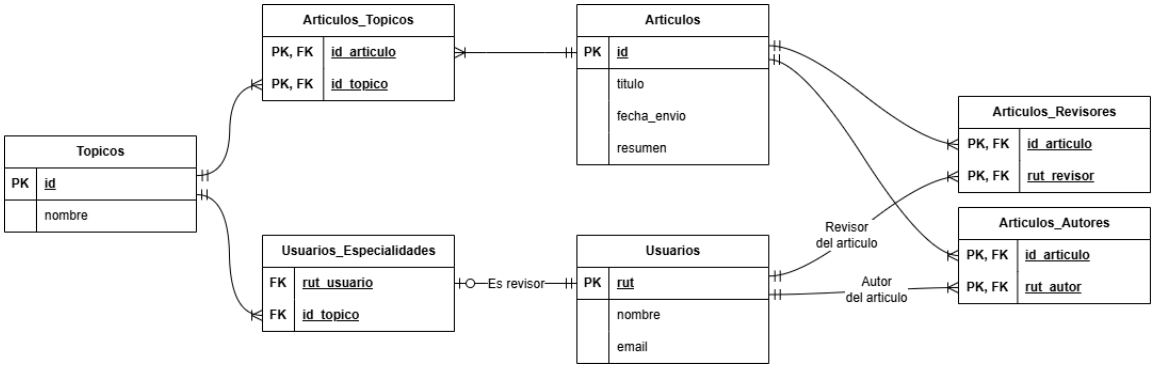
GESCON

Integrante 1: Alejandro Cáceres (202373520-5)
Integrante 2: Miguel Salamanca (202373564-7)

1. Correcciones:

Primero debemos corregir e implementar lo que falta de los Anexos de la Tarea 1, a partir de las vistas dadas en los mismos. Modelo conceptual inicial:



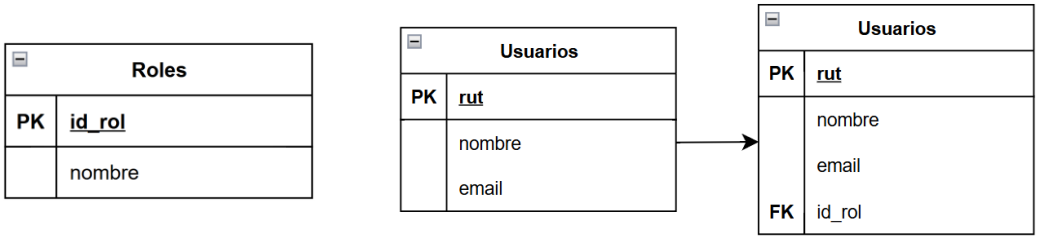
1.0. Roles (EXTRA):

Como en muchos casos un Autor puede hacer ciertas cosas, un Revisor más cosas y el jefe de comité otras, decidimos implementar una Tabla 'Roles'.

La Tabla 'Roles' tiene un '*id_rol*' para identificar que rol es a partir de un id y su respectivo nombre en '*nombre*'.

A partir de esto, se le implemento un FK '*id_rol*' a los Usuarios, el cual almacena su rol. Por defecto (si no se especifica que rol tiene un Usuario al crearlo) se le asigna el rol de Autor.

Aquí la Tabla 'Roles' y los cambios:



- [PK] id_rol -> Roles(id_rol): ID del rol del Usuario.
- Tabla de la Tabla 'Roles' (xD):

Roles	id_rol	nombre	¿Qué hace?
Autor	1	'autor'	Puede enviar, modificar y eliminar Artículos. (puede modificar y eliminar solo los artículos donde sea autor).
Revisor	2	'revisor'	Puede revisar los artículos que se le hayan asignado (ya sea automáticamente o por el jefe de comité).
Jefe de Comité	3	'jefe comité'	Gestiona y asigna a los revisores.

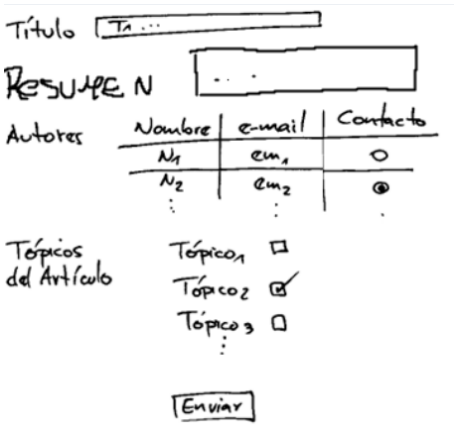
NOTAR: Lo que hace cada uno esta resumido.

Hay que considerar que al subir de rol, puedes seguir haciendo lo que hace un rol de ID inferior.

