Bases de Datos

Características necesarias:

- **Independencia:** fundamental y la más importante a la hora de gestionar la información. Se presentaba como problema principal en los años 60. [John Fisher Kennedy].
- **Redundancia:** cuesta gran cantidad de tiempo de cómputo e integridad. Para ello el uso de registros ficticios con punteros.

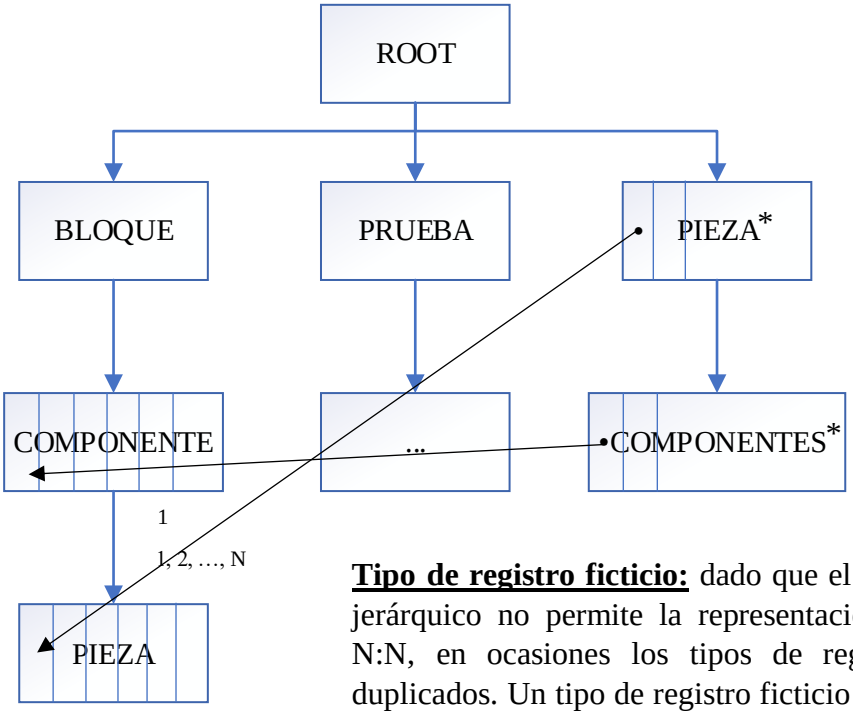
IBM → Estructuras de Datos Jerárquicas → Sistemas de Gestión de Datos → IMS

Tiempo

Modelo navegante a través de punteros mediante la relación "padre - hijo" haciendo uso de los tipos de registro ficticios.

Representación del problema a resolver

Bases de Datos Jerárquicas



Tipo de registro ficticio: dado que el modelo de datos jerárquico no permite la representación de relaciones N:N, en ocasiones los tipos de registro deben ser duplicados. Un tipo de registro ficticio es la duplicación de un tipo de registro existente en el modelo cuyos atributos son punteros a los registros existentes en el tipo de registro original.

Tipos de relación:

- 1:1
- 1:N
- ~~N:N~~

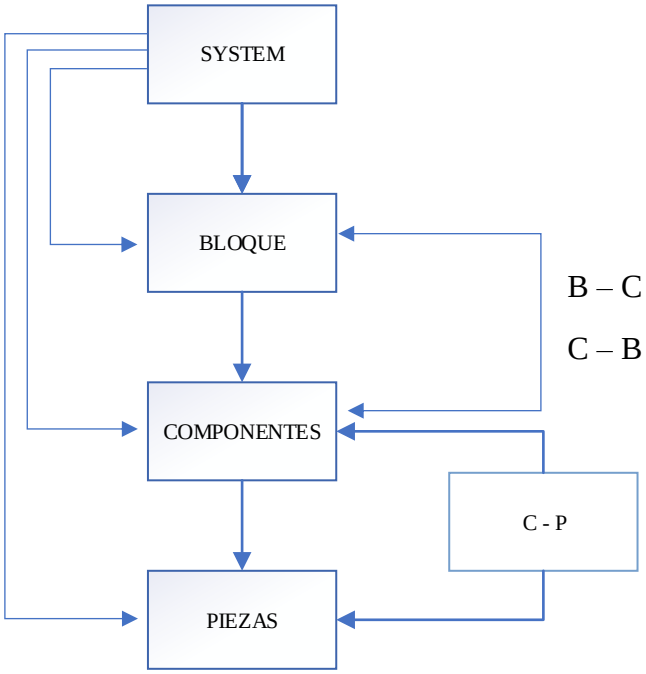
Desarrollo histórico:

- **1960:** Modelo Jerárquico de Red.
- **1970:** Modelo Relacional, desarrollo Oracle.

- **1980:** Modelos de Bases de Datos orientados a objetos basados en modelos relacionales.
- **1990:** Desarrollo del lenguaje SQL.

A finales de los años 60, gracias a un grupo de expertos, se empieza a usar el término de "Base de Datos", gracias a la gestión de datos con gran independencia y una mínima redundancia → DBMS → Codasyl: consorcio de industrias informáticas.

