DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

Escuela Técnica Superior de Ingeniería Informática

etsinf

Bases de Datos y Sistemas de Información

Grado en Ingeniería Informática

Unidad Didáctica 3: Diseño de Bases de Datos Relacionales

Parte 3: Diseño Lógico

(Doc. UD4.3)

Curso 2021/2022

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Para la elaboración de este documento se han consultado los siguientes textos:

- [MCC94] Mota, L.; Celma, M.; Casamayor, J. C.; Bases de datos relacionales: teoría y diseñoSPUPV 767.94, 1994.

- [MP97] Mota, L.; Pastor, M.A.; Diseño conceptual con el modelo Entidad-Relación. Informe Interno DSICID/56/97. 1997

- [CCM03] Celma, M.; Casamayor, J. C.; Mota, L.; Bases de datos relacionales. Pearson, Prentice Hall, 2003.

- [Lar04] Larman, C.; UML y Patrones: una introducción al análisis y diseño orientado a objetos y al proceso unificado (2ª edición). Pearson, Prentice Hall, 2004.

- [BRJ06] Booch, G.; Rumbaugh, J.; Jacobson, I.; El lenguaje unificado de modelado (2ª edición). Pearson, Addsion Wesley, 2006.

- [SR07] Stevens, P.; Pooley, R.; Utilización del UML en Ingeniería del Software con objetos y componentes (2ª edición). Pearson, Addsion Wesley, 2007.

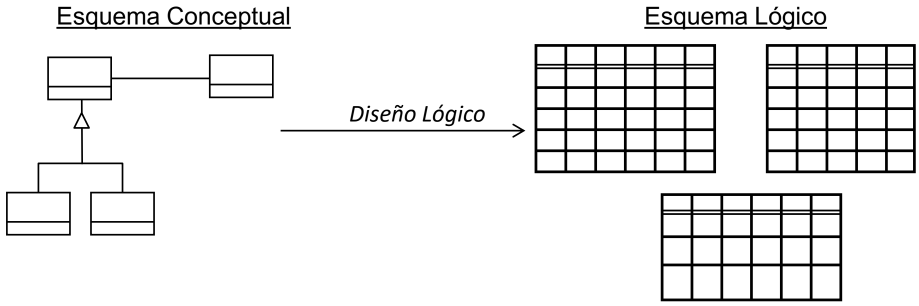
- [Oli01] Olivé, A.; Modelització conceptual de sistemes d'informació. Edicions UPC. 2001.

- [EN12] Ramez Elmasri, Shamkant B. Navathe. Fundamentos de sistemas de bases de datos. Addison-Wesley 2002)

Índice

1 Introducción

El diseño lógico consiste en la transformación del esquema conceptual, que se encuentra descrito con un cierto modelo de datos, en estructuras 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. Así pues, en esta asignatura, el diseño lógico consistirá en transformar un diagrama de clases (esquema conceptual) en un esquema relacional (esquema lógico).



El proceso de obtención de un esquema relacional que represente adecuadamente todos los aspectos estáticos expresados en el diagrama de clases se puede concretar en un conjunto de reglas de transformación. Aquellas propiedades expresadas en el diagrama que no se puedan representar en el esquema relacional deberán ser incluidas en una lista de *restricciones de integridad* para que sean controladas desde programa. El objetivo de este documento es presentar este conjunto de transformaciones.

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

"Elegir el esquema con menos restricciones de integridad añadidas. Ante igualdad de restricciones, elegir el esquema con menos relaciones".

Este criterio se justifica por el hecho de que las restricciones 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.

Para estudiar estas transformaciones, se va a realizar un barrido por los posibles objetos y estructuras de un diagrama de clases, presentándose en cada caso el conjunto de relaciones equivalentes, es decir que expresen la misma realidad que el diagrama.[1]

2 Transformación de las clases

En un diagrama clases, se pueden distinguir cuatro tipos de clases

- Clases fuertes (ni débiles ni especializadas),
- Clases débiles,
- Clases especializadas y
- Clase asociación

A continuación, se muestra cómo se transforman los tres primeros tipos. La trasformación de una clase asociación se realiza al transformar la asociación de la que cuelga.

[1]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.

2.1 Clases fuertes

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

A

a_0: {id}: t_a_0
a_1: {único$_1$}:{0 ..1}:t_a_1
a_2: {1..1}:t_a_2
a_3: {0..1}:t_a_3
a_4: {1..*}:t_a_4
a_5: {0..*}:t_a_5
a_6: {0..1}:
    a_{61}:t_ a_{61}
    a_{62}:t_a_{62}

El esquema relacional equivalente es el siguiente:

A(a_0:t_a_0,a_1:t_a_1,a_2:t_a_2,a_3:t_a_3, a_{61}:t_a_{61},a_{62}:t_a_{62})
   CP:{a_0}
   Úni:{a_1}
   VNN:{a_2}
A4(a_0:t_a_0,a_4:t_a_4)
   CP:{a_0,a_4}
   CAj:{a_0}→A(a_0)
A5(a_0:t_a_0,a_5:t_a_5)
   CP:{a_0,a_5}
   CAj:{a_0}→A(a_0)

Restricciones de integridad:

RI1: Todo valor que aparezca en el atributo a_0 de *A* debe aparecer en el atributo a_0 de *A4*.

Donde:

- Los tipos de los atributos de la relación son los asociados al atributo en la clase.
- El atributo identificador (en el ejemplo a_0) se convierte en la clave primaria de la relación. Si la clase tuviera un identificador formado por más de un atributo, el conjunto de todos ellos formaría la clave primaria de la relación.
- La cardinalidad {1..1} se representan mediante la restricción de valor no nulo (p.e. a_2).
- Las restricciones de unicidad se representan mediante la propiedad de unicidad del modelo relacional (p.e. a_1).
- Los atributos estructurados se descomponen en tantos atributos como campos tenga el registro (p.e. el registro a_6).
- Los atributos cuya cardinalidad máxima es mayor que 1 deben representarse en relaciones independientes ya que pueden tomar varios valores para cada ocurrencia de la clase que está representada en una tupla de la relación (p.e. a_4 y a_5). Cada relación nueva que aparece contendrá como atributos la clave primaria de la relación a la que pertenece y los atributos que tienen cardinalidad máxima mayor que uno. La clave primaria de la nueva relación estará compuesta, en el caso más general, por todos sus atributos, aunque puede haber restricciones que hagan que la clave esté formada por un subconjunto de estos atributos. Además, en el caso de que la cardinalidad mínima del atributo sea 1, habrá que incluir una restricción de integridad añadida al esquema relacional que la exprese (p.e. la RI1 para representar la cardinalidad mínima de a_4).

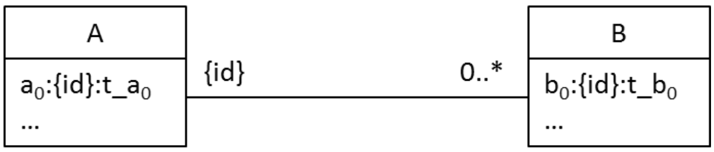
La exigencia de los dos últimos puntos viene del hecho de que en toda relación se debe cumplir que los tipos de datos de todos sus atributos han de ser simples o escalares, es decir, no puede haber tipos más complejos como registros o conjuntos. Utilizando una terminología más precisa, diríamos que toda relación debe estar en *primera forma normal*. La teoría de la normalización se verá en el último apartado del documento.

2.2 Clases débiles

La transformación de una clase débil es análoga a la de una clase que no sea débil; la única diferencia es que es necesario incorporar, como atributos, las claves primarias de las relaciones que representan a las clases gracias a las cuales se identifica. Estos atributos formarán parte de la clave primaria de la relación que represente la clase débil.

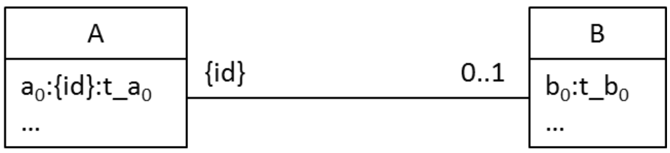
Para ilustrar esta transformación, supóngase los siguientes diagramas y sus correspondientes transformaciones (no se presentan todos los casos posibles):

1)



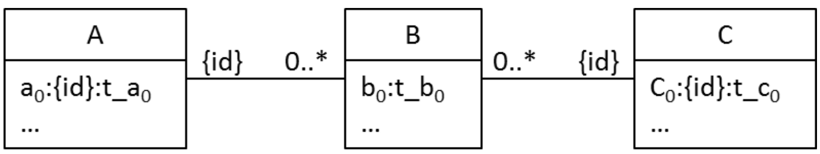
A(a_0:t_a_0,…)
    CP:{a_0}
B(b_0:t_b_0,a_0:t_a_0,…)
    CP:{a_0,b_0}
    CAj:{a_0}→A(a_0)

2)



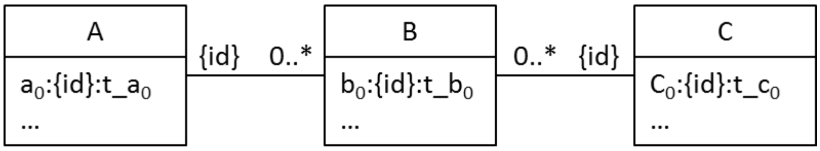
A(a_0:t_a_0,…)
    CP:{a_0}
B(b_0:t_b_0,a_0:t_a_0,…)
    CP:{a_0}
    CAj:{a_0}→A(a_0)

3)



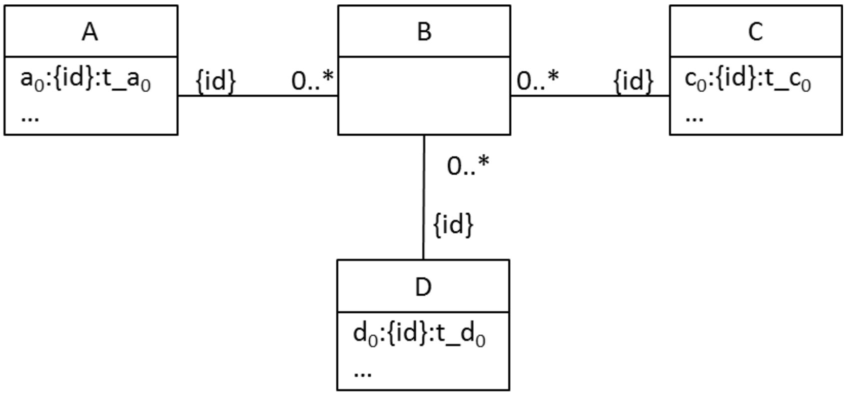
A(a_0:t_a_0,…)
    CP:{a_0}
C(c_0:t_c_0,…)
    CP:{c_0}
B(a_0:t_a_0,c_0:t_c_0,b_0:t_b_0,…)
    CP:{a_0,c_0}
    CAj:{a_0}→A(a_0)
    CAj:{c_0}→C(c_0)

