Diseño de bases de datos

Ficheros y Bases de Datos

31 de julio de 2002

1. Introducción

Este documento contiene preguntas sobre diseño de bases de datos que han ido surgiendo al ver los errores cometidos con más frecuencia en las respuestas de los exámenes. Ya que en esta primera versión el documento contendrá pocas preguntas, se agradecerá enormemente toda colaboración que ayude a completarlo con más preguntas interesantes.

2. Preguntas y respuestas

2.1. *Antes que nada: hablemos "con propiedad"*

Cada tabla tiene siempre una clave primaria. Además PUEDE tener claves alternativas. En caso de que las tenga, siempre hay que indicarlas. Que una clave sea alternativa es casi tan importante como el que sea clave primaria ya que sirve para identificar las filas de la tabla y por lo tanto se debe respetar su unicidad.

Una clave primaria (o una clave alternativa) puede estar formada por un atributo o por varios atributos. Sea como sea, nos referiremos a ella siempre EN SINGULAR. Diremos, por ejemplo, "la clave primaria está formada por las columnas codfac y línea"; "la clave alternativa, formada por nombre y fecha, no acepta nulos" y esto significa que ninguno de sus componentes acepta nulos.

Del mismo modo, y ya que las claves ajenas hacen SIEMPRE referencia a claves primarias COMPLETAS, hablamos de cada una de ellas en singular, aunque sean compuestas. Por ejemplo "la clave ajena está formada por los atributos código y número, no ACEPTA nulos y la regla de borrado ES propagar", lo que significa que ninguno de los dos atributos acepta nulos y que si se borra una fila de la tabla a la que se está haciendo referencia, se propaga el borrado a todas las filas que le hacen referencia mediante esa clave ajena: las que en código y número tienen la misma combinación de valores que hay en la clave primaria de la fila que se borra.

Es un error importante de concepto el hablar de una clave primaria/ajena compuesta como "las claves primarias/ajenas" ya que esto conduce a errores. Por ejemplo, un error típico es dar distintas reglas para cada uno de los atributos que forma parte de una misma clave ajena. Si se habla "con propiedad" y a una clave ajena se le llama así, clave ajena, y no "claves ajenas", es menos problable que se cometa el error de definir varias reglas ¿verdad?

2.2. *¿Cómo se establecen las reglas de las claves ajenas?*

En gran medida, las reglas de las claves ajenas son establecidas por los propietarios de los datos.

Por ejemplo, fijémonos en el caso de las transparencias de clase que aparece cuando reflejamos una relación, en este caso es 1:1, en el esquema lógico de una base de datos:

```
EMPLEADO(codemp, nombre, matrícula, fecha_ini)
VEHíCULO(matrícula, modelo)
```

Tenemos empleados y vehículos. Cada empleado conduce un solo vehículo y cada vehículo es conducido por un solo empleado. En este caso hemos puesto la clave ajena que representa la relación, en la tabla que contiene la información de los empleados, de modo que cada empleado hace referencia al vehículo que conduce. A continuación nos planteamos las preguntas que hay que responder para establecer las reglas de las claves ajenas:

- *¿Acepta nulos la clave ajena?* Que al usuario se pregunta como: *¿puede haber algún empleado que no conduzca ningún vehículo?* Esto, en realidad, no debería preguntarse al usuario ahora, sino que se debería haber preguntado en la recolección de los requisitos y debería aparecer en el esquema conceptual ... ¿dónde? en la cardinalidad mínima con la que participa la entidad empleado en esta relación: si es 0 significa que sí puede haber empleados sin vehículo, por lo que la clave ajena debe aceptar nulos; si es 1 significa que todo empleado debe conducir algún vehículo, por lo que en este caso no debe aceptar nulos. Por lo tanto la respuesta a esta pregunta SIEMPRE aparecerá reflejada en el esquema conceptual como la cardinalidad mínima con la que participa la entidad QUE ACOGE la clave ajena[1].
- *¿Cuál es la regla de borrado?* Que al usuario se pregunta como: *¿qué hay que hacer cuando se intenta borrar un vehículo QUE ES CONDUCIDO por algún empleado?* Las posibles respuestas son:
  - Propagar: se borra el vehículo (se elimina su fila de la tabla) y se borra el empleado (también se borra la fila que hace referencia a ese vehículo).