Es decir, un Revisor puede hacer lo que hace un Autor y un jefe de comité puede hacer lo que hace un Revisor y un Autor. Un Autor no puede hacer lo que un Revisor o un jefe de comité, ya que su rol tiene una id inferior. Un Revisor no puede hacer lo que un jefe de comité puede hacer.

1.1. Enviar Articulo (AUTOR):

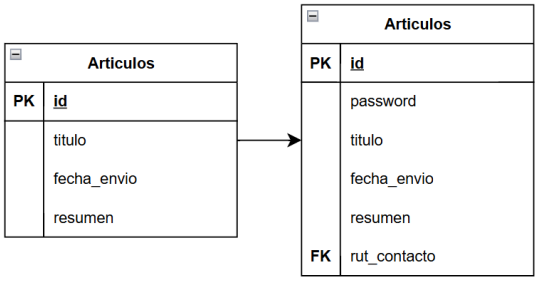
Vista del Autor:



A partir de esta, se modificó la Tabla '*Articulos*', agregando una FK '*rut_contacto*'. Con esto, podemos decir quién es el autor de contacto, el cual recibe el (supuesto) correo enviado por el sistema, con la contraseña '*password*' que tiene el Articulo.

Como supuesto, decidimos que el '*userid*' del Articulo que se le envía al autor de contacto sea el '*id*' del Articulo, aunque con la implementación que hicimos, solo será necesario la contraseña.

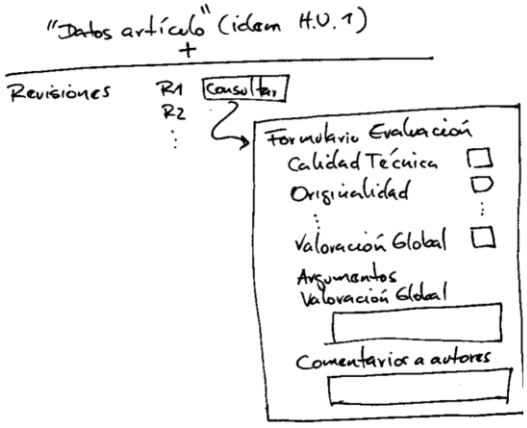
Aquí los cambios:



- [FK] rut_contacto -> Usuarios(rut): Es el autor de contacto.

1.2. Acceso al Artículo (AUTOR):

Vista del Autor:

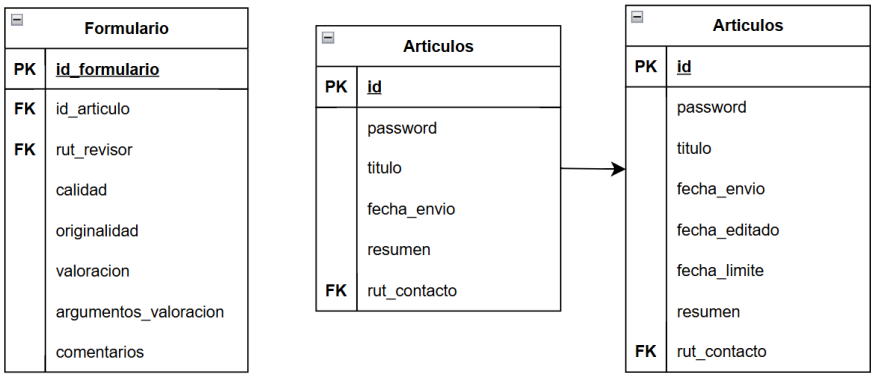


Aquí se implementa la Tabla 'Formulario'. Qué iba en el formulario no se especifica, así que utilizamos como base la imagen del anexo.

Como también se especifica que hay una fecha límite para hacer cambios a un Articulos, se agregó un atributo '*fecha_limite*' a la Tabla 'Articulos'.

También, como extra se agregó un atributo '*fecha_editado*', el cual indica la fecha y hora en que se hizo alguna modificación al Artículo por parte del Autor. Si no se le hizo ninguna modificación, este se mantiene como nulo.

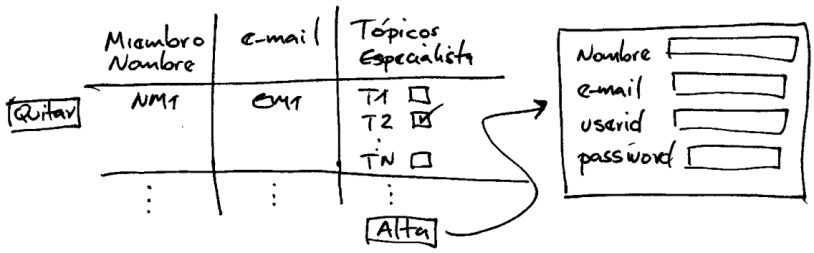
Aquí la Tabla 'Formulario' nueva y los cambios:



- [FK] id_articulo -> Articulo(id): ID del Articulo al que le corresponde el formulario.
- [FK] rut_revisor -> Usuarios(rut): Rut del Revisor (que tiene asignado el Articulo), el cual realizo el formulario.