Bases de Datos en Red

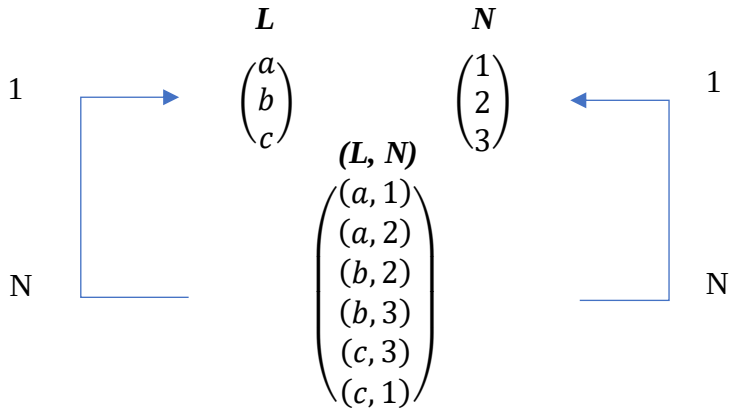


Tipos de relación:

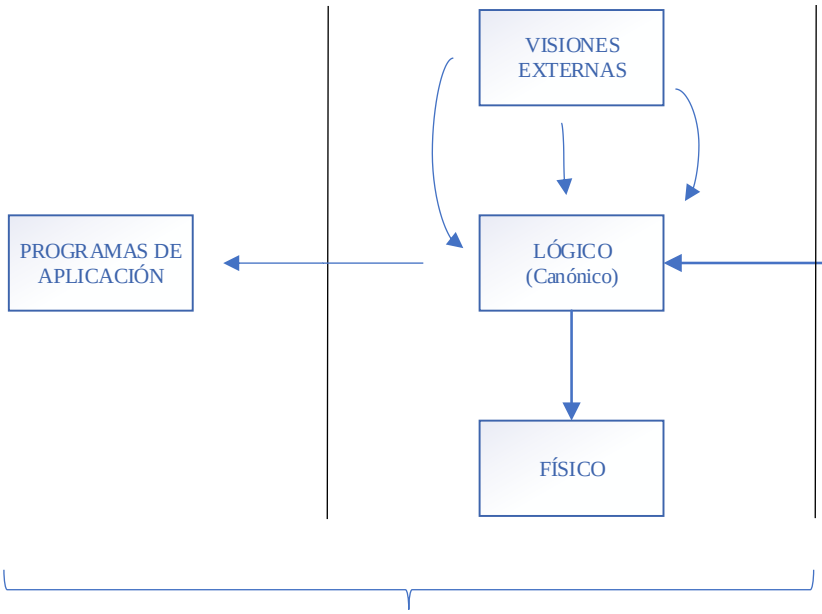
- 1:1
- 1:N

Tipo de registro *Dummy* utilizado en las relaciones N:N en las Bases de Datos en Red. Esta relación se consigue manteniendo una relación 1:N con los dos registros que relaciona mediante dos punteros (uno para cada registro).

Al conjunto de registros de dos tipos se le denomina *set dbtg*, que representa a los objetos del dominio del problema a su vez representados por sus atributos. A la relación entre ellos se le denomina tipo de set, que representa el conjunto de relaciones existentes en el dominio del problema entre esos objetos. En el ejemplo anterior *set dbtg b-c* y *set dbtg c-b*.



NO REDUNDANCIA



INDEPENDENCIA

Si fallamos en la parte conceptual, todo lo que va después también es erróneo.

Características de una Base de Datos

- **Versatilidad:** la información que forma parte del dominio de un problema es única y caracteriza a ese problema o sistema, pueden existir diferentes visiones de esa información.
  Un procedimiento, un programa de aplicación, que maneja la información correspondiente a un problema puede, por tanto, tener en cuenta sólo parte del conjunto de información, mientras que otro procedimiento puede considerar a otro conjunto diferente, o no, de información del mismo problema.

- **Redundancia mínima:** la redundancia implica la existencia de varias copias de un mismo ítem de datos, las cuales pueden, en un momento dado, tener distintos valores.

  Además, hay que tener en cuenta que esta duplicación implica unas necesidades de almacenamiento innecesarias que, aunque el coste de almacenamiento por bit resulte cada vez menor, siempre implicará un coste innecesario.

COPIA DE ÍTEM

Diversas copias	Única copia
Es necesario establecer procedimientos que garanticen la consistencia de la información cuando estas copias sean utilizadas por diferentes procedimientos al mismo tiempo.	Es necesario establecer procedimientos que permitan el acceso a esta copia por varios procedimientos, para garantizar un desempeño aceptable, lo que complica el software en cuestión y la representación del dominio del problema en la base de datos.

- **Desempeño:** las Bases de Datos deben asegurar un tiempo de respuesta adecuado en la comunicación hombre-máquina, permitiendo el acceso simultáneo al mismo o distinto conjunto de ítems de datos por el mismo o distinto procedimiento.

- **Integridad:** hace referencia a la veracidad de los datos almacenados con respecto a la información existente en el dominio del problema que trata la misma. Como los datos de la Base de Datos son manejados por muchos usuarios haciendo uso de muchos procedimientos que tratan los mismos datos de muchas formas, es necesario garantizar que estos datos no sean destruidos ni modificados de forma anómala.

  Pero además de garantizar la integridad de la Base de Datos con respecto a fallos de software, hardware, …, esta integridad debe garantizarse con respecto a la veracidad de los ítems de datos y sus relaciones con respecto al dominio del problema. Así, en una Base de Datos deben establecerse procedimientos que verifiquen que los valores de los datos se ajustan a los requisitos y restricciones extraídas del análisis del problema.

- **Privacidad:** hace referencia a la reserva de la información de la misma a personas no autorizadas.

- **Seguridad:** hace referencia a la capacidad de ésta para proteger los datos contra su pérdida total o parcial por fallos del sistema o por accesos accidentales o intencionados a los mismos.

