Mini-manual de DBDesigner Fork

1. Introducción

Este pequeño manual te ayudará a utilizar la herramienta DBDesigner Fork que puedes descargar desde SourceForge.net. Puedes utilizar esta herramienta para obtener el esquema lógico y el esquema físico de una base de datos relacional, a partir de un esquema conceptual.

Una vez dibujado el esquema conceptual sobre el papel, utiliza la herramienta para representarlo. Verás que la herramienta no dibuja exactamente el esquema conceptual, sino que realiza el diseño lógico relacional transformando el esquema.

El resultado que obtenemos es un conjunto de tablas con claves primarias y claves ajenas. En cada tabla podrás especificar los tipos de los datos y si aceptan nulos o no. Para las claves ajenas podrás definir las reglas de comportamiento que mantienen la integridad referencial. Además, podrás definir los índices que necesites crear. Un índice es una estructura de acceso adicional que se crea sobre una tabla para permitir un acceso más eficiente.

Seleccionando una opción del menú podrás obtener el esquema físico de la base de datos, formado por las sentencias CREATE TABLE y CREATE INDEX necesarias. Este esquema se puede generar para distintos SGBD, entre ellos PostgreSQL, utilizado en la asignatura.

2. Esquema conceptual

Vamos a ilustrar el uso de la herramienta mediante un ejemplo. Comenzaremos con un esquema conceptual donde aparece información de profesores y la docencia que imparten en diversas asignaturas.

Cada asignatura tiene un código (**codasig**), un nombre (**nomasig**), un número de créditos de teoría (**crdteo**) y de prácticas (**crdprac**), y un número de grupos de teoría (**grteo**) y de prácticas (**grprac**). Además, cada asignatura pertenece a un departamento (**depto**).

Los profesores tienen un código (**codprof**), un nombre (**nomprof**), una categoría (ASO, AYU, TEU, TU, CEU, CU) (**categ**), pertenecen a un departamento (**depto**) y cada uno imparte un número determinado de créditos (**creditos**, es la suma de los créditos de docencia del profesor).

Cada profesor puede impartir docencia de varias asignaturas y, de cada una, imparte un número de grupos de teoría (**grteoprof**) y/o de prácticas (**grpracprof**). A partir de estos requisitos, hemos obtenido el esquema conceptual de la figura 1.

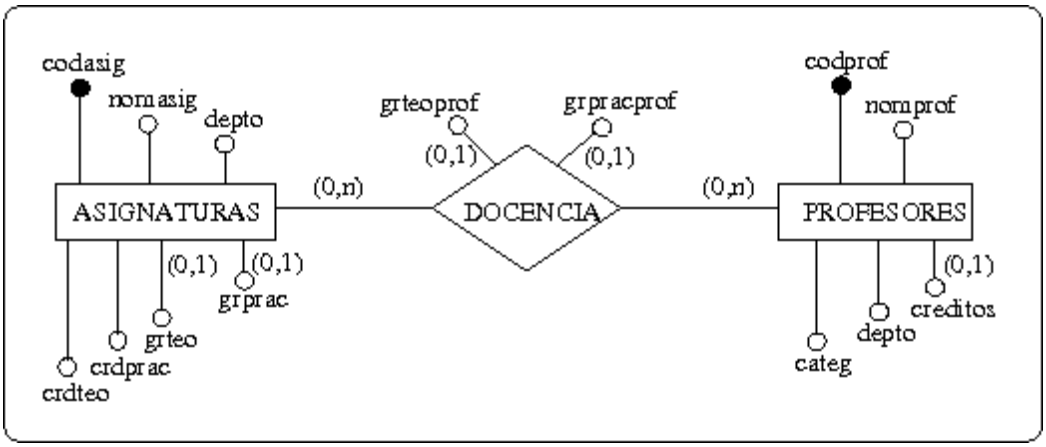

Figura 1. Esquema conceptual.

3. Esquema lógico

Vamos a ver cómo obtener el esquema lógico mediante DBDesigner Fork. Utilizaremos los iconos de la barra de lateral izquierda para crear las entidades (*New Table*). Comenzaremos por la entidad ASIGNATURAS.