[1] Mucho ojo: esto NO es así cuando una relación se representa mediante una tabla aparte. Dedicaremos una pregunta a este caso concreto.

- Restringir: no se puede borrar un vehículo que es conducido por un empleado. En este caso lo normal es pedir que el propietario de los datos especifique un procedimiento a seguir: dar sólo un aviso al usuario; dar un aviso y mostar los datos del empleado dando la posibilidad de cambiarle el vehículo; etc.
- Anular: se borra el vehículo y en la fila del empleado que lo conduce, la clave ajena, que contenía la matrícula del vehículo, se pone a nulo. Esta opción SÓLO es posible cuando la clave ajena acepta nulos.

Pensemos que el usuario decide borrar un vehículo y que ese vehículo es conducido por un empleado. Veamos cómo afecta a la base de datos el borrado de un vehículo con cada una de las reglas:

- Propagar: se borran dos filas, la del vehículo y la del empleado.
- Restringir: no se borra ninguna fila.
- Anular: se borra una fila, la del vehículo, y el valor de una columna de una fila de empleado se pone a nulo.

Evidentemente, cada una de las respuestas tiene unos efectos distintos y, en este caso, será el propietario de la información quien decida qué es lo que quiere hacer.

- *¿Cuál es la regla de modificación?* Que al usuario se pregunta como: *¿qué hay que hacer cuando se intenta modificar la matrícula de un vehículo QUE ES CONDUCIDO por algún empleado?* Las posibles respuestas son:

  - Propagar: se modifica la matrícula del vehículo y en la fila del empleado que lo conduce se cambia en valor de la clave ajena (la matrícula) para que le siga haciendo referencia. Al fin y al cabo, el vehículo es el mismo, sólo ha cambiado el valor de una de sus propiedades ... quizá porque se tecleó mal al introducirla.
  - Restringir: no se puede modificar la matrícula de un vehículo que es conducido por un empleado. En este caso lo normal es pedir que el propietario de los datos especifique un procedimiento a seguir: dar un aviso al usuario; etc.
  - Anular: se modifica la matrícula del vehículo y en la fila del empleado que lo conduce, la clave ajena, que contenía la matrícula del vehículo, se pone a nulo. De nuevo, esta opción SÓLO es posible cuando la clave ajena acepta nulos.

  También en este caso será el usuario quien dé la respuesta. Pero ¿por qué parece que no le damos importancia a esta pregunta? Pues porque en las bases de datos, cuando se implementan físicamente, las claves primarias no suelen ser columnas que representan propiedades de las entidades, sino columnas sin significado que se añaden a propósito para las que se van generando valores únicos cuya función es solamente la de identificar las filas. Ya que este tipo de claves primarias se generan de modo automático, nunca cambian de valor (ni siquiera el usuario necesita saber que existen) por lo que este tipo de operación (modificación) no se realiza.

Hemos empezado esta respuesta diciendo *"En gran medida, las reglas de las claves ajenas son establecidas por los propietarios de los datos."* Esto significa que no siempre será así: hay ocasiones en las que somos nosotros como diseñadores quienes podemos y debemos decidir las reglas de determinadas claves ajenas. Vemos aquí cuáles son esas ocasiones:

- *Jerarquías.* Cuando escogemos representar una jerarquía del modo más general (el que funciona para todo tipo de jerarquía), introducimos una tabla por la entidad genérica y una tabla por cada subentidad. Las tablas correspondientes a las subentidades tienen, cada una, una clave ajena a la tabla correspondiente a la entidad. Esta clave ajena será también una clave candidata (podrá ser la clave primaria o podrá serlo cualquier otro identificador alternativo). Pues bien, esta clave ajena que tiene cada tabla "subentidad" NO acepta nulos y la regla del borrado será SIEMPRE propagar. ¿Por qué es así? Porque para el propietario de la información, la jerarquía de nuestro esquema conceptual es tan solo una clasificación: si quiere borrar una ocurrencia de una entidad ¿se le puede decir que no puede hacerlo (restringir) por el hecho de que esa ocurrencia ha sido clasificada de algún modo? Imaginemos la cara de un usuario del departamento de recursos humanos de un hospital cuando, al intentar dar de baja a uno de los empleados que ha finalizado su contrato, le sale un mensaje en pantalla que dice "no puedes borrar el empleado PORQUE ES un médico". ¡Todos los empleados tendrán alguna clasificación! y eso no es excusa para no poder borrarlos ¿no es cierto?[2].