4)




```
A(a₀:t_a₀,…)
   CP:{a₀}
C(c₀:t_c₀,…)
   CP:{c₀}
B(a₀:t_a₀,c₀:t_c₀,b₀:t_b₀,…)
   CP:{a₀,c₀,b₀}
   CAj:{a₀}→A(a₀)
   CAj:{c₀}→C(c₀)
```

5)



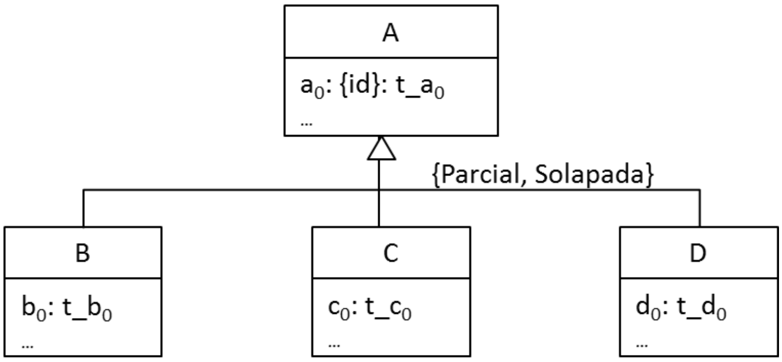

```
   A(a₀:t_a₀,…)
      CP:{a₀}
   C(c₀:t_c₀,…)
      CP:{c₀}
   D(d₀:t_d₀,…)
      CP:{d₀}

B(a₀:t_a₀,c₀:t_c₀,d₀:t_d₀)
      CP:{a₀,c₀,d₀}
      CAj:{a₀}→A(a₀)
      CAj:{c₀}→C(c₀)
      CAj:{d₀}→D(d₀)
```

Hay que darse cuenta de que con estas transformaciones no sólo se representan las clases, sino que también quedan representadas las asociaciones que contribuyen a la identificación de la clase débil.

2.3 Clases especializadas

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



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

```
A(a₀:t_a₀,…)
     CP:{a₀}
B(a₀:t_a₀,b₀:t_b₀,…)
   CP:{a₀}
   CAj:{a₀}→A(a₀)
C(a₀:t_a₀,c₀:t_c₀,…)
   CP:{a₀}
   CAj:{a₀}→A(a₀)
D(a₀:t_a₀,d₀:t_d₀,…)
   CP:{a₀}
   CAj:{a₀}→A(a₀)
```

Obsérvese que la clave primaria de las relaciones donde se representan las clases especializadas se define también como clave ajena a la relación donde está representada la clase general; de esta forma se expresa la restricción de que toda ocurrencia de cualquier clase especializada se corresponde con una ocurrencia de la clase 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: En este caso, el conjunto de relaciones anterior no representa exactamente la generalización/especialización ya que queda por expresar el hecho de que toda ocurrencia *A* tiene que estar asociada con al menos una ocurrencia de alguna subclase. Es necesaria la definición de la siguiente restricción de integridad:

  Restricciones de integridad:

  RI_{Total}: Todo valor que aparezca en el atributo a_0 de *A* debe aparecer en el atributo a_0 de *B*, de *C* o de *D*.

- Disjunta: Para este tipo de especialización, el conjunto de relaciones anterior no expresa la restricción de que las subclases son disjuntas, esto es, el hecho de que cada ocurrencia de la clase general sólo puede estar asociada con una ocurrencia de una subclase. La restricción sería la siguiente:

  Restricciones de integridad:

  $RI_{Disjunta}$: No puede haber un mismo valor en el atributo a_0 de *B* y en *el* a_0 de *C*; ni en el a_0 de *B* y en *el* a_0 de *D*; ni en el atributo a_0 de *C* y en *el* a_0 de *D*.

3 TRANSFORMACIÓN DE LAS ASOCIACIONES

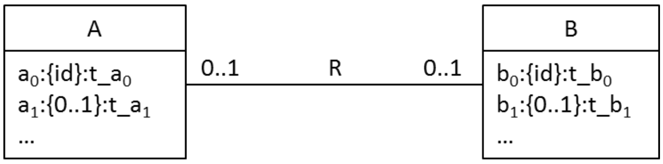
La transformación de las asociaciones depende en gran medida de las cardinalidades máximas y mínimas. A continuación, se presentan varios casos.

3.1 Asociaciones no reflexivas

A continuación, se estudian las posibles asociaciones entre clases distintas.

3.1.1 Asociación 0..1:0..1

Sea el siguiente diagrama de clases:



Este diagrama puede representarse en cuatro esquemas relacionales posibles:[2]

<u>Esquema 1</u>

$A(a_0{:}t\_a_0, a_1{:}t\_a_1, \ldots)$
  $CP{:}\{a_0\}$
$B(b_0{:}t\_b_0, b_1{:}t\_b_1, \ldots, a_0{:}t\_a_0)$
  $CP{:}\{b_0\}$
  $Úni{:}\{a_0\}$
  $CAj{:}\{a_0\} \rightarrow A(a_0)$

Como puede observarse, la representación de la asociación *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 asociación *R* se ha representado junto con la clase *A*:

<u>Esquema 2</u>

$A(a_0{:}t\_a_0, a_1{:}t\_a_1, \ldots, b_0{:}t\_b_0)$
  $CP{:}\{a_0\}$
  $Úni{:}\{b_0\}$
  $CAj{:}\{b_0\} \rightarrow B(b_0)$
$B(b_0{:}t\_b_0, b_1{:}t\_b_1, \ldots)$
  $CP{:}\{b_0\}$

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

<u>Esquema 3</u>

$A(a_0{:}t\_a_0, a_1{:}t\_a_1, \ldots)$
  $CP{:}\{a_0\}$
$B(b_0{:}t\_b_0, b_1{:}t\_b_1, \ldots)$
  $CP{:}\{b_0\}$
$R(b_0{:}t\_b_0, a_0{:}t\_a_0)$
  $CP{:}\{b_0\}$
  $Úni{:}\{a_0\}$
  $VNN{:}\{a_0\}$
  $CAj{:}\{a_0\} \rightarrow A(a_0)$
  $CAj{:}\{b_0\} \rightarrow B(b_0)$

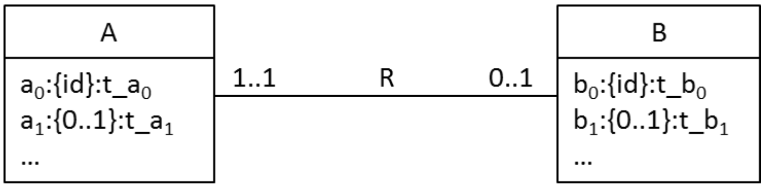
[2] En los demás casos que se van a estudiar, no se mostrarán todos los esquemas posibles sino tan sólo el que se considere mejor o al menos tan bueno como todos los esquemas posibles.

Esquema 4

A(a_0:t_a$_0$,a_1:t_a$_1$,…)
  CP:{a_0}
B(b_0:t_b$_0$,b_1:t_b$_1$,…)
  CP:{b_0}
R(b_0:t_b$_0$,a_0:t_a$_0$)
  CP:{a_0}
  Úni:{b_0}
  VNN:{b_0}
  CAj:{a_0}→A(a_0)
  CAj:{b_0}→B(b_0)

3.1.2 Asociación 1..1: 0..1

Este tipo de asociación se muestra en el siguiente diagrama:



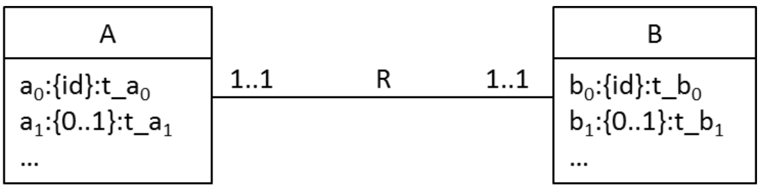
Las relaciones equivalentes a esta estructura son:

A(a_0:t_a$_0$,a_1:t_a$_1$,…)
  CP:{a_0}
B(b_0:t_b$_0$,b_1:t_b$_1$,…,a_0:t_a$_0$)
  CP:{b_0}
  Úni:{a_0}
  VNN:{a_0}
  CAj:{a_0}→A(a_0)

En este caso, como la clase *B* está obligada a relacionarse con la clase *A* a través de la asociación *R*, es posible representar esta conexión mediante la inclusión, en la relación que representa a la clase *B*, 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.

3.1.3 Asociación 1..1:1..1

Este tipo de asociación se muestra en el siguiente diagrama:



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

A-B(a_0:t_a$_0$,a_1:t_a$_1$,…,b_0:t_b$_0$,b_1:t_b$_1$,…)
  CP:{a_0}
  Úni:{b_0}
  VNN:{b_0}

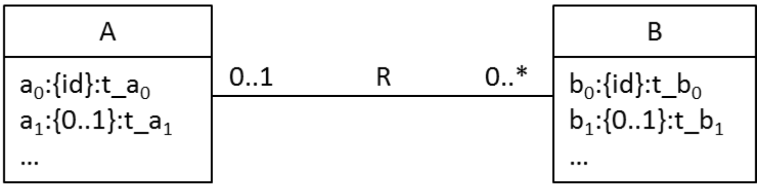
Pese a ser muy compacta (pocas relaciones y sin restricciones), esta solución complica la manipulación de los objetos representados por la clase *B* (al no tener una relación propia) así que en algunos casos será preferible el siguiente esquema, aunque tenga más relaciones:

```
A(a0:t_a0,a1:t_a1,…,b0:t_b0)
   CP:{a0}
   Úni:{b0}
   VNN:{b0}
   CAj:{b0}→B(b0)
B(b0:t_b0,b1:t_b1,…)
   CP:{b0}
   CAj:{b0}→A(b0)
```

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

3.1.4 Asociación 0..1:0..*

El siguiente diagrama muestra una asociación de este tipo:



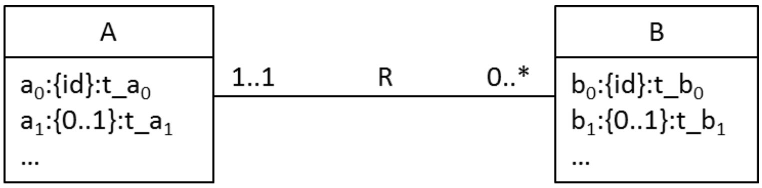
Las relaciones equivalentes a esta estructura son:

```
A(a0:t_a0,a1:t_a1,…)
   CP:{a0}
B(b0:t_b0,b1:t_b1,…,a0:t_a0)
   CP:{b0}
   CAj:{a0}→A(a0)
```

En este caso, se puede observar que, debido a la cardinalidad de la asociación *R,* es posible representar ésta como una clave ajena en la relación que representa la clase con cardinalidad máxima 1 (la clase *B*). Nótese que la posibilidad de que haya ocurrencias de la clase *B* que no tomen parte en la asociación *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.