  Para conseguir estas dos últimas características, una Base de Datos debe satisfacer, al menos, los siguientes requisitos:

  1. Seguridad contra la destrucción de los datos causadas por:
     - El entorno: fuego, robo, inundación…
     - Fallos del sistema, de forma que los datos puedan reconstruirse.
  2. Seguridad contra accesos no autorizados o indebidos a la Base de Datos.

  Deben existir procedimientos de recuperación de la información perdida, como procedimientos que supervisen el acceso a los datos por los usuarios de la Base de Datos.

- **Afinación:** hace referencia a la organización física de la información de la Base de Datos, la cual determina directamente el tiempo de respuesta de los procedimientos que operan sobre la misma.

Para ello la Base de Datos debe ser flexible a la modificación de esta organización física, lo que supone además una migración de los datos según evolucione la Base de Datos, sin que por ello se vean afectados los procedimientos u otras representaciones de los datos pero, sin embargo, se consiga un desempeño más alto.

Granularidad y Ligadura

- **Granularidad:** nivel de detalle en que pueden ser descritas las representaciones externas derivadas de la representación lógica. Una mayor granularidad proporciona una mayor independencia, pero al mismo tiempo una mayor complejidad en el software utilizado para realizar estas representaciones y, por añadidura, en la organización de la información para que estas representaciones puedan ser almacenadas físicamente.

- **Ligadura *(Binding)*:** proceso de vinculación de las diferentes representaciones de la información en la Base de Datos.

    1. **Ligadura lógica:** correspondiente al proceso de vinculación que se produce entre las representaciones externas y la lógica.
    2. **Ligadura física:** correspondiente al proceso de vinculación entre la representación lógica y la física.

La independencia de los datos será mayor cuanto más tardía se realice la ligadura. Si la ligadura se realiza en una fase muy temprana (compilación) implica que los programas de aplicación deberán ser recompilados cada vez que se produzca una modificación de las distintas representaciones de los datos (lógica y física) aunque, por otro lado, el desempeño de los mismos será alto debido a que el tiempo de cómputo que conlleva el proceso de ligadura se consume sólo una vez y no en el arranque del programa o en cada acceso a los datos.

Por otro lado, el que se realice la ligadura en una fase tardía (en cada acceso a los datos) supone que se podrán modificar las representaciones lógica y física de los datos sin que por ello deban traducirse de nuevo a código máquina los programas de aplicación. Sin embargo, el desempeño de estos programas será en principio menor debido a que necesitan un coste computacional añadido para realizar el proceso de ligadura, bien en el arranque del programa o en cada uno de los accesos a los datos.

Definición de Base de Datos

"Una Base de Datos es una colección de archivos relacionados que almacenan tanto una representación abstracta del dominio de un problema del mundo real cuyo manejo resulta de interés para una organización, como los datos correspondientes a la

información acerca del mismo. Tanto la representación como los datos están sujetos a una serie de restricciones, las cuales forman parte del dominio del problema y cuya descripción está también almacenada en esos ficheros."

Es importante diferenciar una Base de Datos de un Sistema de Gestión de Base de Datos (SGBD). Para que la información pueda ser almacenada como se ha descrito y el acceso satisfaga las características exigidas a una base de datos para ser denominada como tal, es necesario que existan una serie de procedimientos (un sistema software) que sea capaz de llevar a cabo tal labor de manera correcta, segura, íntegra y eficiente. A este sistema software es a lo que llamamos SGBD.

Para realizar todas las funciones descritas, el SGBD debe contar con una serie de herramientas:

- **DDL *(Data Definition Language)*:** lenguaje artificial basado en un determinado modelo de datos que permite la representación lógica de los datos.
- **DSDL *(Data Storage Definition Language)*:** se definen los datos correspondientes al dominio de un problema a los dos niveles de abstracción, y a esta definición de los datos se le denomina "Esquema de la Base de Datos".
- **DCL *(Data Control Language)*:** sublenguaje encargado del control y seguridad de los datos. Permite el acceso a la información almacenada en el diccionario de datos, así como el control de la seguridad de los datos.
- **DML *(Data Manipulation Language)*:** lenguaje artificial mediante el cual se realizan dos funciones en la gestión de los datos:
    1. La definición del nivel externo o de usuario de los datos.
    2. La manipulación de los datos.
- **Dictionary Data Language:** además de almacenarse la representación de los datos al nivel externo, lógico y físico, se almacenan un conjunto de reglas que permiten vincular los mismos datos desde un nivel de abstracción y representación a otro. A este conjunto de reglas se le denomina "Mapa de Reglas *(Mapping Rules)*".
- **DBA *(Data Base Administrator)*:**
    1. Define el esquema lógico y físico de la Base de Datos.
    2. Define las visiones de usuario de la Base de Datos.
    3. Controla la privacidad de los datos.
    4. Mantiene los esquemas.
    5. Especifica los procedimientos para el mantenimiento de la seguridad de los datos.
- **Usuarios de la Base de Datos:** componente más del SGBD considerado de gran importancia por los mismos para el correcto funcionamiento de estos. Entre ellos:
    1. Usuarios terminales.
    2. Usuarios técnicos.
    3. Usuarios especializados.
    4. Usuarios críticos.

Representación de los problemas del mundo real

Al conjunto de las propiedades que caracterizan un fenómeno se le denomina datos, y al conjunto de valores que estas propiedades pueden presentar para un determinado fenómeno, junto con el conjunto de las relaciones o dependencias entre las mismas, se le denomina información.

