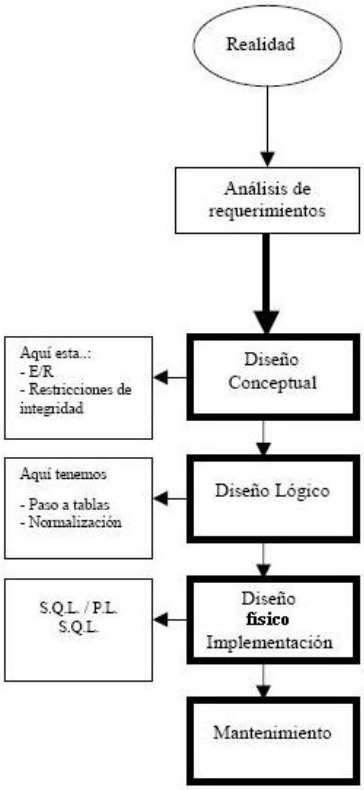
TEMA 5.
DISEÑO LÓGICO

1. INTRODUCCIÓN

El objetivo de las diferentes fases del diseño de base de datos es obtener un conjunto de datos y un conjunto de operaciones sobre ellos que permiten satisfacer las necesidades de la organización.



La creación o diseño de una base de datos para por las siguientes **etapas**:

- **Diseño conceptual:** La realidad se representa por medio de algún tipo de esquema conceptual, diagrama Entidad-Relación. Esta fase es independiente del modelo de base de datos a utilizar.

- **Diseño lógico:** El esquema conceptual se transforma en un esquema lógico aplicando las transformaciones adecuadas. Hay diversos modelos lógicos: *relacional, jerárquico, red, orientado a objetos*. En nuestro caso utilizaremos un ***modelo relacional***.

- **Diseño físico o implementación:** Traducción de las tablas obtenidas en la fase de diseño lógico a estructuras entendibles por un SGBD, mediante el lenguaje apropiado dependiendo del modelo lógico elegido. En nuestro caso relacional → SQL.

2. DISEÑO LÓGICO

El **diseño lógico** consiste en convertir el *esquema conceptual* de datos (obtenido en la fase anterior) en un *esquema lógico* y en un *conjunto de transacciones* que expresan las propiedades estáticas y dinámicas del sistema de información. Para ello se realiza la transformación del modelo E/R al modelo relacional.

El objetivo de este apartado es presentar esas transformaciones. Para ello se supondrá que el sistema de gestión de base de datos relacional (SGBD) soporta para cada relación, la definición de *clave primaria*, *clave ajena*, *restricción de unicidad* y *restricción de valor no nulo*.

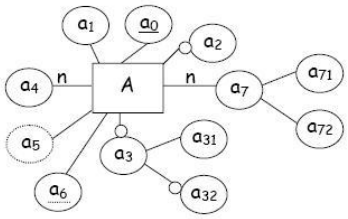
Para realizar el diseño lógico se estudia en primer lugar como se lleva a cabo la *correspondencia entre el diagrama entidad / relación y el esquema lógico relacional*. A continuación, se presenta toda la *teoría de normalización* que permite eliminar las redundancias de datos existentes mediante la detección y eliminación de dependencias funcionales.

3. PASO A TABLAS

3.1. TRANSFORMACIÓN DE ENTIDADES

En un diagrama entidad-relación, se pueden distinguir tres tipos de entidades: *entidades fuertes*, *entidades débiles* y *entidades especializadas*.

3.1.1. Entidad fuerte



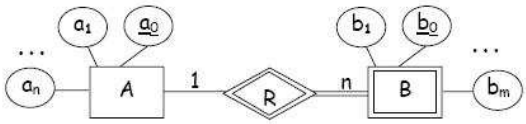
La relación equivalente es la siguiente:

$A(a_0: dom\_a_0, a_1: dom\_a_1, a_2: dom\_a_2, a_3.a_{31}: dom\_a_{31}, a_3.a_{32}: dom\_a_{32}, \{a_4: dom\_a_4\}, a_5: dom\_a_5, a_6: dom\_a_6, \{a_7.a_{71}: dom\_a_{71}, a_7.a_{72}: dom\_a_{72}\})$

Clave Primaria: $\{a_0\}$
Valor No Nulo: $\{a_2\}$
Valor No Nulo: $\{a_3.a_{32}\}$
Único: $\{a_6\}$

