

Tema 3

Diseño de bases de datos (E.T.S.I.A. y F.I.)

Diseño lógico



noviembre, 2006

1 Introducción	1
2 Representación de las estructuras	1
2.1 Transformación de las entidades	2
2.2 Transformación de las relaciones	5
2.3 Transformación de la agregación	16
2.4 Resumen	18
2.5 Ejemplo	19
3 Teoría de la normalización	21
3.1 Conceptos previos	21
3.2 Primera forma normal (1FN)	25
3.3 Segunda forma normal (2FN) con una sola clave	27
3.4 Tercera forma normal (3FN) con una sola clave	29
3.5 Tercera forma normal general	31
3.6 Resumen	34
3.7 Ejemplo	34
4 Transformación de las restricciones de integridad	38
5 Representación de las transacciones	39
6 Cuestiones sobre diseño lógico	43

Este tema se ha elaborado utilizando fundamentalmente los siguientes dos textos:

- *Bases de datos relacionales: teoría y diseño*
Laura Mota, Matilde Celma y Juan Carlos Casamayor
SPUPV-94.767, 1994
- *Fundamentos de Sistemas de bases de datos*
Ramez Elmasri y Shamakant B. Navathe
Addison-Wesley Iberamericana, 2002

1 INTRODUCCIÓN

El diseño lógico consiste en la transformación del esquema conceptual, que se encuentra descrito con un cierto modelo de datos, en estructuras y transacciones descritas en términos del modelo de datos en el cual se base el sistema de gestión de bases de datos que se vaya a utilizar.

En este tema se presenta este diseño lógico teniendo en cuenta que el modelo de datos elegido es el modelo relacional; así, se deberá realizar este proceso para obtener finalmente un esquema relacional y un conjunto de transacciones sobre las relaciones de este esquema. Este proceso se dividirá en dos fases:

- 1) Transformación de los aspectos estáticos del esquema conceptual: para ello se verá cómo es posible transformar cada una de las estructuras de un diagrama entidad-relación, modelo utilizado en el diseño conceptual, en relaciones¹. Esta transformación dará lugar a un primer esquema relacional. Además, en la segunda parte de este tema se estudiará la teoría de la normalización, que permite refinar este primer esquema relacional obteniendo un esquema relacional adecuado desde el punto de vista de su manipulación.
- 2) Transformación de los aspectos dinámicos del esquema conceptual: para ello se verá cómo se obtienen transacciones sobre las relaciones del esquema relacional a partir del análisis de transacciones realizado en la fase de diseño conceptual.

2 REPRESENTACIÓN DE LAS ESTRUCTURAS

El proceso de obtención de un esquema relacional que represente adecuadamente todos los aspectos estáticos expresados en el esquema conceptual (que están descritos en el diagrama entidad-relación y en el conjunto de restricciones de integridad añadidas) consiste en aplicar un conjunto de reglas para transformar el diagrama entidad-relación en un esquema relacional, que constará de un conjunto de relaciones que lo representen adecuadamente. En esta transformación también tienen que ser tratadas las posibles restricciones del esquema conceptual, traduciéndolas a expresiones equivalentes del SQL o, en algunos casos, muy pocos, integrándolas en la propia definición de las relaciones. En otros casos, también pocos, aparecen nuevas restricciones de integridad.

El objetivo de este apartado es presentar estas transformaciones. Para ello se supondrá que el sistema de gestión de bases de datos relacionales soporta, para cada relación, la definición de clave primaria, clave ajena, restricción de unicidad y restricción de valor no nulo sobre atributos.

En algunos casos puede suceder que haya varios esquemas relativos posibles para un mismo esquema conceptual; el criterio de elección que se aplicará cuando esto suceda es el siguiente:

"Elige el esquema con menos restricciones de integridad añadidas. Ante igualdad de restricciones, elige el esquema con menos relaciones".

Este criterio se justifica por el hecho de que las restricciones de integridad suponen, usualmente, controles costosos en tiempo; por otra parte, cuantas menos relaciones tenga el esquema más eficientes serán las operaciones de consulta. Como ya se verá más adelante sólo en algunos casos muy concretos puede no seguirse este criterio.

¹ Aquí el término *relación* hace referencia a la estructura de datos del modelo relacional (en inglés *relation*) y no hay que confundirlo con la relación del modelo entidad-relación (en inglés *relationship*).

Para estudiar estas transformaciones, se va a realizar un barrido por los posibles objetos y estructuras de un diagrama entidad-relación, presentándose en cada caso el conjunto de relaciones equivalentes, es decir que expresen la misma realidad que el diagrama².

2.1 Transformación de las entidades

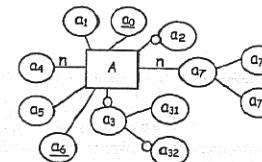
En un diagrama entidad-relación, se pueden distinguir tres clases de entidades:

- Entidades fuertes (ni débiles ni especializadas),
- Entidades débiles, y
- Entidades especializadas.

A continuación se muestra cómo se transforma cada uno de ellos.

2.1.1 Entidades fuertes

Supóngase una entidad, que no sea débil ni especializada, con su conjunto de atributos como se muestra en la siguiente figura:



La relación equivalente es la siguiente:

$A(a_0: \text{dom_}a_0, a_1: \text{dom_}a_1, a_2: \text{dom_}a_2, a_3, a_3: \text{dom_}a_{31}, a_3, a_3: \text{dom_}a_{32}, \{a_4: \text{dom_}a_4\}, a_5: \text{dom_}a_5, a_5: \text{dom_}a_6, \{a_7, a_7: \text{dom_}a_{71}, a_7, a_7: \text{dom_}a_{72}\})$.
 Clave Primaria: {a₀}
 Valor No Nulo: {a₂}
 Valor No Nulo: {a₃, a₃}
 Único: {a₆}

Donde:

- Los dominios de los atributos de la relación se obtienen del anexo del diagrama entidad-relación, donde se asocian dominios a los atributos de las entidades y relaciones.
- El atributo identificador (en el ejemplo a₀) se convierte en la clave primaria de la relación. Si la entidad tuviera un identificador formado por más de un atributo, el conjunto de todos ellos formaría la clave primaria de la relación.
- Las restricciones de valor no nulo (p.e. a₂) y las restricciones de unicidad (p.e. a₆) se representan sin problemas como se muestra en el ejemplo.
- Los atributos multivaluados (p.e. a₃) se incluyen en la relación destacados, cada uno de ellos, entre llaves³. Los problemas que puedan derivarse de la presencia de estos atributos se resolverán en la fase de normalización.
- Los atributos compuestos (p.e. a₃) se representan descomponiéndolos en sus atributos

² Esta equivalencia queda justificada por el hecho de que no hay otro esquema más adecuado. Para estar seguros de esto se propone como ejercicio buscar, en cada caso, esquemas alternativos mejores que el que se propone.

³ Con esta notación se destaca el hecho de que ese atributo toma para cada tupla múltiples valores.

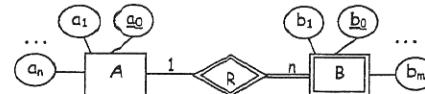
componentes. Esta descomposición podría posponerse hasta la fase de normalización, pero el realizarla directamente permite especificar algunas restricciones de valor no nulo sobre los atributos componentes. Por otra parte, si no hay problemas de ambigüedad con los nombres de los atributos, podrá utilizarse como nombre el descriptor del campo (a_{31}, a_{32}, \dots). Obsérvese cómo se ha representado el atributo a_7 que es un atributo compuesto multivaluado.

- Los atributos derivados (p.e. a_8) se incluyen en la relación como un atributo más, debiendo especificarse su fórmula de derivación como restricción.

2.1.2 Entidades débiles

La transformación de una entidad débil es análoga a la de una entidad que no sea débil; la única diferencia es que es necesario incorporar, como atributos, las claves primarias de las relaciones que representan a las entidades gracias a las cuales se identifica.

Para ilustrar esta transformación, supóngase el siguiente diagrama entidad-relación:

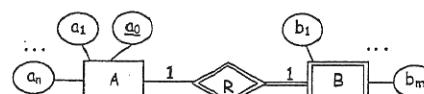


Las relaciones correspondientes son:

$A(a_0: \text{dom_}a_0, a_1: \text{dom_}a_1, \dots, a_n: \text{dom_}a_n)$
Clave Primaria: $\{a_0\}$

$B(b_0: \text{dom_}b_0, b_1: \text{dom_}b_1, \dots, b_m: \text{dom_}b_m, a_0: \text{dom_}a_0)$
Clave Primaria: $\{a_0, b_0\}$
Clave Ajena: $\{a_0\}$ hace referencia a A

Existe otro diagrama en el que puede aparecer una restricción de identificación; es el siguiente:



cuyo esquema relacional equivalente es:

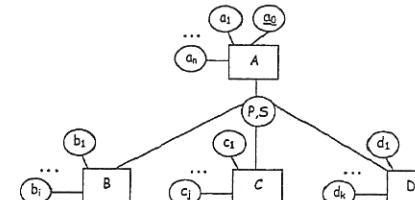
$A(a_0: \text{dom_}a_0, a_1: \text{dom_}a_1, \dots, a_n: \text{dom_}a_n)$
Clave Primaria: $\{a_0\}$

$B(b_1: \text{dom_}b_1, \dots, b_m: \text{dom_}b_m, a_0: \text{dom_}a_0)$
Clave Primaria: $\{a_0\}$
Clave Ajena: $\{a_0\}$ hace referencia a A

Hay que darse cuenta de que con estas transformaciones no sólo se representan las dos entidades, sino que también queda representada la relación R.

2.1.3 Entidades especializadas

Sólo cuando la especialización es *Parcial* y *Solapada* existe una transformación en relaciones totalmente adecuada; en los demás casos es necesario incluir ciertas restricciones de integridad que permitan la definición exacta de estos objetos. La transformación consiste en definir una relación para cada entidad especializada que incluye los atributos propios y también la clave primaria de la relación que representa a la entidad general, que pasa a ser también la clave primaria de la relación. Sea, por ejemplo, el siguiente diagrama entidad-relación:



El conjunto de relaciones que representan ese esquema es el siguiente:

$A(a_0: \text{dom_}a_0, a_1: \text{dom_}a_1, \dots, a_n: \text{dom_}a_n)$
Clave Primaria: $\{a_0\}$

$B(a_0: \text{dom_}a_0, b_1: \text{dom_}b_1, \dots, b_i: \text{dom_}b_i)$
Clave Primaria: $\{a_0\}$
Clave Ajena: $\{a_0\}$ hace referencia a A

$C(a_0: \text{dom_}a_0, c_1: \text{dom_}c_1, \dots, c_j: \text{dom_}c_j)$
Clave Primaria: $\{a_0\}$
Clave Ajena: $\{a_0\}$ hace referencia a A

$D(a_0: \text{dom_}a_0, d_1: \text{dom_}d_1, \dots, d_k: \text{dom_}d_k)$
Clave Primaria: $\{a_0\}$
Clave Ajena: $\{a_0\}$ hace referencia a A

Obsérvese que la clave primaria de las relaciones donde se representan las entidades especializadas se define también como clave ajena a la relación donde está representada la entidad general; de esta forma se expresa la restricción de que toda ocurrencia de cualquier entidad especializada corresponde a una ocurrencia de la entidad general.

En caso de que la generalización sea de otro tipo, el esquema relacional que se debe definir es el mismo, pero es necesaria la inclusión de algunas restricciones de integridad como se muestra a continuación:

- Total y Solapada: En este caso el conjunto de relaciones anterior no representa exactamente el objeto general ya que queda por expresar el hecho de que toda ocurrencia de la entidad general A tiene que estar asociada con al menos una ocurrencia de alguna entidad especializada. Es necesaria la definición de una restricción de integridad como se muestra:

```

CREATE ASSERTION Total CHECK
  NOT EXISTS (SELECT Ax.a0 FROM A Ax
  WHERE Ax.a0 NOT IN (SELECT Bx.a0 FROM B Bx UNION
  SELECT Cx.a0 FROM C Cx UNION
  SELECT Dx.a0 FROM D Dx))
  
```

- Parcial y Disjunta: Para este tipo de especialización, el conjunto de relaciones anterior no expresa la restricción de que las entidades especializadas son disjuntas, esto es, el hecho de que cada ocurrencia de la entidad general sólo puede estar asociada con una ocurrencia de una entidad especializada. La expresión en SQL que representa esta restricción es la siguiente:

```
CREATE ASSERTION Disjunta CHECK
```

NOT EXISTS (SELECT B.AO FROM B WHERE
 B.AO IN (SELECT C.AO FROM C) OR B.AO
 IN (SELECT D.AO FROM D) UNION SELECT C.AO FROM C
 WHERE C.AO IN (SELECT D.AO FROM D))

- Total y Disjunta: En este caso es necesaria la inclusión de las dos restricciones para obtener un esquema relacional que represente exactamente este tipo de objeto general.

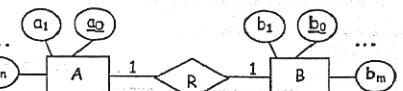
2.2 Transformación de las relaciones

Las transformaciones de las relaciones se van a estudiar atendiendo al grado y a las cardinalidades máximas y mínimas de la relación a transformar. Se van a considerar diecisiete casos que abarcan relaciones binarias, reflexivas y ternarias; después de presentar todos estos casos se estudiará la presencia de atributos en la relación.

2.2.1 Relaciones binarias

2.2.1.1 Relación binaria 1:1 sin restricciones de existencia

Sea el siguiente diagrama entidad-relación:



Las relaciones equivalentes a esta estructura son:

Esquema 1

A($a_0: \text{dom}_{a_0}, a_1: \text{dom}_{a_1}, \dots, a_n: \text{dom}_{a_n}$)
 Clave Primaria: $\{a_0\}$

B($b_0: \text{dom}_{b_0}, b_1: \text{dom}_{b_1}, \dots, b_m: \text{dom}_{b_m}, a_0: \text{dom}_{a_0}$)
 Clave Primaria: $\{b_0\}$
 Único: $\{a_0\}$
 Clave Ajena: $\{a_0\}$ hace referencia a A

Como puede observarse, la representación de la relación R se realiza mediante la inclusión en la relación B de una clave ajena a_0 que hace referencia a la relación A y que además se define también con restricción de unicidad para representar correctamente que la cardinalidad máxima de A es 1.

Este esquema relacional no es el único posible; también podría elegirse el siguiente esquema en el que la relación R se ha representado junto con A:

Esquema 2

A($a_0: \text{dom}_{a_0}, a_1: \text{dom}_{a_1}, \dots, a_n: \text{dom}_{a_n}, b_0: \text{dom}_{b_0}$)
 Clave Primaria: $\{a_0\}$
 Único: $\{b_0\}$

⁴ Sería más adecuado que esta relación se llamara B-R ya que en ella se representan la entidad B y la relación R, sin embargo por comodidad en todo el documento se ha elegido exclusivamente el nombre de la entidad.

Clave Ajena: $\{b_0\}$ hace referencia a B

B($b_0: \text{dom}_{b_0}, b_1: \text{dom}_{b_1}, \dots, b_m: \text{dom}_{b_m}$)

Clave Primaria: $\{b_0\}$

Además de estos dos esquemas, también podrían diseñarse los siguientes que representan correctamente la realidad modelada en el anterior diagrama entidad-relación pero que son menos adecuados que los primeros por constar de una relación más⁵:

Esquema 3

A($a_0: \text{dom}_{a_0}, a_1: \text{dom}_{a_1}, \dots, a_n: \text{dom}_{a_n}$)
 Clave Primaria: $\{a_0\}$

B($b_0: \text{dom}_{b_0}, b_1: \text{dom}_{b_1}, \dots, b_m: \text{dom}_{b_m}$)
 Clave Primaria: $\{b_0\}$

R($b_0: \text{dom}_{b_0}, a_0: \text{dom}_{a_0}$)
 Clave Primaria: $\{b_0\}$

Único: $\{a_0\}$

Valor No Nulo: $\{a_0\}$

Clave Ajena: $\{a_0\}$ hace referencia a A
 Clave Ajena: $\{b_0\}$ hace referencia a B

Esquema 4

A($a_0: \text{dom}_{a_0}, a_1: \text{dom}_{a_1}, \dots, a_n: \text{dom}_{a_n}$)
 Clave Primaria: $\{a_0\}$

B($b_0: \text{dom}_{b_0}, b_1: \text{dom}_{b_1}, \dots, b_m: \text{dom}_{b_m}$)
 Clave Primaria: $\{b_0\}$

R($b_0: \text{dom}_{b_0}, a_0: \text{dom}_{a_0}$)
 Clave Primaria: $\{a_0\}$

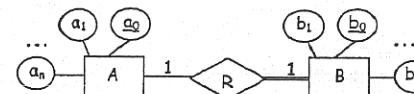
Único: $\{b_0\}$

Valor No Nulo: $\{b_0\}$

Clave Ajena: $\{a_0\}$ hace referencia a A
 Clave Ajena: $\{b_0\}$ hace referencia a B

2.2.1.2 Relación binaria 1:1 con una restricción de existencia

Este tipo de relación se muestra en el siguiente diagrama:



Las relaciones equivalentes a esta estructura son:

A($a_0: \text{dom}_{a_0}, a_1: \text{dom}_{a_1}, \dots, a_n: \text{dom}_{a_n}$)
 Clave Primaria: $\{a_0\}$

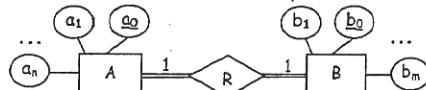
⁵ En los demás casos que se van a estudiar, no se mostrarán todas las esquemas posibles sino tan sólo el que se considere mejor o al menos tan bueno como todos los esquemas posibles.