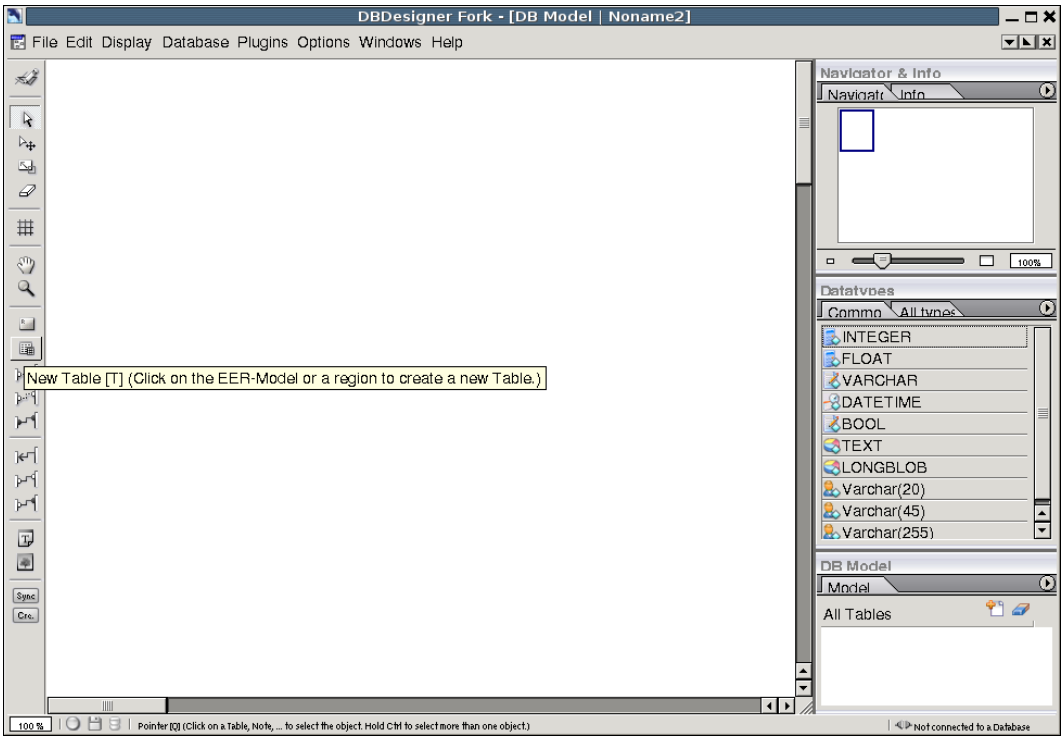

Figura 2. Crear una entidad (New Table).

Cada entidad del esquema conceptual da lugar a una tabla, así que la herramienta crea esa tabla. Pinchamos sobre ella para acceder a la ventana que permite darle nombre y definir las columnas.

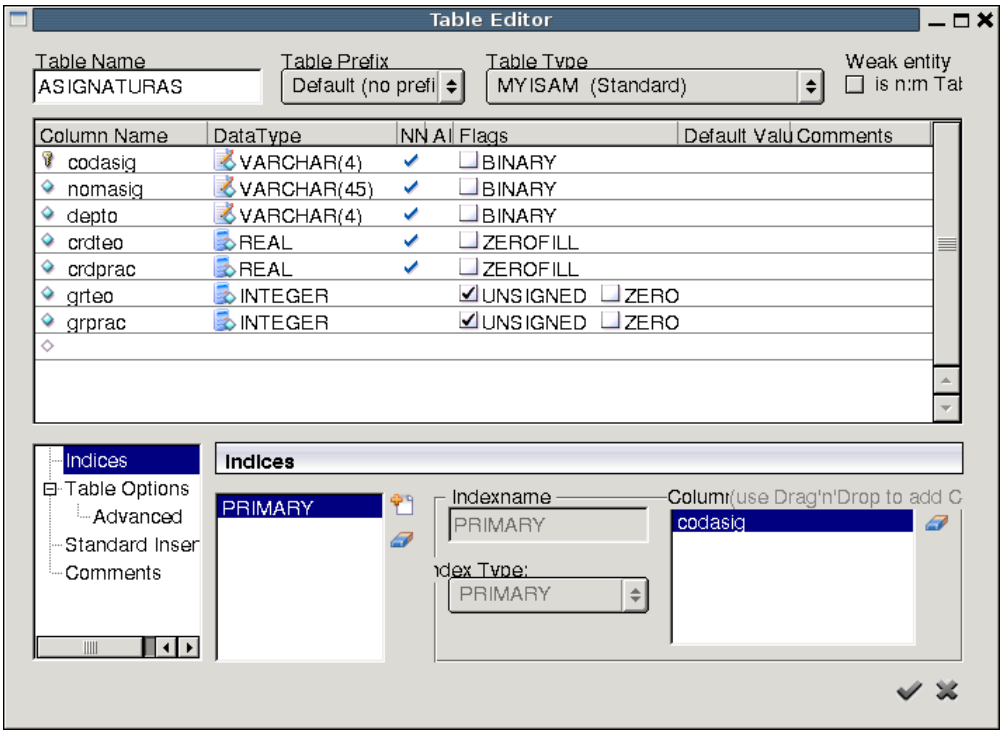

Figura 3. Dar nombre a una tabla y definir sus columnas.

La propiedad NN se debe seleccionar para las columnas que no deban admitir nulos. La propiedad AI se utiliza para hacer columnas autoincrementativas en MySQL. La clave primaria está formada por las columnas marcadas con una llave dorada. La llave se activa y desactiva pinchando sobre el icono que precede al nombre de la columna.