- *Entidades débiles.* Una entidad débil es aquella que necesita del identificador de otra para formar su propio identificador. Por ejemplo, pensemos en el esquema conceptual correspondiente a la base de datos de prácticas e imaginemos que hemos dibujado como entidad las líneas de factura. El identificador de esta entidad está compuesto por el identificador de la entidad facturas y el número de la línea. La entidad líneas de factura es una entidad débil: no tiene sentido que exista una línea sin factura, es decir, la línea depende de la factura para existir. Cuando obtengamos las tablas correspondientes a estas dos entidades tendremos dos tablas, facturas y líneas de factura. En líneas de factura la clave primaria contiene una clave ajena a la tabla de facturas (codfac). Por formar parte de la clave primaria, la clave ajena no aceptará nulos, algo que siempre ocurrirá con las entidades débiles. La regla de borrado será SIEMPRE propagar: si se borra una factura, se propagará el borrado a sus líneas. No tiene sentido decirle al usuario que no puede borrar una factura porque tiene líneas ... ¿cuándo se ha visto una factura sin líneas donde se reflejen los cargos que contiene?

- *Atributos con múltiples valores.* Cuando una entidad tiene un atributo que puede tener varios valores (cuando la cardinalidad máxima del atributo es n), tras la normalización, se tiene una tabla que contendrá los distintos valores del atributo para cada ocurrencia de la entidad. Por ejemplo, podemos tener una entidad empleado con un atributo con múltiples valores en donde se indiquen los títulos académicos que tiene cada empleado.

[2]Otra cosa es que el médico no se pueda borrar porque tenga que participar en varias operaciones que se han programado, pero entonces entrará en juego la regla de otra clave ajena y será ella la que excuse la restricción de borrado.

Este atributo dará lugar a una tabla que tendrá una clave ajena a la tabla de empleados; esta clave ajena formará parte de la clave primaria junto con el nombre del título. Esta clave ajena NO apceptará nulos y la regla de borrado será SIEMPRE propagar: en realidad esta tabla ha aparecido porque en el modelo relacional es así como se representan los atributos con múltiples valores, mediante una nueva tabla. Para el usuario, el empleado es una entidad, pero en la base de datos la información se ha "repartido" en varias tablas. Lo que no se puede hacer es decirle al usuario que no puede borrar un empleado porque tiene títulos (esto sería restringir). Lo que se debe hacer es que cuando un usuario intenta borrar una ocurrencia de una entidad, se debe propagar el borrado de ésta a cualquier otra tabla que almacene propiedades el empleado que hayan surgido a causa de atributos con múltiples valores.

2.3. *Cuando se representa una relación mediante una tabla aparte ¿cómo se establecen las reglas de las claves ajenas de esta tabla?*

Como se ha visto, hay tres formas de representar las relaciones binarias en una base de datos relacional:

- Con las relaciones 1:1 podemos integrar en una sola tabla los datos de las dos entidades que participan en la relación. Sabemos que una ocurrencia de una entidad se relaciona con una ocurrencia de la otra entidad porque "comparten" la misma fila. En este caso, dos entidades y su relación 1:1 dan lugar a una sola tabla.
- Otro modo de representar relaciones es mediante una clave ajena en la tabla que corresponde a una de las entidades que participan en la relación. Si la relación es 1:1, la clave ajena puede estar tanto en una tabla como en la otra (pero sólo en una de ellas). Si la relación es 1:n, la clave ajena SÓLO se puede situar en la tabla que corresponde a la entidad de la parte del 1 (las reglas de esta clave ajena se han explicado en la pregunta anterior). En cualquier caso, las dos entidades y la relación dan lugar a dos tablas, una de ellas con una clave ajena a la otra.
- El tercer modo de representar las relaciones, que se puede utilizar con cualquier tipo de relación (1:1, 1:n, n:n), consiste en introducir una nueva tabla: habrá una tabla por cada entidad participante en la relación y una tercera tabla que representa a la relación en sí. Esta tabla tiene dos claves ajenas, una a cada una de las tablas de las entidades.