B($b_0: \text{dom_}b_0, b_1: \text{dom_}b_1, \dots, b_m: \text{dom_}b_m, a_0: \text{dom_}a_0$)
Clave Primaria: $\{b_0\}$
Único: $\{a_0\}$
Valor No Nulo: $\{a_0\}$
Clave Ajena: $\{a_0\}$ hace referencia a A

En este caso, como la entidad B está obligada a relacionarse con la entidad A a través de la relación R, es posible representar esta relación mediante la inclusión, en la relación que representa a la entidad B⁶, de una clave ajena con restricción de valor no nulo para representar la restricción de existencia y con restricción de unicidad para representar correctamente la cardinalidad máxima de A.

2.2.1.3 Relación binaria 1:1 con dos restricciones de existencia

El siguiente diagrama muestra una estructura de este tipo:



El esquema relacional equivalente a esta estructura contiene una única relación que incluye toda la información representada por las dos entidades y por la relación⁷:

A-B($a_0: \text{dom_}a_0, a_1: \text{dom_}a_1, \dots, a_n: \text{dom_}a_n, b_0: \text{dom_}b_0, b_1: \text{dom_}b_1, \dots, b_m: \text{dom_}b_m$)
Clave Primaria: $\{a_0\}$
Único: $\{b_0\}$
Valor No Nulo: $\{b_0\}$

Como se puede observar, la representación de esta estructura da lugar a una única relación, no siendo posible representar independientemente las dos entidades A y B sin tener que añadir restricciones de integridad. Sin embargo, esta solución complica la manipulación de los objetos representados por la entidad B (al no tener una relación propia) así que en algunos casos será preferible el siguiente esquema aunque tenga más relaciones y también restricciones de integridad:

A($a_0: \text{dom_}a_0, a_1: \text{dom_}a_1, \dots, a_n: \text{dom_}a_n, b_0: \text{dom_}b_0$)
Clave Primaria: $\{a_0\}$

Único: $\{b_0\}$

Valor No Nulo: $\{b_0\}$

Clave Ajena: $\{b_0\}$

B($b_0: \text{dom_}b_0, b_1: \text{dom_}b_1, \dots, b_m: \text{dom_}b_m$)
Clave Primaria: $\{b_0\}$

En este caso la elección de un esquema u otro dependerá de cada caso concreto.

La restricción de existencia de la entidad B respecto a la relación R podría representarse con la siguiente expresión:

```
CREATE ASSERTION Restr_existencia CHECK
NOT EXISTS (SELECT *
```

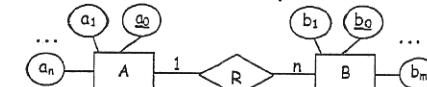
⁶ Piénsese que en este caso si se representa la relación en A habría que añadir una restricción de integridad para representar la restricción de existencia de B.

⁷ Este es probablemente el único caso en el que no se define una relación por cada entidad del diagrama.

```
FROM B Bx
WHERE NOT EXISTS (SELECT * FROM A Ax
WHERE Bx.b_0 = Ax.b_0))
```

2.2.1.4 Relación binaria 1:M sin restricciones de existencia

El siguiente diagrama muestra una relación de este tipo:



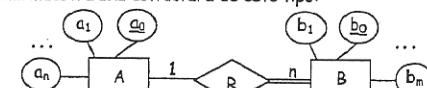
Las relaciones equivalentes a esta estructura son:

A($a_0: \text{dom_}a_0, a_1: \text{dom_}a_1, \dots, a_n: \text{dom_}a_n$)
Clave Primaria: $\{a_0\}$
B($b_0: \text{dom_}b_0, b_1: \text{dom_}b_1, \dots, b_m: \text{dom_}b_m, a_0: \text{dom_}a_0$)
Clave Primaria: $\{b_0\}$
Clave Ajena: $\{a_0\}$ hace referencia a A

En este caso, se puede observar que debido a la cardinalidad de la relación binaria R es posible representar ésta como una clave ajena en la relación que representa la entidad con cardinalidad máxima 1 (la entidad A). Nótese que la posibilidad de que haya ocurrencias de la entidad B que no tomen parte en la relación R está bien trasladada al esquema relacional, ya que es posible que en tuplas de la relación B el atributo correspondiente a la clave ajena tenga valor nulo.

2.2.1.5 Relación binaria 1:M con una restricción de existencia sobre la entidad de cardinalidad máxima 1

El siguiente diagrama muestra una estructura de este tipo:



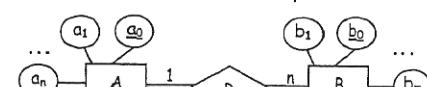
Las relaciones equivalentes a esta estructura son:

A($a_0: \text{dom_}a_0, a_1: \text{dom_}a_1, \dots, a_n: \text{dom_}a_n$)
Clave Primaria: $\{a_0\}$
B($b_0: \text{dom_}b_0, b_1: \text{dom_}b_1, \dots, b_m: \text{dom_}b_m, a_0: \text{dom_}a_0$)
Clave Primaria: $\{b_0\}$
Clave Ajena: $\{a_0\}$ hace referencia a A
Valor No Nulo: $\{a_0\}$

En este caso, la restricción de existencia de la entidad B se representa por la definición de una restricción de valor no nulo sobre la clave ajena de la relación B que representa la relación binaria R.

2.2.1.6 Relación binaria 1:M con dos restricciones de existencia

El siguiente diagrama muestra una estructura de este tipo:



Las relaciones equivalentes a esta estructura son:

$A(a_0: \text{dom}_a_0, a_1: \text{dom}_a_1, \dots, a_n: \text{dom}_a_n)$
Clave Primaria: $\{a_0\}$

$B(b_0: \text{dom}_b_0, b_1: \text{dom}_b_1, \dots, b_m: \text{dom}_b_m, a_0: \text{dom}_a_0)$
Clave Primaria: $\{b_0\}$
Clave Ajena: $\{a_0\}$ hace referencia a A
Valor No Nulo: $\{a_0\}$

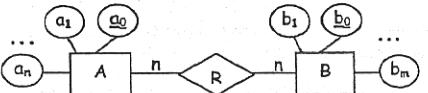
Como puede apreciarse, este esquema es idéntico al de caso anterior, por tanto la restricción de existencia sobre la entidad A no está representada. Esta restricción habría que representarla mediante una expresión del SQL:

```
CREATE ASSERTION Restr_existencia CHECK
    NOT EXISTS (SELECT * FROM A Ax
        WHERE NOT EXISTS (SELECT * FROM B Bx
            WHERE Bx.a_0 = Ax.a_0))
```

El diseño del caso de una relación binaria 1:M con una única restricción de existencia sobre la entidad con cardinalidad máxima n se realiza de forma equivalente al caso de una relación binaria 1:M sin restricciones de existencia, siendo por tanto también necesaria en este caso la utilización de una expresión del SQL para representar de nuevo la restricción de existencia.

2.2.1.7 Relación binaria M:M sin restricciones de existencia

Este tipo de estructura se muestra en el siguiente diagrama:



Las relaciones equivalentes a esta estructura son:

$A(a_0: \text{dom}_a_0, a_1: \text{dom}_a_1, \dots, a_n: \text{dom}_a_n)$
Clave Primaria: $\{a_0\}$

$B(b_0: \text{dom}_b_0, b_1: \text{dom}_b_1, \dots, b_m: \text{dom}_b_m)$
Clave Primaria: $\{b_0\}$

$R(a_0: \text{dom}_a_0, b_0: \text{dom}_b_0)$
Clave Primaria: $\{a_0, b_0\}$
Clave Ajena: $\{a_0\}$ hace referencia a A
Clave Ajena: $\{b_0\}$ hace referencia a B

En este caso, debido a que las dos cardinalidades máximas son n, es necesario la representación de la relación binaria R mediante una relación independiente. La clave primaria de esta relación es el par $\{a_0, b_0\}$ ya que es el único conjunto de atributos de R que satisface las condiciones exigidas a una clave primaria⁸.

Los casos de relaciones binarias M:M en los que aparecen restricciones de existencia se diseñan de manera análoga, siendo necesario escribir expresiones en SQL que representen las restricciones de

⁸ Repásense estas condiciones en el libro de la asignatura de Bases de datos.

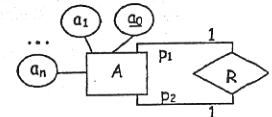
existencia.

2.2.2 Relaciones reflexivas

Estas relaciones son difíciles de asimilar por lo que para entender sus transformaciones se recomienda compararlas con las binarias normales de igual cardinalidad ya que la estrategia seguida es la misma.

2.2.2.1 Relación binaria reflexiva 1:1 sin restricciones de existencia

El diagrama de la derecha muestra una relación de este tipo y la entidad asociada. En la figura, p₁ y p₂ son dos etiquetas que indican el papel que cada ocurrencia de A juega en la relación.



La relación equivalente a esta estructura es:

$A(a_0: \text{dom}_a_0, a_1: \text{dom}_a_1, \dots, a_n: \text{dom}_a_n, a_0-p_2: \text{dom}_a_0)$
Clave Primaria: $\{a_0\}$
Único: $\{a_0-p_2\}$
Clave Ajena: $\{a_0-p_2\}$ hace referencia a A

Dado que estas relaciones son difíciles de entender, a continuación se aclara su significado con un ejemplo. Sea una instancia del esquema anterior:

a ₀	a ₁	...	a ₀ -p ₂
1	b	...	
2	r	...	1
3	n	...	2

Las tres tuplas que contiene la relación nos proporcionan la siguiente información:

- Hay tres objetos de tipo A identificados por a₀=1, a₀=2 y a₀=3.
- Hay dos instancias de la relación R, representadas en las dos últimas tuplas, una que relaciona al objeto 2 (jugando el papel p₁) con el objeto 1 (jugando el papel p₂); y otra que relaciona al objeto con 3 (jugando el papel p₁) con el objeto 2 (jugando el papel p₂).

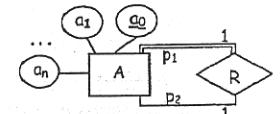
Así pues, en este caso, la relación R queda representada mediante la clave ajena a₀-p₂ en la relación que representa a la entidad. Además, esta clave ajena también se define con restricción de unicidad para representar que la cardinalidad máxima de la entidad A jugando el papel p₂ es 1.

2.2.2.2 Relación reflexiva 1:1 con una restricción de existencia

El diagrama de la derecha muestra una relación de este tipo y la entidad asociada.

La relación equivalente a esta estructura es:

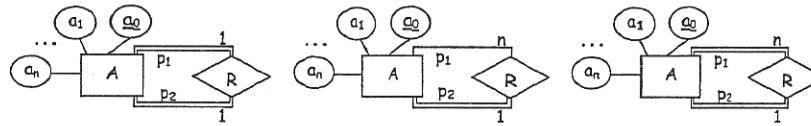
$A(a_0: \text{dom}_a_0, a_1: \text{dom}_a_1, \dots, a_n: \text{dom}_a_n, a_0-p_2: \text{dom}_a_0)$
Clave Primaria: $\{a_0\}$
Único: $\{a_0-p_2\}$
Clave Ajena: $\{a_0-p_2\}$ hace referencia a A
Valor No Nulo: $\{a_0-p_2\}$



En este caso, la restricción de existencia se representa con la definición de la restricción de valor no nulo sobre la clave ajena a₀-p₂.

Curiosamente, este diagrama entidad-relación puede representar exactamente la misma realidad que los tres siguientes por lo que su transformación dará lugar a la misma relación. Esta afirmación puede comprobarse cuando se intenta encontrar una extensión de estos cuatro diagramas que no pueda

representarse por alguno de los otros dos.

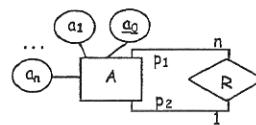


2.2.2.3 Relación reflexiva 1:M sin restricciones de existencia

La figura muestra una relación de este tipo y la entidad asociada por la relación.

La relación equivalente a esta estructura es:

$A(a_0: \text{dom}_a_0, a_1: \text{dom}_a_1, \dots, a_n: \text{dom}_a_n, a_0.p_2: \text{dom}_a_0)$
Clave Primaria: $\{a_0\}$
Clave Ajena: $\{a_0.p_2\}$ hace referencia a A

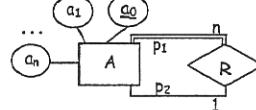


2.2.2.4 Relación reflexiva 1:M con una restricción de existencia

El diagrama muestra una relación de este tipo y la entidad asociada por la relación.

La relación equivalente a esta estructura es:

$A(a_0: \text{dom}_a_0, a_1: \text{dom}_a_1, \dots, a_n: \text{dom}_a_n, a_0.p_2: \text{dom}_a_0)$
Clave Primaria: $\{a_0\}$
Clave Ajena: $\{a_0.p_2\}$ hace referencia a A
Valor No Nulo: $\{a_0.p_2\}$



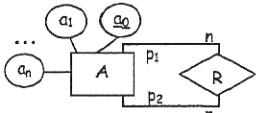
La restricción de existencia queda representada mediante la definición de la restricción de valor no nulo sobre la clave ajena $a_0.p_2$ que representa la relación R.

2.2.2.5 Relación reflexiva M:M sin restricciones de existencia

El diagrama presenta este tipo de relación junto con la entidad asociada.

Las relaciones equivalentes a esta estructura son:

$A(a_0: \text{dom}_a_0, a_1: \text{dom}_a_1, \dots, a_n: \text{dom}_a_n)$
Clave Primaria: $\{a_0\}$
 $R(a_0.p_1: \text{dom}_a_0, a_0.p_2: \text{dom}_a_0)$
Clave Primaria: $\{a_0.p_1, a_0.p_2\}$
Clave Ajena: $\{a_0.p_1\}$ hace referencia a A
Clave Ajena: $\{a_0.p_2\}$ hace referencia a A



En este caso, debido a que la relación reflexiva R tiene cardinalidad máxima M:M, es necesario utilizar una relación independiente para representarla. Nótese cómo en este caso hay en R dos claves ajena que se refieren a la misma relación, A. Si hubiera definida alguna restricción de existencia, también sería necesaria la inclusión de una expresión en SQL. La presencia de restricciones de existencia en las relaciones reflexivas es, sin embargo, poco frecuente.

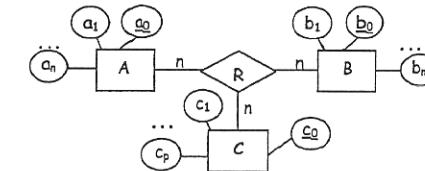
2.2.3 Relaciones ternarias

En este apartado se muestran cuatro casos de relaciones ternarias. En ninguno de ellos se considera

la presencia de restricciones de existencia ya que cuando una relación es de grado mayor que dos siempre se representa independientemente de las entidades y son necesarias restricciones añadidas. En todos los casos la transformación de las entidades sigue el mismo patrón por lo que sólo se incluyen en el primer caso.

2.2.3.1 Relación ternaria M:M:M sin restricciones de existencia

Este tipo de relación se muestra en el siguiente diagrama:



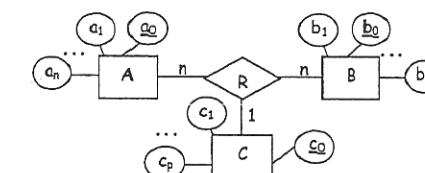
Las relaciones equivalentes a esta estructura son:

$A(a_0: \text{dom}_a_0, a_1: \text{dom}_a_1, \dots, a_n: \text{dom}_a_n)$
Clave Primaria: $\{a_0\}$
 $B(b_0: \text{dom}_b_0, b_1: \text{dom}_b_1, \dots, b_m: \text{dom}_b_m)$
Clave Primaria: $\{b_0\}$
 $C(c_0: \text{dom}_c_0, c_1: \text{dom}_c_1, \dots, c_p: \text{dom}_c_p)$
Clave Primaria: $\{c_0\}$
 $R(a_0: \text{dom}_a_0, b_0: \text{dom}_b_0, c_0: \text{dom}_c_0)$
Clave Primaria: $\{a_0, b_0, c_0\}$
Clave Ajena: $\{a_0\}$ hace referencia a A
Clave Ajena: $\{b_0\}$ hace referencia a B
Clave Ajena: $\{c_0\}$ hace referencia a C

Para este tipo de cardinalidad máxima, la clave primaria de la relación R debe ser el atributo compuesto por las tres claves primarias de las tres relaciones que representan a las tres entidades. (Nótese que la definición de cualquier otra clave primaria da lugar a una mala representación de la relación R).

2.2.3.2 Relación ternaria 1:M:M sin restricciones de existencia

Este tipo de estructura se muestra en el diagrama.



En este caso la relación R es la siguiente:

$R(a_0: \text{dom}_a_0, b_0: \text{dom}_b_0, c_0: \text{dom}_c_0)$
Clave Primaria: $\{a_0, b_0\}$
Valor No Nulo: $\{c_0\}$

- Clave Ajena: $\{a_0\}$ hace referencia a A
 Clave Ajena: $\{b_0\}$ hace referencia a B
 Clave Ajena: $\{c_0\}$ hace referencia a C

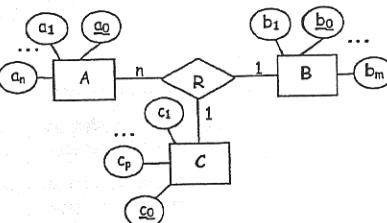
Hay que llamar la atención sobre la definición de una restricción de valor no nulo sobre el atributo c_0 , que es la clave primaria de la relación C . La razón para esta definición es la necesidad de evitar que puedan aparecer tuplas de la relación R que tengan valor nulo para el atributo c_0 , ya que estas tuplas no representan ocurrencias reales de la relación R del diagrama entidad-relación. Así, como regla general, se deberá definir una restricción de valor no nulo sobre todo atributo correspondiente a una clave ajena que no esté presente en la clave primaria de la relación que representa a la relación ternaria del esquema entidad-relación.