1.3. Gestión de Revisores (JEFE DE COMITÉ):

Vista del jefe de comité:

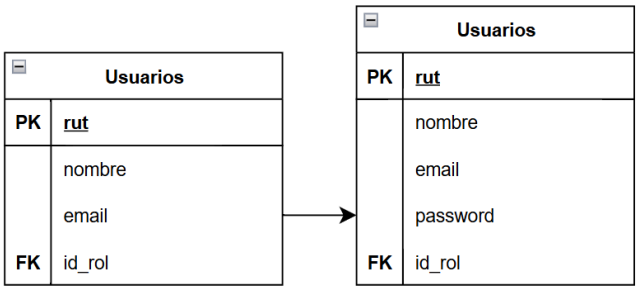


Aquí el jefe de comité puede añadir, modificar, eliminar y consultar los Revisores. Directamente de aquí no se hicieron cambios al modelo conceptual, pero si se agregaron cosas.

Como un Revisor debe tener un '*userid*' y '*password*', decidimos darle uno a los Usuarios en general. Entonces, la forma de verificarse es con un sistema de login que ya se pide para esta Tarea 2.

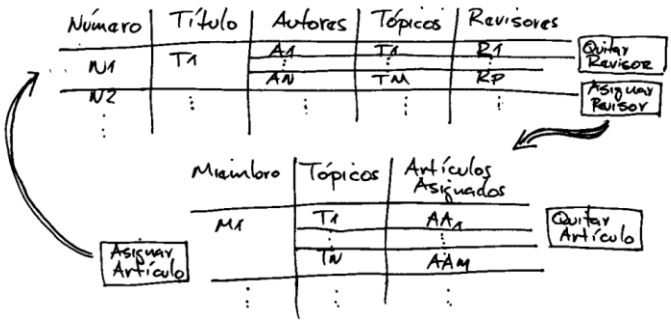
Como supuesto, el '*userid*' para los Usuarios será simplemente el '*rut*' y agregamos el atributo '*password*' que almacena la contraseña del Usuario.

Aquí los cambios:



1.4. Asignación de Artículos a Revisores (JEFE DE COMITÉ):

Vista del jefe de comité:

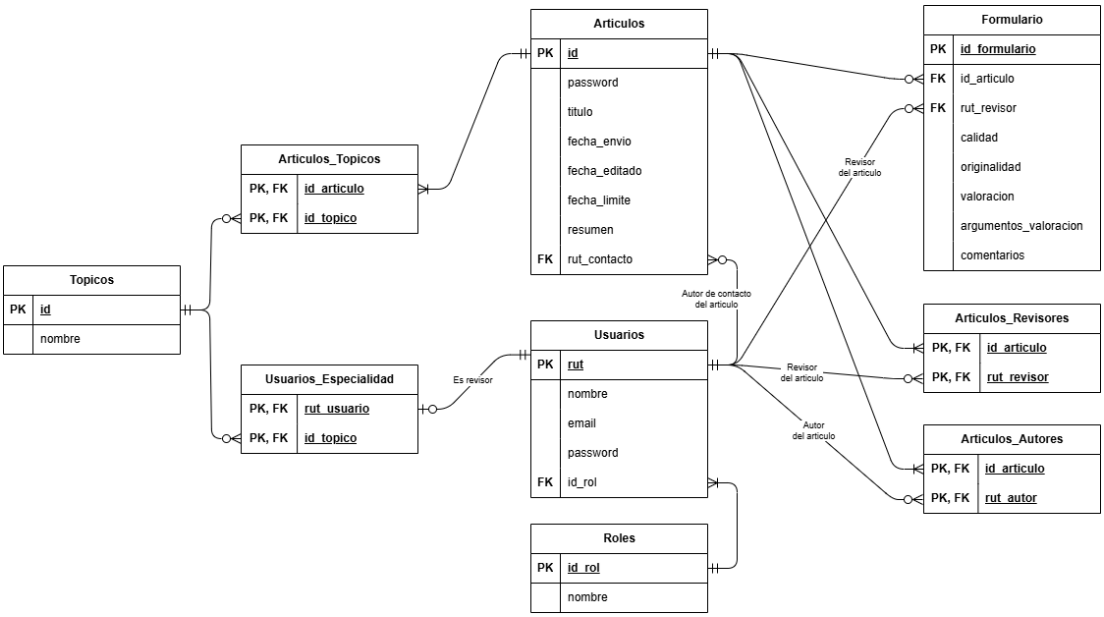


Aquí el jefe de comité asigna o elimina Revisores a los Artículos. El sistema ya asigna Revisores por defecto, pero aun así el jefe de comité puede hacer cambios manualmente en caso de ser necesario.