3.1.5 Asociación 1..1:0..*

El siguiente diagrama muestra una estructura de este tipo:



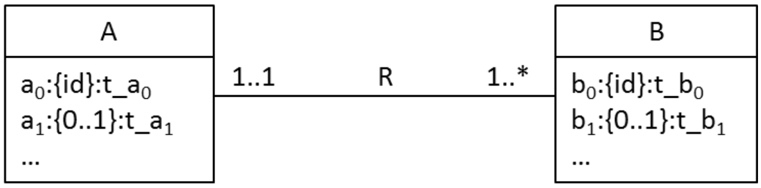
Las relaciones equivalentes a esta estructura son:

```
A(a0:t_a0,a1:t_a1,…)
   CP:{a0}
B(b0:t_b0,b1:t_b1,…,a0:t_a0)
   CP:{b0}
   CAj:{a0}→A(a0)
   VNN:{a0}
```

En este caso, la cardinalidad mínima de la clase *B* (esto es, la restricción de existencia de *B* respecto a *R*) se representa por la definición de una restricción de valor no nulo sobre la clave ajena de la relación *B* que representa la asociación *R*.

3.1.6 Asociación 1..1:1..*

El siguiente diagrama muestra una estructura de este tipo:



Las relaciones equivalentes a esta estructura son:

```
A(a0:t_a0,a1:t_a1,…)
   CP:{a0}
B(b0:t_b0,b1:t_b1,…,a0:t_a0)
   CP:{b0}
   CAj:{a0}→A(a0)
   VNN:{a0}
```

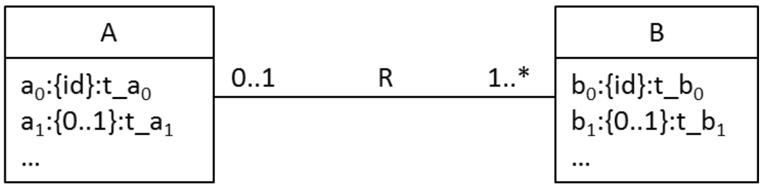
Como puede apreciarse, este esquema es idéntico al del caso anterior, por tanto, la restricción de existencia sobre la clase *A* no está representada. Esta restricción habría que incluirla en la lista:

<u>`Restricciones de integridad:`</u>

`RI1: Todo valor que aparezca en el atributo` a_0 `de` *A* `debe aparecer en el atributo` a_0 `de` *B*.

3.1.7 Asociación 0..1:1..*

El siguiente diagrama muestra una estructura de este tipo:



Las relaciones equivalentes a esta estructura son:

```
A(a0:t_a0,a1:t_a1,…)
   CP:{a0}
B(b0:t_b0,b1:t_b1,…,a0:t_a0)
   CP:{b0}
   CAj:{a0}→A(a0)
```

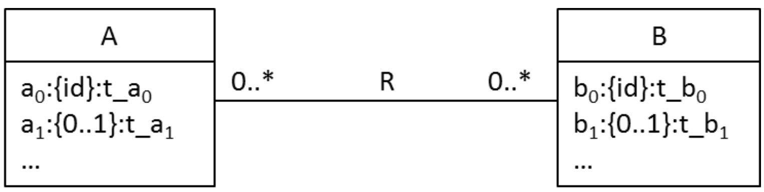
Como puede apreciarse, este esquema es idéntico al del caso 0..1:0..*, sin que quede representada la restricción de existencia de la clase *A*. Esta restricción habría que incluirla en la lista:

<u>`Restricciones de integridad:`</u>

`RI1: Todo valor que aparezca en el atributo` a_0 `de` *A* `debe aparecer en el atributo` a_0 `de` *B*.

3.1.8 Asociación 0..*:0..*

Este tipo de estructura se muestra en el siguiente diagrama:



Las relaciones equivalentes a esta estructura son:

```
A(a0:t_a0,a1:t_a1,…)
   CP:{a0}
B(b0:t_b0,b1:t_b1,…)
   CP:{b0}
R(a0:t_a0,b0:t_b0)
   CP:{a0,b0}
   CAj:{a0}→A(a0)
   CAj:{b0}→B(b0)
```

En este caso, debido a que las dos cardinalidades máximas son *, es necesario la representación de la asociación *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 primaria.

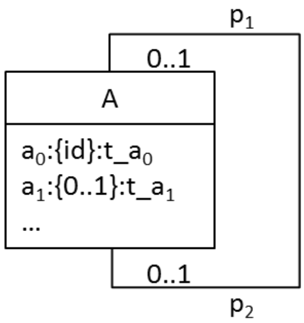
Los casos de asociaciones en las que las dos cardinalidades máximas son * en los que aparecen restricciones de existencia se diseñan de manera análoga, siendo necesario añadir restricciones para representar las restricciones de existencia.

3.2 Asociaciones reflexivas

Estas asociaciones son difíciles de asimilar por lo que para entender sus transformaciones se recomienda compararlas con las asociaciones no reflexivas de igual cardinalidad ya que la estrategia seguida es la misma. Un detalle importante a recordar es que las claves ajenas no tienen por qué tener el mismo nombre en la relación en la que se define que en la relación a la que se hace referencia. En el caso de las relaciones reflexivas, con mucha frecuencia, es necesario que se tengan nombres distintos.

3.2.1 Asociación reflexiva 0..1:0..1

En el siguiente diagrama se muestra una asociación de este tipo y la clase asociada. En la figura, p_1 y p_2 son dos etiquetas que indican el rol que cada ocurrencia de *A* juega en la asociación.



La relación equivalente a esta estructura es:

```
A(a0:t_a0,a1:t_a1,…,a0_p2:t_a0)
   CP:{a0}
   Úni:{a0_p2}
   CAj:{a0_p2}→A(a0)
```

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

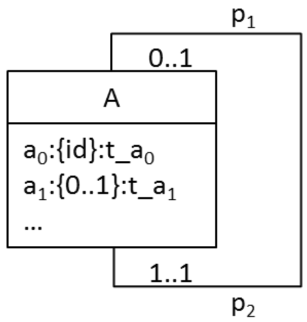
Las tres tuplas que contiene la relación nos proporcionan la siguiente información:

- Hay tres objetos de la clase *A* identificados por $a_0 = 1$, $a_0 = 2$ y $a_0 = 3$.
- Hay dos instancias de la relación *R*, representadas en las dos últimas tuplas, una que relaciona al objeto 2 (jugando el papel p_1) con el objeto 1 (jugando el papel p_2); y otra que relaciona al objeto con 3 (jugando el papel p_1) con el objeto 2 (jugando el papel p_2).

Así pues, en este caso, la asociación *R* queda representada mediante la clave ajena $a_0\_p_2$ en la relación que representa a la clase. Además, esta clave ajena también se define con restricción de unicidad para representar que la cardinalidad máxima de la clase *A* jugando el papel p_2 es 1.

3.2.2 Asociación 0..1:1..1

En el siguiente diagrama se muestra una asociación de este tipo y la clase asociada.



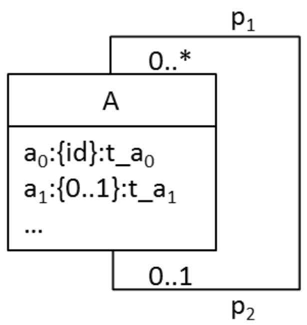
La relación equivalente a esta estructura es:

```
A(a₀:t_a₀,a₁:t_a₁,…,a₀_p₂:t_a₀)
   CP:{a₀}
   Úni:{a₀_p₂}
   CAj:{a₀_p₂}→A(a₀)
   VNN:{a₀_p₂}
```

En este caso, la restricción de existencia se representa con la definición de la restricción de valor no nulo sobre la clave ajena $a_0\_p_2$.[3]

3.2.3 Asociación 0..*:0..1

En el siguiente diagrama se muestra una asociación de este tipo y la clase asociada.



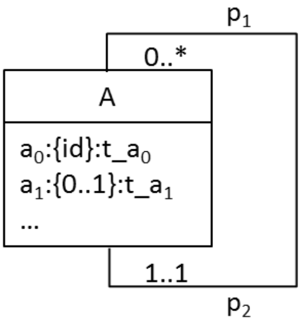
La relación equivalente a esta estructura es:

```
A(a₀:t_a₀,a₁:t_a₁,…,a₀_p₂:t_a₀)
   CP:{a₀}
   CAj:{a₀_p₂}→A(a₀)
```

[3] Curiosamente, esta asociación representa exactamente la misma realidad que asociaciones de cardinalidad: 1..1:1..1, 1..*:0..1 y 1..*:1..1 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.

3.2.4 Asociación 0..*: 1..1

En el siguiente diagrama se muestra una asociación de este tipo y la clase asociada.



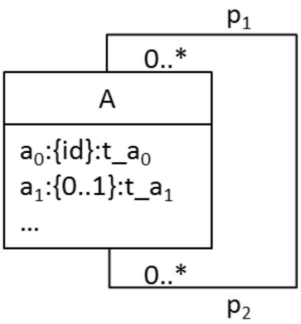
La relación equivalente a esta estructura es:

```
A(a₀:t_a₀,a₁:t_a₁,…,a₀_p₂:t_a₀)
   CP:{a₀}
   CAj:{a₀_p₂}→A(a₀)
   VNN:{a₀_p₂}
```

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 asociación *R*.

3.2.5 Asociación 0..*:0..*

En el siguiente diagrama se muestra una asociación de este tipo y la clase asociada.



Las relaciones equivalentes a esta estructura son:

```
A(a₀:t_a₀,a₁:t_a₁,…)
   CP:{a₀}
R(a₀_p₁:t_a₀,a₀_p₂:t_a₀)
   CP:{a₀_p₁,a₀_p₂}
   CAj:{a₀_p₁}→A(a₀)
   CAj:{a₀_p₂}→A(a₀)
```

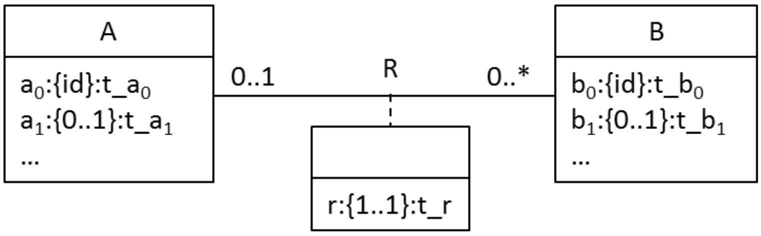
En este caso, debido a que la asociación reflexiva *R* tiene ambas cardinalidades máximas con valor *, es necesario utilizar una relación independiente para representarla. Nótese cómo en este caso hay en *R* dos claves ajenas que se refieren a la misma relación *A*. Si hubiera definida alguna restricción de existencia, también sería necesaria la inclusión de restricciones de integridad. La presencia de restricciones de existencia en las relaciones reflexivas es, sin embargo, poco frecuente.