2.2.3.3 Relación ternaria 1:1:M sin restricciones de existencia

El diagrama muestra una estructura de este tipo.

En este caso la relación R es la siguiente:

- $R(a_0: \text{dom}_a_0, b_0: \text{dom}_b_0, c_0: \text{dom}_c_0)$
 Clave Primaria: $\{a_0, b_0\}$
 Único: $\{a_0, c_0\}$
 Valor No Nulo: $\{c_0\}$
 Clave Ajena: $\{a_0\}$ hace referencia a A
 Clave Ajena: $\{b_0\}$ hace referencia a B
 Clave Ajena: $\{c_0\}$ hace referencia a C



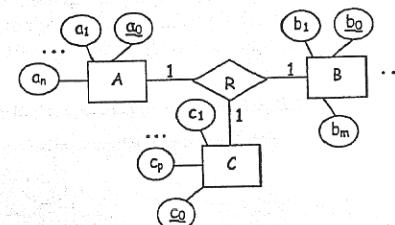
Debido a la presencia de dos cardinalidades máximas 1, existen dos claves candidatas para la relación R : cualquiera de las dos puede ser clave primaria, teniendo que definirse para la otra una restricción de unicidad, además de la correspondiente restricción de valor no nulo sobre la clave ajena que no forma parte de la clave primaria.

2.2.3.4 Relación ternaria 1:1:1 sin restricciones de existencia

Este tipo de estructura se muestra en el diagrama de la derecha.

En este caso la relación R es la siguiente:

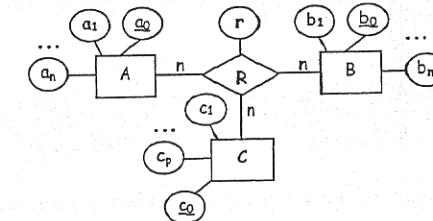
- $R(a_0: \text{dom}_a_0, b_0: \text{dom}_b_0, c_0: \text{dom}_c_0)$
 Clave Primaria: $\{a_0, b_0\}$
 Único: $\{a_0, c_0\}$
 Único: $\{b_0, c_0\}$
 Valor No Nulo: $\{c_0\}$
 Clave Ajena: $\{a_0\}$ hace referencia a A
 Clave Ajena: $\{b_0\}$ hace referencia a B
 Clave Ajena: $\{c_0\}$ hace referencia a C



En este caso, existen tres cardinalidades máximas 1, por lo que hay tres claves candidatas para la relación R : cualquiera de las tres puede ser clave primaria teniendo que definirse para las otras dos una restricción de unicidad.

2.2.4 Atributos en las relaciones

Cuando una relación entre entidades, por ejemplo R , tiene atributos propios, éstos deben incluirse siempre en la relación donde se represente R . Por ejemplo, sea el siguiente diagrama entidad-relación:



Las relaciones equivalentes a esta estructura son:

$A(a_0: \text{dom}_a_0, a_1: \text{dom}_a_1, \dots, a_n: \text{dom}_a_n)$

Clave Primaria: $\{a_0\}$

$B(b_0: \text{dom}_b_0, b_1: \text{dom}_b_1, \dots, b_m: \text{dom}_b_m)$

Clave Primaria: $\{b_0\}$

$C(c_0: \text{dom}_c_0, c_1: \text{dom}_c_1, \dots, c_p: \text{dom}_c_p)$

Clave Primaria: $\{c_0\}$

$R(a_0: \text{dom}_a_0, b_0: \text{dom}_b_0, c_0: \text{dom}_c_0, r: \text{dom}_r)$

Clave Primaria: $\{a_0, b_0, c_0\}$

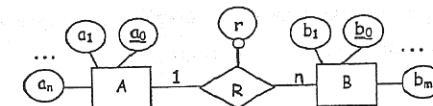
Clave Ajena: $\{a_0\}$ hace referencia a A

Clave Ajena: $\{b_0\}$ hace referencia a B

Clave Ajena: $\{c_0\}$ hace referencia a C

La presencia de atributos en las relaciones puede suponer la aparición de restricciones de integridad que hagan que un esquema relacional que sería adecuado sin los atributos deje de serlo por su presencia. Esta situación se ilustra a continuación. Se considerarán tres casos distintos:

a)



Si consideramos como esquema relacional de partida el diseñado en el apartado 2.2.1.4 añadiendo el atributo r en la relación B (que es en la que se representa R) el esquema quedaría como sigue:

$A(a_0: \text{dom}_a_0, a_1: \text{dom}_a_1, \dots, a_n: \text{dom}_a_n)$

Clave Primaria: $\{a_0\}$

$B(b_0: \text{dom}_b_0, b_1: \text{dom}_b_1, \dots, b_m: \text{dom}_b_m, r: \text{dom}_r)$

Clave Primaria: $\{b_0\}$

Clave Ajena: $\{a_0\}$ hace referencia a A

En este esquema no se puede especificar que el atributo r tiene restricción de valor no nulo ya que no todas las ocurrencias de B tienen que tener valor en el atributo r sino sólo aquéllas que se relacionan con una ocurrencia de A . Así pues, para que este esquema relacional sea correcto se debe añadir la siguiente restricción de integridad:

```
CREATE ASSERTION atributo_no_nulo_en_relación CHECK
  NOT EXISTS (SELECT *
    FROM B BX
```

WHERE ($B.x.a_0$ IS NOT NULL AND $B.x.r$ IS NOT NULL)

OR

(B.x.a₀ IS NULL AND B.x.r IS NOT NULL))

Al añadir una restricción de integridad, este esquema relacional deja de ser el más adecuado ya que hay otro que, aunque tiene una relación más, no necesita restricciones añadidas. Este esquema es el siguiente:

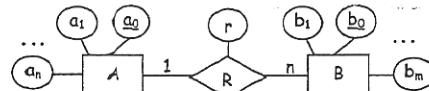
A(a₀; dom_a₀, a₁; dom_a₁, ..., a_n; dom_a_n)
Clave Primaria: {a₀}B(b₀; dom_b₀, b₁; dom_b₁, ..., b_m; dom_b_m)
Clave Primaria: {b₀}R(b₀; dom_b₀, a₀; dom_a₀, r; dom_r)
Clave Primaria: {b₀}Valor No Nulo: {a₀}

Valor No Nulo: {r}

Clave Ajena: {a₀} hace referencia a AClave Ajena: {b₀} hace referencia a B

Estos problemas derivados de la presencia de atributos en la relación y del hecho de que tengan o no tengan restricción de valor no nulo sólo se presentan cuando la relación se ha representado en el esquema relacional junto con algún otro objeto y no en una relación independiente. Para aclarar más este punto, a continuación se presentan algunos diagramas con su esquema relacional más adecuado⁹:

b)



Las relaciones que se obtendrían siguiendo la transformación general serían las mismas que en el caso anterior, pero habría que incluir la restricción de integridad siguiente:

```

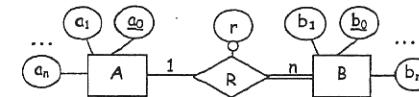
CREATE ASSERTION atributo_en_relación CHECK
NOT EXISTS(SELECT * FROM B BX
WHERE (BX.a0 IS NULL AND BX.r IS NOT NULL))
  
```

por lo que también en este caso es más adecuado el siguiente esquema:

A(a₀; dom_a₀, a₁; dom_a₁, ..., a_n; dom_a_n)
Clave Primaria: {a₀}B(b₀; dom_b₀, b₁; dom_b₁, ..., b_m; dom_b_m)
Clave Primaria: {b₀}R(b₀; dom_b₀, a₀; dom_a₀, r; dom_r)
Clave Primaria: {b₀}Valor No Nulo: {a₀}Clave Ajena: {a₀} hace referencia a AClave Ajena: {b₀} hace referencia a B

⁹ Se deja como ejercicio el desestimar otros posibles esquemas relacionales.

c)

A(a₀; dom_a₀, a₁; dom_a₁, ..., a_n; dom_a_n)Clave Primaria: {a₀}B(b₀; dom_b₀, b₁; dom_b₁, ..., b_m; dom_b_m, a₀; dom_a₀, r; dom_r)Clave Primaria: {b₀}Valor No Nulo: {a₀}

Valor No Nulo: {r}

Clave Ajena: {a₀} hace referencia a A

En esta situación en la que la entidad B tiene restricción de existencia respecto a R, dado que, sobre el atributo a₀ de la relación B resultante se define una restricción de valor no nulo, lo más adecuado es representarlo como se ha hecho en los casos en los que no había atributos en la relación, es decir, la relación R junto a la entidad B, tanto si el atributo r tiene restricción de valor no nulo como si no.

2.3 Transformación de la agregación

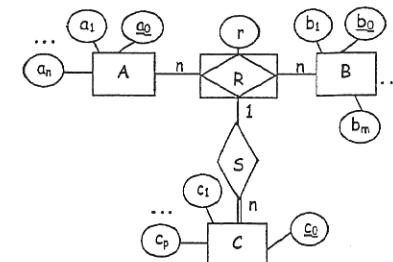
Generalizar el diseño de los objetos agragados que pueden aparecer en un diagrama entidad-relación es complicado ya que éstos pueden estar definidos sobre relaciones de cualquier grado y, por otro lado, tomar parte en cualquier otra relación del diagrama. Por ello se verá primero cómo se realiza este diseño para un caso concreto y luego se dará una guía general para cualquier tipo de agragación.

Supóngase que el diagrama de la figura de la derecha es el diagrama a diseñar.

Como este diagrama contiene una agragación, se deberá iniciar el diseño transformando la parte del diagrama en la que está representada la agragación. Las relaciones que se obtienen son:

A(a₀; dom_a₀, a₁; dom_a₁, ..., a_n; dom_a_n)Clave Primaria: {a₀}B(b₀; dom_b₀, b₁; dom_b₁, ..., b_m; dom_b_m)Clave Primaria: {b₀}R(a₀; dom_a₀, b₀; dom_b₀, r; dom_r)Clave Primaria: {a₀, b₀}Clave Ajena: {a₀} hace referencia a AClave Ajena: {b₀} hace referencia a B

Claramente, la relación que está representando al objeto agragado es R, ya que cada tupla de la relación R expresa que una ocurrencia de la entidad A y otra de B están asociadas. La clave de esta relación es pues la clave o identificador del objeto agragado en el esquema relacional. En nuestro



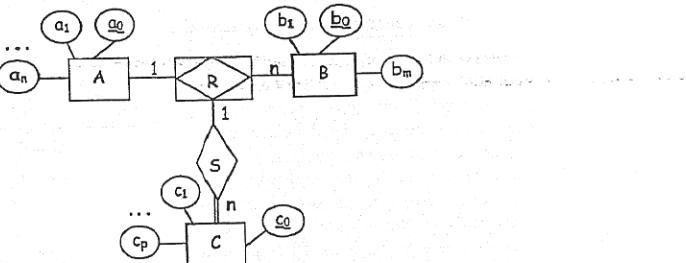
ejemplo esta clave es el atributo compuesto $\{a_0, b_0\}$. Esta clave es el identificador del objeto agregado y deberá ser usada como tal cuando se diseñen las relaciones en las cuales toma parte el objeto agregado.

El segundo paso consiste en transformar el resto del diagrama, es decir, las relaciones en las cuales toman parte el objeto agregado y el resto de entidades y relaciones del diagrama. En el ejemplo anterior, consiste en transformar la relación S en la cual toman parte el objeto agregado y la entidad C , y diseñar también la propia entidad C . Ya que la relación S es una relación binaria entre el objeto agregado y C con cardinalidad máxima 1:M y C sufre una restricción de existencia, se debe obtener la relación siguiente (ver apartado 2.2.1.5):

$C(c_0: \text{dom}_c_0, c_1: \text{dom}_c_1, \dots, c_p: \text{dom}_c_p, a_0: \text{dom}_a_0, b_0: \text{dom}_b_0)$
 Clave Primaria: $\{c_0\}$
 Clave Ajena: $\{a_0, b_0\}$ hace referencia a R
 Valor No Nulo: $\{a_0, b_0\}$

Esta relación representa tanto la entidad C como la relación S . Esta última se representa mediante una clave ajena $\{a_0, b_0\}$ que hace referencia a la relación en la cual está representado el objeto agregado, es decir R .

En otros casos la agregación de una relación puede hacernos reconsiderar el diseño propuesto para la relación sin agregar. Esto se ilustra en el siguiente ejemplo:



Las relaciones que se obtienen al transformar la parte superior del diagrama, teniendo en cuenta las normas explicadas en el apartado 2.2.1.4 son las siguientes:

$A(a_0: \text{dom}_a_0, a_1: \text{dom}_a_1, \dots, a_n: \text{dom}_a_n)$
 Clave Primaria: $\{a_0\}$
 $B(b_0: \text{dom}_b_0, b_1: \text{dom}_b_1, \dots, b_m: \text{dom}_b_m, a_0: \text{dom}_a_0)$
 Clave Primaria: $\{b_0\}$
 Clave Ajena: $\{a_0\}$ hace referencia a A

Ahora es necesario identificar qué relación está representando al objeto agregado. En este caso no hay una relación independiente sino que R está representada en B . El segundo paso consiste como antes en transformar el resto del diagrama:

$C(c_0: \text{dom}_c_0, c_1: \text{dom}_c_1, \dots, c_p: \text{dom}_c_p, b_0: \text{dom}_b_0)$
 Clave Primaria: $\{c_0\}$
 Clave Ajena: $\{b_0\}$ hace referencia a B
 Valor No Nulo: $\{b_0\}$

Ahora bien, este esquema no es adecuado ya que podría darse el caso de que una ocurrencia de C se relacionara con una ocurrencia de B que no se relaciona con ninguna ocurrencia de A , por lo que para que

el esquema fuese correcto habría que añadir la siguiente restricción de integridad:

```

CREATE ASSERTION Agregado CHECK
NOT EXISTS (SELECT Cx.c0 FROM C Cx WHERE
EXISTS (SELECT Bx.b0 FROM B Bx
WHERE (BX.b0=Cx.b0 AND BX.a0 IS NULL)))
  
```

De nuevo la aparición de la restricción hace que este esquema no sea el más deseable ya que se puede encontrar otro con una relación más pero sin restricciones añadidas. Este esquema es el que se muestra:

$A(a_0: \text{dom}_a_0, a_1: \text{dom}_a_1, \dots, a_n: \text{dom}_a_n)$
 Clave Primaria: $\{a_0\}$
 $B(b_0: \text{dom}_b_0, b_1: \text{dom}_b_1, \dots, b_m: \text{dom}_b_m)$
 Clave Primaria: $\{b_0\}$
 $R(b_0: \text{dom}_b_0, a_0: \text{dom}_a_0)$
 Clave Primaria: $\{b_0\}$
 Valor No Nulo: $\{a_0\}$
 Clave Ajena: $\{a_0\}$ hace referencia a A
 Clave Ajena: $\{b_0\}$ hace referencia a B
 $C(c_0: \text{dom}_c_0, c_1: \text{dom}_c_1, \dots, c_p: \text{dom}_c_p, b_0: \text{dom}_b_0)$
 Clave Primaria: $\{c_0\}$
 Clave Ajena: $\{b_0\}$ hace referencia a R
 Valor No Nulo: $\{b_0\}$

Resumiendo, la guía para realizar el diseño de los objetos agregados se puede concretar en los tres siguientes puntos:

- 1) Realizar la transformación del subesquema correspondiente al objeto agregado. En el supuesto de que existieran varios, se realizaría el diseño del más simple al más complejo. En este punto hay que tener en cuenta los problemas que pueden aparecer si la relación que se agrega se representa junto con alguna entidad como se ha ilustrado en el ejemplo.
- 2) Identificar la relación que representa al objeto agregado y la clave de la misma en las relaciones obtenidas en el punto anterior.
- 3) Continuar con la transformación del resto del diagrama. Las participaciones del objeto agregado en relaciones se resolverán teniendo en cuenta el punto anterior, es decir, dónde se encuentra representado el objeto agregado y cuál es su clave.

2.4 Resumen

El conjunto de transformaciones que se ha presentado permite traducir cualquier objeto de un diagrama entidad-relación a un conjunto de definiciones de relaciones, y, en su caso, de expresiones en SQL (que expresarían las propiedades del diagrama entidad-relación que no tienen una transformación exacta al modelo relacional). Los siguientes puntos constituyen una guía para la aplicación de estas reglas:

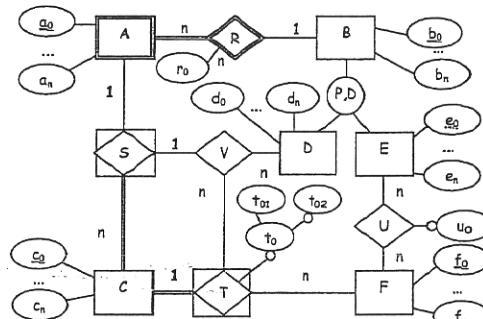
- 1) Es aconsejable, iniciar la transformación por las entidades fuertes y después transformar el resto de entidades. En estas relaciones se definirán también, cuando sea posible, las relaciones con al menos una cardinalidad máxima a 1. Una vez estén todas las entidades transformadas a

relaciones en el esquema relacional, continuar por las demás relaciones del diagrama.