3.1.2. Entidad débil

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

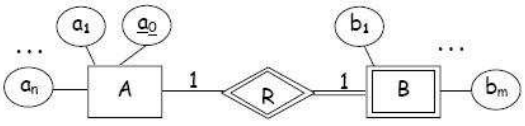


Las relaciones correspondientes son:

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

$B(b_0: dom\_b_0, b_1: dom\_b_1, \ldots, b_m: dom\_b_m, a_0: dom\_a_0)$
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:



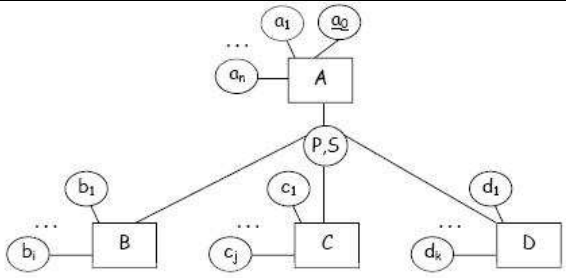
cuyo esquema relacional equivalente es:

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

$B(b_1: dom\_b_1, \ldots, b_m: dom\_b_m, a_0: dom\_a_0)$
Clave Primaria: $\{a_0\}$
Clave Ajena: $\{a_0\}$ hace referencia a A

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

3.1.3. Entidad especializadas



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

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

$B(a_0: dom\_a_0, b_1: dom\_b_1, \ldots, b_i: dom\_b_i)$
Clave Primaria: $\{a_0\}$
Clave Ajena: $\{a_0\}$ hace referencia a A

$C(a_0: dom\_a_0, c_1: dom\_c_1, \ldots, c_j: dom\_c_j)$
Clave Primaria: $\{a_0\}$
Clave Ajena: $\{a_0\}$ hace referencia a A

$D(a_0: dom\_a_0, d_1: dom\_d_1, \ldots, d_k: dom\_d_k)$
Clave Primaria: $\{a_0\}$
Clave Ajena: $\{a_0\}$ hace referencia a A

Sólo cuando la especialización es *Parcial* y *Solapada* existe una transformación en relaciones totalmente adecuada; en los demás casos es necesario introducir ciertas restricciones de integridad.

Total: tenemos que expresar que el hecho de que toda ocurrencia de la entidad general A tiene que estar asociada con al menos una ocurrencia de alguna entidad especializada. La definición en SQL es la siguiente:

```
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))
```

Disjunta: cada ocurrencia de la entidad general sólo puede estar asociada con una ocurrencia de una entidad especializada.

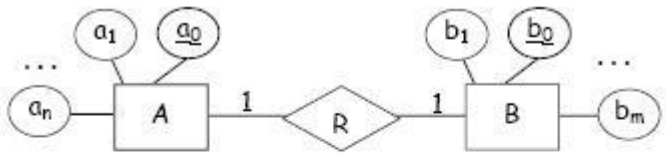
```
CREATE ASSERTION Disjunta CHECK
      NOT EXISTS (SELECT * FROM B Bx, C Cx, D Dx
                  WHERE Dx.a0= Bx.a0 OR Dx.a0= Cx.a0 OR Bx.a0= Cx.a0)
```

3.2. TRANSFORMACIÓN DE RELACIONES

La transformación de las relaciones se va a estudiar atendiendo al grado y a las cardinalidades máximas y mínimas de la relación a transformar. Por último se presentará el caso de la presencia de atributos en la relación.

3.2.1. Relaciones binarias

- **Relación binaria 1:1 sin restricciones de existencia**



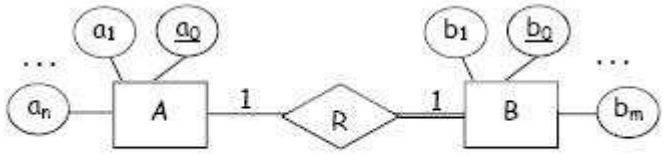
Esquema

A(a_0: dom_a_0, a_1: dom_a_1,..., a_n: dom_a_n)
Clave Primaria: {a_0}

B (b_0: dom_b_0, b_1: dom_b_1,..., b_m: dom_b_m, a_0: dom_a_0)
Clave Primaria: {b_0}
Único: {a_0}
Clave Ajena: {a_0} hace referencia a A

Como puede observarse, la relación R se representa 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 con restricción de unicidad para representar las cardinalidad máxima que es 1.