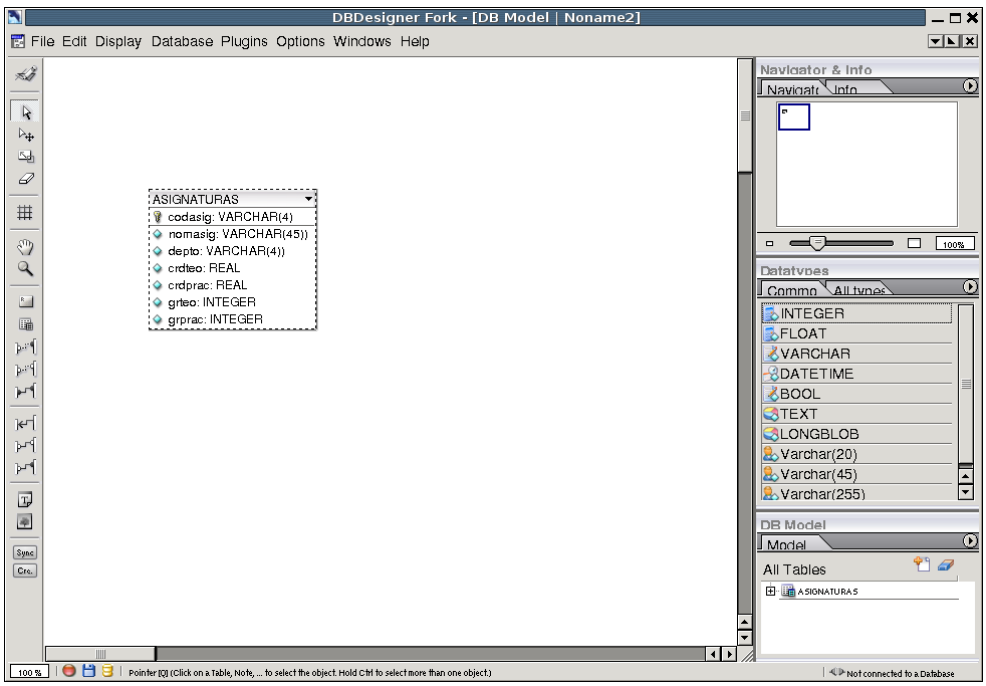

Figura 4. Tabla creada para la entidad ASIGNATURAS.

Tras cerrar la ventana de definición de la tabla ASIGNATURAS deberemos crear la tabla correspondiente a la entidad PROFESORES.

Vamos ahora con las relaciones. En el esquema conceptual de la figura 1 tenemos una relación de muchos a muchos entre las dos entidades: un profesor puede impartir muchas asignaturas y una asignatura puede ser impartida por muchos profesores. En la figura 5 podemos ver el icono que nos permite crear este tipo de relación, a la que se denomina n:m (muchos a muchos).

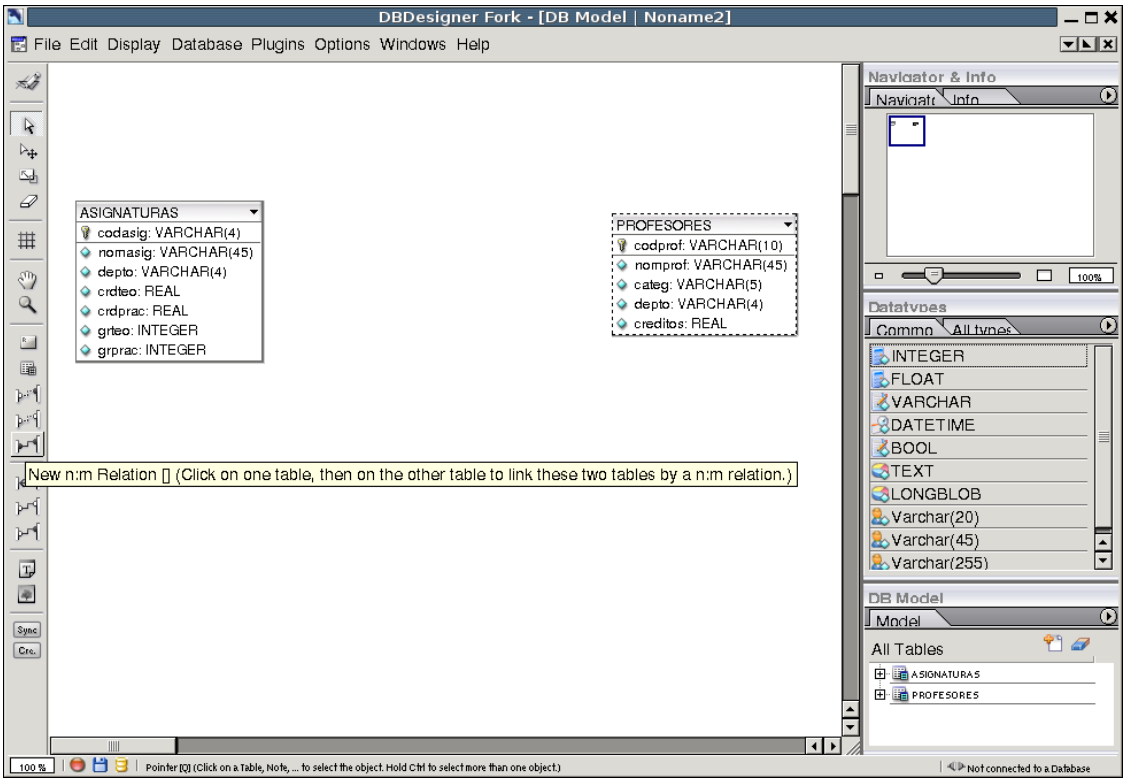

Figura 5. Elemento de la barra lateral para crear relaciones de muchos a muchos.

Una vez seleccionado el tipo de relación, se pincha primero en una entidad y después en la otra. Entonces la herramienta crea una nueva tabla para la relación. La nueva tabla toma como claves ajenas las claves primarias de las tablas participantes en la relación y las señala como clave primaria. Hay que tener en cuenta que la clave primaria de esta nueva tabla no siempre ha de ser la combinación de las claves ajenas, esto dependerá del significado de la relación, por lo que es posible que necesitemos cambiarla.