3.3 Atributos de enlace

Cuando una asociación entre clases tiene atributos de enlace, éstos deben incluirse *siempre* en la relación donde se represente la asociación.

La presencia de atributos en las asociaciones 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:

1)



Si consideramos como esquema relacional de partida el diseñado en el apartado 3.1.4 añadiendo el atributo *r* en la relación *B* (que es en la que se representa *R*) el esquema quedaría como sigue:

```
A(a0:t_a0,a1:t_a1,…, an:t_an)
  CP:{a0}
B(b0:t_b0,b1:t_b1,…, bm:t_bm, a0:t_a0,r:t_r)
  CP:{b0}
  CAj:{a0}→A(a0)
```

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.

Restricciones de integridad:

RI1: En toda tupla de *B* se debe cumplir que si el atributo a_0 es no nulo el atributo *r* también tiene que ser no nulo y además no puede haber ninguna tupla de *B* en la que a_0 sea nulo y *r* distinto de nulo.

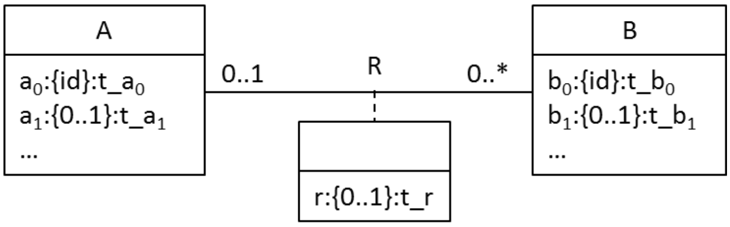
Al añadir una restricción de integridad, este esquema relacional deja de ser el más adecuado ya que hay otro que, aunque tiene una relación más, no necesita restricciones añadidas. Este esquema es el siguiente:

```
A(a0:t_a0,a1:t_a1,…, an:t_an)
  CP:{a0}
B(b0:t_b0,b1:t_b1,…, bm:t_bm)
  CP:{b0}
R(b0:t_b0,a0:t_a0,r:t_r)
  CP:{b0}
  VNN:{a0}
  VNN:{r}
  CAj:{a0}→A(a0)
  CAj:{b0}→B(b0)
```

Estos problemas derivados de la existencia de atributos de enlace y del hecho de que tengan o no tengan restricción de valor no nulo sólo se presentan cuando la asociación 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 adecuado[4]:

[4] Se deja como ejercicio el desestimar otros posibles esquemas relacionales.

2)



Las relaciones que se obtendrían siguiendo la transformación general serían las mismas que en el caso anterior,

$A(a_0{:}t\_a_0, a_1{:}t\_a_1, \ldots, a_n{:}t\_a_n)$
CP:{a_0}

$B(b_0{:}t\_b_0, b_1{:}t\_b_1, \ldots, b_m{:}t\_b_m, a_0{:}t\_a_0, r{:}t\_r)$
CP:{b_0}
CAj:{a_0}$\rightarrow$A(a_0)

pero habría que incluir la restricción de integridad siguiente:

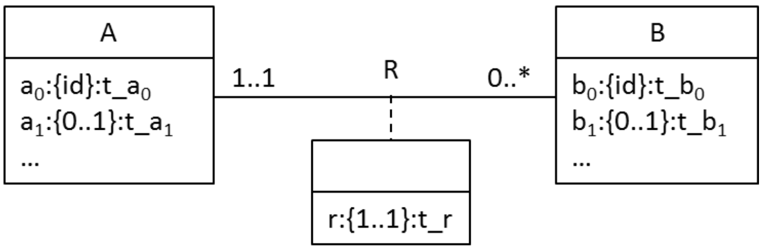
Restricciones de integridad:

RI1: No puede existir una tupla en B que tenga el atributo a_0 nulo y el atributo r distinto de nulo.

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

$A(a_0{:}t\_a_0, a_1{:}t\_a_1, \ldots, a_n{:}t\_a_n)$
CP:{a_0}
$B(b_0{:}t\_b_0, b_1{:}t\_b_1, \ldots, b_m{:}t\_b_m)$
CP:{b_0}
$R(b_0{:}t\_b_0, a_0{:}t\_a_0, r{:}t\_r)$
CP:{b_0}
VNN:{a_0}
CAj:{a_0}$\rightarrow$A(a_0)
CAj:{b_0}$\rightarrow$B(b_0)

3)



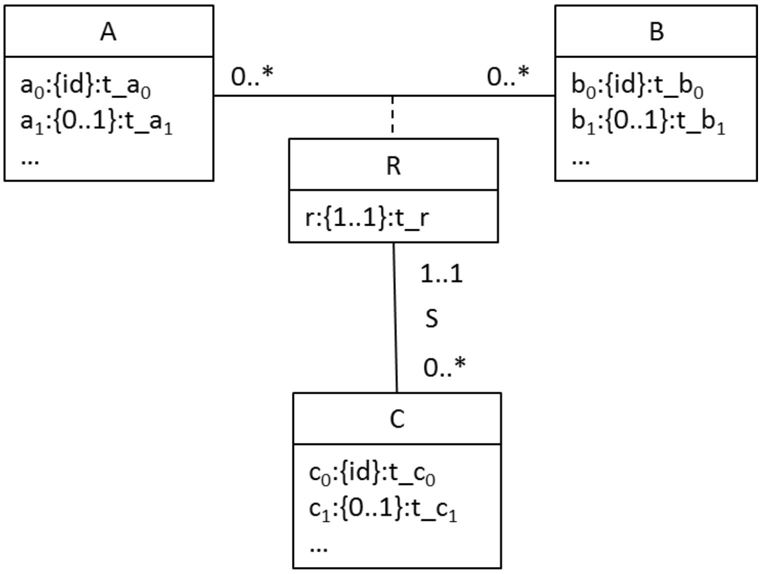
$A(a_0{:}t\_a_0, a_1{:}t\_a_1, \ldots, a_n{:}t\_a_n)$
CP:{a_0}
$B(b_0{:}t\_b_0, b_1{:}t\_b_1, \ldots, b_m{:}t\_b_m, a_0{:}t\_a_0, r{:}t\_r)$
CP:{b_0}
VNN:{a_0}
VNN:{r}
CAj:{a_0}$\rightarrow$A

En esta situación en la que la entidad *B* tiene restricción de existencia respecto a *R*, dado que, sobre el atributo a_0 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 asociación, es decir, la relación *R* junto a la entidad *B*, tanto si el atributo *r* tiene restricción de valor no nulo como si no.

3.4 Transformación de la asociación cuando se asocia con clases asociación

En este apartado se presentan algunos ejemplos que ilustran cómo realizar la transformación de una asociación cuando en ésta participa una clase asociación. Contemplar todos los posibles casos es complicado ya que pueden aparecer asociaciones de cualquier grado y, por otro lado, también se puede asociar con cualquier otra clase del diagrama de clases. Por ello se verán un par de ejemplos en los que se da este tipo de asociación.

Supóngase el diagrama de la figura:



Se deberá iniciar el diseño transformando la parte del diagrama en la que está representada la asociación *R* entre *A* y *B*. Hasta que esta parte no se resuelva, no se podrá analizar la asociación *S*. Las relaciones que se obtienen son:

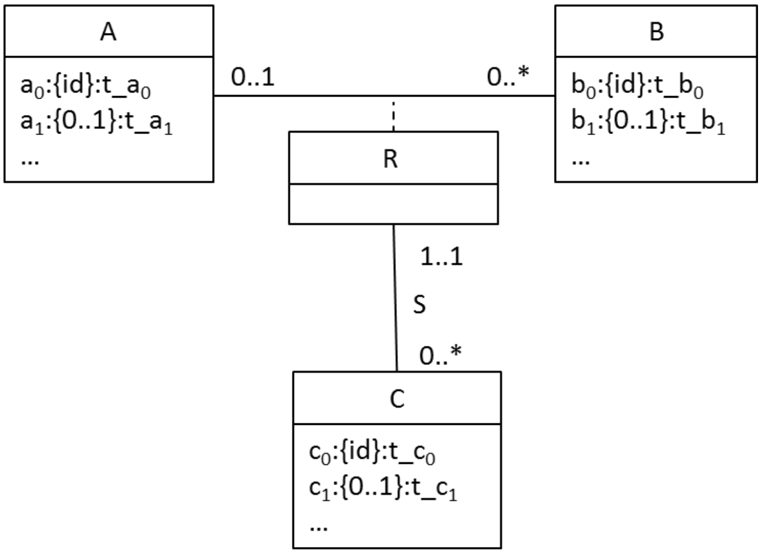
```
A(a0:t_a0,a1:t_a1,…)
  CP:{a0}
B(b0:t_b0,b1:t_b1,…)
  CP:{b0}
R(a0:t_a0,b0:t_b0,r:t_r)
  CP:{a0,b0}
  CAj:{a0}→A(a0)
  CAj:{b0}→B(b0)
  VNN:{r}
```

El segundo paso consiste en transformar el resto del diagrama, es decir, las asociaciones en las cuales interviene *R* así como el resto de clases y asociaciones del diagrama. En el ejemplo, consiste en transformar la asociación *S* en la cual toman parte *R* y *C*, y diseñar también la propia entidad *C*. Ya que la relación *S* es una asociación binaria entre *R* y *C* con cardinalidad máxima 1..* y *C* sufre una restricción de existencia, se debe obtener la relación siguiente:

```
C(c0:t_c0,c1:t_c1,…, a0:t_a0,b0:t_b0)
  CP:{c0}
  CAj:{a0,b0}→R(a0,b0)
  VNN:{a0,b0}
```

Esta relación representa tanto la clase *C* como la asociación *S*. Esta última se representa mediante una clave ajena {a_0, b_0} que hace referencia a la relación en la cual está representada la asociación *R*.

En otros casos, como se ilustra en el siguiente ejemplo, el tipo de asociación entre una clase asociación, puede hacernos variar el criterio general estudiado para el caso de la transformación de una asociación con esa cardinalidad.