- 2) En presencia de objetos agregados iniciar la transformación por los objetos agregados (del más interior al más exterior). Una vez estén los objetos agregados transformados y sabiendo dónde se encuentran representados en el esquema relacional obtenido hasta ese momento, continuar la transformación del resto del diagrama. Para cualquier participación de un objeto agregado en una relación se deberá utilizar la representación de ese objeto que se identificó en el paso anterior.

2.5 Ejemplo

Sea el diagrama entidad-relación que se muestra en la figura.



A continuación se va a realizar el diseño lógico aplicando las transformaciones que se han presentado. En primer lugar se transforman la entidad fuerte **B** y la débil **A**.

B(b₀; dom_{_b0}, ..., b_n; dom_{_bn})
Clave Primaria: {b₀}

A(a₀; dom_{_a0}, ..., a_n; dom_{_an}, b₀; dom_{_b0}, {r₀; dom_{_r0}})
Clave Primaria: {b₀, a₀}
Clave Ajena: {b₀} hace referencia a B

Ahora ya se puede transformar la entidad **C**, en la relación que la representa se representará también la relación **S** ya que la cardinalidad máxima de **C** en esa relación es 1:

C(c₀; dom_{_c0}, ..., c_n; dom_{_cn}, b₀; dom_{_b0}, a₀; dom_{_a0})
Clave Primaria: {c₀}
Valor No Nulo: {b₀, a₀}
Clave Ajena: {b₀, a₀} hace referencia a A

La relación que representa **F** es la siguiente:

F(f₀; dom_{_f0}, ..., f_n; dom_{_fn})
Clave Primaria: {f₀}

Para representar las dos entidades especializadas se definen dos relaciones junto con una restricción de integridad:

D(d₀; dom_{_d0}, ..., d_n; dom_{_dn}, b₀; dom_{_b0})
Clave Primaria: {b₀}

Clave Ajena: {b₀} hace referencia a B

E(e₀; dom_{_e0}, ..., e_n; dom_{_en}, b₀; dom_{_b0})
Clave Primaria: {b₀}
Clave Ajena: {b₀} hace referencia a B
Único: {e₀}

Restricción de integridad:

```

CREATE ASSERTION Disjunta CHECK
    NOT EXISTS (SELECT * FROM D Dx, E Ex
    WHERE Dx.b0= Ex.b0)
  
```

La relación **U** se representa por separado al ser de tipo M:M:

U(f₀; dom_{_f0}, b₀; dom_{_b0}, u₀; dom_{_u0})
Clave Primaria: {f₀, b₀}
Valor No Nulo: {u₀}
Clave Ajena: {f₀} hace referencia a F
Clave Ajena: {b₀} hace referencia a E

La relación **T** se podría representar junto a **F**, aunque como tiene un atributo con restricción de valor no nulo y **F** no tiene restricción de existencia respecto a **T**, esto obligaría a añadir una restricción en SQL. Por tanto, es mejor definir una relación propia para **T** cuya clave primaria será el atributo **t₀** ya que la cardinalidad máxima de **F** es 1. Habrá que añadir la restricción de valor no nulo para **c₀** porque define la ocurrencia de **C** que participa en **T**.

T(c₀; dom_{_c0}, f₀; dom_{_f0}, t₀₁; dom_{_t01}, t₀₂; dom_{_t02})
Clave Primaria: {f₀}
Valor No Nulo: {c₀, t₀₂}
Clave Ajena: {c₀} hace referencia a C
Clave Ajena: {f₀} hace referencia a F

Además, como hay una restricción de existencia en **C** respecto a **T** se debe añadir una restricción en SQL:

```

CREATE ASSERTION Restr_existencia CHECK
    NOT EXISTS (SELECT *
    FROM C Cx
    WHERE NOT EXISTS (SELECT * FROM T Tx
    WHERE Cx.c0 = Tx.c0))
  
```

Por último hay que definir una relación para representar a **V**, que es una relación ternaria entre dos objetos agregados y uno especializado. Para saber qué atributos incluir en **V**, hay que averiguar en qué relación están representados cada uno de los objetos: **S** en la relación **C**, y **D** y **T** por separado. Como clave primaria de la relación, dadas las cardinalidades máximas, se elige la unión de las claves primarias de las relaciones **D** y **T**. El resto de claves primarias, **c₀** en este caso, debe tener restricción de valor no nulo.

V(c₀; dom_{_c0}, f₀; dom_{_f0}, b₀; dom_{_b0})
Clave Primaria: {f₀, b₀}
Valor No Nulo: {c₀}
Clave Ajena: {c₀} hace referencia a C
Clave Ajena: {f₀} hace referencia a T
Clave Ajena: {b₀} hace referencia a D

3 TEORÍA DE LA NORMALIZACIÓN

Antes de considerarlo definitivo, el esquema relacional obtenido con las transformaciones presentadas en el apartado 2 debe ser revisado para comprobar que se encuentra adecuadamente diseñado. Para ello se utilizarán los conceptos de la teoría de la normalización. La aplicación de esta teoría hay que entenderla como una fase de refinamiento que se aplica al esquema relacional obtenido para comprobar que no existe ningún problema de manipulación.

En este apartado se va a presentar esta teoría de la normalización que consiste en un conjunto de conceptos que permiten conocer el grado de corrección de un esquema de base de datos relacional. La normalización de datos puede considerarse, de esta forma, como un proceso durante el cual los esquemas de relación insatisfactorios se descomponen repartiendo sus atributos entre esquemas de relación más pequeños que poseen propiedades deseables.

Antes de iniciar la presentación de esta teoría, se introducen unos conceptos previos necesarios para la posterior definición de las *formas normales*.

3.1 Conceptos previos

Dependencia funcional

Sea R un esquema de relación cuyo conjunto de atributos es $A = \{A_1, A_2, \dots, A_n\}$; una dependencia funcional entre dos conjuntos de atributos X e Y ($X \subseteq A, Y \subseteq A$), denotado por $X \rightarrow Y$, especifica una restricción sobre las posibles tuplas que podrían aparecer en una extensión de R . La restricción es la siguiente: "para cualquier par de tuplas t_1 y t_2 posibles en R tales que $t_1[X]^{10} = t_2[X]$ entonces se debe cumplir $t_1[Y] = t_2[Y]$ ". Se suele decir entonces que X determina a Y ; alternativamente, que Y depende funcionalmente de X .

Por ejemplo, sea el siguiente esquema de relación:

`Ha_escrito(DNI: dom_dni, nombre: dom_nom, ISBN: dom_IS, título: dom_tit, pesetas: dom_pes)`
`CP: {DNI, ISBN}`

en el que cada tupla representa la siguiente información: "el escritor de D.N.I. *DNI* que se llama *nombre* ha escrito, solo o en colaboración, el libro de código *ISBN* y de título *título* y ha recibido por ese trabajo la cantidad de *pesetas* *pesetas*".

Entre los atributos de esta relación, se pueden detectar las siguientes dependencias funcionales:

1. $\{DNI\} \rightarrow \{\text{nombre}\}^{11}$
2. $\{DNI, ISBN\} \rightarrow \{\text{nombre}\}$
3. $\{DNI, título\} \rightarrow \{\text{nombre}\}$
4. $\{DNI, pesetas\} \rightarrow \{\text{nombre}\}$
5. $\{DNI, ISBN, título\} \rightarrow \{\text{nombre}\}$
6. $\{DNI, ISBN, pesetas\} \rightarrow \{\text{nombre}\}$
7. $\{DNI, título, pesetas\} \rightarrow \{\text{nombre}\}$
8. $\{DNI, ISBN, título, pesetas\} \rightarrow \{\text{nombre}\}$
9. $\{ISBN\} \rightarrow \{\text{título}\}$

¹⁰ Con esta notación se representa el valor que la tupla t_i toma en los atributos del conjunto X .

¹¹ Es decir, para un valor de *DNI* sólo existe asociado un posible valor del atributo *nombre*.

10. $\{ISBN, DNI\} \rightarrow \{\text{título}\}$
11. $\{ISBN, \text{nombre}\} \rightarrow \{\text{título}\}$
12. $\{ISBN, pesetas\} \rightarrow \{\text{título}\}$
13. $\{ISBN, DNI, \text{nombre}\} \rightarrow \{\text{título}\}$
14. $\{ISBN, DNI, pesetas\} \rightarrow \{\text{título}\}$
15. $\{ISBN, \text{nombre}, pesetas\} \rightarrow \{\text{título}\}$
16. $\{ISBN, DNI, \text{nombre}, pesetas\} \rightarrow \{\text{título}\}$
17. $\{DNI, ISBN\} \rightarrow \{\text{pesetas}\}$
18. $\{DNI, ISBN, \text{nombre}\} \rightarrow \{\text{pesetas}\}$
19. $\{DNI, ISBN, \text{título}\} \rightarrow \{\text{pesetas}\}$
20. $\{DNI, ISBN, \text{nombre}, \text{título}\} \rightarrow \{\text{pesetas}\}$

Aunque pudiera parecerlo, este conjunto de dependencias funcionales no está completo ya que por ejemplo no aparece la siguiente: $\{DNI, ISBN\} \rightarrow \{\text{título, nombre}\}$. Esto es debido a que esta dependencia funcional, así como bastantes de la lista anterior, pueden deducirse a partir de un conjunto más pequeño (incluso de varias formas) al que llamaremos *conjunto de interés*. Aunque esta deducción podría realizarse simplemente razonando, existen seis reglas de inferencia que, a partir de ese conjunto que más tarde se definirá, permiten la obtención de todas las dependencias posibles entre un conjunto de atributos.

Sea R de nuevo un esquema de relación cuyo conjunto de atributos es $A = \{A_1, A_2, \dots, A_n\}$, y sean X, Y y Z subconjuntos de A . Las reglas de inferencia son las siguientes¹²:

- (R1) Regla reflexiva: si $Y \subseteq X$ entonces $X \rightarrow Y$
- (R2) Regla de aumento: si $X \rightarrow Y$ entonces $X \cup Z \rightarrow Y \cup Z$ y $X \cup Z \rightarrow Y$
- (R3) Regla transitiva: si $X \rightarrow Y$ y $Y \rightarrow Z$ entonces $X \rightarrow Z$ (a esta dependencia funcional se le suele denominar *dependencia transitiva*)
- (R4) Regla de descomposición: si $X \rightarrow Y \cup Z$ entonces $X \rightarrow Y$ y $X \rightarrow Z$
- (R5) Regla de unión o aditiva: si $X \rightarrow Y$ y $X \rightarrow Z$ entonces $X \rightarrow Y \cup Z$
- (R6) Regla pseudotransitiva: si $X \rightarrow Y$ y $W \cup Y \rightarrow Z$ entonces $W \cup X \rightarrow Z$

Todas estas reglas pueden demostrarse a partir de la definición de dependencia funcional, por demostración directa o por contradicción.

Aplicando estas reglas al ejemplo, podrían obtenerse las siguientes dependencias funcionales (se consideran las 20 anteriores dependencias funcionales y las que se vayan obteniendo).

Regla	Dado que ... (Premisa)	Entonces... (Conclusión)
R1	$\{DNI\} \subseteq \{DNI, ISBN\}$	$\{DNI, ISBN\} \rightarrow \{DNI\}$ (21)
R2	$\{DNI\} \rightarrow \{\text{nombre}\}$	(1) $\{DNI, \text{título}\} \rightarrow \{\text{nombre, título}\}$ (22) $\{DNI, \text{título}\} \rightarrow \{\text{nombre}\}$ (3)
R3	$\{DNI, ISBN\} \rightarrow \{DNI\}$	(21) $\{DNI, ISBN\} \rightarrow \{\text{nombre}\}$ (2)
	$\{DNI\} \rightarrow \{\text{nombre}\}$	(1)
R4	$\{DNI, \text{título}\} \rightarrow \{\text{nombre, título}\}$	(22) $\{DNI, \text{título}\} \rightarrow \{\text{nombre}\}$ (3)

¹² Las tres primeras reglas se suelen conocer con el nombre de *Reglas de Inferencia de Armstrong*.

Regla	Dado que ... (Premisa)	Entonces... (Conclusión)
		$\{\text{DNI}, \text{título}\} \rightarrow \{\text{título}\}$ (23)
R5	$\{\text{DNI}, \text{ISBN}, \text{pesetas}\} \rightarrow \{\text{nombre}\}$ (6) $\{\text{DNI}, \text{ISBN}, \text{pesetas}\} \rightarrow \{\text{título}\}$ (14)	$\{\text{DNI}, \text{ISBN}, \text{pesetas}\} \rightarrow \{\text{nombre}, \text{título}\}$ (24)
R6	$\{\text{DNI}\} \rightarrow \{\text{nombre}\}$ (1) $\{\text{ISBN}, \text{nombre}\} \rightarrow \{\text{título}\}$ (11)	$\{\text{ISBN}, \text{DNI}\} \rightarrow \{\text{título}\}$ (10)

Antes de pasar a la siguiente definición, hay que hacer hincapié en que las dependencias funcionales se definen por la semántica de los atributos y no deben deducirse de la observación de una extensión concreta de una relación. Supóngase que en un instante determinado la información referente a escritores es la siguiente:

DNI	nombre	ISBN	título	pesetas
17.897.569	Pepe Pérez	1254567W	El corsario	236.563
54.325.658	Juan Gómez	458264R	Pistas	25.369

Pese a que, en estos momentos, sólo hay un escritor de nombre *Pepe Pérez* y un solo escritor de nombre *Juan Gómez* no se puede afirmar que exista la dependencia funcional $\{\text{nombre}\} \rightarrow \{\text{DNI}\}$.

Dependencia funcional completa

Una dependencia funcional entre dos conjuntos de atributos $X \rightarrow Y$ es completa si la eliminación de cualquier atributo A_i de X hace que la dependencia deje de existir, es decir si $\forall A_i / A_i \in X$ se cumple que Y no depende funcionalmente de $(X - \{A_i\})$.

Se dice entonces que Y depende funcionalmente de forma completa de X o también que Y depende completamente de X .

De las 24 dependencias funcionales del ejemplo anterior son completas las siguientes¹³:

1. $\{\text{DNI}\} \rightarrow \{\text{nombre}\}$
9. $\{\text{ISBN}\} \rightarrow \{\text{título}\}$
17. $\{\text{DNI}, \text{ISBN}\} \rightarrow \{\text{pesetas}\}$
22. $\{\text{DNI}, \text{título}\} \rightarrow \{\text{nombre}, \text{título}\}$

Conjunto de interés

Un conjunto de interés de una relación es un conjunto de dependencias funcionales que basta considerar para normalizar esa relación eliminando problemas de redundancia. A partir de este conjunto, utilizando las reglas anteriores, se debe poder obtener todas dependencias funcionales existentes entre los atributos de la relación. Este conjunto, sea CI , cumple las siguientes condiciones: si $X \rightarrow Y \in CI$ entonces:

1. Y consta de un solo elemento (i.e. las dependencias funcionales 22 y 24 no formarán parte de CI), y
2. Y depende completamente de X (i.e. las dependencias funcionales 2..8, 10..16, 18.. 21 y 23 no formarán parte de CI).

¹³ Estas dependencias pueden obtenerse aplicando la definición que se acaba de dar.

3. El conjunto es minimal. Esto significa que si se elimina alguna dependencia funcional de CI ya no obtiene el mismo conjunto dependencias funcionales al aplicar las reglas de inferencia.

De acuerdo a esta definición, el conjunto de interés del ejemplo que se está utilizando es el siguiente:

$$CI = \{\{\text{DNI}\} \rightarrow \{\text{nombre}\}, \{\text{ISBN}\} \rightarrow \{\text{título}\}, \{\text{DNI}, \text{ISBN}\} \rightarrow \{\text{pesetas}\}\}$$

Para la normalización de las relaciones sólo se considerarán las dependencias funcionales de este conjunto.

El conjunto de interés de una relación no es único. Sea, por ejemplo, el siguiente esquema de relación que almacena información sobre los socios de un gimnasio:

Socio(num_socio: dom_num, nombre: dom_nombre, edad: dom_edad, DNI: dom_dni)
Clave Primaria: {num_socio}
Único: {DNI}

Entre cuyos atributos existen las siguientes dependencias funcionales completas:

- | | | |
|--|--|---|
| - $\{\text{DNI}\} \rightarrow \{\text{nombre}\}$ | - $\{\text{DNI}\} \rightarrow \{\text{edad}\}$ | - $\{\text{DNI}\} \rightarrow \{\text{num_socio}\}$ |
| - $\{\text{num_socio}\} \rightarrow \{\text{nombre}\}$ | - $\{\text{num_socio}\} \rightarrow \{\text{edad}\}$ | - $\{\text{num_socio}\} \rightarrow \{\text{DNI}\}$ |

Asociados a esta relación existen dos conjuntos de interés posibles:

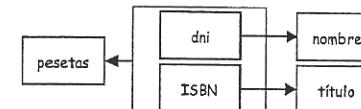
$$CI_1 = \{\{\text{DNI}\} \rightarrow \{\text{nombre}\}, \{\text{DNI}\} \rightarrow \{\text{edad}\}, \{\text{DNI}\} \rightarrow \{\text{num_socio}\}, \{\text{num_socio}\} \rightarrow \{\text{DNI}\}\}$$

$$CI_2 = \{\{\text{num_socio}\} \rightarrow \{\text{nombre}\}, \{\text{num_socio}\} \rightarrow \{\text{edad}\}, \{\text{DNI}\} \rightarrow \{\text{num_socio}\}, \{\text{num_socio}\} \rightarrow \{\text{DNI}\}\}$$

Diagrama de dependencias funcionales

Permite una representación gráfica de las dependencias funcionales facilitando su estudio. Aunque no está muy estandarizado, estos diagramas utilizan cajas para encmarcar los atributos o conjuntos de atributos y flechas para denotar la dependencia funcional. Normalmente sólo se representan las dependencias funcionales del conjunto de interés.