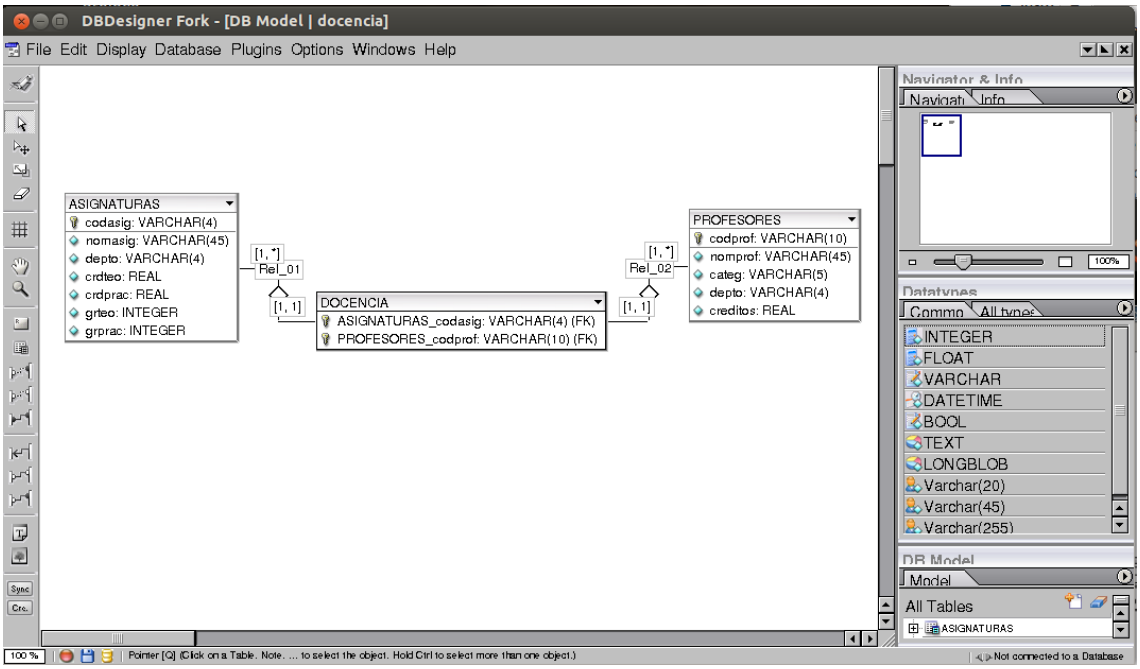

Figura 6. Tabla creada por la herramienta al establecer la relación n:m.

La nueva tabla se puede editar para cambiar el nombre y añadir las columnas correspondientes a los atributos de la relación.

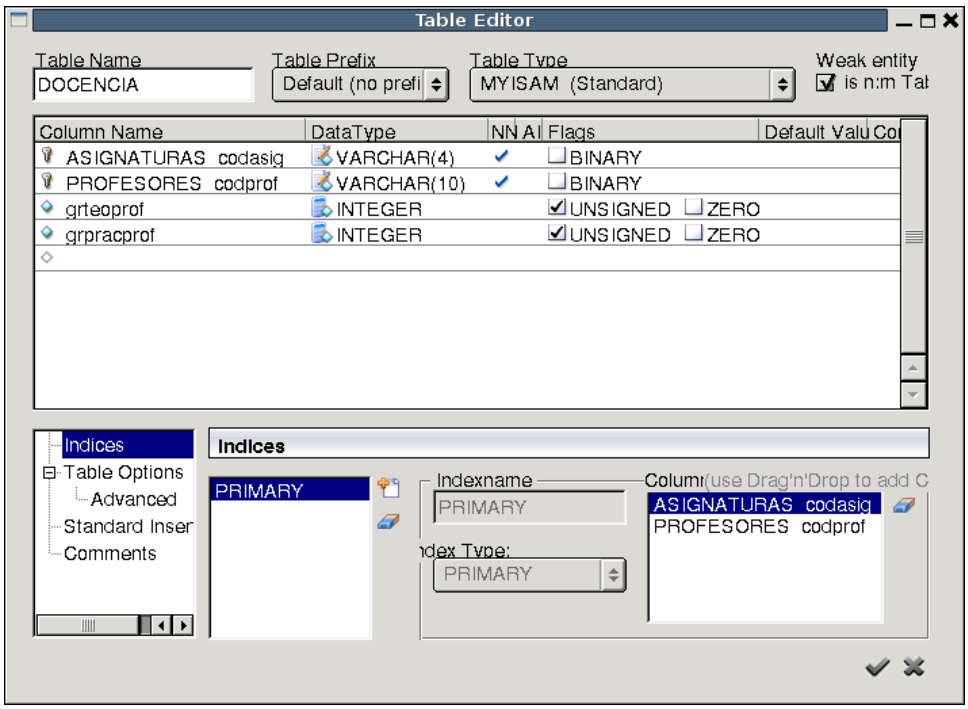

Figura 7. Edición de la tabla correspondiente a la relación n:m.