En este caso la clase asociación *R* no tiene relación propia ya que según se ha mostrado en el apartado 3.1.4, ésta asociación puede representarse en la relación de la clase *B*. Las relaciones que se obtienen teniendo en cuenta las normas explicadas, son las siguientes:

```
A(a0:t_a0,a1:t_a1,…n)
  CP:{a0}
B(b0:t_b0,b1:t_b1,…,a0:t_a0)
        CP:{b0}
        CAj:{a0}→A
```

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(c0:t_c0,c1:t_c1,…,b0:t_b0)
  CP:{c0}
  CAj:{b0}→B
  VNN:{b0}
```

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:

<u>Restricciones de integridad:</u>

RI1: No puede existir una tupla en *C* que se relacione con una tupla de *B* que no esté relacionada con *A*.

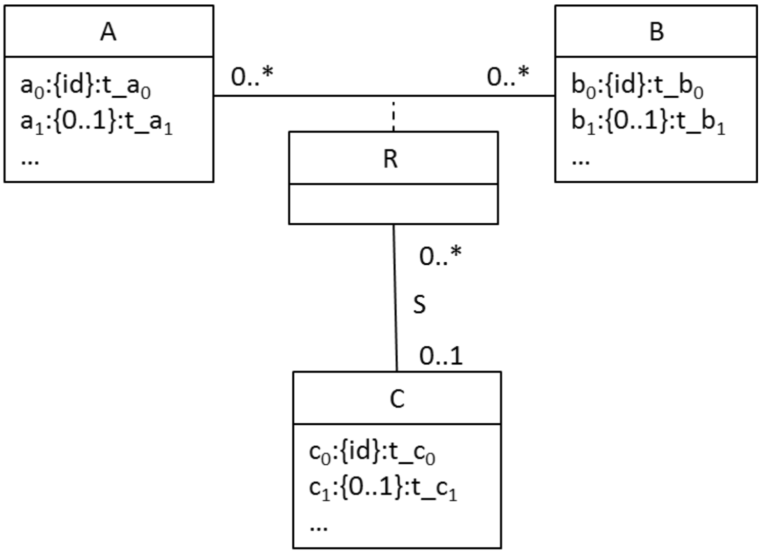
De nuevo la aparición de la restricción hace que este esquema no sea el más deseable ya que se puede encontrar otro con una relación más, pero sin restricciones añadidas. Este esquema es el que se muestra:

```
A(a0:t_a0,a1:t_a1,…)
  CP:{a0}
B(b0:t_b0,b1:t_b1,…)
  CP:{b0}
R(b0:t_b0,a0:t_a0)
  CP:{b0}
  VNN:{a0}
  CAj:{a0}→A(a0)
  CAj:{b0}→B(b0)
```

```
C(c0:t_c0,c1:t_c1,…,b0:t_b0)
  CP:{c0}
  CAj:{b0}→R(b0)
  VNN:{b0}
```

A continuación, se incluyen algunos ejemplos más.

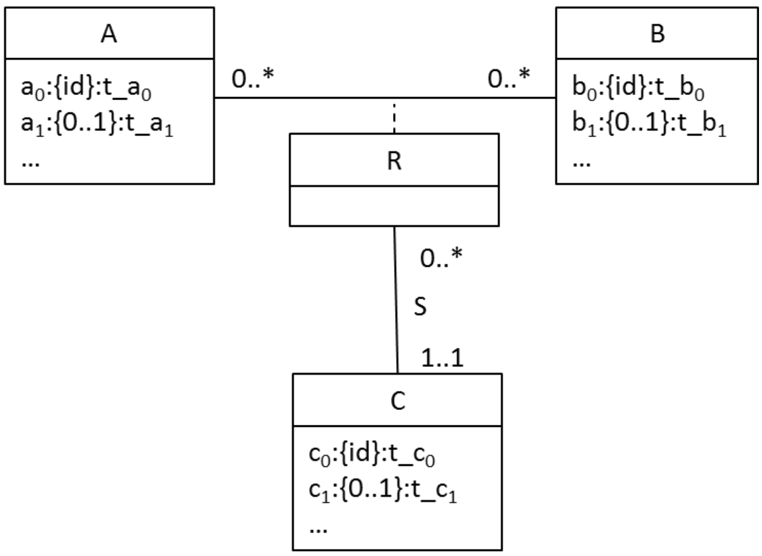
1)



El esquema relacional asociado es el siguiente:

```
A(a0:t_a0,a1:t_a1,…)
  CP:{a0}
B(b0:t_b0,b1:t_b1,…)
  CP:{b0}
C(c0:t_c0,c1:t_c1,…)
  CP:{c0}
R(a0:t_a0,b0:t_b0,c0:t_c0)
  CP:{a0,b0}
  CAj:{a0}→A(a0)
  CAj:{b0}→B(b0)
  CAj:{c0}→C(c0)/*representa la asociación S*/
```

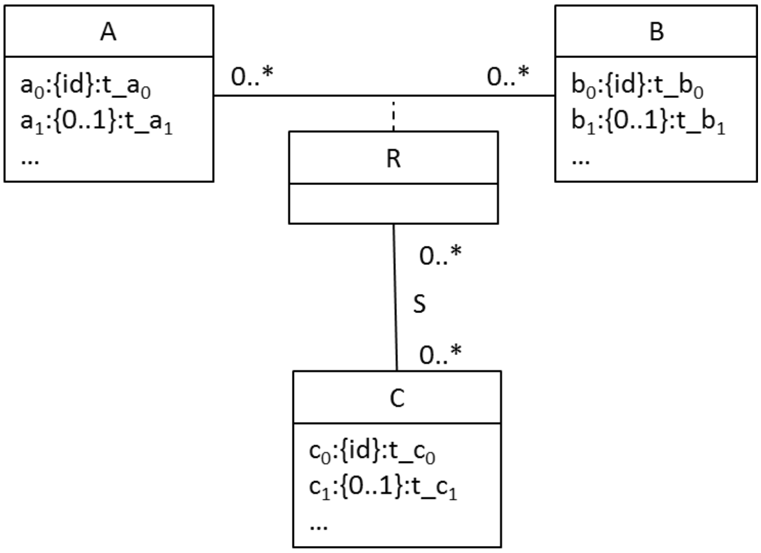
2)



El esquema relacional asociado es el siguiente:

```
A(a₀:t_a₀,a₁:t_a₁,…)
  CP:{a₀}
B(b₀:t_b₀,b₁:t_b₁,…)
  CP:{b₀}
C(c₀:t_c₀,c₁:t_c₁,…)
  CP:{c₀}
R(a₀:t_a₀,b₀:t_b₀,c₀:t_c₀)
  CP:{a₀,b₀}
  CAj:{a₀}→A(a₀)
  CAj:{b₀}→B(b₀)
  CAj:{c₀}→C(c₀) /*representa la asociación S*/
  VNN:{c₀}       /*Representa la cardinalidad mínima de R en S*/
```

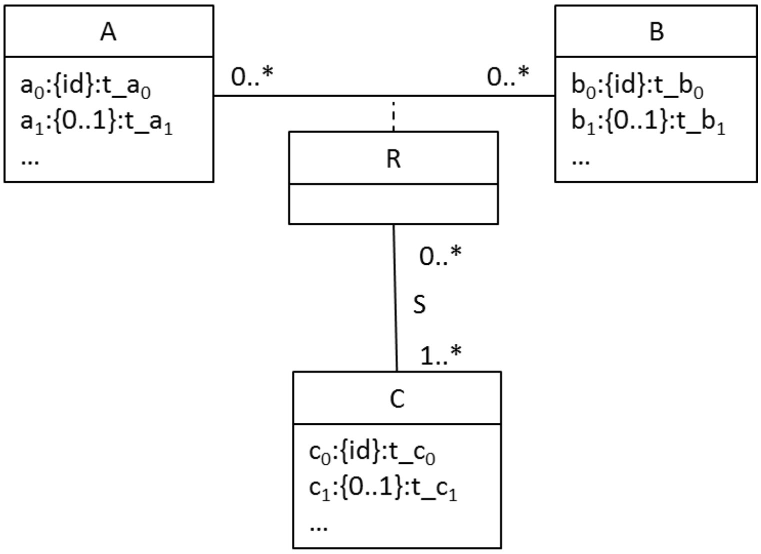
3)



El esquema relacional asociado es el siguiente:

```
A(a₀:t_a₀,a₁:t_a₁,…)
  CP:{a₀}
B(b₀:t_b₀,b₁:t_b₁,…)
  CP:{b₀}
C(c₀:t_c₀,c₁:t_c₁,…)
  CP:{c₀}
R(a₀:t_a₀,b₀:t_b₀)
  CP:{a₀,b₀}
  CAj:{a₀}→A(a₀)
  CAj:{b₀}→B(b₀)
S(a₀:t_a₀,b₀:t_b₀,c₀:t_c₀)
  CP:{a₀,b₀,c₀}
  CAj:{a₀,b₀}→R(a₀,b₀)
  CAj:{c₀}→C(c₀)
```

4)



El esquema relacional asociado es el siguiente:

```
A(a0:t_a0,a1:t_a1,…)
  CP:{a0}
B(b0:t_b0,b1:t_b1,…)
  CP:{b0}
C(c0:t_c0,c1:t_c1,…)
  CP:{c0}
R(a0:t_a0,b0:t_b0)
  CP:{a0,b0}
  CAj:{a0}→A(a0)
  CAj:{b0}→B(b0)
S(a0:t_a0,b0:t_b0,c0:t_c0)
  CP:{a0,b0,c0}
  CAj:{a0,b0}→R(a0,b0)
  CAj:{c0}→C(c0)
```

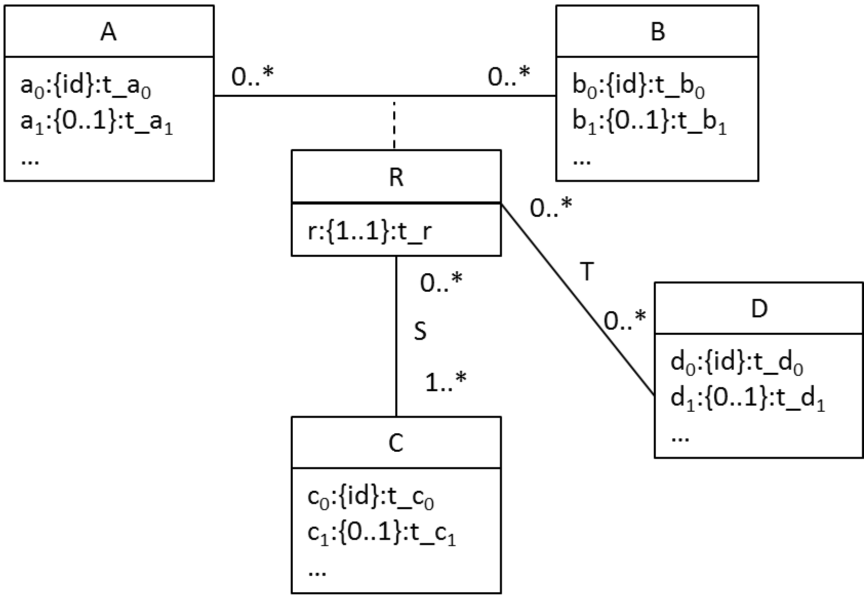
Restricciones de integridad:

RI1: No puede existir una tupla en *R* tal que su clave principal, (a_0,b_0), no aparezca en los atributos (a_0,b_0) de *S*.

En este caso es necesario añadir la restricción de integridad RI1 para asegurar que no toda ocurrencia de la asociación *R* participa en la asociación *S*; existe un esquema más adecuado entonces que consiste en eliminar la relación *R* ya que todo lo que aporta está también en *S* y además elimina la restricción, el esquema quedaría como sigue:

```
A(a0:t_a0,a1:t_a1,…)
  CP:{a0}
B(b0:t_b0,b1:t_b1,…)
  CP:{b0}
C(c0:t_c0,c1:t_c1,…)
  CP:{c0}
RyS(a0:t_a0,b0:t_b0,c0:t_c0)
  CP:{a0,b0,c0}
  CAj:{a0}→A(a0)
  CAj:{b0}→B(b0)
  CAj:{c0}→C(c0)
```

Esta simplificación no es posible en el caso de que la clase asociación *R* participe en otras asociaciones distintas a *S* o tenga atributos. Este caso se ilustra a continuación:



El esquema relacional asociado es el siguiente:

```
A(a0:t_a0,a1:t_a1,…)
  CP:{a0}
B(b0:t_b0,b1:t_b1,…)
  CP:{b0}
C(c0:t_c0,c1:t_c1,…)
  CP:{c0}
D(d0:t_c0,d1:t_d1,…)
  CP:{d0}
R(a0:t_a0,b0:t_b0,r:t_r)
  CP:{a0,b0}
  CAj:{a0}→A(a0)
  CAj:{b0}→B(b0)
  VNN:{r}
S(a0:t_a0,b0:t_b0,c0:t_c0)
  CP:{a0,b0,c0}
  CAj:{a0,b0}→R(a0,b0)
  CAj:{c0}→C(c0)
T(a0:t_a0,b0:t_b0,d0:t_d0)
  CP:{a0,b0,d0}
  CAj:{a0,b0}→R(a0,b0)
  CAj:{d0}→D(d0)
```

Restricciones de integridad:

RI1: No puede existir una tupla en *R* tal que su clave principal, (a_0,b_0), no aparezca en los atributos (a_0,b_0) de *S*.

En este caso la relación *R* no puede desaparecer por dos motivos:

- Tiene una asociación con *D* que no se podría representar.
- Tiene un atributo propio, *r*.

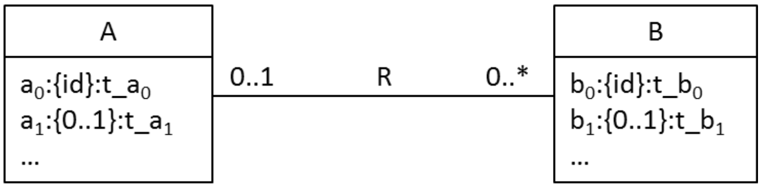
3.5 Elección de las directrices de las claves ajenas

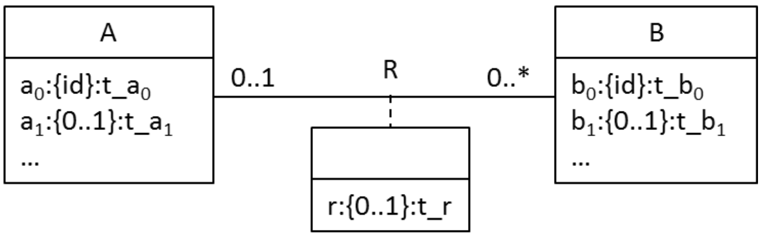
Como ya se estudió en la unidad didáctica 1, la definición de una clave ajena implica la especificación de las directrices de borrado y de actualización para que el SGBD pueda, si así se desea, restaurar la integridad referencial cuando ésta se viola.

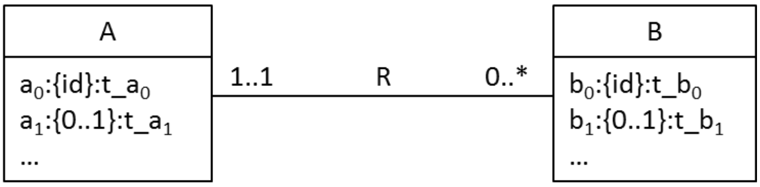
Dado que las claves ajenas permiten representar vínculos entre los objetos del sistema (al conectar las relaciones donde están representados estos objetos), estas directrices permitirán también la representación de algunas restricciones de integridad incluidas en el esquema conceptual o incluso algunas detectadas en análisis del sistema.

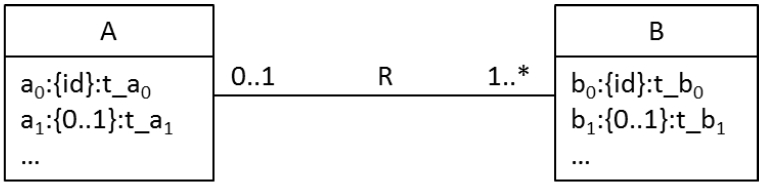
A continuación, se muestra el paso de algunos diagramas de clases a esquemas relacionales y se proponen unas directrices de borrado cuya elección viene condicionada por lo que representa la clave ajena. En la exposición, cuando haya varias opciones razonables se incluirán todas, sin embargo, cuando el diagrama represente un sistema real habrá que elegir una (la que suponga el comportamiento más adecuado para el problema en concreto).

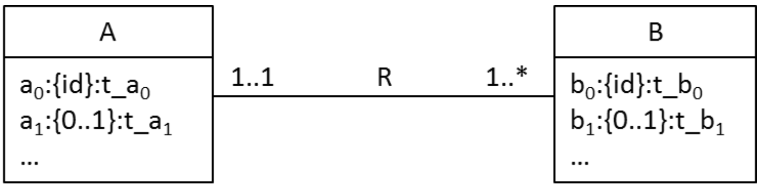
En el caso de la directriz de modificación, en general, se opta por la directriz en cascada por lo que no se va a especificar; esto no significa que otras directrices no sean aceptables, pero vendrán determinadas por cada caso en concreto, no por la estructura del diagrama de clases.

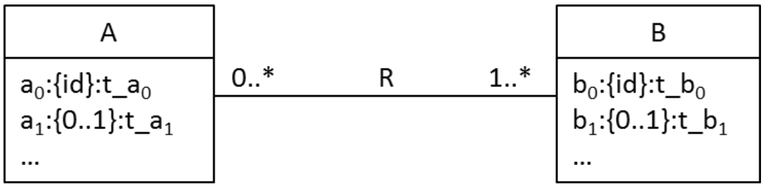
<table>
<tr><td>Esquema Conceptual</td><td>
</td></tr>
<tr><td>Esquema Relacional</td><td>A(a_0: t_a_0, a_1: t_a_1,…)
  CP: {a_0}
B(b_0: t_b_0, b_1: t_b_1,…, a_0: t_a_0)
  CP: {b_0}
  CAj: {a_0}→ A
    BORRADO {NULOS|RESTRICTIVO|CASCADA}</td></tr>
<tr><td>Comentarios</td><td>En este caso las tres directrices son aceptables y su elección dependerá del comportamiento que se desee que tenga el sistema ante borrados de tuplas de la relación A.</td></tr>
</table>

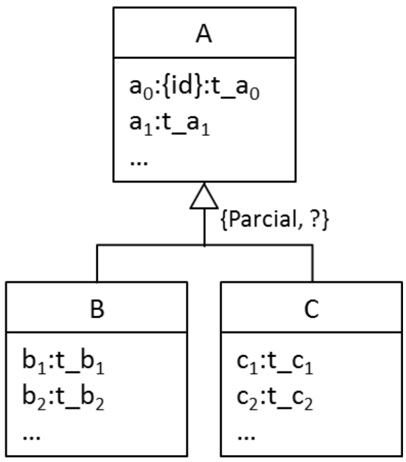
<table>
<tr><td>Esquema Conceptual</td><td>
</td></tr>
<tr><td>Esquema Relacional</td><td>A(a0: t_a_0, a_1: t_a_1,…)
  CP: {a0}
B(b0: t_b_0, b_1: t_b_1,…)
  CP: {b0}
R(b0: t_b_0, a_0: t_a_0, r: t_r)
  CP: {b_0}
  VNN: {a_0}
  CAj: {a_0}→ A
    BORRADO {RESTRICTIVO|CASCADA}
  CAj: {b_0}→ B
    BORRADO {RESTRICTIVO|CASCADA}</td></tr>
<tr><td>Comentarios</td><td>En este caso dado que relación R representa una relación entre A y B el borrado a nulos no es posible.</td></tr>
</table>

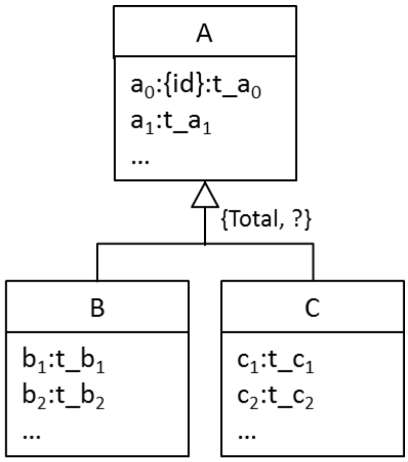
<table>
<tr><td>Esquema Conceptual</td><td colspan="2">
</td></tr>
<tr><td>Esquema Relacional</td><td colspan="2">A(a_0: t_a_0, a_1: t_a_1,…)
CP: {a_0}
B(b_0: t_b_0, b_1: t_b_1,…, a_0: t_a_0)
CP: {b_0}
CAj: {a_0}→A
BORRADO{RESTRICTIVO|CASCADA}
VNN: {a_0}</td></tr>
<tr><td>Comentarios</td><td colspan="2">En este caso el borrado a nulos no es posible por la restricción de existencia B en R que a nivel lógico implica que toda tupla de B debe tener un valor no nulo en el atributo a_0.</td></tr>
</table>

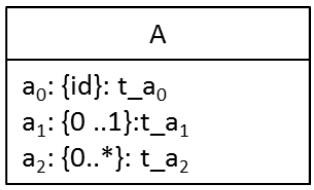
<table>
<tr><td>Esquema Conceptual</td><td colspan="2">
</td></tr>
<tr><td>Esquema Relacional</td><td>A(a_0: t_a_0, a_1: t_a_1,…)
CP: {a_0}
B(b_0: t_b_0, b_1: t_b_1,…, a_0: t_a_0)
CP: {b_0}
CAj: {a_0}→A
BORRADO{NULOS|CASCADA}</td><td>RI:
Todo valor de a_0 de la relación A aparece al menos una vez en el atributo a_0 de la relación B</td></tr>
<tr><td>Comentarios</td><td colspan="2">En este caso el borrado restrictivo no es posible por la restricción de existencia de la entidad A en R que a nivel lógico implica que toda tupla de A aparece referenciada en B.</td></tr>
</table>

<table>
<tr><td>Esquema Conceptual</td><td colspan="2">
</td></tr>
<tr><td>Esquema Relacional</td><td>A(a_0: t_a_0, a_1: t_a_1,…)
CP: {a_0}
B(b_0: t_b_0, b_1: t_b_1,…, a_0: t_a_0)
CP: {b_0}
CAj: {a_0}→A
BORRADO CASCADA
VNN: {a_0}</td><td>RI:
Todo valor de a_0 de la relación A aparece al menos una vez en el atributo a_0 de la relación B.</td></tr>
<tr><td>Comentarios</td><td colspan="2">En este caso el borrado restrictivo no es posible por la restricción de existencia de la entidad A en R que a nivel lógico implica que toda tupla de A debe aparecer referenciada en B. El borrado a nulos tampoco es posible por la restricción de existencia B en R que a nivel lógico implica que toda tupla de B debe tener un valor no nulo en el atributo a_0.</td></tr>
</table>

<table>
<tr><td>Esquema Conceptual</td><td colspan="2">
</td></tr>
<tr><td>Esquema Relacional</td><td>A(a_0: t_a_0, a_1: t_a_1,…)
CP: {a_0}
B(b_0: t_b_0, b_1: t_b_1,…)
CP: {b_0}
R(a_0: t_a_0, b_0: t_b_0)
CP: {a_0,b_0}
CAj: {a_0}→A
BORRADO CASCADA
CAj: {b_0}→B
BORRADO{CASCADA|RESTRICTIVO}</td><td>RI:
Todo valor de a_0 de la relación A aparece al menos una vez en el atributo a_0 de la relación R</td></tr>
<tr><td>Comentarios</td><td colspan="2">En este caso el borrado restrictivo de A no es posible por la restricción de existencia que tiene en R que a nivel lógico implica que toda tupla de A debe aparecer referenciada en R. El borrado a nulos tampoco es posible porque la clave ajena forma parte de la clave primaria de R.</td></tr>
</table>

<table>
<tr><td>Esquema Conceptual</td><td>
</td></tr>
<tr><td>Esquema Relacional</td><td>A(a_0: t_a_1, a_1: t_a_1,…)
CP: {a_0}
B(a_0: t_a_0, b_1: t_b_1, b_2: t_b_2,…)
CP: {a_0}
CAj: {a_0} → A
BORRADO {RESTRICTIVO|CASCADA}
C(a_0: t_a_0, c_1: t_c_1, c_2: t_c_2,…)
CP: {a_0}
CAj: {a_0} → A
BORRADO {RESTRICTIVO|CASCADA}</td></tr>