El diagrama de dependencias funcionales del ejemplo que se está estudiando sería el siguiente:



Clave de una relación

Sea R un esquema de relación cuyo conjunto de atributos es $A = \{A_1, A_2, \dots, A_n\}$, y sea C un subconjunto de atributos de ese esquema ($C \subseteq A$): se dice que C es una clave de R si C es la clave primaria de R o bien si C tiene una restricción de unicidad.

Teniendo en cuenta lo que supone ser clave, es evidente que todos los atributos de una relación dependen funcionalmente de cada clave que hay en la relación.

Visto desde otra perspectiva, si R es un esquema de relación cuyo conjunto de atributos es $A = \{A_1, A_2, \dots, A_n\}$, cualquier subconjunto de atributos de ese esquema ($C \subseteq A$) del que dependan funcionalmente todos los demás atributos (i.e. el subconjunto $A - C$) debe ser, o bien la clave primaria de la relación o bien tener restricción de unicidad.

En el ejemplo sólo hay una clave que es {DNI, ISBN}.

Atributo primo

Sea R un esquema de relación cuyo conjunto de atributos es $A = \{A_1, A_2, \dots, A_n\}$, un atributo A se dice que es primo si forma parte de alguna clave de R .

En el ejemplo que está desarrollándose hay dos atributos primos: DNI e ISBN.

Por comodidad, cuando un atributo no sea primo se utilizará la expresión no-primo.

3.2 Primera forma normal (1FN)

Definición de la 1FN

Una relación R está en 1FN si sus atributos sólo pueden tomar valores atómicos (simples, indivisibles).

Así pues, la 1FN prohíbe tener un conjunto de valores, una tupla (o registro) de valores o una combinación de ambos como valor de un atributo para una tupla individual.

Problemas que resuelve

Sea el siguiente esquema relacional:

Proveedor(vcod: dom_vcod, nombre: dom_nom, teléfonos: dom_conj_tel¹⁴, dir:dom_dir)
Clave Primaria: {vcod}

Donde los dominios son los siguientes:

- dom_vcod: entero
- dom_conj_tel: conjunto de dom_tel
- dom_calle: cad(40)
- dom_ciudad: cad(20)
- dom_dir: registro de [calle: dom_calle; número: dom_número; ciudad: dom_ciudad]
- dom_nom: cad(40)
- dom_tel: cad(9)
- dom_número: entero

y donde cada tupla representa la siguiente información: "el proveedor de código vcod se llama nombre vive en la dirección dir y se le puede localizar en cualquier número del conjunto teléfonos".

Una extensión de ese esquema de relación podría ser la siguiente:

Proveedor

vcod	nombre	teléfonos	Dir
V1	Pepe	(96 3233258, 964 523844 979 568987, 987 456123)	Paz 7, Valencia
V2	Juan	(96 3852741, 910147258)	Eolo 3, Castellón
V3	Eva	(987 456 312)	F. Lorca 2, Utiel

Los problemas del uso de relaciones que no están en primera forma normal vienen provocados por la necesidad de utilizar operadores asociados a tipos de datos estructurados. Por ejemplo, la información contenida en los atributos teléfonos y dir obligaría a utilizar operadores asociados a estos tipos de datos, ya sean conjuntos, listas, registros, etc. Todo ello provoca la aparición de múltiples problemas.

¹⁴ Otra forma equivalente de destacar que un proveedor tiene varios teléfonos sería utilizando la notación introducida al presentar la transformación de los atributos multivaluados en el apartado 2.1.1 es decir {teléfono:dom_tel}.

que aconsejan no utilizar relaciones que no estén en primera forma normal.

Paso a 1FN

La técnica para transformar una relación que no está en 1FN a una relación en 1FN es muy simple. Supóngase que la relación R no está en 1FN. Por tanto, R ha de tener un atributo A cuyo dominio asociado es:

- un conjunto,
- un registro.

Estas dos situaciones aparecen cuando el objeto del diagrama entidad-relación del cual proviene R tiene un atributo multivalorado o estructurado, respectivamente. Para el caso a) la solución es extraer de R el atributo A y crear una nueva relación R' que contiene la clave primaria de R y el atributo A al que se le asocia el dominio de los elementos del conjunto en la relación R . Así en el ejemplo se crearía una nueva relación que podría llamarse Lista_tel cuyo esquema sería {vcod:dom_vcod, teléfono: dom_tel}. En el caso b) la solución es sustituir en la relación R el atributo A por los atributos que corresponden a los campos del registro y cuyo dominio asociado es el de los correspondientes campos. En el ejemplo, sustituiríamos el par dir: dom_dir por los pares calle: dom_calle, número: dom_número y ciudad: dom_ciudad.

Con este cambio, la información representada en la extensión anterior se representaría como sigue:

Proveedor

vcod	nombre	calle	número	ciudad
V1	Pepe	Paz	7	Valencia
V2	Juan	Eolo	3	Castellón
V3	Eva	F. Lorca	2	Utiel

Lista_tel

vcod	teléfonos
V1	96 3233258
V1	964 523844
V1	979 568987
V1	987 456123
V2	96 3852741
V2	910 147258
V3	987 456 312

Resulta evidente que el atributo vcod no puede ser la clave primaria de la nueva relación así que hay que estudiar las dependencias funcionales entre sus atributos para escoger una clave primaria. En el ejemplo, si suponemos que un teléfono no puede ser compartido por varios proveedores¹⁵ (i.e. {teléfono} → {vcod}) entonces la clave primaria es {teléfono} quedando el esquema de relación como sigue:

Lista_tel(vcod: dom_vcod, teléfono: dom_tel)

Clave Primaria: {teléfono}

Clave Ajena: {vcod} hace referencia a Proveedor

Valor No Nulo: {vcod}

¹⁵ Si se supusiera que varios proveedores pueden tener un mismo teléfono, entonces la clave primaria sería {vcod, teléfono}.

El exigir que una relación esté en primera forma normal resuelve los problemas mencionados anteriormente; sin embargo, las relaciones que están en primera forma normal siguen provocando algunos problemas que son resueltos por formas normales superiores.

Reflexión sobre la 1FN

La primera forma normal se considera hoy en día parte de la definición formal de relación ya que los dominios asociados a los atributos debe ser simples; sin embargo, parece ser que la tendencia actual es enriquecer los lenguajes de manipulación para relajar esta restricción y permitir dominios complejos.

En los apartados siguientes se presentan la segunda y tercera formas normales suponiendo que sólo existe una clave¹⁶ (la clave primaria) para generalizarla después.

3.3 Segunda forma normal (2FN) con una sola clave

Definición de la 2FN

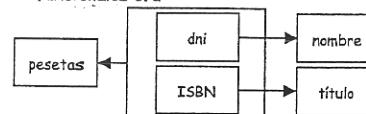
Una relación R está en 2FN si: está en 1FN y todo atributo no-primo depende funcionalmente de forma completa de la clave primaria de R .

Problemas que resuelve

Sea de nuevo el esquema relacional $Ha_escrito$ del apartado 3.1:

$Ha_escrito(DNI: dom_dni, nombre: dom_nom, ISBN: dom_IS, título: dom_tit, pesetas: dom_pes)$
 $CP: \{DNI, ISBN\}$

cuyo diagrama de dependencias funcionales era:



Si se observa este diagrama fácilmente se puede apreciar que los atributos no-primos *nombre* y *título* de la relación no dependen completamente de la clave primaria, lo que provoca, como se puede observar en la siguiente extensión, la presencia de redundancia:

DNI	nombre	ISBN	título	pesetas
17.897.569	Pepe Pérez	1254567W	El corsario	236.563
17.897.569	Pepe Pérez	458264R	Pistas	100.000
54.325.658	Juan Gómez	458264R	Pistas	250.000
54.325.658	Juan Gómez	1254567W	El corsario	25.369
15.236.588	Maria Bur	8524697Y	Dependencias	132.566

Como consecuencia de esa redundancia, la manipulación de esta relación presenta los siguientes problemas:

- **Inserción:** no puede darse de alta un autor que no haya escrito ningún libro ni tampoco introducir la información de un libro si no se sabe quién lo ha escrito.
- **Borrado:** el borrado de una tupla puede provocar a su vez el borrado de toda la información de un autor si éste sólo ha escrito un libro o de un libro si sólo tiene un autor.

¹⁶ Recuérdese la definición dada en el apartado 3.1.

- **Actualización:** la redundancia asociada a los atributos *nombre* y *título* provoca que la actualización de los mismos sea mucho más costosa.

Paso a 2FN

Supóngase que la relación R cuya clave primaria es CP está en 1FN pero no en 2FN. De este hecho y de la definición de 2FN se puede deducir que la clave CP consta de más de un atributo y que existe al menos un atributo (o conjunto de atributos) no-primo A de R que no depende completamente de CP . Por tanto existe un subconjunto propio de CP , llámeselo S , tal que A depende completamente de S . Por otra parte, es posible que exista un atributo B de R que sí dependa completamente de CP . La técnica consiste en obtener un conjunto de relaciones proyectando la relación R adecuadamente, teniendo en cuenta cuáles son las dependencias no completas que aparecen en R . En el supuesto anterior se deben hacer las siguientes proyecciones: $R1=R[S, A]$ y $R2=R[CP, B]$ ¹⁷. La clave primaria de $R1$ es S y la clave primaria de $R2$ es CP ; además $R2$ tiene una clave ajena que hace referencia a $R1$. Así, las dos relaciones resultantes quedan en 2FN. Esta forma de realizar las proyecciones y la definición de clave ajena en $R2$ asegura que la información contenida en R , por un lado, y en $R1$ y $R2$ es idéntica.

Aplicando esta transformación a la relación $Ha_escrito$ se obtendría el siguiente esquema relacional:

$Autor(DNI: dom_dni, nombre: dom_nom)$
Clave Primaria: {DNI}

$Libro(ISBN: dom_IS, título: dom_tit)$
Clave Primaria: {ISBN}

$Ha_escrito(DNI: dom_dni, ISBN: dom_IS, pesetas: dom_pes)$
Clave Primaria: {DNI, ISBN}
Clave Ajena: {DNI} hace referencia a Autor
Clave Ajena: {ISBN} hace referencia a Libro

Y la información antes presentada se distribuiría de la siguiente forma:

Autor	
17.897.569	Pepe Pérez
54.325.658	Juan Gómez
15.236.588	Maria Bur

Libro	
ISBN	título
1254567W	El corsario
458264R	Pistas
8524697Y	Dependencias

Ha_escrito

DNI	ISBN	pesetas
17.897.569	1254567W	236.563
17.897.569	458264R	100.000
54.325.658	458264R	250.000
54.325.658	1254567W	25.369

¹⁷ En caso de no existir el atributo B de R esta segunda proyección sería $R2=R[CP]$.

15.236.588	8524697Y	132.566
------------	----------	---------

Los problemas de manipulación que aparecían en la relación *Ha_escrito* ya no se dan en el nuevo esquema relacional ya que:

- **Inserción:** es posible realizar la inserción de información referente a un autor o a un libro de forma independiente.
- **Borrado:** el borrado del único libro escrito por un autor no implica el borrado de la información del autor, ni el borrado de un autor que haya escrito en solitario un libro supone el borrado de la información del libro.
- **Actualización:** la redundancia que existía en los atributos *nombre* y *título* ha desaparecido, por tanto la actualización de esta información no presenta los problemas vistos anteriormente.

Aunque el paso de una relación a 2FN elimina algunos problemas, todavía pueden presentarse otros que dificultan la manipulación de la misma. Algunos de estos problemas se resuelven con la tercera forma normal.

3.4 Tercera forma normal (3FN) con una sola clave

Definición de la 3FN

Una relación *R* está en 3FN si está en 2FN y ningún atributo no-primo depende funcionalmente de forma transitiva de la clave primaria.

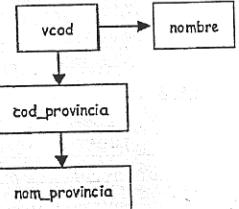
Es decir, en una relación en 3FN no hay dependencias funcionales entre atributos no-primos.

Problemas que resuelve

Sea el siguiente esquema de relación:

Proveedor(vcod: dom_vcod, nombre: dom_nom, nom_provincia: dom_nomp, cod_provincia: dom_codp)
Clave Primaria: {vcod}

en el que cada tupla almacena información del código, nombre y provincia de residencia de un proveedor y cuyo diagrama de dependencias funcionales es el siguiente:



Si se analiza el diagrama anterior, se puede observar que existe una dependencia funcional entre dos atributos no-primos, *cod_provincia* y *nom_provincia*¹⁸. Esta dependencia funcional induce la dependencia funcional $\{vcod\} \rightarrow \{nom_provincia\}$ de forma transitiva ya que se puede obtener con la regla de inferencia R3 a partir de las dependencias funcionales $\{vcod\} \rightarrow \{cod_provincia\}$ y $\{cod_provincia\} \rightarrow \{nom_provincia\}$.

¹⁸ Esta dependencia se detecta ya que las provincias tienen códigos únicos.

Los problemas de la manipulación de esta relación son causados por esa dependencia funcional transitiva que supone la presencia de redundancia, como se puede ver en la siguiente extensión de la relación *Proveedor*, ya que cada valor del atributo *cod_provincia* sólo puede tener asociado un valor del atributo *nom_provincia*:

vcod	nombre	cod_provincia	nom_provincia
V1	Pepe	46	Valencia
V2	Juan	12	Castellón
V3	Eva	46	Valencia

Debido a la redundancia inducida por esta dependencia, la manipulación de esta relación presenta los siguientes problemas:

- **Inserción:** no puede introducirse la información de cuál es el código de cada provincia hasta que no existe un proveedor en esa provincia
- **Borrado:** el borrado de un proveedor puede implicar implícitamente la pérdida de la información de provincia si el proveedor es el único en esa provincia.
- **Actualización:** la modificación del código de una provincia hay que realizarla tantas veces como proveedores viven en esa ciudad, ya que esa información es redundante y está replicada.

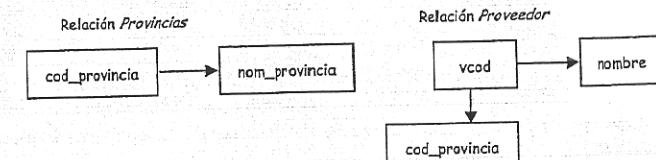
Paso a 3 FN

La técnica para transformar una relación que está en 2FN pero no en 3FN en un conjunto de relaciones que están en 3FN es la siguiente: supóngase que *R* es una relación con clave primaria *CP* que está en 2FN y no en 3FN. Supóngase que *B* depende funcionalmente de *A*. Las relaciones que se obtienen como proyección de *R* son $R1=R[CP, CA]$ (donde *C* es el resto de atributos no-primos de *R*) y $R2=R[A, B]$. La clave primaria de *R1* es *CP* y además *A* es clave ajena que hace referencia a *R2*. La clave primaria de *R2* es *A*. De esta forma se asegura que no existe pérdida de información en la descomposición.

Así, en el ejemplo en el paso de la relación *Proveedor* a 3FN se obtendrán dos relaciones:

Proveedor(vcod: dom_vcod, nombre: dom_nom, cod_provincia: dom_codp)
Clave Primaria: {vcod}
Clave Ajena: {cod_provincia} hace referencia a Provincia
Provincia(cod_provincia: dom_codp, nom_provincia: dom_nomp)
Clave Primaria: {cod_provincia}

El diagrama de dependencias funcionales de estas relaciones, en el que se puede observar que no hay dependencias transitivas, es el siguiente:



Las siguientes dos tablas muestran como queda la información:

vcod	nombre	cod_provincia
V1	Pepe	46
V2	Juan	12
V3	Eva	46

Provincia

cod_provincia	nom_provincia
46	Valencia
12	Castellón

Los problemas que existían en la manipulación de la relación original no aparecen en la manipulación de estas dos relaciones:

- Inserción: es posible realizar la inserción de una nueva información de una cierta provincia.
- Borrado: el borrado del único proveedor que vive en una provincia no implica la pérdida de la información que indica en qué provincia se encuentra esa ciudad.
- Actualización: el problema que existía con la actualización de la información ya no existe al no haber redundancia.

3.5 Tercera forma normal general¹⁹

Las relaciones en 3FN presentan ciertas anomalías cuando tienen más de una clave. Estas anomalías son:

- Relaciones que, a pesar de no estar en 3FN, no presentan redundancias.
- Relaciones que están en 3FN y presentan redundancias.

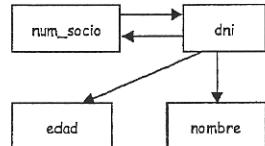
Como ejemplo del primer problema, sea el esquema de relación que almacena información sobre los socios de un gimnasio:

Socio(num_socio: dom_num, nombre: dom_nombre, edad: dom_edad, DNI:dom_dni)

Clave Primaria: {num_socio}

Único: {DNI}

Y sea su diagrama de dependencias funcionales el que se muestra:



Si se estudian sus dependencias funcionales, esta relación no está en 3FN (según la definición del apartado 3.4²⁰) ya que hay dependencias funcionales transitivas: el atributo {nombre} depende de forma transitiva de la clave principal de la relación ($\{num_socio\} \rightarrow \{DNI\}$ y de $\{DNI\} \rightarrow \{nombre\}$) se puede deducir por la regla R3 la dependencia $\{num_socio\} \rightarrow \{nombre\}$). Sin embargo, la transitividad aparece como consecuencia de la existencia de varias claves lo que no da lugar a problemas como los anteriormente comentados ya que en este caso no hay redundancia de información. Una posible extensión de esa relación sería la siguiente:

Socios

¹⁹ Existe una Segunda Forma Normal General cuya definición puede consultarse en el texto de Elmasri y Navathe. En estos apuntes no se ha incluido al no ser necesaria para la 3FN general.