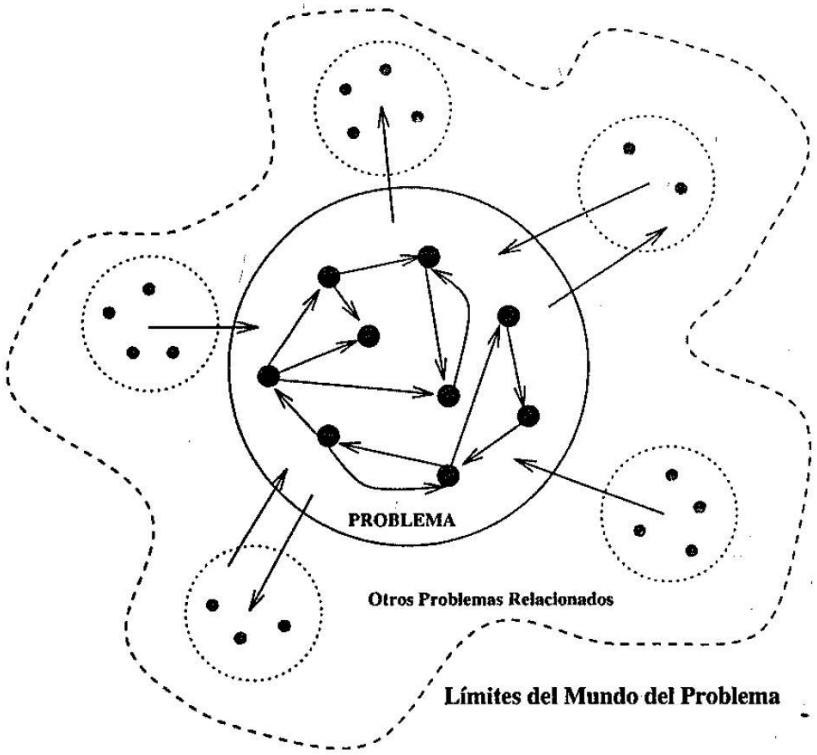
Un modelo es entonces una representación abstracta y holística (que considera algo como un todo) de un determinado fenómeno natural. Cualquier dato que se desee medir para un determinado fenómeno será definido por unos dominios, que son los datos básicos que lo caracterizan.

Así pues, un Modelo de Datos es una unidad de abstracción mediante la cual puede describirse un fenómeno real o abstracto, es decir, se describen las características estáticas y dinámicas.

Los problemas del mundo real

En base a lo anterior, mediante el uso de un modelo de datos puede ser representado cualquier problema sobre el que se desea obtener información para un conocimiento y/o solución.

El primer paso a seguir en la determinación de los límites del problema para poder así simplificarlo:

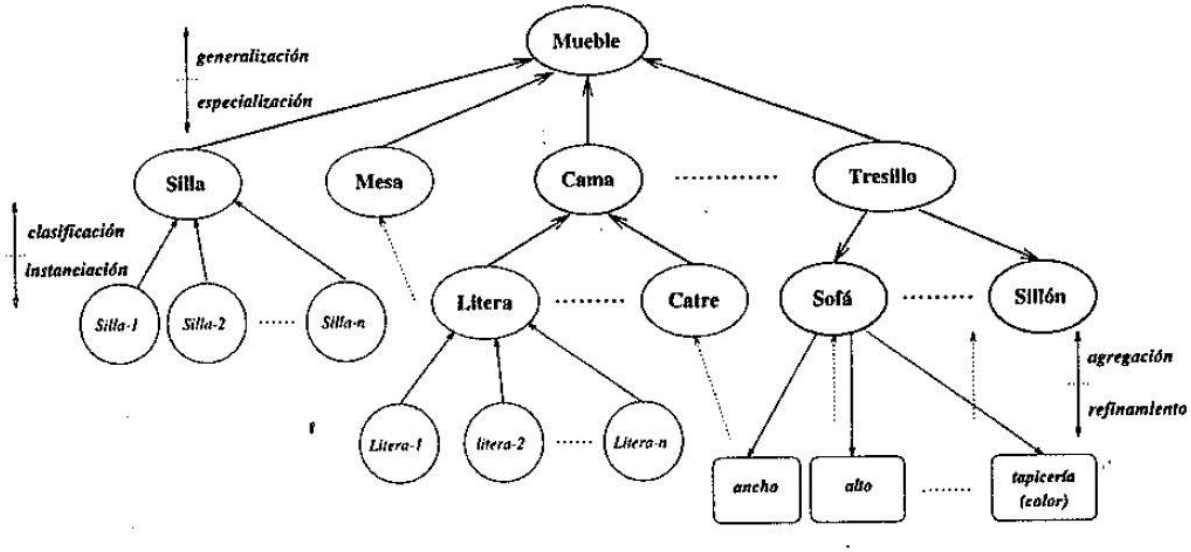


En este proceso se determinan las propiedades de interés, dejando otras propiedades que puedan afectar o verse afectadas por el sistema. Este proceso de

simplificación es innato al proceso mental del ser humano y está basado en la capacidad de abstracción.

La abstracción

La abstracción es la capacidad mediante la cual una serie de objetos se categorizan en un nuevo objeto mediante una función de pertenencia. Al nuevo objeto se le denomina clase o tipo de objeto, y todos los elementos categorizados en esta clase tienen propiedades comunes, las cuales caracterizan la clase. En la definición de los datos, la abstracción es utilizada de dos formas: generalización y agregación.



es_un

- La **generalización** es la abstracción por la cual un conjunto de clases de objetos puede ser visto como una nueva clase de objetos más general. A los procesos inversos se les denomina **especialización** e **instanciación**.

parte_de

- La **agregación**, por otra parte, es la capacidad de considerar un objeto basándose en los elementos que lo constituyen. A proceso inverso se le denomina **refinamiento**.