Es este tercer caso el que nos ocupa en esta pregunta: ¿cómo se establecen las reglas de las claves ajenas? Independientemente de cuál sea la clave primaria de esta nueva tabla, NINGUNA de las dos claves ajenas acepta nulos: cada fila de esta tabla representa una ocurrencia de una relación entre dos entidades ¿qué dos entidades? aquellas a las que referencian ambas claves ajenas. Si una ocurrencia concreta de una entidad no se relaciona con ninguna ocurrencia de la otra entidad ¿qué tenemos? ...¡nada! no hay ninguna fila en esta tabla que haga referencia a esa ocurrencia que no participa en la relación. Si imaginamos las relaciones

como largos hilos cuyos extremos se atan a las ocurrencias de las entidades participantes, esta tabla almacena hilos: los hilos que no existen, no se almacenan. Esto implica que ninguna de las claves ajenas acepte nulos, sea cual sea la clave primaria y sean cuales sean las cardinalidades mínimas con las que participan las entidades en la relación.

Sobre la regla del borrado no podemos asegurar nada, será normalmente eel usuario quien determine qué quiere hacer. En el ejemplo de las transparencias donde se utiliza una tabla aparte para guardar las citas que tienen los pacientes con los médicos, podemos ver cómo por mucho que nos calentemos la cabeza, la respuesta es del propietario de los datos. Pensemos en la clave ajena al paciente: no se puede borrar un paciente si tiene citas (restrigir), porque si se permite el borrado y se propaga a las citas, entonces se desaprovechan las citas que éste tenía; pero claro, si el paciente no va a venir de todos modos, igual se puede restrigir el borrado, pero pedir que se saque un listado de pacientes en lista de espera para ver si quieren aprovechar el hueco que deja el paciente; pero ¿habrá lista de espera? .... En fin, que como no somos del gremio, no sabemos todo lo que se puede hacer, así que mejor preguntamos ¿no parece lo más adecuado?

El siguiente ejemplo intenta aclarar un poco todo lo explicado en esta pregunta. Recuperamos el ejemplo de los empleados y los vehículos que conducen (relación 1:1). Imaginemos que hemos decidido poner una clave ajena en el empleado para expresar la relación con el vehículo que conduce:

```
EMPLEADO(codemp, nombre, matrícula, fecha_ini)
VEHíCULO(matrícula, modelo)
```

Imaginemos también que la clave ajena (y fecha_ini, que es el atributo de la relación) acepta nulos porque la participación del empleado en la relación tiene cardinalidad mínima 0. ¿Qué consulta hay que escribir en SQL para obtener los datos de los empleados que NO conducen ningún vehículo (es decir, aquellos que no participan en la relación)?

```
SELECT codemp, nombre
FROM EMPLEADO
WHERE matrícula IS NULL;
```

Bien, veamos cómo debemos plantear esta misma consulta cuando se escoge representar la relación como una tabla aparte:

```
EMPLEADO(codemp, nombre)
VEHíCULO(matrícula, modelo)
CONDUCE(codemp, matrícula, fecha_ini)
```

Repetimos la pregunta: ¿Qué consulta hay que escribir en SQL para obtener los datos de los empleados que NO conducen ningún vehículo (aquellos que no participan en la relación)?

```
SELECT codemp, nombre
FROM EMPLEADO
WHERE codemp NOT IN
     (SELECT codemp FROM CONDUCE);
```

2.4. *¿Alguna pista para saber si se ha hecho bien la normalización?*

Buenoooo, en primer lugar hay que fijarse en que las dependencias funcionales no deseadas han desaparecido. Estas dependencias vienen dadas por las flechas que no salen de la clave primaria o que salen sólo de un subconjunto de ella. Las únicas dependencias que deben quedar (flechas) son las que salen de la clave primaria COMPLETA.

Al normalizar una tabla (2FN y 3FN) lo que hacemos es obtener distintas proyecciones de ella para repartir sus columnas en varias tablas de modo que se eliminen las dependencias no deseadas (que no son más que redundancias de datos). Por lo tanto, el conjunto de tablas que obtenemos al normalizar debe permitir recuperar la tabla original (la que no estaba en 3FN) haciendo concatenaciones (*joins*). Si nos fijamos en los ejemplos de las transparencias:

```
INSCRIPCION(estudiante, actividad, precio)
```

Tenemos que la actividad determina el precio y al normalizar obtenemos:

```
INSCRIPCION(estudiante, actividad)
ACTIVIDAD(actividad, precio)
```