²⁰ Formalmente, a esta relación no se le puede aplicar esa definición ya que tiene más de una clave. Se hace para destacar las situaciones que hay que corregir y que se resuelven con la 3FNG.

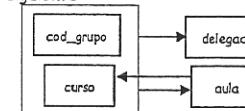
num_socio	Nombre	edad	DNI
S1	Pepe Pérez	33	17.897.569
S2	Juan Gómez	40	54.325.658
S3	Maria Bur	25	15.236.588

Considerando qué información se puede almacenar y las claves que tiene la relación puede apreciarse que no hay redundancia y que por lo tanto la relación, pese a no estar en 3FN, tiene buenas propiedades. Más adelante se generaliza la definición de 3FN para contemplar esta posibilidad.

Como ejemplo de la segunda anomalía, sea el siguiente esquema de relación:

Colegio(curso: dom_cur, cod_grupo: dom_gru, delegado: dom_nom, aula: dom_aula)
Clave Primaria: {curso, cod_grupo}²¹

en el que cada tupla representa la siguiente información: "el grupo de código cod_grupo del curso curso tiene como delegado al alumno de nombre delegado y recibe sus clases en el aula aula". Y supóngase que en este colegio, por cuestiones de equipamiento de aulas, se cumple la restricción de que en un aula determinada sólo se puede impartir clase a grupos de un mismo curso (i.e. {aula} → {curso}); el diagrama de dependencias funcionales es el siguiente:



Estudiando ese diagrama de dependencias se puede apreciar que esa relación está en 3FN, sin embargo, en ella puede aparecer redundancia como se ve en la siguiente extensión:

Colegio

Curso	Cod_grupo	Delegado	Aula
1	A	Juanito	2
1	B	Jorgito	2
1	C	Jaimito	4
2	A	Carmen	3

ya que cada vez que el atributo aula tome el valor 2, el atributo curso debe tomar el valor 1.

Esta redundancia es debida a que la relación tiene otra clave que es {aula, cod_grupo}. Esta clave se descubre si se comprueba que los demás atributos de la relación ({delegado, curso}) dependen funcionalmente de ese conjunto. Esto se muestra a continuación:

- ¿ {aula, cod_grupo} → {delegado}?

A partir de la dependencia funcional {aula} → {curso} y con la regla del aumento (R2) se obtiene $\{aula, cod_grupo\} \rightarrow \{curso, cod_grupo\}$; con esta dependencia funcional y con $\{curso, cod_grupo\} \rightarrow \{delegado\}$ por la regla transitiva (R3) se tiene que $\{aula, cod_grupo\} \rightarrow \{delegado\}$

- ¿ {aula, cod_grupo} → {curso}?

A partir de la dependencia funcional {aula} → {curso} y con la regla del aumento (R2) se obtiene

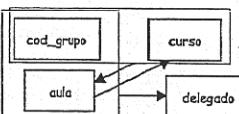
²¹ El hecho de que la clave primaria sea el par {curso, cod_grupo} destaca que puede haber dos grupos con el mismo código en distintos cursos (p.e. 1 A, 2 A...).

$\{aula, cod_grupo\} \rightarrow \{curso\}$.

Así pues, esta relación está en 3FN pero presenta redundancia. Podría pensarse entonces que si se considera la relación con la otra clave como clave primaria y se estudian sus dependencias funcionales quizás se encuentre una solución al problema. La relación quedaría entonces como sigue:

Colegio(curso: dom_cur, cod_grupo: dom_gru, delegado: dom_nom, aula: dom_aula)
Clave Primaria: {aula, cod_grupo}

y sus dependencias se muestran en el siguiente diagrama:



Esta relación no está en 2FN ya que {curso} no depende completamente de la clave principal. Si, pasamos a 2FN mediante proyecciones tal como se ilustró en el apartado 3.3 se obtienen las siguientes relaciones:

Colegio(aula: dom_aula, cod_grupo: dom_gru, delegado: dom_nom)
Clave Primaria: {aula, cod_grupo}
Clave Ajena: {aula} hace referencia a Clases
Clases(curso: dom_cur, aula: dom_aula)
Clave Primaria: {aula}

con lo que se elimina la redundancia como se ve en las extensiones:

Colegio		
Aula	Cod_grupo	Delegado
2	A	Juanito
2	B	Jorgito
4	C	Jaimito
3	A	Carmen

Clases	
Curso	Aula
1	2
1	4
2	3

pero se pierde información ya que no se representa la dependencia funcional $\{curso, cod_grupo\} \rightarrow \{aula\}$; es decir, se puede dar el caso de que un grupo de un curso recibe sus clases en más de un aula como se muestra en la siguiente extensión:

Colegio		
Aula	cod_grupo	Delegado
2	A	Juanito
4	A	Jorgito

Clases	
curso	aula
1	2
1	4

en la que se puede apreciar que el grupo A de primer curso recibe sus clases en las aulas 2 y 4.

Para controlar las dos situaciones ilustradas, se generaliza la definición de la 3FN que se aplica cuando una relación tiene más de una clave:

Definición general de la 3FNG²²

²² Esta definición puede aplicarse directamente para comprobar si una relación está en 3FN sin que sea necesario pasarlo primero a 2FN.

Una relación está en 3FNG si para toda dependencia $X \rightarrow A$ (donde A es un único atributo) se cumple una de las dos condiciones siguientes:

- 1) X contiene alguna clave, o
- 2) A es un atributo primo.

Si se aplica esta definición a las relaciones *Socio* y *Colegio* se puede apreciar que ambas están en 3FNG.

Para aclarar esta definición se puede considerar que la primera condición permite la presencia de cualquier dependencia funcional definida por una clave de la relación (en la parte izquierda) ya que como el valor de éstas no puede repetirse en ninguna extensión de la relación esas dependencias funcionales no introducen redundancia.

La segunda condición permite la presencia de ciertas dependencias funcionales (que no se permiten en formas normales más rigurosas) que pueden introducir problemas de redundancia pero que no se pueden resolver con la estrategia de proyecciones que se ha estado utilizando hasta ahora. Ante esta situación la 3FNG permite las dependencias funcionales transitivas cuando el miembro derecho de la misma forma parte de una clave; la redundancia que aparece se puede controlar con una restricción de integridad que, en el ejemplo que se está desarrollando, sería:

```

CREATE ASSERTION Aulas
CHECK NOT EXISTS (SELECT * FROM Colegio Cx, Colegio Cy
                  WHERE Cx.aula = Cy.aula AND Cx.curso ≠ Cy.curso)
  
```

Para comprobar que una relación está en 3FNG general no es necesario comprobar que esté en 2FN.

3.6 Resumen

Falta ahora incluir una guía de cómo utilizar la teoría de la normalización. Sea E un esquema relacional, cada relación R de E debe normalizarse como sigue:

- a) Comprobar que R está en 1FN. Si no lo está, transformar R a un conjunto $C1$ de relaciones que estén en 1FN.
- b) Obtener, para cada relación de $C1$, el conjunto de interés de dependencias funcionales y encontrar todas las claves.
- c) Comprobar si las relaciones de $C1$ con una sola clave están en 2FN. Si alguna relación no está en 2FN debe descomponerse en un conjunto de relaciones que estén en 2FN. Comprobar si las relaciones en 2FN están en 3FN aplicando la definición más sencilla, la que no lo esté deberá descomponerse.
- d) Comprobar si las relaciones de $C1$ con más de una clave están en 3FNG aplicando la definición más general. En caso de no estarlo deberá descomponerse en un conjunto de relaciones que sí lo estén.

3.7 Ejemplo

Sea la siguiente definición de dominios:

```

dom_numcli: entero;
dom_nif: cadena(9);
dom_nombre: cadena(50);
dom_dirección: cadena(30);
dom_teléfono: entero;
  
```

```

dom_ciudad: cadena(30);
dom_provincia: cadena(30);
dom_tarjeta: Registro n°tar: cadena(15),
    tipo_t: cadena(10),
    comisión: real,
    capital_límite: real;
dom_varias_tarjetas: conjunto de dom_tarjeta;
dom_cuenta: Registro n°cc: cadena(15),
    tipo_c: cadena(10),
    saldo: real,
    tarjetas: dom_varias_tarjetas;
dom_varias_cuentas: conjunto de dom_cuenta;

```

Y sea el siguiente esquema de relación:

```

Cliente(numcli: dom_numcli, nif: dom_nif, nombre: dom_nombre, dirección: dom_dirección,
nom_ciudad: dom_ciudad, provincia: dom_provincia, telefono: dom_telefono,
cuentas: dom_varias_cuentas)
CP: {numcli}
UNI: {nif}
VNN: {numcli, nif, nombre, dirección, telefono, nom_ciudad, provincia, cuentas}

```

Transforme a tercera forma normal la anterior relación teniendo en cuenta las siguientes restricciones:

- No hay dos ciudades con el mismo nombre y una ciudad está en una sola provincia.
- Una cuenta debe ser necesariamente de un cliente y sólo uno.
- Una cuenta se identifica por su *nºcc*, es de un tipo y tiene un saldo determinado.
- Una tarjeta se identifica por su *nºtar*, debe ser de un tipo, tiene una comisión, un capital límite y debe pertenecer a una sola cuenta.
- La comisión de una tarjeta depende del tipo de tarjeta.

A continuación se ilustra el paso a 3FN por etapas; en cada una de estas etapas las relaciones obtenidas se han destacado en negrita.

Paso a 1^a Forma Normal

Para resolver el problema derivado del dominio de tipo conjunto del atributo *cuentas* la relación *Cliente* se divide en dos:

```

Cliente(numcli: dom_numcli, nif: dom_nif, nombre: dom_nombre, dirección: dom_dirección,
telefono: dom_telefono, nom_ciudad: dom_ciudad, provincia: dom_provincia)
CP: {numcli}
UNI: {nif}
VNN: {numcli, nif, nombre, dirección, telefono, nom_ciudad, provincia}

```

```

Cuentas(numcli: dom_numcli, cuentas: dom_varias_cuentas)
    ↓ Eliminando el dominio de tipo conjunto y cambiando el nombre de la relación23.
Cuenta(numcli: dom_numcli, cuenta: dom_cuenta)
    ↓ Eliminando el dominio de tipo registro.

```

²³ Este cambio no es obligatorio pero sí aconsejable ya que en la relación *Cuenta* cada tupla va a almacenar la información de una cuenta corriente nada más.

```

Cuenta(numcli: dom_numcli, n°cc: cadena(15), tipo_c: cadena(10), saldo: real,
tarjetas: dom_varias_tarjetas)
    ↓ Buscando la Clave Principal.
Cuenta(numcli: dom_numcli, n°cc: cadena(15), tipo_c: cadena(10), saldo: real,
tarjetas: dom_varias_tarjetas)
CP: {n°cc}
CAj: {numcli} → Cliente
VNN: {numcli, tipo_c, saldo, tarjetas}

```

La relación *Cliente* está en 1FN ya que los dominios de sus atributos no son de tipo conjunto ni de tipo registro, sin embargo la relación *Cuenta* tiene un atributo de tipo conjunto que hay que eliminar por lo que se divide en dos relaciones.

```

Cuenta(numcli: dom_numcli, n°cc: cadena(15), tipo_c: cadena(10), saldo: real)
CP: {n°cc}
CAj: {numcli} → Cliente
VNN: {numcli, tipo_c, saldo}
Tarjetas(n°cc: cadena(15), tarjetas: dom_varias_tarjetas)
    ↓ Eliminando el dominio de tipo conjunto y cambiando el nombre de la relación.
Tarjeta(n°cc: cadena(15), n°tar: cadena(15), tipo_t: cadena(10), comisión: real, capital_límite: real)
    ↓ Buscando la Clave Principal.
Tarjeta(n°cc: cadena(15), n°tar: cadena(15), tipo_t: cadena(10), comisión: real, capital_límite: real)
CP: {n°tar}
CAj: {n°cc} → Cuenta
VNN: {n°cc, tipo_t, comisión, capital_límite}

```

Las relaciones *Cuenta* y *Tarjeta* ya están en 1FN.

Paso a 2^a Forma Normal

Ninguna relación de las obtenidas en el paso a 1FN tiene una clave compuesta por lo que todas ellas (*Cliente*, *Cuenta* y *Tarjeta*) están en 2FN.

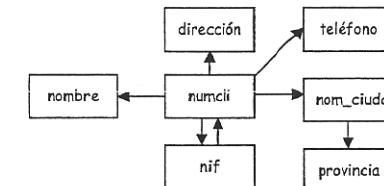
Paso a 3^a Forma Normal

```

Cliente(numcli: dom_numcli, nif: dom_nif, nombre: dom_nombre, dirección: dom_dirección,
telefono: dom_telefono, nom_ciudad: dom_ciudad, provincia: dom_provincia)
CP: {numcli}
UNI: {nif}
VNN: {numcli, nif, nombre, dirección, telefono, nom_ciudad, provincia}

```

El diagrama de dependencias funcionales de esta relación es el siguiente:



Esta relación tiene dos claves así que hay que aplicar la definición de 3FN General; la única dependencia funcional no permitida es la que existe entre *nom_ciudad* y *provincia*. Para resolver el problema se divide la relación en dos:

```

Cliente(numcli: dom_numcli, nif: dom_nif, nombre: dom_nombre, dirección: dom_dirección,

```

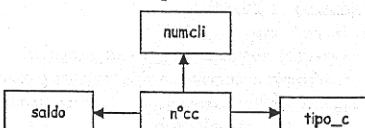
teléfono: dom_telefono, nom_ciudad: dom_ciu
 CP: {numcli}
 UNI: {nif}
 VNN: {numcli, nif, nombre, dirección, telefono, nom_ciudad}
 CAj: {nom_ciudad} → Ciudad
 Ciudad(nom_ciudad: dom_ciu, provincia: dom_prov)
 CP: {nom_ciudad}
 VNN: {provincia}

En cuanto a la relación Cuenta:

Cuenta(numcli: dom_numcli, n°cc: cadena(15), tipo_c: cadena(10), saldo: real)
 CP: {n°cc}
 CAj: {numcli} → Cliente

VNN: {numcli, tipo_c, saldo}

el diagrama de dependencias funcionales es el siguiente:



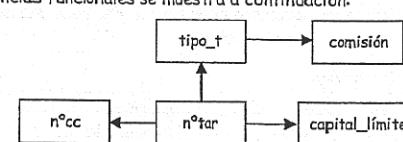
En él se puede apreciar que todas las dependencias funcionales son admisibles.

Por último, la relación Tarjeta:

Tarjeta(n°cc: cadena(15), n°tar: cadena(15), tipo_t: cadena(10), comisión: real, capital_límite: real)
 CP: {n°tar}
 CAj: {n°cc} → Cuenta

VNN: {n°cc, tipo_t, comisión, capital_límite}

El diagrama de dependencias funcionales se muestra a continuación:



La dependencia funcional entre tipo_t y comisión no es admisible; para resolver el problema se divide la relación en dos:

Tarjeta(n°cc: cadena(15), n°tar: cadena(15), tipo_t: cadena(10), capital_límite: real)
 CP: {n°tar}
 CAj: {n°cc} → Cuenta

CAj: {tipo_t} → Comisión

VNN: {n°cc, tipo_t, capital_límite}

Comisión(tipo_t: cadena(10), comisión: real)

CP: {tipo_t}

VNN: {comisión}

Así pues el conjunto de relaciones definitivo en 3FN es el siguiente:

Cliente(numcli: dom_numcli, nif: dom_nif, nombre: dom_nombre, dirección: dom_dirección,
 telefono: dom_telefono, nom_ciudad: dom_ciu)
 CP: {numcli}

UNI: {nif}
 VNN: {numcli, nif, nombre, dirección, telefono, nom_ciudad}
 CAj: {nom_ciudad} → Ciudad

Ciudad(nom_ciudad: dom_ciu, provincia: dom_prov)
 CP: {nom_ciudad}
 VNN: {provincia}

Cuenta(numcli: dom_numcli, n°cc: cadena(15), tipo_c: cadena(10), saldo: real)
 CP: {n°cc}

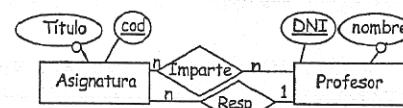
CAj: {numcli} → Cliente
 VNN: {numcli, tipo_c, saldo}

Tarjeta(n°cc: cadena(15), n°tar: cadena(15), tipo_t: cadena(10), capital_límite: real)
 CP: {n°tar}
 CAj: {n°cc} → Cuenta
 CAj: {tipo_t} → Comisión
 VNN: {n°cc, tipo_t, capital_límite}

Comisión(tipo_t: cadena(10), comisión: real)
 CP: {tipo_t}
 VNN: {comisión}

4 TRANSFORMACIÓN DE LAS RESTRICCIONES DE INTEGRIDAD

Otra parte importante del esquema conceptual además del diagrama entidad-relación es el conjunto de restricciones de integridad que completan el diagrama entidad-relación para que se ajuste a la realidad que representa. Evidentemente, estas restricciones de integridad deben trasladarse también al esquema lógico (relacional) obtenido después de aplicar la Teoría de la Normalización. En la mayoría de los casos no se pueden integrar directamente en la definición de las relaciones y deben ser traducidas a expresiones en SQL; para ello hay que determinar en qué relaciones se representan todos los objetos del diagrama ER que se ven afectados por cada restricción, y formular la misma restricción sobre dichas relaciones. A continuación se incluye un ejemplo:



Restricción de integridad: "Todo responsable de una asignatura ha de impartirla."