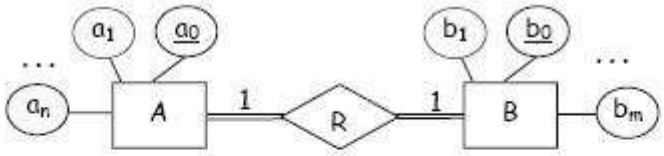
- **Relación binaria 1:1 con una restricción de existencia**



A(a_0: dom_a_0, a_1: dom_a_1,..., a_n: dom_a_n)
Clave Primaria: {a_0}

B(b_0: dom_b_0, b_1: dom_b_1,..., b_m: dom_b_m, a_0: dom_a_0)
Clave Primaria: {b_0}
Único: {a_0}
Valor No Nulo: {a_0}
Clave Ajena: {a_0} hace referencia a A

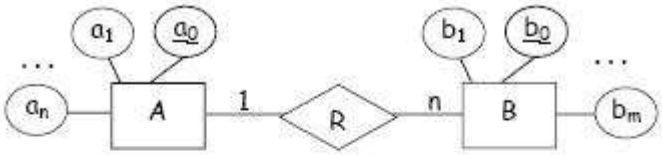
- **Relación binaria 1:1 con dos restricciones de existencia**



A-B(a_0:dom_a_0, a_1:dom_a_1,..., a_n:dom_a_n, b_0: dom_b_0, b_1: dom_b_1,..., b_m: 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. Esta solución complica la manipulación de los objetos representados, así que en algunos casos será necesario representar con relaciones y restricciones de integridad.

- **Relación binario 1:M sin restricciones de existencia**

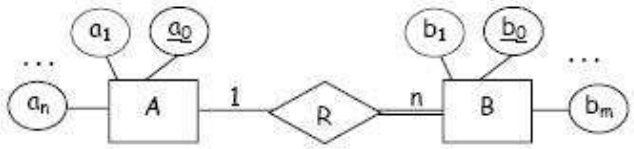


A(a_0: dom_a_0, a_1: dom_a_1,..., a_n: dom_a_n)
Clave Primaria: {a_0}

B(b_0: dom_b_0, b_1: dom_b_1,..., b_m: dom_b_m, a_0: 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 representarla como una clave ajena en la entidad con cardinalidad máxima N.

- **Relación binario 1:M con una restricción de existencia sobre N**

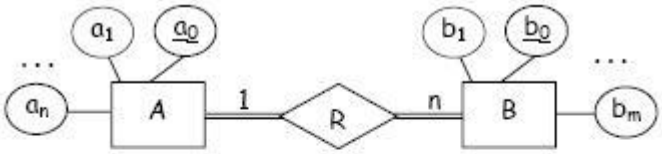


A(a_0: dom_a_0, a_1: dom_a_1,..., a_n: dom_a_n)
Clave Primaria: {a_0}

B(b_0: dom_b_0, b_1: dom_b_1,..., b_m: dom_b_m, a_0: 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 mediante la restricción de valor no nulo sobre la clave ajena que representa la relación R.

- **Relación binario 1:M con dos restricciones de existencia**



A(a_0: dom_a_0, a_1: dom_a_1,..., a_n: dom_a_n)
Clave Primaria: {a_0}

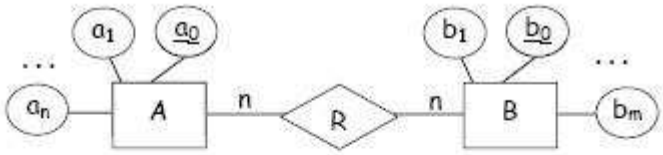
B(b_0: dom_b_0, b_1: dom_b_1,..., b_m: dom_b_m, a_0: dom_a_0)
Clave Primaria: {b_0}
Clave Ajena: {a_0} hace referencia a A
Valor No Nulo: {a_0}

Como puede apreciarse, este esquema es idéntico al caso anterior, por tanto la restricción de existencia sobre la relación A no está representada. Utilizaremos el lenguaje SQL para representar la restricción.

```
CREATE ASSERTION Restr_existencia CHECK
    NOT EXISTS (SELECT * FROM A Ax
                WHERE NOT EXISTS (SELECT * FROM B Bx
                                  WHERE Bx.a0 = Ax.a0))
```

El caso de una *relación binaria 1:M con una única restricción de existencia sobre 1* se realiza de forma equivalente al caso de una relación binaria 1:M sin restricciones, siendo necesaria la expresión del cálculo relacional para representar la restricción.