El modelo Entidad – Interrelación

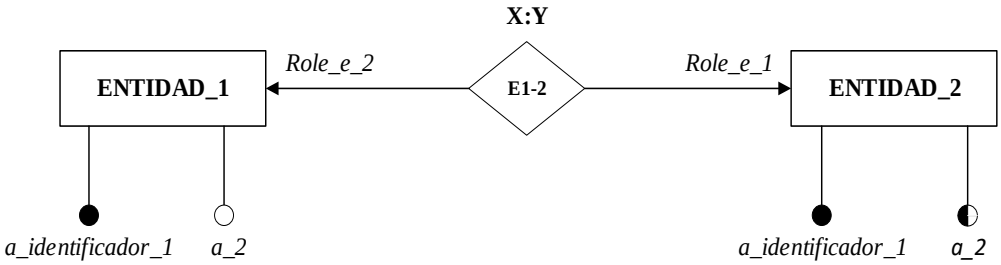
El modelo Entidad – Interrelación (E-R) fue propuesto por Peter Chen para la representación conceptual de los problemas y como medio para representar la visión de un sistema de forma global.

Las características actuales de este modelo permiten la representación de cualquier tipo de sistema y a cualquier nivel de abstracción o refinamiento.

Está soportado en la representación de los datos haciendo uso de grafos y tablas. Mediante un conjunto de símbolos, y haciendo uso de un conjunto reducido de reglas, son representados los elementos que forman parte del sistema y las relaciones existentes entre ellos.

Conceptos:

- **Conjunto:** agregación de una serie de objetos elementales mediante una función de pertenencia.
- **Relación:** un conjunto que representa correspondencia entre dos o más conjuntos.
- **Intención:** clasificación de una serie de elementos individuales en un tipo o clase de objeto al que se ha denominado conjunto o relación (Descripción del tipo de objeto).
- **Extensión:** descripción de un tipo o clase de objetos (el conjunto o la relación).
- **Dominio:** conjuntos cuyos elementos son homogéneos.
- **Atributo:** se denomina atributo de un dominio a la intención de ese dominio, y el valor del atributo será la extensión del dominio. Un atributo identifica la semántica de un dominio para la descripción de un problema, es decir, el significado de un dato.
    1. **Atributos simples.**
    2. **Atributos múltiples.**
    3. **Atributos compuestos.**
- **Entidad:** es un tipo de objeto (un conjunto) definido en base a la agregación de una serie de atributos. La intención de una entidad es denominada tipo de entidad y representa el posible conjunto de objetos definidos en base a la agregación de un mismo conjunto de atributos.
- **Interrelación:** representa la relación existente entre entidades, denominándose tipo de interrelación a la intención de la relación existente entre dos tipos de entidad.

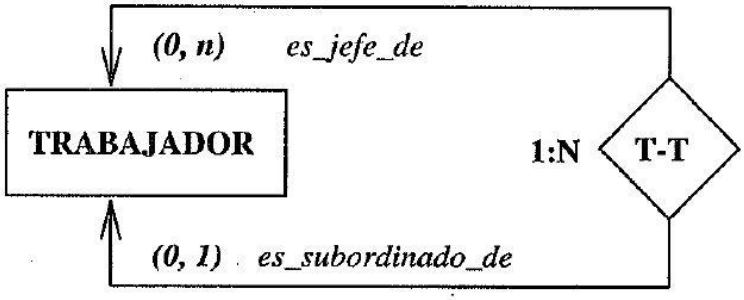


X:Y

1:1
1:N
N:N

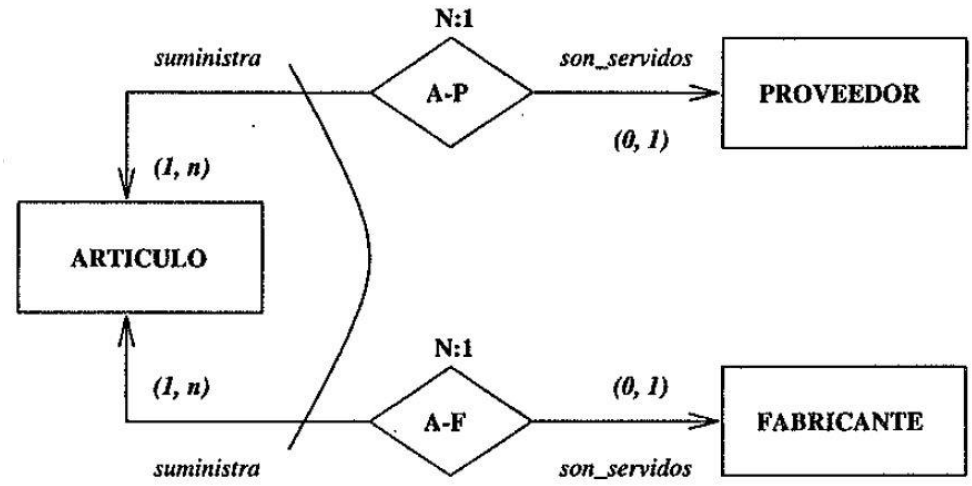
Interrelaciones reflexivas

Las interrelaciones reflexivas son relaciones unarias y, por tanto, consideran que en el tipo de interrelación se ve involucrado un único tipo de entidad.