El esquema relacional equivalente es el siguiente:

Asignatura(cod: dom_cod, Título: dom_Título, DNI: dom_DNI)

Clave Primaria: {cod}

Clave Ajena: {DNI} hace referencia a Profesor

Profesor(DNI: dom_DNI, nombre: dom_nombre)

Clave Primaria: {DNI}

Imparte(cod: dom_cod, DNI: dom_DNI)

Clave Primaria: {cod, DNI}

Clave Ajena: {cod} hace referencia a Asignatura

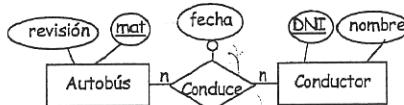
Clave Ajena: {DNI} hace referencia a Profesor

Restricción de integridad

```
CREATE ASSERTION Responsable-Imparte
  CHECK NOT EXISTS (SELECT * FROM Asignatura Ax
    WHERE Ax.DNI IS NOT NULL AND
      NOT EXISTS (SELECT * FROM Imparte Iy
        WHERE Iy.DNI = Ax.DNI AND Iy.cod = Ax.cod))
```

Como puede apreciarse, debido a que la relación *Resp* del esquema ER se representa en el esquema lógico sobre la relación *Asignatura*, la restricción en SQL afecta a esta última relación.

Sin embargo, hay algunos casos excepcionales en los que esas restricciones pueden integrarse en el esquema relacional como se ilustra a continuación. Sea el siguiente esquema conceptual:



Restricción de integridad: "Un conductor no puede conducir dos autobuses en la misma fecha."

El esquema relacional equivalente es el siguiente:

Autobús(mat; dom_mat, revisión; dom_revisión)

Clave Primaria: {mat}

Conductor(DNI; dom_DNI, nombre; dom_nombre)

Clave Primaria: {DNI}

Conduce(mat; dom_mat, DNI; dom_DNI, fecha; dom_fecha)

Clave Primaria: {mat, DNI}

Único: {DNI, fecha}

VNN: {fecha}

Clave Ajena: {mat} hace referencia a Autobús

Clave Ajena: {DNI} hace referencia a Conductor

En él puede observarse que la restricción se ha integrado en la propia definición de las relaciones mediante la especificación de una restricción de unicidad para el conjunto de atributos {DNI, fecha}.

En resumen, en esta fase se debe estudiar el conjunto de restricciones de integridad para integrarlas, si es posible, en la propia definición de las relaciones; las que no se puedan integrar deben traducirse a SQL.

5 REPRESENTACIÓN DE LAS TRANSACCIONES

Como última tarea del diseño lógico, es necesario abordar el diseño de los aspectos dinámicos sobre el modelo relacional. Este aspecto se ha abordado de manera muy superficial en el modelo conceptual y se ha limitado a un análisis de las transacciones y a la especificación de un esquema simplificado de las mismas.

Para abordar el diseño de transacciones sobre el esquema relacional se partirá del análisis de transacciones realizado sobre el esquema conceptual, para así, poder obtener y especificar un nuevo conjunto de transacciones sobre el esquema relacional.

Al igual que se ha hecho en la transformación de la parte estática, se supondrá que el sistema de gestión de bases de datos soporta la definición de claves primarias, claves ajenas, restricciones de unicidad, restricciones de valor no nulo sobre atributos y demás restricciones expresadas con la cláusula CREATE ASSERTION.

El método que se seguirá para la transformación de un esquema de transacción, definido en el esquema conceptual, a una transacción sobre el esquema lógico se realizará en tres pasos:

- 1) Investigar la transacción para concretar sobre qué objetos (entidades o relaciones) actúa y con qué operación (Inserción, borrado o modificación).
- 2) Determinar en qué relaciones del esquema relacional se encuentran representados esos objetos y qué operación debe utilizarse.
- 3) Analizar el esquema simplificado de transacciones del modelo conceptual y transformarlo en un conjunto de transacciones sobre las correspondientes relaciones del esquema relacional.

A continuación se presenta el lenguaje transaccional para la especificación de las transacciones sobre el esquema relacional. El conjunto de operaciones básicas permitidas será:

```
operación_básica ::= sentencia_insert |  
                     sentencia_delete |  
                     sentencia_update
```

utilizando para ello las operaciones del SQL estándar.

La sintaxis de definición de una transacción del esquema lógico que se utilizará es la siguiente:

```
TRANSACCIÓN nombre_transacción (parámetro: tipo, ...)  
INICIO_TRANSACCIÓN
```

/*conjunto de operaciones básicas que componen la transacción expresadas
como una secuencia, y en las que se pueden utilizar mecanismos de iteración,
selección, diálogo con el usuario, mensaje, interrupción de la transacción, ...*/

```
FIN_TRANSACCIÓN
```

donde *tipo* puede ser un dominio o tipo de datos básico del SQL y *variable* es un nombre de variable que se utiliza en la transacción.

Se puede combinar un conjunto de instrucciones de las formas que se muestran²⁴:

- **Secuencia:** instrucciones separadas por ";"
*instrucción1; ...; instrucción n*²⁵;
- **Iteración:**
 - a) Iterar una misma operación para todos los elementos de un conjunto.
Para cada valor de conjunto hacer instrucción 1; ...; instrucción n fin_for
 - b) Iterar mientras se cumpla una condición.
 - 1) Mientras condición hacer instrucción 1;...; instrucción n fin_mientras
 - 2) Repetir instrucción 1;..., instrucción n hasta condición
- **Selección:**
 - 1) Si condición entonces instrucción 1;...; instrucción n

²⁴ No se ha considerado necesario definir rigurosamente la sintaxis de las transacciones ya que como puede verse el lenguaje es un pseudocódigo fácil de leer

²⁵ Utilizamos *n* para representar un valor indefinido que puede ser distinto en cada caso.

[Si no instrucción 1;...; instrucción n] fin_si

2) Caso término_dominio_escalar

valor 1: instrucción 1;...; instrucción n;

...

valor n: instrucción 1;...; instrucción n

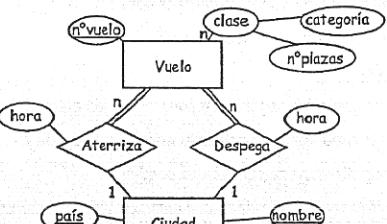
fin_caso

Además de las operaciones básicas presentadas, en las transacciones también podrán aparecer las siguientes instrucciones:

- Lectura: leer(variable)
- Mensajes hacia el usuario: Mensaje ("texto del MENSAJE")
- Anulación de una transacción: Anular: acaba la transacción sin que provoque cambios en ninguna estructura.
- Confirmación de una transacción: Confirmar: da como buenos todos los cambios realizados por la transacción.
- Comentarios: /*comentario ... */
- Llamadas a otras transacciones.

Para poder consultar el resultado de la ejecución de una transacción, se va a suponer la existencia de una variable del sistema de nombre *error* donde el sistema pueda dejar un valor que indique cómo ha terminado la transacción. Para simplificar se va a considerar que este parámetro puede tomar los siguientes valores: *error = -1* que indica que la transacción ha violado alguna restricción de integridad y *error = n (n ≥ 0)* que indica que la transacción ha finalizado sin problemas (con éxito) y que se han visto afectadas *n* tuplas. Esta variable se podrá consultar en las transacciones permitiendo al diseñador anular o confirmar la transacción.

Para ilustrar estas transformaciones, se va a hacer uso del siguiente diagrama entidad-relación que representa la información de los vuelos entre ciudades:



RI: Para un vuelo determinado las categorías para dicho vuelo no se pueden repetir

Sea también la transacción que permite dar de alta un nuevo vuelo:

Transacción para insertar en vuelo

- Insertar en Vuelo
 - ⇒ Insertar en Aterrizada
 - ⇒ Insertar en Despegada

El esquema relacional equivalente se obtiene siguiendo las directrices presentadas en el tema:

Ciudad(nombre: dom_nom, país: dom_pa)

Clave Primaria: (nombre, país)

Vuelo(n°_vuelo: dom_nº, hora_des: dom_hora, ciu_des: dom_nom, país_des: dom_pa, hora_ate: dom_hora, ciu_ate: dom_nom, país_ate: dom_pa)

Clave Primaria: (n°_vuelo)

Clave Ajena: (ciu_des, país_des) hace referencia a Ciudad

f(ciu_des) = nombre

f(país_des) = país

Valor No Nulo: (ciu_des, país_des)

Clave Ajena: (ciu_ate, país_ate) hace referencia a Ciudad

f(ciu_ate) = nombre

f(país_ate) = país

Valor No Nulo: (ciu_ate, país_ate)

Vuelo_Clase(n°_vuelo: dom_nº, categoría: dom_cat, n°plazas: dom_num)

Clave Primaria: {n°_vuelo, categoría}

Clave Ajena: { n°_vuelo } hace referencia a Vuelo.

La transacción se puede escribir en términos lógicos substituyendo cada una de las tres operaciones por una operación sobre el esquema lógico; además, hay que incluir un bucle de repetición que permita asignar las plazas que hay de cada clase (ya que esa información se guarda en otra relación). Para ello hay que tener en cuenta que las relaciones *Aterrizada* y *Despegada* se han representado junto con la entidad *Vuelo* y que por tanto las operaciones de inserción se convierten en operaciones de modificación. Una primera versión de la transacción sería la siguiente:

```

TRANSAKCION Alta_Vuelo(nx: dom_nº,hdx: dom_hora, cdx: dom_nom, pdx: dom_pa,
                      hax: dom_hora, cax: dom_nom, pax: dom_pa, [catx: dom_cat, npx: dom_num])
INICIO_TRANSACCION
  INSERT INTO VUELO (n°_vuelo) VALUES (nx)
  UPDATE VUELO SET ciu_des = cdx, país_des = pdx, hora_des = hdx
  WHERE n°_vuelo = nx
  UPDATE VUELO SET ciu_ate = cax, país_ate = pax, hora_ate = hax
  WHERE n°_vuelo = nx
  Para cada valor del conjunto {catx, npx} hacer
    INSERT INTO VUELO_CLASE VALUES (nx, catx, npx)
FIN_TRANSACCION
  
```

Aunque, si se observa con atención existe otra que condensa las tres primeras operaciones en una sola quedando la transacción como sigue:

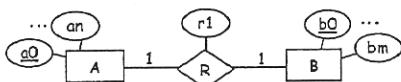
```

TRANSAKCION Alta_Vuelo(nx: dom_nº,hdx: dom_hora, cdx: dom_nom, pdx: dom_pa, hax:
                      dom_hora,
                      cax: dom_nom, pax: dom_pa, [catx: dom_cat, npx: dom_num])
INICIO_TRANSACCION
  INSERT INTO VUELO VALUES (nx, hdx, cdx, pdx, hax, cax, pax)
  Para cada valor del conjunto {catx, npx} hacer
    INSERT INTO VUELO_CLASE VALUES (nx, catx, npx)
FIN_TRANSACCION
  
```

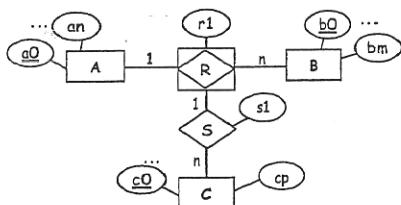
Con la representación de las transacciones con este lenguaje termina la fase de diseño lógico.

6 CUESTIONES SOBRE DISEÑO LÓGICO

1. Suponiendo un Sistema de Gestión de Bases de Datos (SGBD) en el que la definición de una restricción de unicidad implice también la restricción de valor no nulo, ¿cuál sería el esquema relacional más adecuado para el siguiente diagrama entidad-relación? Justifique la respuesta.

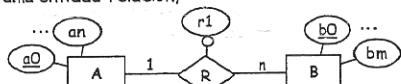


2. ¿Cuál sería el esquema relacional más adecuado para el siguiente diagrama entidad-relación?



Justifique la respuesta.

3. Dado el siguiente diagrama entidad-relación,



Y los siguientes esquemas relationales:

ESQUEMA 1:

$A(a_0: \text{dom_}a_0, \dots, a_n: \text{dom_}a_n)$
 $CP: \{a_0\}$
 $B(b_0: \text{dom_}b_0, \dots, b_m: \text{dom_}b_m, a_0: \text{dom_}a_0, r_1: \text{dom_}r_1)$
 $CP: \{b_0\}$
 $CA_j: \{a_0\} \rightarrow A$

ESQUEMA 2:

$A(a_0: \text{dom_}a_0, \dots, a_n: \text{dom_}a_n)$
 $CP: \{a_0\}$
 $B(b_0: \text{dom_}b_0, \dots, b_m: \text{dom_}b_m, a_0: \text{dom_}a_0, r_1: \text{dom_}r_1)$
 $CP: \{b_0\}$
 $CA_j: \{a_0\} \rightarrow A$
 $RI: \forall Bx (B(Bx) \wedge \neg \text{nulo}(Bx.a_0) \rightarrow \neg \text{nulo}(Bx.r_1))$

ESQUEMA 3:

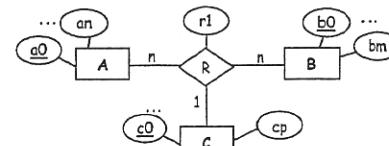
$A(a_0: \text{dom_}a_0, \dots, a_n: \text{dom_}a_n)$
 $CP: \{a_0\}$
 $B(b_0: \text{dom_}b_0, \dots, b_m: \text{dom_}b_m, a_0: \text{dom_}a_0, r_1: \text{dom_}r_1)$
 $CP: \{b_0\}$
 $CA_j: \{a_0\} \rightarrow A$
 $VNN: \{r_1\}$

ESQUEMA 4:

$A(a_0: \text{dom_}a_0, \dots, a_n: \text{dom_}a_n)$
 $CP: \{a_0\}$
 $B(b_0: \text{dom_}b_0, \dots, b_m: \text{dom_}b_m)$
 $CP: \{b_0\}$
 $R(b_0: \text{dom_}b_0, a_0: \text{dom_}a_0, r_1: \text{dom_}r_1)$
 $CP: \{b_0\}$
 $CA_j: \{a_0\} \rightarrow A$
 $CA_j: \{b_0\} \rightarrow B$
 $VNN: \{a_0\}$
 $VNN: \{r_1\}$

¿Son los cuatro esquemas anteriores transformaciones correctas del diagrama entidad-relación?. De los que considere correctos, ¿cuál es el más adecuado?. Justifique sus respuestas.

4. Sea el siguiente diagrama entidad-relación:

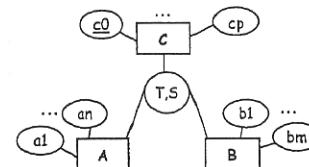


En cuyo esquema relacional equivalente, la relación R es:

$R(a_0: \text{dom_}a_0, b_0: \text{dom_}b_0, c_0: \text{dom_}c_0, r_1: \text{dom_}r_1)$
 $CP: \{a_0, b_0\}$
 $CA_j: \{a_0\} \rightarrow A$
 $CA_j: \{b_0\} \rightarrow B$
 $CA_j: \{c_0\} \rightarrow C$
 $VNN: \{c_0\}$

¿Es realmente necesario definir el atributo c_0 con la propiedad de valor no nulo?. Justifique la respuesta.

5. Sea el siguiente diagrama entidad-relación:



Y el siguiente esquema relacional:

$C(c_0: \text{dom_}c_0, \dots, c_p: \text{dom_}c_p)$
 $CP: \{c_0\}$
 $A(c_0: \text{dom_}c_0, a_1: \text{dom_}a_1, \dots, a_n: \text{dom_}a_n)$
 $CP: \{c_0\}$
 $CA_j: \{c_0\} \rightarrow C$
 $B(c_0: \text{dom_}c_0, b_1: \text{dom_}b_1, \dots, b_m: \text{dom_}b_m)$
 $CP: \{c_0\}$
 $CA_j: \{c_0\} \rightarrow C$

- ¿Queda representado en este esquema el hecho de que la generalización es solapada?. Justifique la respuesta.
- ¿Por qué no queda expresada la propiedad de totalidad en el esquema relacional? ¿Se resolvería el problema cambiando la relación C por la que se muestra?

$C(c_0: \text{dom_}c_0, \dots, c_p: \text{dom_}c_p)$

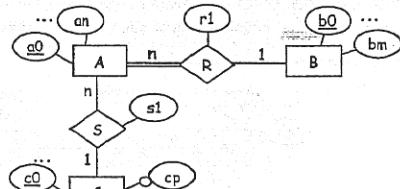
$CP: \{c_0\}$

$CA_1: \{c_0\} \rightarrow A$

$CA_2: \{c_0\} \rightarrow B$

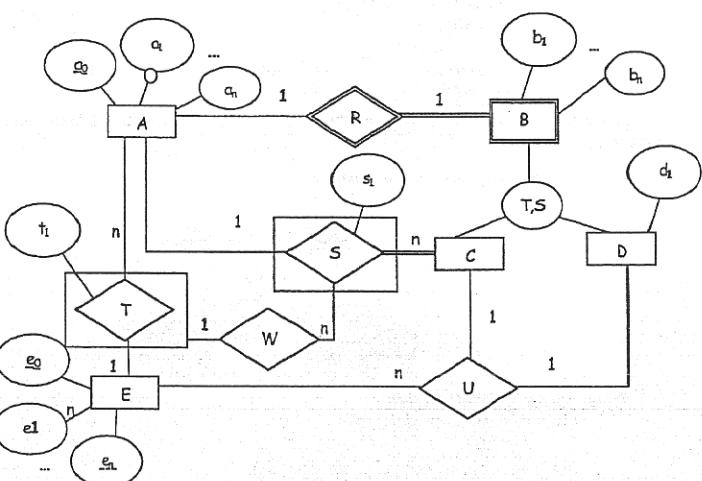
Justifique la respuesta.