<tr><td>Comentarios</td><td>En este caso el borrado a nulos no es posible ya que el atributo a_0 es la clave primaria de la relación B. El borrado en cascada representa el borrado de ocurrencias de las entidades especializadas cuando se borra en la entidad general. El borrado restrictivo supone que sólo se puede borrar una ocurrencia de la entidad general si no está especializada.</td></tr>
</table>

<table>
<tr><td>Esquema Conceptual</td><td colspan="2">
</td></tr>
<tr><td>Esquema Relacional</td><td>A(a_0: t_a_1, a_1: t_a_1,…)
  CP: {a_0}
B(a_0: t_a_0, b_1: t_b_1, b_2: t_b_2,…)
  CP: {a_0}
  CAj: {a_0} → A
    BORRADO CASCADA
C(a_0: t_a_0, c_1: t_c_1, c_2: t_c_2,…)
  CP: {a_0}
  CAj: {a_0} → A
    BORRADO CASCADA</td><td>RI:
Todo valor de a_0 de la relación A aparece al menos una vez en el atributo a_0 de la relación B o en el atributo a_0 de la relación C.</td></tr>
<tr><td>Comentarios</td><td colspan="2">En este caso el borrado a nulos no es posible ya que el atributo a_0 es la clave primaria de la relación B. El borrado restrictivo no es posible ya que toda ocurrencia de la entidad general debe estar en al menos una de las especializaciones.</td></tr>
</table>

<table>
<tr><td>Esquema Conceptual</td><td>
</td></tr>
<tr><td>Esquema Relacional</td><td>A(a_0: t_a_1, a_1: t_a_1)
  CP: {a_0}
A2(a_0: t_a_0, a_2: t_a_2)
  CP: {a_0}
  CAj: {a_0} → A
    BORRADO CASCADA</td></tr>
<tr><td>Comentarios</td><td>En este caso la única directriz razonable es el borrado en cascada ya que las tuplas de A2 almacenan el valor de una propiedad de A y por lo tanto deben desaparecer cuando desaparezca la tupla a la que hacen referencia.</td></tr>
</table>

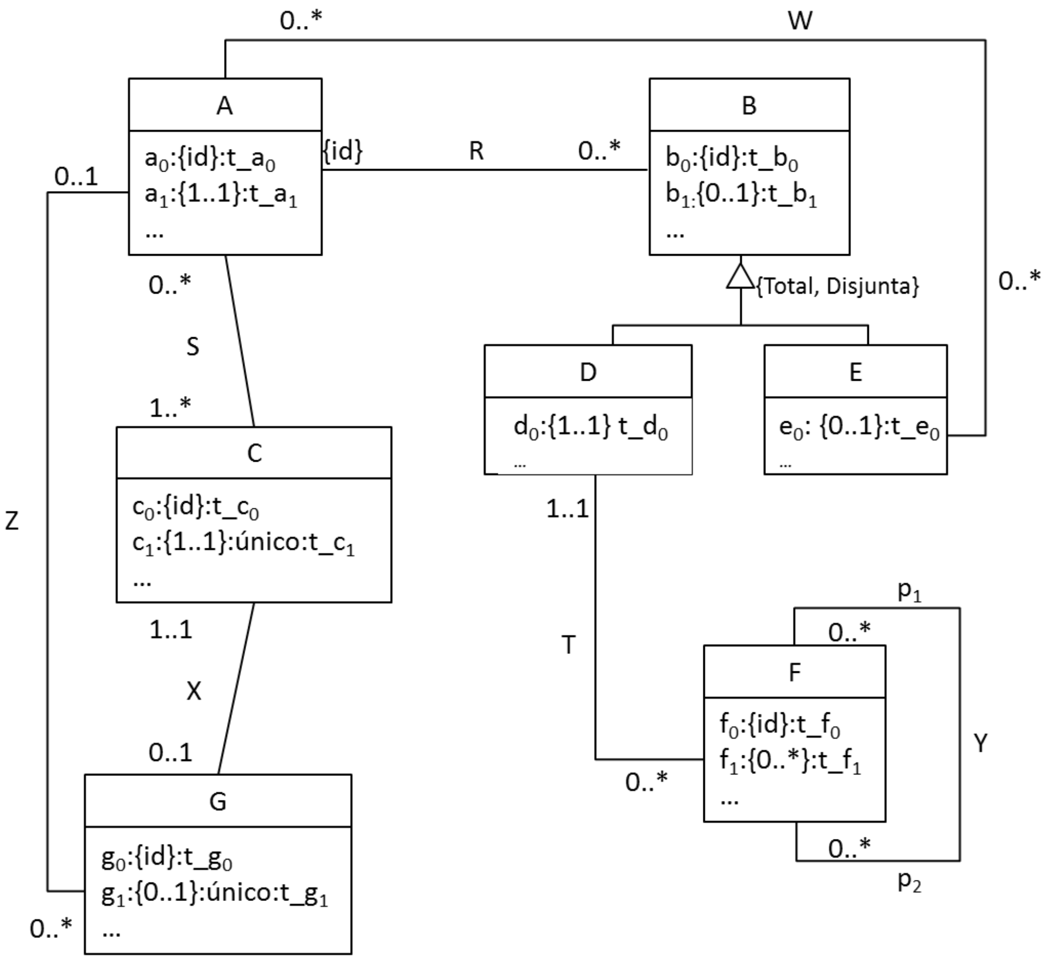
Hoy en día los SGBD no suelen tener implementadas todas las directrices de clave ajena. Cuando la directriz deseada no esté disponible en el SGBD que se vaya a utilizar[5], será necesario simular el comportamiento deseado por programa, esto podrá hacerse siguiendo dos estrategias:

[5] El Access no tiene borrado ni modificación a nulos y el ORACLE además de estas dos tampoco tiene modificación en cascada.

- Definiendo la comprobación de la integridad como diferida (UD2.3) y simulando la directriz deseada por programa (esto se podría hacer así en ORACLE).
- No especificando la clave ajena debiendo definirse una restricción para controlar la integridad referencial y las directrices deseadas (en ACCES se utilizaría esta estrategia).

4 EJEMPLO DE TRANSFORMACIÓN

Sea el siguiente diagrama de clases:



Aplicando las transformaciones que se han presentado, se obtendría el siguiente esquema relacional (recuérdese que las directrices de restauración de la integridad referencial por defecto son "borrado y modificación restrictivos").

A(a_0:t_a_0,a_1:t_a_1,…)
  CP:{a_0}
  VNN:{a_1}
B(a_0:t_a_0,b_0:t_b_0,b_1:t_b_1,…)
  CP:{a_0,b_0}
  CAj:{a_0}→A(a_0)
    Modificación en cascada
C(c_0:t_c_0,c_1:t_c_1,…)
  CP:{c_0}
  VNN:{c_1}
  Úni:{c_1}
D(a_0:t_a_0,b_0:t_b_0,d_0:t_d_0,…)
  CP:{a_0,b_0}
  VNN:{d_0}
  CAj:{a_0,b_0}→B(a_0,b_0)
    Borrado en cascada
    Modificación en cascada
E(a_0_E:t_a_0,b_0_E:t_b_0,e_0:t_e_0,…)
  CP:{a_0_E,b_0_E }
  CAj:{a_0_E,b_0_E }→B
    Borrado en cascada
    Modificación en cascada
F(f_0:t_f_0,…,a_0_D:t_a_0,b_0_D:t_b_0)
  CP:{f_0}
  CAj:{a_0_D,b_0_D}→D(a_0,b_0)
    Modificación en cascada
  VNN:{a_0_D,b_0_D}
F1(f_0:t_f_0,f_1:t_f_1)
  CP:{f_0,f_1}
  CAj:{f_0}→F(f_0)
    Borrado en cascada
    Modificación en cascada
G(g_0:t_g_0,g_1:t_g_1,…,c_0:t_c_0,a_0:t_a_0)
  CP:{g_0}
  Úni:{g_1}
  CAj:{c_0}→C(c_0)
    Borrado en cascada
    Modificación en cascada
  VNN:{c_0}
  Úni:{c_0}
  CAj:{a_0}→A(a_0)
    Modificación en cascada
S(a_0:t_a_0,c_0:t_c_0)
  CP:{a_0,c_0}
  CAj:{a_0}→A(a_0)
    Borrado en cascada
    Modificación en cascada
  CAj:{c_0}→C(c_0)
    Modificación en cascada
W(a_0:t_a_0,a_0_E:t_a_0,b_0_E:t_b_0)
  CP:{a_0,a_0_E,b_0_E}
  CAj:{a_0}→A(a_0)
    Modificación en cascada
  CAj:{a_0_E,b_0_E}→E(a_0_E,b_0_E)
    Modificación en cascada
Y(f_0_p_1:t_f_0,f_0_p_2:t_f_0)
  CP:{f_0_p_1,f_0_p_2}
  CAj:{f_0_p_1}→F(f_0)
    Modificación en cascada
  CAj:{f_0_p_2}→F(f_0)
    Modificación en cascada

Restricciones de integridad:

RI1: Restricción de existencia de *A* en *S*: Todo valor del atributo a_0 de la relación *A* debe aparecer en el atributo a_0 de la relación *S*.
RI2: Especialización total: Todo par de valores de los atributos (a_0,b_0) que aparezca en una tupla de la relación *B* debe aparecer en una tupla de la relación *D* o en una tupla de la relación *E*.
RI3: Especialización disjunta: No puede haber un par de valores de los atributos (a_0,b_0) que aparezca a la vez en las relaciones *D* y *E*.

5 Teoría de la Normalización

Antes de considerarlo definitivo, el esquema relacional obtenido con las transformaciones presentadas 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*.

En este documento se van a presentar sólo las tres primeras formas normales. Formas normales más avanzadas se estudiarán en otras asignaturas de alguna de las ramas específicas.

5.1 Conceptos previos

Dependencia funcional

Sea R un esquema de relación cuyo conjunto de atributos es $A = \{A_1, A_2, \ldots, A_n\}$; una dependencia funcional entre dos conjuntos de atributos X e Y ($X \subseteq A$, $Y \subseteq A$, $X \neq Y$), 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 $t_1[X]$[6] $= t_2[X]$ entonces se debe cumplir $t_1[Y] = t_2[Y]$". Se suele decir entonces que X determina a Y o, 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*".

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

1. {DNI} → {nombre}[7]
2. {DNI, ISBN} → {nombre}
3. {DNI, título} → {nombre}
4. {DNI, pesetas} → {nombre}
5. {DNI, ISBN, título} → {nombre}
6. {DNI, ISBN, pesetas} → {nombre}
7. {DNI, título, pesetas} → {nombre}
8. {DNI, ISBN, título, pesetas} → {nombre}
9. {ISBN} → {título}
10. {ISBN, DNI } → {título}
11. {ISBN, nombre} → {título}
12. {ISBN, pesetas} → {título}
13. {ISBN, DNI, nombre} → {título}
14. {ISBN, DNI, pesetas} → {título}
15. {ISBN, nombre, pesetas} → {título}
16. {ISBN, DNI, nombre, pesetas} → {título}
17. {DNI, ISBN} → {pesetas}
18. {DNI, ISBN, nombre} → {pesetas}
19. {DNI, ISBN, título} → {pesetas}
20. {DNI, ISBN, nombre, título} → {pesetas}

Aunque pudiera parecerlo, este conjunto de dependencias funcionales no está completo ya que

[6] Con esta notación se representa el valor que la tupla t_1 toma en los atributos del conjunto *X*.

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

por ejemplo no aparece la siguiente: {DNI, ISBN} → {título, nombre}.

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 {nombre} → {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.*

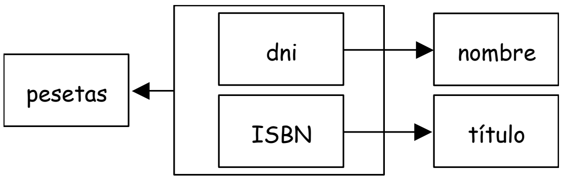
De las 20 dependencias funcionales del ejemplo anterior son completas las siguientes[8]:

1. {DNI} → {nombre}
9. {ISBN} → {título}