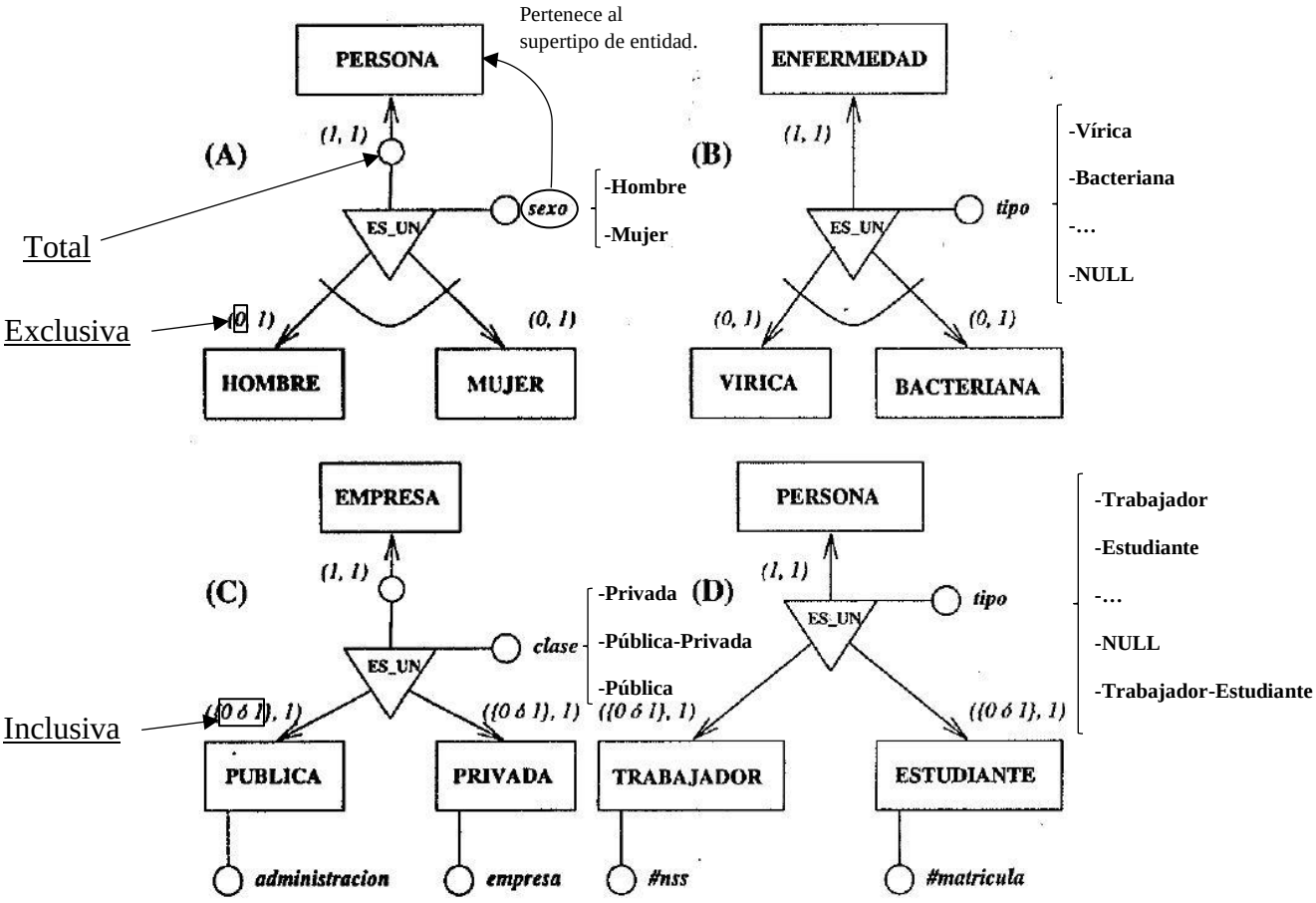


Interrelaciones exclusivas

Dos o más tipos de relación son exclusivos respecto de un tipo de entidad que participa en ambos, si cada instancia del tipo de entidad sólo puede participar en uno de los tipos de relación.



Interrelaciones jerárquicas



A	B	C	D
- Exclusiva - Total	- Exclusiva - Parcial	- Inclusiva - Total	- Inclusiva - Parcial

El modelo Relacional

- Simplicidad
- Versatilidad

Dada una serie de conjuntos $D_1, D_2, D_3, \ldots, D_n$, no necesariamente distintos, se dice que R es una relación entre estos *n* conjuntos si es un conjunto de *n* tuplas no ordenadas $(d_1, d_2, d_3, \ldots, d_n)$ tales que $d_1 \in D_1, d_2 \in D_2, d_3 \in D_3, \ldots, d_n \in D_n$. A los conjuntos $D_1, D_2, D_3, \ldots, D_n$ se les denomina dominios de R, y el valor de *n* es el grado de la relación R.

- Al número de tuplas de una relación en un instante dado se le denomina cardinalidad de la relación.
- Al número de columnas de una relación se le denomina grado / orden de una relación.