- **Relación binaria M:M sin restricciones de existencia**



A(a_0: dom_a_0, a_1: dom_a_1,..., a_n: dom_a_n)
Clave Primaria: {a_0}

B(b_0: dom_b_0, b_1: dom_b_1,..., b_m: dom_b_m)
Clave Primaria: {b_0}

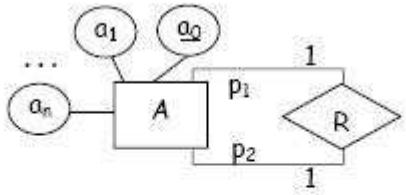
R(a_0: dom_a_0, b_0: dom_b_0)
Clave Primaria: {a_0, b_0}
Clave Ajena: {a_0} hace referencia a A
Clave Ajena: {b_0} hace referencia a B

En este caso, debido a que las dos cardinalidades máximas son N, es necesaria la representación de la relación binario R mediante una relación independiente. La clave primaria de esta relación es el par {a_0, b_0} para satisfacer las condiciones exigidas.

Los casos de relaciones binaria M:M en los que aparecen restricciones de existencia se diseñan de forma análoga, siendo necesario escribir las restricciones de existencia en lenguaje SQL.

3.2.2. Relaciones reflexivas

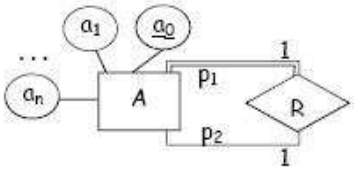
- **Relación reflexiva 1:1 sin restricciones de existencia**



A(a_0: dom_a_0, a_1: dom_a_1,..., a_n: dom_a_n, a_0_p_2: dom_a_0)
Clave Primaria: {a_0}
Único: {a_0_p_2}
Clave Ajena: {a_0_p_2} hace referencia a A

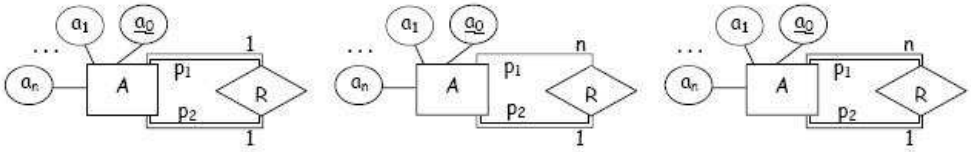
En la figura, p_1 y p_2 son dos etiquetas que indican el papel que cada ocurrencia de A juega en la relación. En este caso, la relación R queda correctamente representada mediante la clave ajena {a_0_p_2} en la relación que representa a la entidad. Además, esta clave también se define con restricción de unicidad para representar la cardinalidad máxima 1.

- **Relación reflexiva 1:1 con una restricción de existencia**

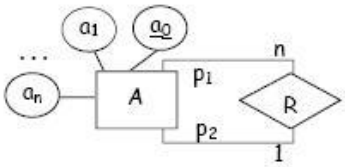


A(a_0: dom_a_0, a_1: dom_a_1,..., a_n: dom_a_n, a_0_p_2: dom_a_0)
Clave Primaria: {a_0}
Único: {a_0_p_2}
Clave Ajena: {a_0_p_2} hace referencia a A
Valor No Nulo: {a_0_p_2}

Curiosamente, este diagrama E-R puede representar la misma realidad que los tres siguientes:

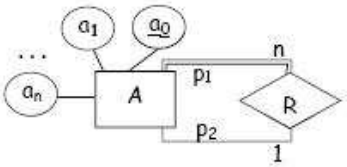


- **Relación reflexiva 1:M sin restricciones de existencia**



A(a_0: dom_a_0, a_1: dom_a_1,..., a_n: dom_a_n, a_0_p_2: dom_a_0)
Clave Primaria: {a_0}
Clave Ajena: {a_0_p_2} hace referencia a A

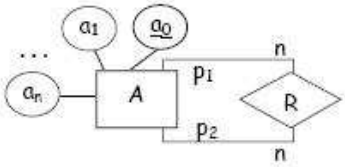
- **Relación reflexiva 1:M con una restricción de existencia sobre N**



A(a_0: dom_a_0, a_1: dom_a_1,..., a_n: dom_a_n, a_0_p_2: 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 valor no nulo sobre la clave ajena {a_0_p_2} que representa la relación R.