17. {DNI, ISBN } → {pesetas}

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 enmarcar los atributos o conjuntos de atributos y flechas para denotar la dependencia funcional. Normalmente sólo se representan las dependencias funcionales completas.

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 $A = \{A_1, A_2, \ldots, A_n\}$, 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.[9]

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 $A = \{A_1, A_2, \ldots, A_n\}$, cualquier subconjunto de atributos de ese esquema ($C \subseteq A$) del que dependan

[8] Estas dependencias pueden obtenerse aplicando la definición que se acaba de dar.

[9] Aunque sólo se van a considerar relaciones que no tienen restricciones de unicidad, se ha conservado la definición genérica de clave por coherencia.

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 $A = \{A_1, A_2, \ldots, A_n\}$, 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*.

5.2 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_tel[10], dir:dom_dir)
CP: {vcod}

Donde los dominios son los siguientes:

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

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

[10] 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}.

múltiples problemas 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:

a) un conjunto, o
b) un registro.

Estas dos situaciones aparecen cuando el objeto del diagrama entidad-relación del cual proviene *A* tiene un atributo multivaluado 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_núm* y *ciudad: dom_ciu*.

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 un teléfono no puede ser compartido por varios proveedores (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)
CP: {teléfono}
CAj: {vcod} → Proveedor
VNN: {vcod}

Por otra parte, si varios proveedores pueden tener un mismo teléfono entonces la clave primaria sería {*vcod,teléfono*} quedando el esquema de relación como sigue:

Lista_tel(vcod: dom_vcod, teléfono: dom_tel)
CP: {teléfono,vcod}
CAj: {vcod} → Proveedor

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 clave (la clave primaria).

5.3 Segunda forma normal (2FN)

Definición de la 2FN

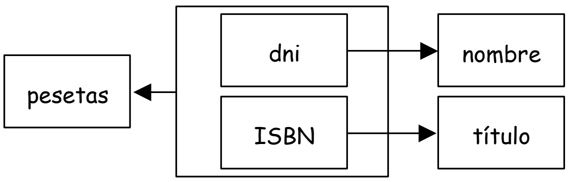
Una relación *R* está en 2FN si está en 1FN y todo atributo no-primo depende funcionalmente de forma completa de toda clave de *R*.

Problemas que resuelve

Sea de nuevo el esquema relacional *Ha_escrito* del apartado 5.1:

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	María 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.
- 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*, llámese *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: *R*1=*R*[*S*, *A*] y *R*2=R[*CP*, *B*][11]. La clave primaria de *R*1 es *S* y la clave primaria de *R*2 es *CP*; además *R*2 tiene una clave ajena que hace referencia a *R*1. Así, las dos relaciones resultantes quedan en 2FN. Esta forma de realizar las proyecciones y la definición de clave ajena en *R*2 asegura que la información contenida en *R*, por un lado, y en *R*1 y *R*2 es idéntica.

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

Autor(DNI: dom_dni, nombre: dom_nom)
CP: {DNI}

Libro(ISBN: dom_IS, título: dom_tit)
CP: {ISBN}

Ha _escrito(DNI: dom_dni, ISBN: dom_IS, pesetas: dom_pes)
CP: {DNI, ISBN}
CAj: {DNI} $\rightarrow$ Autor
CAj: {ISBN} $\rightarrow$ Libro

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

Autor

DNI	nombre
17.897.569	Pepe Pérez
54.325.658	Juan Gómez
15.236.588	María 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
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:

[11] En caso de no existir el atributo *B* de *R,* esta segunda proyección sería *R*2=*R*[CP].

- 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.

5.4 Tercera forma normal (3FN)

Definición de la 3FN

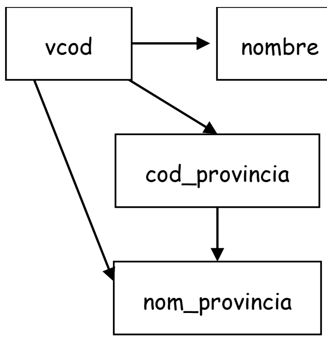
Una relación *R* está en 3FN si está en 2FN y no hay dependencias funcionales entre atributos no-primos.

Problemas que resuelve

Sea el siguiente esquema de relación:

Proveedor(vcod: dom_vcod, nombre: dom_nom, nom_provincia: dom_nomp,
cod_provincia: dom_codp)
CP: {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*[12].

Los problemas de la manipulación de esta relación son causados por esa dependencia funcional que implica la posible 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

[12] Esta dependencia se detecta ya que las provincias tiene códigos únicos.

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 exista 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 nombre de una provincia hay que realizarla tantas veces como proveedores vivan en esa ciudad, ya que esa información es redundante y está replicada.

Paso a 3 FN

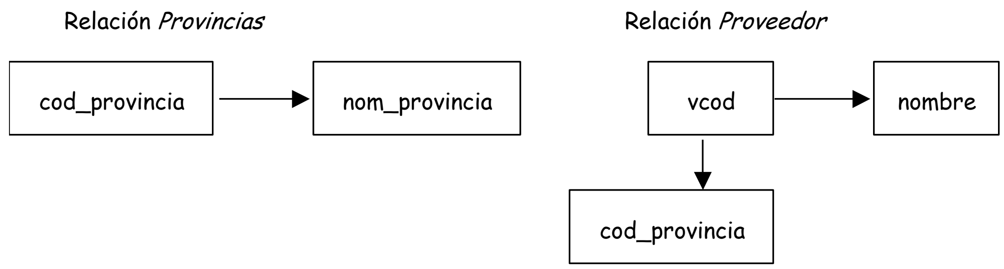
La técnica para transformar una relación que está en 2FN pero no en 3FN en un conjunto de relaciones que están en 3FN es la siguiente: supóngase que *R* es una relación con clave primaria *CP* que está en 2FN y no en 3FN. Supóngase que *B* y *A* son dos atributos no-primos y que *B* depende funcionalmente de *A*. Las relaciones que se obtienen como proyección de *R* son *R*1=*R*[*CP,C,A*] (donde *C* es el resto de atributos no-primos de *R*) y *R*2=*R*[*A,B*]. La clave primaria de *R*1 es *CP* y además *A* es clave ajena que hace referencia a *R*2. La clave primaria de *R*2 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)
CP: {vcod}
CAj: {cod_provincia} → Provincia

Provincia(cod_provincia: dom_codp, nom_provincia: dom_nomp)
CP: {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:

Proveedor

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.

5.5 Reflexión sobre la Teoría de la Normalización

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 *C*1 de relaciones que estén en 1FN.

b) Comprobar si las relaciones de *C1* están en 2FN. Si alguna relación no está en 2FN debe descomponerse en un conjunto de relaciones, sea *C*2, que estén en 2FN.

c) Comprobar si las relaciones de *C*2 están en 3FN. Si alguna relación no está en 3FN, debe descomponerse debe descomponerse en un conjunto que sí lo esté.

Como reflexión final, hay que decir que si, el esquema relacional que se está considerando se ha obtenido como transformación de un diseño conceptual correcto, la teoría de la normalización sólo será necesaria para resolver el problema de los atributos con cardinalidad máxima *. Sin embargo, es importante conocer las formas normales para corregir esquemas relacionales incorrectos y también porque en ocasiones, puede llegar a ser razonable, por cuestiones de eficiencia, trabajar con esquemas relacionales que no estén en 3FN[13], pero en este caso es fundamental saber qué problemas pueden presentar y cómo controlarlos.

[13] Esto se ve en estudios muy avanzados de bases de datos.