Ejemplo de relación:

grado / orden

cardinalidad

ALUMNO	*matricula#*	*nombre*	*apellidos*	*curso*	*nota*
	3456	José	Pérez de la Lastra	1	5.25
	0101	María	Antúnez Sastre	2	7.80
	8743	Lourdes	Sánchez Argote	1	4.50
	1234	Antonio	Soria Madrid	3	6.35
	5674	Luis	González Silos	1	3.20
	0678	Pilar	Alcántara Badajoz	2	5.50
	0345	Dolores	Almiz Márquez	3	7.30
	2985	Manuel	Rives Fuentes	3	3.50

Alumno ≡ *(matrícula, nombre, apellidos, curso, nota)*

- **Clave principal:** *nombre_clave_principal*
- **Clave secundaria:** *nombre_clave_secundaria*

NOTA: No pueden existir dos relaciones (columnas) con el mismo nombre.

Reglas de integridad

- **Integridad de clave:** ningún atributo que forme parte de la clave candidata de una relación podrá tomar valores nulos para ninguna tupla de esa relación.

- **Integridad de referencia:** sea *D* un dominio primario, y sea R_1 una relación con un atributo $R_1.a$ definido en el dominio *D*, entonces, en cualquier instante dado, cada valor de $R_1.a$ en R_1 debe ser nulo o bien igual a algún valor *V*, el cual existe, en ese instante, para un atributo $R_2.b$ definido en el mismo dominio *D* sobre la relación R_2 y en la cual está definido como clave primaria.
  A aquellos atributos $R_1.a$ que satisfacen esta regla de integridad se les denomina **Foreign Key (FK),** los cuales, junto con las claves primarias, proporcionan al modelo los mecanismos adecuados para representar las relaciones existentes entre los objetos del dominio del problema.

- **Integridad de dominio:** como, por ejemplo, restricción de valores permitidos para los atributos que forman parte de las relaciones existente (valor máximo y mínimo, lista de valores…).

Dependencias funcionales

- **Reflexiva:**

R.b ⊆ R.a

R.a → R.b

R ≡ {a, b}

- **Aumento:**

R.a → R.b

R.(a + c) → R.(b + c)

R ≡ {a, b, c}

- **Transitiva:**

R.a → R.b

R.b → R.c

R.a → R.c

R ≡ {a, b, c}

Axiomas de Amstrong

- **Unión:**

R.a → R.b

R.b → R.c

R.a → R.(b + c)

R ≡ {a, b, c}

- **Pseudo – transitiva:**

R.a → R.b

R.(b + c) → R.d

R.(a + c) → R.d

R ≡ {a, b, c, d}

- **Descomposición:**

R.a → R.b

R.c ⊆ R.b

R.a → R.c

R ≡ {a, b, c}

Forma Normal 1 (FN1)

A	B	C
a_1	b_1	c_1
a_2	b_2	c_2
a_3	b_3	c_3

Una relación R satisface la FN1 si, y solo si, cumple:

1- No hay orden ni de filas ni de columnas.
2- No hay filas duplicadas. Para ello exige clave primaria.
3- Sólo un valor para el atributo de cada fila – y – columna. No atributos nulos en caso de ser clave primaria.

Ejemplo:

Supóngase que un diseñador desea guardar los nombres y los números telefónicos de los clientes. Se procede a definir una tabla de cliente como la que sigue:

Clientes

ID_Cliente	Nombre	Apellido	Teléfono
123	Rachel	Ingram	555-861-2025
456	James	Wright	555-403-1659
789	Cesar	Dure	555-808-9633

Supongamos que es un requisito guardar múltiples números de teléfono para algunos clientes. La manera más simple de hacer esto es permitir que el campo “Teléfono” contenga más de un valor en cualquier registro dado:

Clientes

ID_Cliente	Nombre	Apellido	Teléfono
123	Rachel	Ingram	555-861-2025
456	James	Wright	555-403-1659 555-776-4100
789	Cesar	Dure	555-808-9633

Asumiendo, sin embargo, que la columna “Teléfono” está definida en algún tipo de dominio de número telefónico (por ejemplo, el dominio de cadenas de 12 caracteres de longitud), la representación de arriba no está en FN1. La FN1 prohíbe a un campo contener más de un valor de su dominio de columna.

Un diseño que está inequívocamente en FN1 hace uso de dos tablas: una tabla de cliente y una tabla de teléfono del cliente:

Clientes

ID_Cliente	Nombre	Apellido
123	Rachel	Ingram
456	James	Wright
789	Cesar	Dure

Teléfono - Clientes

ID_Cliente	Teléfono
123	555-861-2025
456	555-403-1659
456	555-776-4100
789	555-808-9633

Este diseño no permite grupos repetidos de número telefónicos. En lugar de eso, cada enlace Cliente-a-Teléfono aparece en su propio registro. Es valioso notar que este diseño cumple los requisitos adicionales para la FN2 Y FN3.

Forma Normal 2 (FN2)

Una relación R satisface la FN2 si, y sólo si, satisface la FN1 y cada atributo de la relación depende funcionalmente de forma completa de la clave primaria de esa relación.