Fíjate que la relación de muchos a muchos se ha descompuesto en dos relaciones de uno a muchos y que debemos editarlas para establecer las reglas de comportamiento de las claves ajenas.

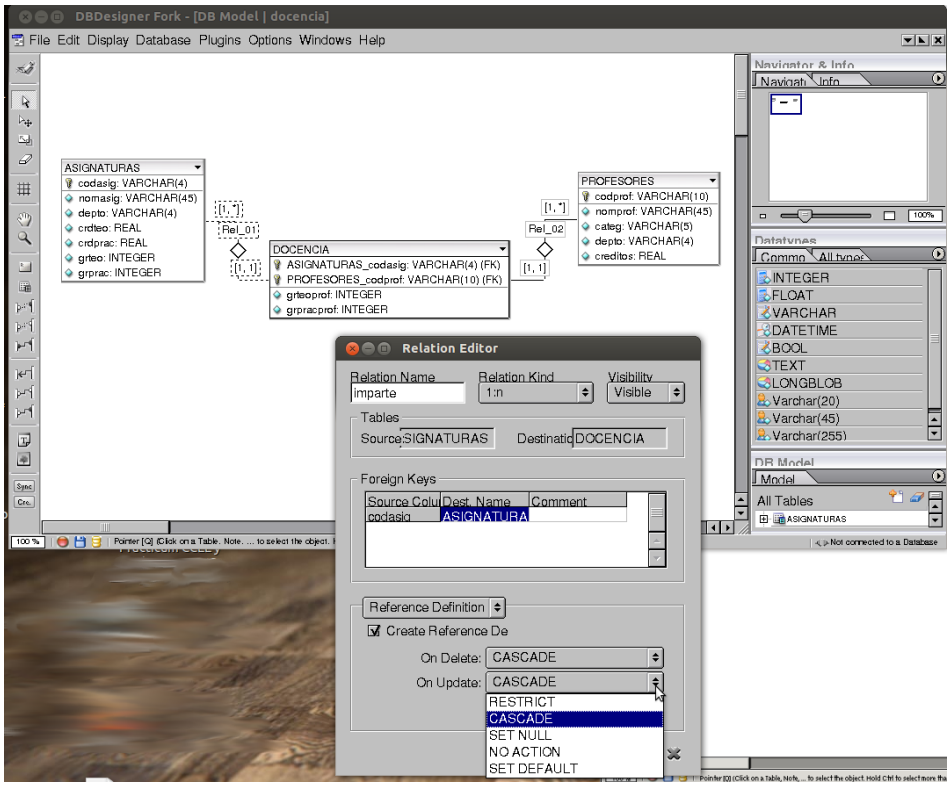

Figura 8. Edición de las relaciones para establecer las reglas de las claves ajenas.

El esquema lógico que hemos obtenido mediante DBDesigner Fork, correspondiente al esquema conceptual de la figura 1 es el siguiente:

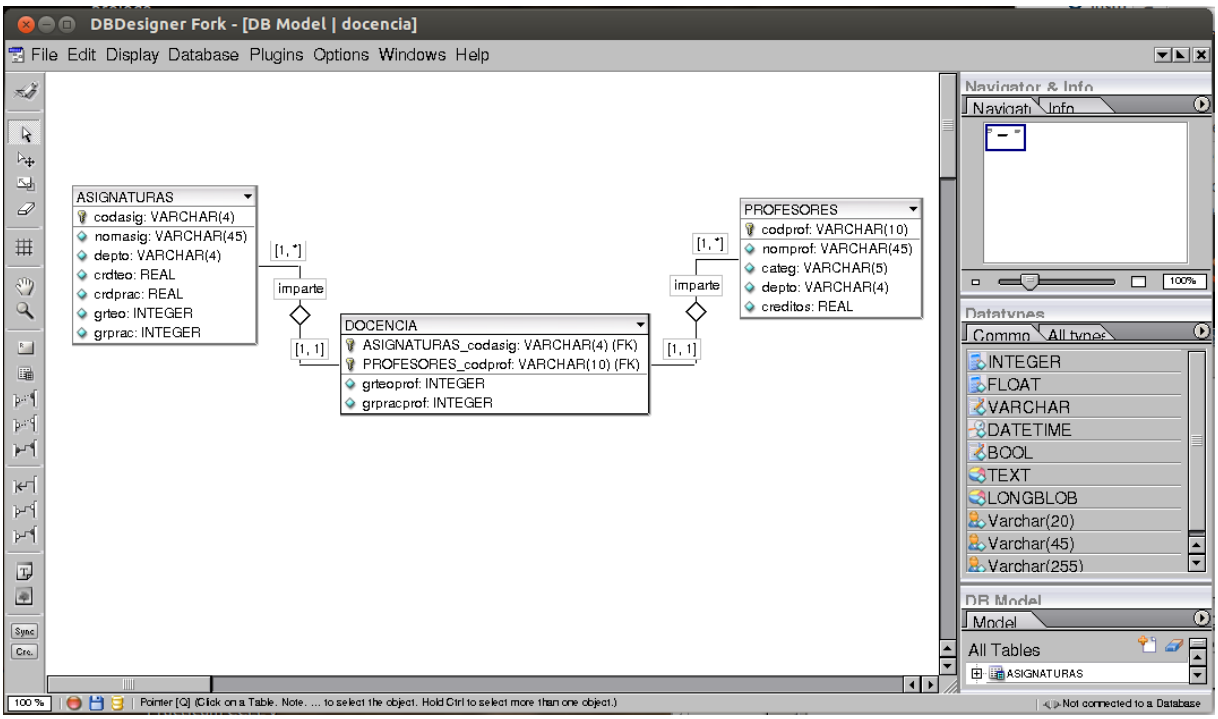

Figura 9. Esquema lógico generado por la herramienta.