- **Relación reflexiva M:M sin restricciones de existencia**



A(a_0: dom_a_0, a_1: dom_a_1,..., a_n: dom_a_n)
Clave Primaria: {a_0}

R(a_0_p_1: dom_a_0, a_0_p_2: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

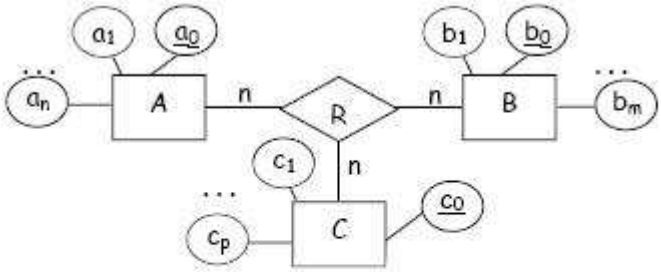
En este caso, debido a que la relación reflexiva E tiene cardinalidad máxima M:M, es necesario utilizar una relación independiente para representarla. Si hubiera definida alguna restricción de existencia, también sería necesaria la inclusión de una expresión en lenguaje SQL. La presencia de restricciones de existencia en las relaciones reflexivas es, sin embargo, poco frecuente.

3.2.3. Relaciones ternarias

En este apartado se muestran cuatro casos. 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 mediante restricciones.

Las **relaciones ternarias** siempre generan una tabla para cada entidad más una cuarta tabla para representar la relación ternaria.

- **Relación ternaria M:M:M sin restricciones de existencia**



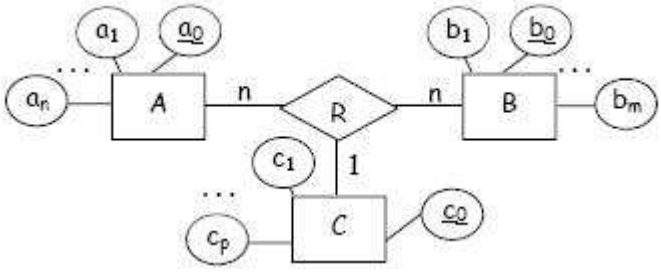
A(a_0: dom_a_0, a_1: dom_a_1,..., a_n: dom_a_n)
Clave Primaria: {a_0}

B(b_0: dom_b_0, b_1: dom_b_1,..., b_m: dom_b_m)
Clave Primaria: {b_0}

C(c_0: dom_c_0, c_1: dom_c_1,..., c_L: dom_c_p)
Clave Primaria: {c_0}

R(a_0: dom_a_0, b_0: dom_b_0, c_0: 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

- **Relación ternaria 1:M:M sin restricciones de existencia**



R(a_0: dom_a_0, b_0: dom_b_0, c_0: 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

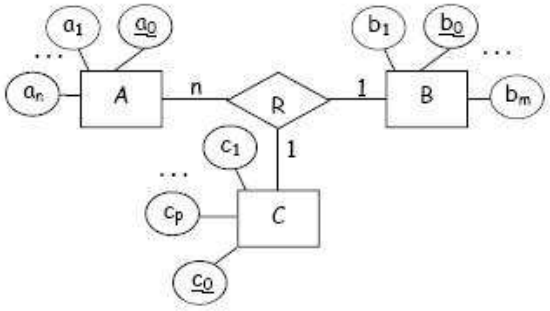
Restricción de VNN sobre el atributo {c_0}, que es la clave primaria de la entidad C. Como regla general, se deberá definir una restricción de VNN sobre todo atributo correspondiente a una clave ajena que no esté presente en la CP de la relación.