De aquí no se hacen cambios al modelo conceptual.

1.5. Modelo Conceptual Corregido:

Con todos los cambios, obtuvimos este modelo conceptual:



Este será el que normalizaremos a 3FN.

2. Normalización:

Para convertir el modelo a 3FN, debemos primero convertirlo en 1FN, luego 2FN y finalmente 3FN.

NOTAR: En verdad no se cambió nada, ya que el modelo conceptual que creamos ya era 3FN, pero aun así aquí están los pasos que seguimos:

2.1. Primera Forma Normal (1FN):

Para convertirlo en 1FN, debemos **eliminar grupos repetitivos**. Cada tabla de modelo conceptual tiene un propósito y no hay grupos repetidos.

Por lo tanto, ya es 1FN. Podemos pasar al siguiente.

2.2. Segunda Forma Normal (2FN):

Para convertirlo en 2FN, hay que **eliminar dependencias parciales**. Esto quiere decir que los atributos no clave deben depender de la clave primaria.

Por ejemplo, en nuestro modelo conceptual, a partir del 'id_articulo' -> Articulos(id) podemos obtener la información del articulo y no esta repetida en otras tablas. Lo mismo con '*rut*', '*id_topico*', etc.

Como no hay dependencias parciales en nuestro modelo conceptual, podemos decir que ya es 2FN. Podemos continuar.

2.3. Tercera Forma Normal (3FN):

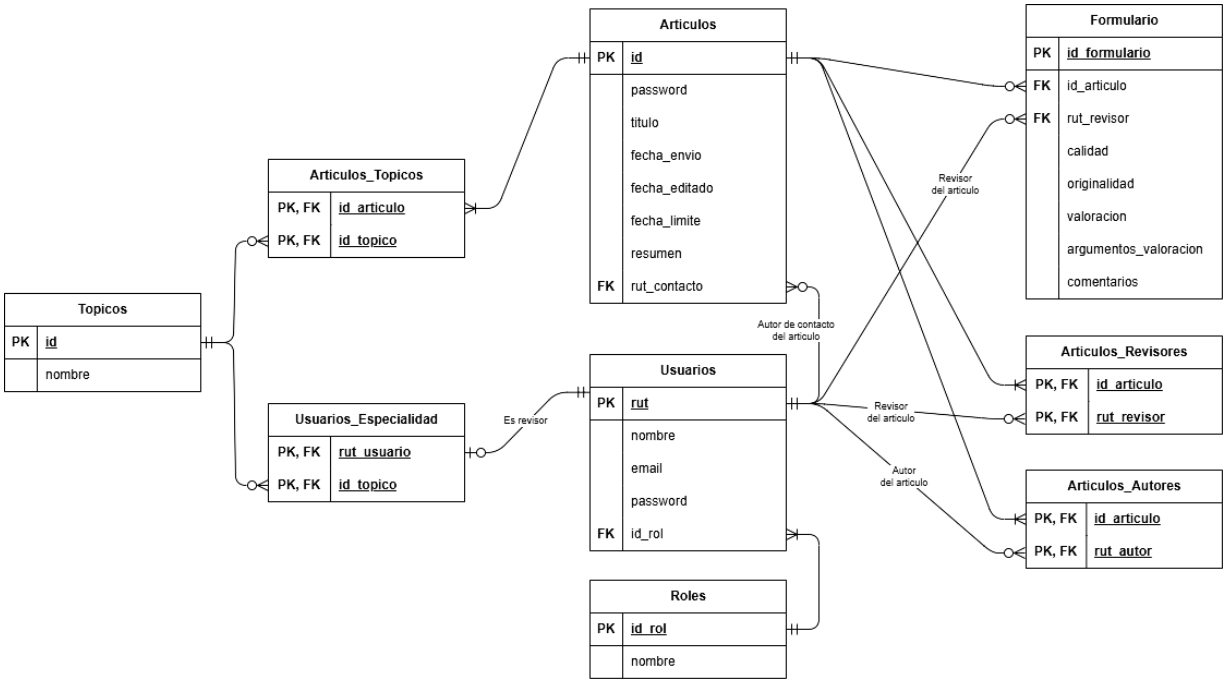
Finalmente, para que esté en 3FN, hay que **eliminar dependencias transitivas**. Un atributo no clave no debería depender de otro atributo no clave.

Como no hay dependencias transitivas en nuestro modelo conceptual, podemos decir que este ya está en 3FN.

2.4. Modelo Conceptual Normalizado:

Como se vio, el modelo conceptual no tuvo cambios.

Aun así, aquí está el modelo conceptual que usaremos para nuestra base de datos:



(Es igual!! xD)