Fíjate que las claves ajenas las ha puesto automáticamente la herramienta. Por eso es importante que no las pongamos en los esquemas conceptuales, porque sino estarían

repetidas.

4. Esquema físico

Para obtener el esquema físico correspondiente al esquema lógico de la figura 9 debemos escoger la opción *Export* del menú *File* y seleccionar la opción de crear un script de SQL.

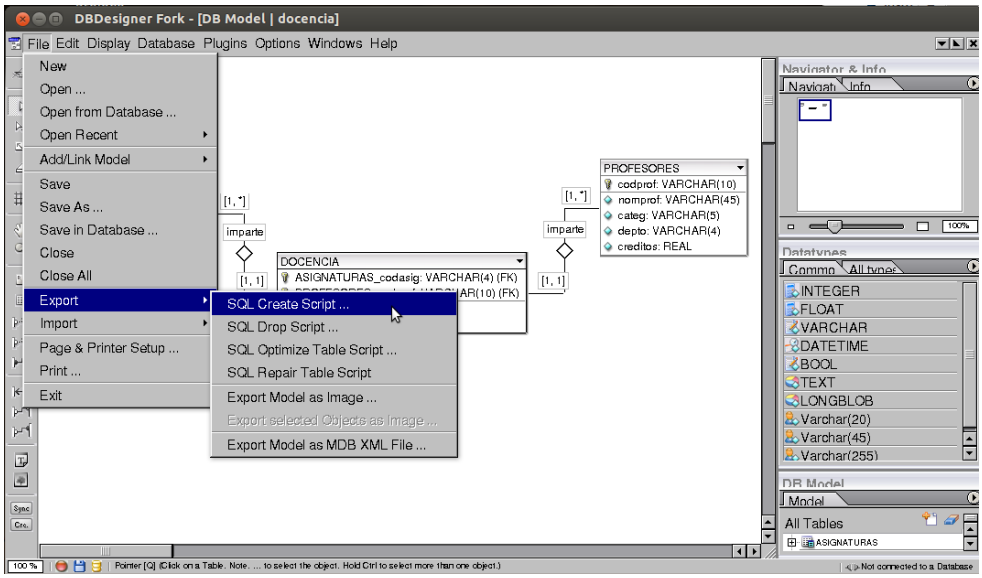

Figura 10. Generación del esquema físico en SQL.

En la ventana que aparece se debe escoger el SGBD para el que se deben generar las sentencias SQL, que en nuestro caso es PostgreSQL.

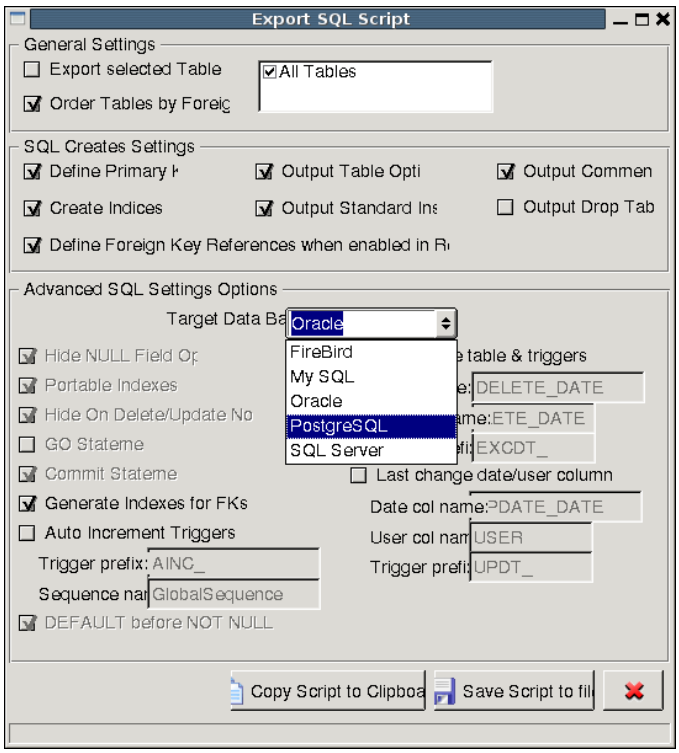

Figura 11. Selección del SGBD en el que se van a crear las tablas.