Las proyecciones que hemos hecho son:

```
ACTIVIDAD := INSCRIPCION[actividad, precio]
INSCRIPCION := INSCRIPCION[estudiante, actividad]
```

Y podemos recuperar la tabla original concatenando las tablas obtenidas:

```
INSCRIPCION := ACTIVIDAD JOIN INSCRIPCION
```

Algo que nos puede ayudar también para ver si vamos por buen camino es que la clave primaria de cada tabla será distinta. Fijémonos: la tabla original de inscripciones tenía 3500 filas, que corresponden a las inscripciones de 2800 estudiantes en 32 actividades distintas. ¿Cuántas filas tiene la nueva tabla de inscripciones? ¿y cuántas filas tiene la nueva tabla de actividades? Piensa las respuestas a estas dos preguntas y luego míralas en la nota al pie[3]

[3] 3500 inscripciones y 32 actividades.

La tabla de inscripciones mantiene su clave primaria y, por lo tanto, mantiene su número de filas. La tabla de actividades tiene como clave primaria la columna de la que salía una flecha no deseada en la tabla original ¿cuántas filas tiene? tantas como actividades distintas existen.

Si no hacemos bien las proyecciones y tenemos más de una tabla con la misma clave primaria, las flechas no deseadas seguirán estando presentes, quizás en otra tabla, pero saliendo de donde no queremos.

Quizá sea de ayuda el pensarlo así: las flechas siempre han de salir de la clave primaria completa. Si hay flechas que no cumplen esto, hay que sacarlas ¿cómo? el atributo (o atributos) del que sale la flecha será la clave primaria de una nueva tabla (¿acaso no es sólo de ellas de donde han de salir las flechas?) y los atributos a los que llega la fecha se MUEVEN de la tabla original a la nueva tabla. Ya que hemos descompuesto una tabla en varios pedazos "verticales" (hemos sacado algunas columnas a otra tabla), tenemos que representar la relación que guardan para poder recuperar la tabla original. Como siempre, el modo de representar la relación es mediante una clave ajena: ese atributo (o atributos) del que antes salía una flecha no deseada es ahora una clave ajena a la nueva tabla en donde él mismo ejerce el papel de clave primaria.

2.5. *¿Qué errores se consideran "graves"?*

Se consideran graves todos aquellos errores que se producen porque hay conceptos que no se han asimilado bien. Y si hay, normalmente, unas pocas formas de hacerlo bien, hay infinitas formas de hacerlo mal ;(

Comentamos aquí algunos errores que han aparecido con frecuencia en el último examen que se ha hecho, el de junio de 2002.

- Representar una relación 1:n con una clave ajena en la parte de la n. Concretamente, un error cometido en el examen ha sido poner en el administrador una clave ajena a la comunidad de vecinos: de este modo, cada administrador sólo podría administrar una comunidad y una misma comunidad podría ser administrada por varios administradores.

- Escoger mal la clave primaria de una tabla que representa una relación. Cuando se ha obtenido una tabla aparte para la relación gestiona, un error cometido es que se ha escogido como clave primaria la combinación de la clave ajena al administrador y la clave ajena a la comunidad de vecinos, lo que está expresando una relación de muchos a muchos entre ambas entidades, cuando en realidad es de uno a muchos. Tampoco es correcto escoger como clave primaria la combinación de la clave ajena al administrador junto a los honorarios: en este caso estamos representando que un mismo administrador puede cobrar distintos honorarios a una misma comunidad.

Un consejo: una vez escogida la clave primaria piensa "qué se traga": si se traga de más es malo; si se traga de menos, también es malo. En este caso, la clave primaria debe ser sólo la clave ajena a la comunidad: de este modo cada comunidad sólo aparecerá una vez (será gestionada por un solo administrador) y un administrador podrá aparecer muchas veces (gestionar varias comunidades). Cada vez que un administrador aparece con una comunidad, tiene unos honorarios que es lo que cobra al año por sus servicios.

- Escoger mal la clave primaria cuando una entidad tiene varios identificadores. ¿Y qué es un identificador? Una "bolita negra". Si una entidad tiene varias bolitas negras, cada una de ellas será una clave candidata en el diseño lógico. Una de esas claves candidatas, SOLO UNA será escogida como clave primaria y el resto serán denominadas claves alternativas. El error que se comete a veces es que se toma como clave primaria la combinación de todas las bolitas negras que tiene la entidad. Muchas veces una bolita negra "engancha" a varios atributos. En este caso, la clave candidata está formada por todos esos atributos a los que engancha.

