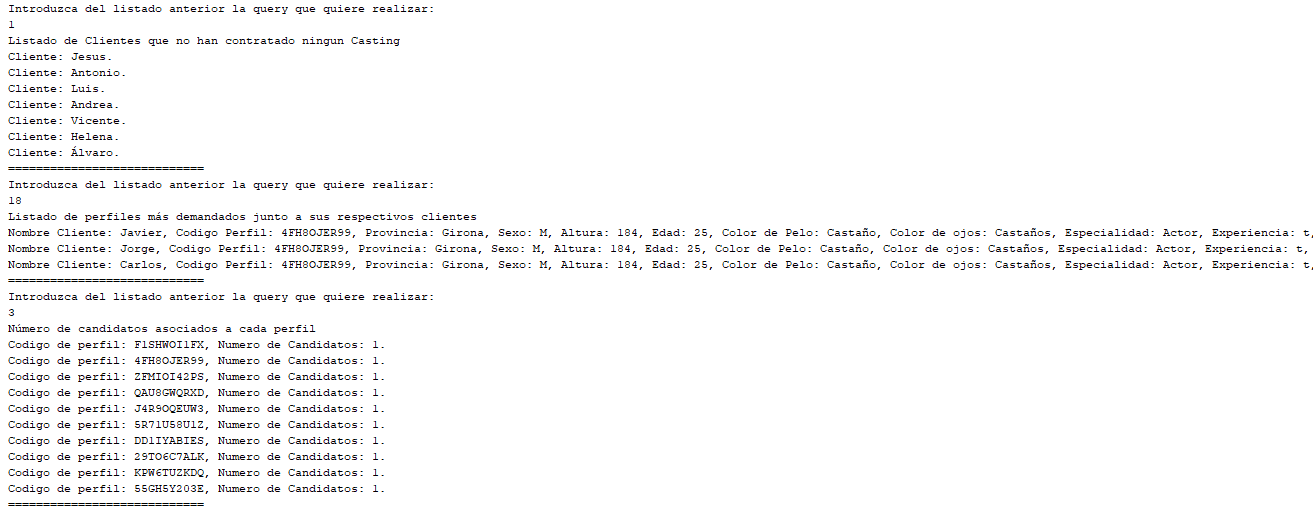


## Acceso desde Java

La aplicación JavaDatabase se desarrolla en 3 fases:

1º Autentificación del usuario con contraseña. Cuando se introduzcan el programa intentará conectarse a la base de datos PL3.

2º Consultas fijas. El usuario podrá elegir entre 20 querys introduciendo en el sistema un número entre el 1 y el 20 que ejecutarán y mostrarán por pantalla los resultados tras buscarlos en la base de datos. Esta fase se encuentra en un bucle el cual se terminará de ejecutar tras introducir la opción 21 en el programa.

Ejemplo de ejecución de algunas instrucciones SQL:

3º Desconexión y fin del programa.