El esquema físico se puede copiar al portapapeles o guardar en un fichero. El que ha generado

el esquema lógico de la figura 9 es el que se muestra a continuación. Este fichero se puede modificar (por ejemplo, para añadir restricciones tipo CHECK) y después ejecutar desde *psql*.

```
CREATE TABLE PROFESORES (
  codprof VARCHAR(10)   NOT NULL ,
  nomprof VARCHAR(45)   NOT NULL ,
  categ VARCHAR(5)    NOT NULL ,
  depto VARCHAR(4)    NOT NULL ,
  creditos REAL       ,
PRIMARY KEY(codprof));
CREATE TABLE ASIGNATURAS (
  codasig VARCHAR(4)    NOT NULL ,
  nomasig VARCHAR(45)    NOT NULL ,
  depto VARCHAR(4)    NOT NULL ,
  crdteo REAL    NOT NULL ,
  crdprac REAL    NOT NULL ,
  grteo INTEGER     ,
  grprac INTEGER       ,
PRIMARY KEY(codasig));
CREATE TABLE DOCENCIA (
  ASIGNATURAS_codasig VARCHAR(4)    NOT NULL ,
  PROFESORES_codprof VARCHAR(10)    NOT NULL ,
  grteoprof INTEGER     ,
  grpracprof INTEGER       ,
PRIMARY KEY(ASIGNATURAS_codasig, PROFESORES_codprof),
  FOREIGN KEY(ASIGNATURAS_codasig)
    REFERENCES ASIGNATURAS(codasig)
      ON DELETE CASCADE
      ON UPDATE CASCADE,
  FOREIGN KEY(PROFESORES_codprof)
    REFERENCES PROFESORES(codprof)
      ON DELETE CASCADE
      ON UPDATE CASCADE);
CREATE INDEX IFK_Rel_01 ON DOCENCIA (ASIGNATURAS_codasig);
CREATE INDEX IFK_Rel_02 ON DOCENCIA (PROFESORES_codprof);
```

5. Tipos de relaciones

En la figura 12 puedes ver los distintos tipos de relaciones que podemos dibujar.

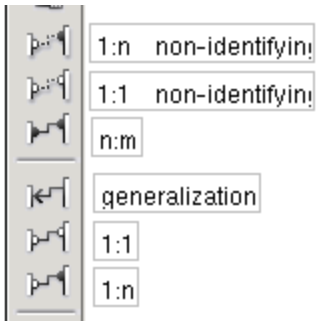

Figura 12. Tipos de relaciones.

Las relaciones de línea discontinua (*non-identifying*) son para dibujarlas entre entidades fuertes, que son aquellas cuyas ocurrencias se identifican mediante atributos de la propia entidad. La herramienta las transforma en el esquema lógico incluyendo una clave ajena en la tabla de la entidad que hace el papel de hijo, a la tabla que hace el papel de padre. Una vez seleccionada la relación a dibujar, debes pinchar primero en la tabla padre y después en la tabla hijo.

Las relaciones con línea continua 1:1 o 1:n se utilizan cuando la entidad hijo es una entidad débil. En este caso, la herramienta también las transforma incluyendo una clave ajena de la tabla hijo a la tabla padre, pero haciendo además que la clave ajena forme parte de la clave primaria de la tabla hijo (al ser débil, requiere atributos del padre para identificarse).

Las relaciones n:m las hemos visto en el ejemplo que hemos desarrollado en los apartados anteriores. Cuando establecemos una de estas relaciones, la herramienta crea una nueva tabla para la entidad y toma como clave primaria la combinación de las dos claves ajenas que tiene a las entidades participantes. Si queremos incluir más columnas en la clave primaria, debemos editar esta tabla y seleccionar la llave dorada en ellas. Por otro lado, si la clave primaria de la tabla correspondiente a la relación no debe contener alguna de las claves ajenas que se han incluido, debemos trabajar un poco más. Los pasos a seguir son los siguientes:

- Editar la relación con la entidad cuya clave ajena no debe formar parte de la clave primaria y cambiar el tipo de relación: es 1:n y hay que cambiarla a 1:n *non-identifying*.
- Editar la tabla y en el índice llamado *PRIMARY* borrar la clave ajena que no debe ser parte de la clave primaria (se selecciona del cuadro de la derecha y se pincha en la goma de borrar).

6. Jerarquías de generalización

En la figura 12 también se muestra el icono que permite dibujar jerarquías de generalización (*generalization*). A continuación puedes ver cómo se dibuja una jerarquía mediante el ejemplo de la figura 13. Esta jerarquía hace una clasificación de cuentas bancarias, de las que se conoce el número de cuenta y el saldo. De las que son cuentas corrientes se conoce también el límite de descubierto; de las que son cuentas de ahorro se conoce el interés anual.

La jerarquía es (1,1) porque una cuenta sólo puede pertenecer a uno de los subconjuntos y ha de hacerlo obligatoriamente.

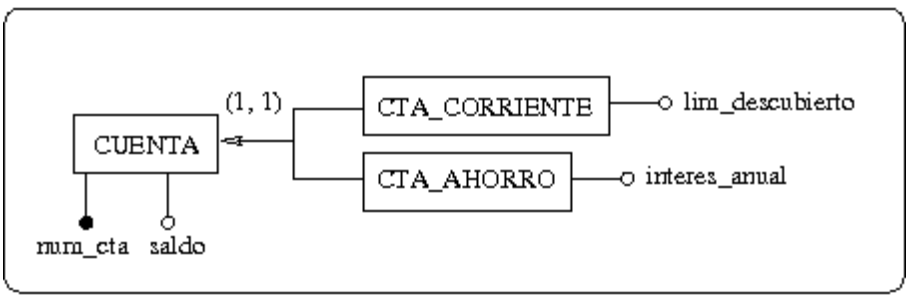

Figura 13. Jerarquía de generalización.