- **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: dom_a_0, b_0: dom_b_0, c_0: 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*



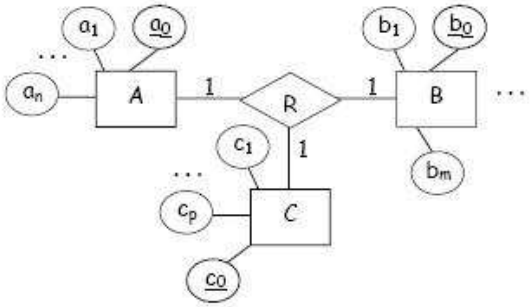
Debido a la existencia de dos cardinalidades máximas 1, existen dos claves candidatas para la relación R, cualquiera de las dos puede ser clave primaria, teniendo que definirse para la otra una restricción de unicidad, además de la restricción VNN sobre la clave ajena que no forma parte de la clave primaria.

- **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:

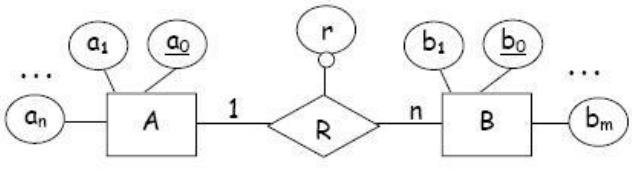
R(a_0: dom_a_0, b_0: dom_b_0, c_0: dom_c_0)
Clave Primaria: {a_0, b_0}
Único: {a_0, c_0}
Único: {b_0, c_0}
Valor No Nulo: {c_0}
Clave Ajena: {a_0} hace referencia a *A*
Clave Ajena: {b_0} hace referencia a *B*
Clave Ajena: {c_0} hace referencia a *C*



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 restricciones de unicidad.

3.2.4. Atributos en las relaciones

Cuando una relación R tiene atributos propios, éstos deben incluirse siempre en la relación donde se represente R. 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.



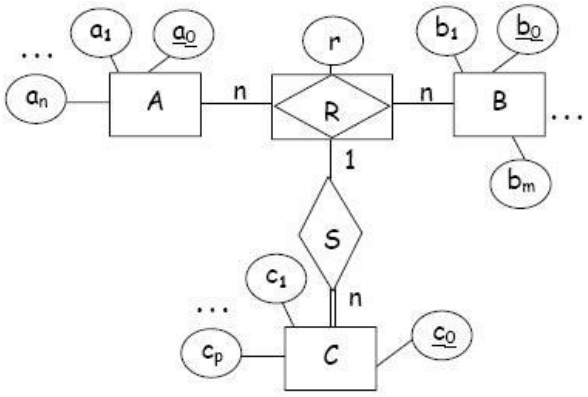
A(a_0: dom_a_0, a_1: dom_a_1,..., a_n: dom_a_n)
Clave Primaria: {a_0}

B(b_0: dom_b_0, b_1: dom_b_1,..., b_m: dom_b_m)
Clave Primaria: {b_0}

R(b_0: dom_b_0, a_0: dom_a_0, r: dom_r)
Clave Primaria: {b_0}
Valor No Nulo: {a_0}
Valor No Nulo: {r}
Clave Ajena: {a_0} hace referencia a A
Clave Ajena: {b_0} hace referencia a B

3.3. TRANSFORMACIÓN DE LA AGREGACIÓN

Generalizar el diseño de los objetos agregados 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.



A(a_0: dom_a_0, a_1: dom_a_1,..., a_n: dom_a_n)
Clave Primaria: {a_0}

B(b_0: dom_b_0, b_1: dom_b_1,..., b_m: dom_b_m)
Clave Primaria: {b_0}

R(a_0: dom_a_0, b_0: dom_b_0, r: dom_r)
Clave Primaria: {a_0, b_0}
Clave Ajena: {a_0} hace referencia a A
Clave Ajena: {b_0} hace referencia a B

C(c_0: dom_c_0, c_1: dom_c_1,..., c_p: dom_c_p, a_0: dom_a_0, b_0: dom_b_0)
Clave Primaria: {c_0}
Clave Ajena: {a_0, b_0} hace referencia a R
Valor No Nulo: {a_0, b_0}

Claramente, la relación que está representando al objeto agregado es R. La clave de esta relación es {a_0, b_0}. Esta clave es el identificador del objeto agregado y será usada como tal cuando se diseñen las relaciones en las cuales toma parte el objeto agregado.

La relación C representa tanto a 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.