- Especificar como clave alternativa algo que realmente no lo es. Son claves alternativas aquellos identificadores que aparecen en el esquema conceptual que después no hemos escogido como clave primaria. Si una entidad sólo tiene una bolita negra, esa será la clave primaria de la tabla correspondiente y no hará falta inventarse ninguna clave alternativa: si no la hay, no la hay. Otro error cometido es escoger como clave alternativa un subconjunto de la clave primaria. Este es un error importante de concepto: una clave primaria cumple las propiedades de unicidad y minimalidad (o irreducibilidad). Si se dice que una clave alternativa es un subconjunto de la clave primaria, se está diciendo que la clave primaria no cumple la propiedad de minimalidad: ¡mediante un subconjunto de ella se puede seguir identificando de modo único!

- Especificar para una clave ajena compuesta, las reglas de nulos y borrado para cada uno de los atributos que la forman. Incluso si se especifican exactamente las mismas reglas para cada atributo, se está cometiendo un error importante de concepto: la clave ajena es UNA (aunque compuesta por varios atributos) y como una se ha de comportar: se especifica si acepta o no acepta nulos (y esto será para todos sus atributos a la vez: o todos son nulos o ninguno lo es) y se especifica su comportamiento ante el borrado de filas en la tabla a la que referencia.

- Especificar como regla de borrado anular cuando la clave ajena no acepta nulos. No hacen falta más comentarios ¿verdad?

- Colocar un atributo compuesto con cardinalidad máxima 1 como una tabla dentro de la tabla correspondiente a la entidad. Fijémonos en el ejemplo de las transparencias donde aparece una entidad libro con distintos tipos de atributos. Si el atributo compuesto título (que tiene cardinalidad máxima 1) se representa como una tabla dentro de libro tenemos:

```
LIBRO(isbn, editorial, idioma, TITULO(título_ppal, subtítulo), ...)
```

¿Qué indica esta tabla TITULO? Que para cada libro hay varios títulos (cosa que no refleja el esquema conceptual). ¿Cuál es la clave primaria de TITULO (toda tabla debe tener una clave primaria, aunque esté dentro de otra)? ¿título_ppal? Bueno, vale, cuando esta tabla salga de LIBRO para pasar a 1FN tendremos esta tabla:

```
TITULO(isbn, título_ppal, subtítulo)
```

¿Qué se traga esta clave primaria? Se traga varios títulos para un mismo libro, cada título con su subtítulo correspondiente ¿Es eso lo que se ve en el esquema conceptual? No, según el esquema cada libro tiene un título y un subtítulo. Sólo colocaremos una tabla dentro de otra cuando tengamos atributos con cardinalidad máxima n, ya sean simples o compuestos.

- Cambiar la clave primaria de una tabla conforme ésta va evolucionando durante el diseño lógico. En esta etapa lo que hacemos es proceder por partes: en primer lugar nos fijamos en las entidades y obtenemos una tabla para cada una de ellas. Después nos fijamos en las relaciones. Si una entidad participa en varias relaciones es posible que tenga que acoger varias claves ajenas. Un error importante es el hecho de cambiar la clave primaria conforme se van añadiendo claves ajenas a una tabla, añadiendo esos nuevos atributos a la clave primaria o, simplemente, cambiándola por otra. Si una tabla guarda ocurrencias de una entidad, por ejemplo estudiantes, si éstos se identifican por el DNI, por el hecho de tener un tutor de estancias en prácticas (es una relación) o por estar matriculado en una determinada titulación (es otra relación), no cambia su modo de identificarlo: seguirá siendo único por su DNI ¿verdad? pues la clave primaria de la tabla de los estudiantes permanecerá igual aunque la tabla vaya acogiendo columnas como claves ajenas.

- Inventar atributos no es conveniente porque no se suele especificar todo lo necesario para que puedan ser interpretados por quien debe hacerlo. En caso de que se haga, se debe justificar porqué se introduce el nuevo atributo, cuál es su dominio (básicamente, su rango de valores) y si acepta nulos. Si el atributo pasa a ser la clave primaria, habrá que especificar el resto de claves alternativas (al menos tendrá una, la que el resto de las personas hubieran escogido :). Si no se hace así, se está dejando de representar información importante en el esquema lógico.