Para dibujar la jerarquía con la herramienta debemos dibujar, en primer lugar, la entidad genérica y las entidades subconjunto, como se muestra en la figura 14. Fíjate que los subconjuntos no tienen clave primaria: en las jerarquías se hereda el identificador de la entidad genérica.

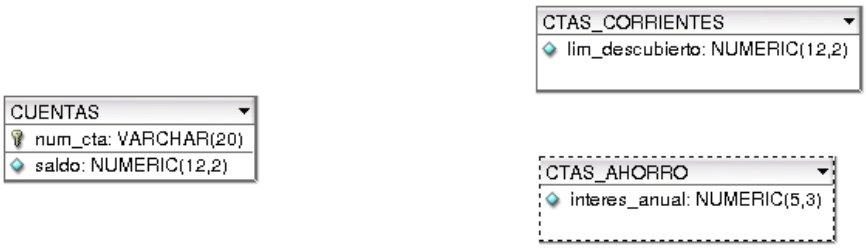

Figura 14. Entidad y subentidades de una jerarquía de generalización.

A continuación, se establece la jerarquía mediante el icono que contiene una flecha (ver figura 12). Puesto que hay dos subconjuntos, debemos dibujar las dos flechas. Para cada flecha, se pincha primero en la entidad genérica (hace el papel de padre) y después en una entidad subconjunto (hijo).

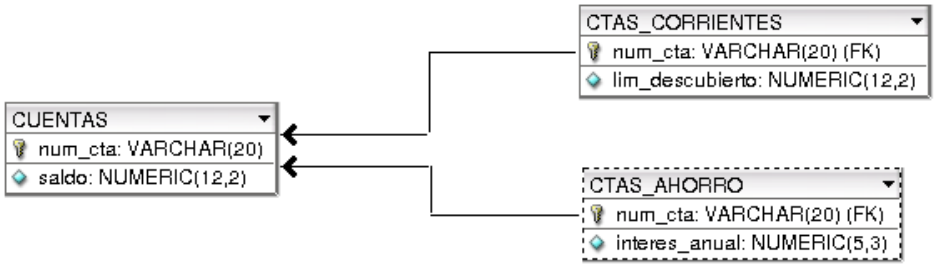

Figura 15. Establecer la jerarquía de generalización.

Por último, establecemos las reglas de las claves ajenas pinchando sobre cada una de las flechas. En las jerarquías se debe escoger la regla de propagar el borrado y la modificación.

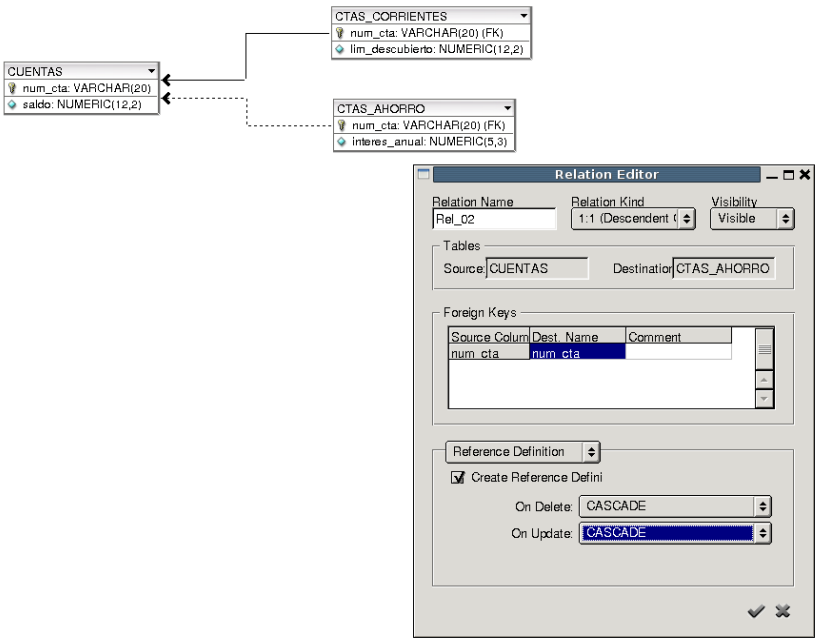

Figura 16. Claves ajenas en la jerarquía de generalización.

7. Opciones de configuración

Si la notación que te aparece en la herramienta no coincide con la que has visto en este manual, puedes cambiarla desde el menú *Display*. Escoge la opción EER [1,n] en *Notation*. Desde el mismo menú puedes determinar el detalle con el que se han de mostrar las columnas en las tablas, *Table Columns*. Por ejemplo, puedes escoger no mostrar los detalles de los tipos de datos para tener un diagrama más compacto (*Physical Schema Level*).

Hay otras opciones de configuración que puedes cambiar desde el menú *Options*. Estas opciones las puedes cambiar en el esquema que estás editando, *Model Options,* o cambiarlas en DBDesigner Fork para todos los modelos con los que trabajes en adelante, *DBDesigner Options*. Una de las opciones que quizá quieras cambiar es la que da nombre a las claves ajenas, *Default Model Options*. Fíjate que, por defecto, se añade el nombre de la tabla al que hace referencia la clave ajena. Además, puedes fijar un prefijo o un sufijo para que se añada siempre a las claves ajenas.