6. Obtenga el esquema relacional equivalente al siguiente diagrama entidad-relación:



7. Sobre el esquema definido en la cuestión 6, escriba la transacción que permitiría insertar una nueva tupla en la relación A.

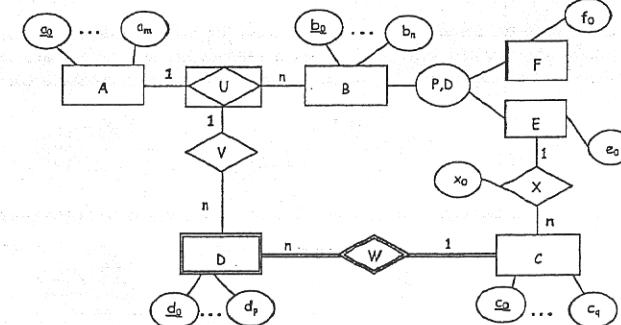
8. Obtenga el esquema relacional equivalente al siguiente diagrama entidad-relación, incluyendo las restricciones necesarias:



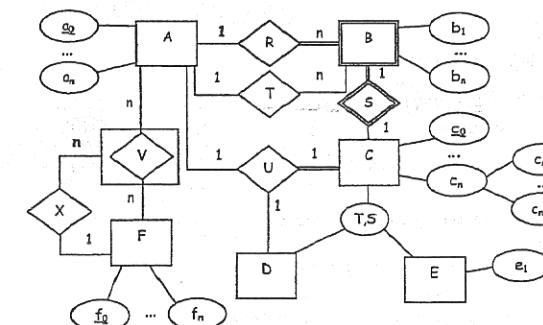
9. Sobre el esquema definido en la cuestión 8, escriba la transacción de borrado de C, que sea restrictivo respecto a U y en cascada respecto al resto.

10. A partir del siguiente diagrama entidad-relación, realice el diseño lógico, obteniendo un conjunto

de relaciones en tercera forma normal incluyendo las restricciones necesarias.



11. Realice el diseño lógico, obteniendo un conjunto de relaciones en tercera forma normal, incluyendo las restricciones necesarias.



12. Para cada uno de los siguientes esquemas relationales, obtenga todos los diagramas entidad-relación que sean equivalentes²⁶. (NOTA: Las soluciones no debe incluir restricciones en lenguaje natural, es decir todo lo que se especifica en el esquema relacional, debe expresarse gráficamente.)

Esquema 1

$A(a_0: \text{dom_}a_0, a_1: \text{dom_}a_1)$

$CP: \{a_0\}$

$B(b_0: \text{dom_}b_0, b_1: \text{dom_}b_1, a_0: \text{dom_}a_0, c_0: \text{dom_}c_0)$

$CP: \{b_0\}$

$\text{Uni: } \{a_0\}$

$CA_1: \{a_0\} \rightarrow A$

$CA_2: \{c_0\} \rightarrow C$

²⁶ Entendiendo por equivalentes que su transformación al modelo relacional generaría el esquema que se presenta.

$C(c0: \text{dom_c}0, c1: \text{dom_c}1)$
 $CP: \{c0\}$

$M(c0: \text{dom_c}0, m: \text{dom_m}, n: \text{dom_n})$
 $CP: \{c0, m\}$
 $CA_j: \{c0\} \rightarrow C$

R.I.: Toda ocurrencia de B , si se relaciona con C también lo hace con A .

Esquema 2

$A(a0: \text{dom_a}0, a1: \text{dom_a}1, b0: \text{dom_b}0)$
 $CP: \{a0\}$

$CA_j: \{b0\} \rightarrow B$
 $VNN: \{b0\}$
 $Uni: \{b0\}$

$B(b0: \text{dom_b}0, b1: \text{dom_b}1)$
 $CP: \{b0\}$

$R(b0: \text{dom_b}0, a0: \text{dom_a}0)$
 $CP: \{b0, a0\}$
 $CA_j: \{a0\} \rightarrow A$
 $CA_j: \{b0\} \rightarrow B$

R.I.: Toda ocurrencia de A debe participar en la relación R .

Esquema 3

$A(a0: \text{dom_a}0, a1: \text{dom_a}1, a2: \text{dom_a}2, a3: \text{dom_a}3)$
 $CP: \{a0\}$

$B(b0: \text{dom_b}0, b1: \text{dom_b}1, a0: \text{dom_a}0, c0: \text{dom_c}0)$
 $CP: \{b0\}$
 $CA_j: \{a0\} \rightarrow A$
 $CA_j: \{c0\} \rightarrow R$

$C(c0: \text{dom_c}0, c1: \text{dom_c}1, c2: \text{dom_c}2)$
 $CP: \{c0\}$

$R(b0: \text{dom_b}0, a0: \text{dom_a}0, c0: \text{dom_c}0)$
 $CP: \{c0\}$
 $Uni: \{a0\}$
 $VNN: \{a0\}$
 $CA_j: \{b0\} \rightarrow B$
 $CA_j: \{a0\} \rightarrow A$
 $CA_j: \{c0\} \rightarrow C$

Esquema 4

$A(a0: \text{dom_a}0, a1: \text{dom_a}1)$
 $CP: \{a0\}$

$B(b0: \text{dom_a}0, b1: \text{dom_b}1, b2: \text{dom_b}2, b3: \text{dom_a}0)$
 $CP: \{b0\}$
 $CA_j: \{b0\} \rightarrow A$
 $CA_j: \{b3\} \rightarrow A$

$C(c0: \text{dom_c}0, c1: \text{dom_c}1)$

$CP: \{c0\}$
 $CA_j: \{c0\} \rightarrow C$

$D(d0: \text{dom_d}0, d1: \text{dom_d}1, d2: \text{dom_a}0)$

$CP: \{d0, d2\}$
 $CA_j: \{d2\} \rightarrow C$

$E(e0: \text{dom_e}0, e1: \text{dom_e}1)$

$CP: \{e0, e1\}$
 $CA_j: \{e0, e1\} \rightarrow E$
 $CA_j: \{e0\} \rightarrow A$

R.I.: Las ocurrencias de B no pueden pertenecer a C y viceversa.

13. Sea el siguiente esquema relacional:

$R(a: \text{entero}, b: \text{carácter}, c: \text{real})$
 $CP: \{a, b\}$

donde la relación R no está en 2ª Forma Normal ya que c depende funcionalmente de a . Describa los problemas de manipulación que tiene dicha relación.

14. Para cada uno de los siguientes diagramas entidad-relación, realice el diseño lógico, obteniendo un conjunto de relaciones en tercera forma normal incluyendo las restricciones necesarias.

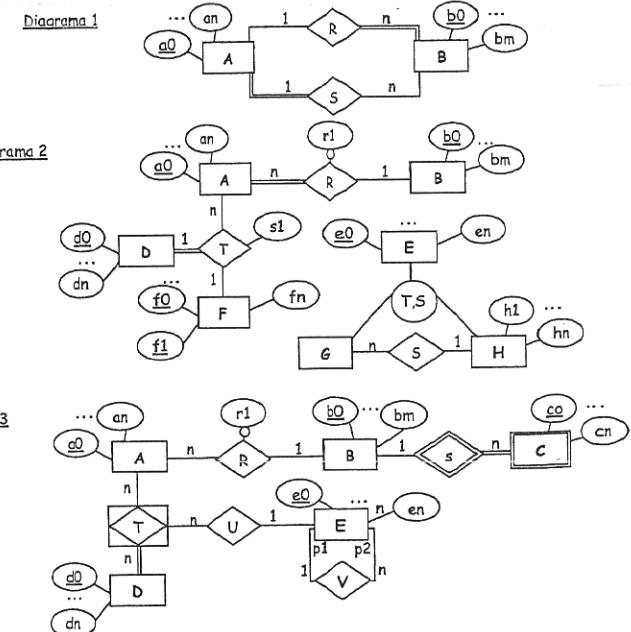


Diagrama 4

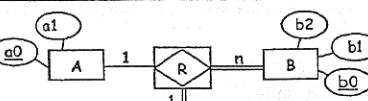
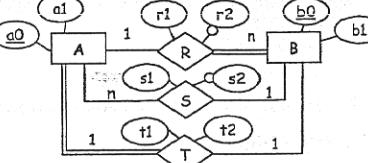
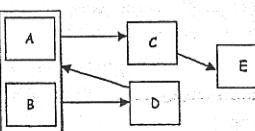


Diagrama 5

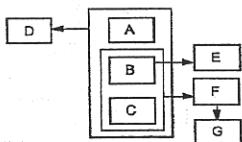


15. Dada una relación R con atributos A, B, C, D, E todos no nulos, y dado el siguiente conjunto de dependencias funcionales:



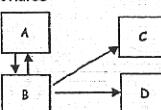
Obtenga un conjunto equivalente de relaciones en tercera forma normal.

16. Sea R la siguiente relación: $R(A:\text{dom_A}, B:\text{dom_B}, C:\text{dom_C}, D:\text{dom_D}, E:\text{dom_E}, F:\text{dom_F}, G:\text{dom_G})$ entre cuyos atributos existen las dependencias funcionales que se muestran en el siguiente diagrama:



Obtenga un esquema relacional equivalente en 3^a Forma Normal.

17. Dados los atributos A, B, C, D , suponiendo que todos tienen restricción de valor no nulo, y dado el siguiente conjunto de dependencias funcionales:



Proponga un esquema en 3FN, con el número mínimo de relaciones, explicando el porqué de la solución.

18. Sea la siguiente definición de dominios:

`dom_resp: entero;`
`dom_nom: cadena(20);`

`dom_cod_ciudad: entero;`
`dom_nivel: 1..10;`
`dom_destino: Conjunto de dom_ciudad;`
`dom_ciudad: Registro`
`cod_ciudad: dom_cod_ciudad;`
`nombre: dom_nombre`
`nivel: dom_nivel`

Y sea el siguiente esquema de relación:

`Representante(cod_resp: dom_resp, nombre: dom_nombre, destino: dom_destino)`
`CP: {cod_resp}`

Normalizar la relación anterior sabiendo que cada ciudad tiene asignada un único representante y que no hay dos ciudades con el mismo código.

19. Sea la siguiente definición de dominios.

`dom_cod_disco: cadena (10);`
`dom_título: cadena (15);`
`dom_año: cadena (4);`
`dom_cod_casa_discos: cadena (20);`
`dom_dir_casa_discos: cadena (20);`
`intérprete: Registro`
`nombre: cadena (15),`
`nación: cadena (10);`
`dom_intérprete: Conjunto de intérprete;`
`canción: Registro`
`código: cadena (10),`
`nombre: cadena (20),`
`estilo: cadena (15),`
`duración: real,`
`intérpretes: dom_intérprete;`
`dom_canción: Conjunto de canción;`

Y sea el siguiente esquema de relación:

`Disco(cod_disco: dom_cod_disco, título: dom_título, año: dom_año,`
`cod_casa_discos: dom_cod_casa_discos, dir_casa_discos: dom_dir_casa_discos,`
`canciones: dom_canción)`
`CP: {cod_disco}`

Teniendo en cuenta las restricciones que se exponen, transfórmela a tercera forma normal.

- No existen dos intérpretes con el mismo nombre.
- Un intérprete sólo tiene una nación.
- Una canción puede estar en más de un disco. El nombre y el estilo dependen del código de la canción, mientras que la duración e intérpretes dependen del disco en que esté cada canción.
- Un intérprete puede tener varias canciones pero al menos debe tener una.
- Una casa de discos sólo tiene una dirección.
- El título del disco depende del código de disco.

20. Sea la siguiente definición de dominios:

`dom_cod_pelicula: cadena (5);`
`dom_salas: Conjunto de registros de`
`(nombre: cadena (20),`

dirección: cadena (40),
 capacidad: entero,
 sesiones: conjunto de sesión: dom_sesión
 semanas: entero)
 dom_año: entero
 dom_sesión: cadena (5);
 dom_título: cadena(50);
 dom_director: cadena (30);
 dom_productor: cadena (30);
 dom_duración: cadena (5);
 dom_actor: Conjunto de registros de
 (nombre: cadena(30),
 nacionalidad: cadena (10),
 papel: cadena (20));

Y sea el siguiente esquema de relación

Película(cod_película: dom_cod_película, salas: dom_salas, título: dom_título, año: dom_año,
 director: dom_director, productor: dom_productor, duración: dom_duración,
 actores: dom_actor)
 CP: {cod_película}

donde *semanas* indica el número de semanas que permanece cada película en cada sala y teniendo en cuenta las restricciones que se exponen, transfórmela a tercera forma normal.

- Todos los atributos tienen restricción de valor no nulo.
- Los actores se identifican por el nombre.
- La nacionalidad del actor depende de su nombre.
- Las salas se identifican por su nombre.
- La dirección y capacidad de una sala dependen de su nombre.
- En una sala se pueden exponer varias películas en diferentes sesiones y cada película se puede pasar en una o varias sesiones.
- Un actor puede aparecer en varias películas realizando un papel en cada una de ellas.

21. Normalizar la relación *R* a 3FN teniendo en cuenta las restricciones y dependencias funcionales que se muestran.

domA: char(10);
 domB: char(10);
 domC: char(10);
 domD: registro de
 {D1: char(10),
 D2: char(10)}
 R(A: domA, B: domB, C: domC, D: domD, E: domE)
 CP: {A}

domE: conjunto de registro de
 {E1: char(10);
 E2: char(10);
 E3: char(10);}

Todos los atributos tienen restricción de valor no nulo.
 Dependencias funcionales
 {C} → {A}
 {E1} → {A}, {E1} → {E2}, {E1} → {E3}, {E2} → {E3}, {E2} → {E1}
 {B} → {D}

22. En la planta de maternidad de un Hospital se controla la ocupación de las habitaciones por parte de

la madre y de sus bebé/s mediante la estructura Habitación. Normalizarla a 3FN teniendo en cuenta las restricciones que se acompañan.

Habitación(número: dom_numero, dni_madre: dom_dni, nom_madre: dom_nom,
 fecha-ingreso: dom_fecha, medicación_madre: dom_medicación, cunas: dom_cunas)

CP: {número}
 UNI: {dni_madre}
 VNN: {dni_madre, nom_madre, fecha-ingreso}

dom_numero: cadena (5)
 dom_dni: cadena(10);
 dom_nom: cadena(50);
 dom_fecha: cadena(8);
 dom_hora: cadena(5);
 dom_nºcuna: entero;
 dom_medicación: Conjunto de registro de (código: dom_numero,

descripción: dom_nom,
 dosis: dom_nom);

dom_cunas: conjunto de registro de (nºcuna: dom_nºcuna,
 nombre_bebe: dom_nom,
 fecha_nacimiento: dom_fecha,
 hora: dom_hora,
 tipo_alimentación: dom_nom,
 observaciones: dom_nom);

Restricciones:

- La dosis de un medicamento puede ser diferente para cada madre.
- Las cunas se identifican por el nº cuna y el número de habitación.
- Los medicamentos se identifican por el código, del cuál depende funcionalmente la descripción.
- Todo bebé debe tener una y solamente una madre.

23. La Inspección Técnica de un Vehículo se controla mediante la estructura que se muestra. Normalizarla a 3FN teniendo en cuenta las restricciones que se acompañan.

Inspección_técnica(matrícula: dom_matrícula, fecha: dom_fecha, marca: dom_marca,
 modelo: dom_modelo, año_fabricación: dom_año_fabricación,
 diagnósticos: dom_diagnósticos, importe_a_pagar: dom_importe_a_pagar)

CP: {matrícula, fecha}
 dom_matrícula: cadena(10);
 dom_fecha: cadena(8);
 dom_marca: cadena(15);
 dom_modelo: cadena(10);
 dom_año_fabricación: entero;
 dom_diagnóstico: Registro de (código: cadena(5),
 descripción: cadena(500),
 valoración: cadena(100));
 dom_diagnósticos: conjunto de dom_diagnóstico;
 dom_importe_a_pagar: real;

Restricciones:

- Todos los atributos tienen restricción de valor no nulo.
- La marca, el modelo y el año_fabricación de un vehículo vienen determinados por la matrícula.
- La descripción de un diagnóstico depende del código, mientras que la valoración depende del código, del vehículo que se inspecciona, y de la fecha.

26. Dada las siguientes definiciones de dominios, la relación y el conjunto de restricciones que la acompañan, transfórmese a un conjunto de relaciones en tercera forma normal.

dom_nss: cadena(20)
dom_cocama: entero
dom_coddoctor: entero
dom_nomdoctor: cadena(50)
dom_equipo: Conjunto de dom_ATS²⁷

dom_ats: Registro de
cod_ATS: entero,
nom_ATS: cadena (20)
dom_evolucion: Conjunto de dom_anotación
dom_anotación: Registro de
fecha: date
observaciones: cadena(200)
prescripción: cadena(200)
Hospitalización(nss: dom_nss, cod_cama: dom_cocama, cod_doctor: dom_coddoctor,
nom_doctor: dom_nomdoctor, equipo_ATS: dom_equipo, evolución: dom_evolucion)
CP: {nss, cod_cama}
VNN: {equipo_ATS, nom_doctor, cod_doctor, evolución}

Restricciones:

- No hay dos ATS con el mismo código
- No hay dos doctores con el mismo código
- Todo paciente ha de estar atendido por algún doctor, y por algún ATS.
- La evolución del paciente en una cama se almacena como máximo una vez al día

²⁷ ATS: Asistente Técnico Sanitario.