Matrícula ≡ *(dni, asignatura, nombre, apellidos, nota, curso, aula, lugar)*

R.(dni, asignatura) → { R.nombre, R.apellidos, R.nota, R.curso, R.aula, R.lugar }

R.dni → { R.nombre, R.apellidos }

R.asignatura → R.curso

Cuando no se satisface la FN2, se crea una nueva tabla con los atributos que me crean esos problemas:

Imparte ≡ *(asignatura, curso)*

FK → Matrícula-$_2$ ≡ *(dni, asignatura, nombre, apellidos, nota, aula, lugar)*

Imparte.asignatura = Matrícula-$_2$.asignatura

Alumno ≡ *(dni, nombre, apellidos)*

FK → Matrícula-$_3$ ≡ *(dni, asignatura, nota, aula, lugar)*

Alumno.dni = Matrícula-$_3$.dni

Ejemplo:

Supóngase una tabla describiendo las habilidades de los empleados en una empresa:

Habilidades

Empleado*	Habilidad*	Lugar_actual_de_trabajo
Jones	Mecanografía	114 Main Street
Jones	Taquigrafía	114 Main Street
Jones	Tallado	114 Main Street
Bravo	Limpieza	73 Industrial Way
Ellis	Alquimia	73 Industrial Way
Ellis	Malabarismo	73 Industrial Way
Harrison	Limpieza	73 Industrial Way

La única clave candidata de la tabla es {*Empleado, Habilidad*}. El atributo restante, *Lugar_actual_de_trabajo*, es dependiente solo en parte de la clave candidata, llamada *Empleado*. Por lo tanto, la tabla no está en FN2. Se observa la redundancia en la manera en que son representados los Lugares actuales de trabajo: obtenemos 3 veces que Jones trabaja en *114 Main Street*, y dos veces que Ellis trabaja en *73 Industrial Way*. Esta redundancia hace a la tabla vulnerable a anomalías de actualización: por ejemplo, es posible actualizar el lugar de trabajo de Jones en sus registros "Mecanografía" y "Taquigrafía" y no actualizar su registro "Tallado". Los datos resultantes implicarían respuestas contradictorias a la pregunta *"¿Cuál es el lugar de trabajo de Jones?"*.

Una alternativa en 2FN a este diseño representaría la misma información en dos tablas:

Empleados

Empleado*	Lugar_actual_de_trabajo
Jones	114 Main Street
Bravo	73 Industrial Way
Ellis	73 Industrial Way
Harrison	73 Industrial Way

Habilidades

Empleado*	Habilidad*
Jones	Mecanografía
Jones	Taquigrafía
Jones	Tallado
Bravo	Limpieza
Ellis	Alquimia
Ellis	Malabarismo
Harrison	Limpieza

Formal Normal 3 (FN3)

Una relación R satisface la FN3 si, y sólo si, satisface la segunda forma normal y cada atributo no primo de la relación no depende funcionalmente de forma transitiva de la clave primaria de esa relación. Es decir, no pueden existir dependencias entre los atributos que no forman parte de la clave primaria de la relación R, ya que crearía redundancia.

FK
Ubicación ≡ *(aula, lugar)*
Matrícula-4 ≡ *(dni, asignatura, nota, aula)*
Ubicación.aula = Matrícula-4.aula

El atributo *aula* no es clave primaria de la relación y, además, determina al atributo lugar, por lo que, de forma transitiva, *lugar* esta dependiendo de forma funcional de la clave principal de la relación Matrícula-4.

Ejemplo:

Un ejemplo de una tabla en FN2 que falla en satisfacer los requisitos de la FN3 es:

Ganadores

Torneo*	Año*	Ganador	Fecha_nacimiento
Indiana Invitational	1998	Al Fredrickson	21/07/1975
Cleveland Open	1999	Bob Albertson	28/09/1968
Des Moines Masters	1999	Al Fredrickson	21/07/1975
Indiana Invitational	1999	Chip Masterson	14/03/1977

La única clave candidata es {*Torneo, Año*}. La violación de la FN3 ocurre porque el atributo no primario *Fecha_nacimiento* es dependiente transitivamente de {*Torneo, Año*} vía el atributo no primario *Ganador*. El hecho de que la *Fecha_nacimiento* es funcionalmente dependiente en el *Ganador* hace la tabla vulnerable a inconsistencias lógicas, pues no hay nada que impida a la misma persona ser mostrada con diferentes fechas de nacimiento en diversos registros. Para expresar los mismos hechos sin violar la FN3, es necesario dividir la tabla en dos:

Ganadores

Torneo*	Año*	Ganador
Indiana Invitational	1998	Al Fredrickson
Cleveland Open	1999	Bob Albertson
Des Moines Masters	1999	Al Fredrickson
Indiana Invitational	1999	Chip Masterson

Fecha_nacimiento

Ganador*	Fecha_nacimiento
Al Fredrickson	21/07/1975
Bob Albertson	28/09/1968
Al Fredrickson	21/07/1975
Chip Masterson	14/03/1977

Forma Normal de Boyce – Codd (FNBC)

La FNBC se basa en el concepto de Determinante Funcional. Se denomina Determinante Funcional a uno o a un conjunto de atributos de una relación R del cual depende funcionalmente de forma completa algún otro atributo de la relación.

Una relación R satisface la FNBC si, y sólo si, se encuentra en FN1, y cada determinante funcional es una clave candidata de la relación R.

Matrícula-$_5$ ≡ *(dni, asignatura, nombre, apellidos, nota, curso, aula, lugar)*

$$\text{R.(dni, asignatura)} \rightarrow \begin{cases} \text{R.nota} \\ \text{R.curso} \\ \text{R.aula} \\ \text{R.lugar} \end{cases} \qquad \text{R.(asignatura, nombre, apellidos)} \rightarrow \begin{cases} \text{R.nota} \\ \text{R.curso} \\ \text{R.aula} \\ \text{R.lugar} \end{cases}$$

R.(dni, asignatura) ↔ R.(asignatura, nombre, apellidos)

Determinantes funcionales:

R.aula → R.lugar

R.asignatura → R.curso

R.(dni, asignatura) → R.nota

R.(dni, asignatura) → R.aula

R.(dni, asignatura) → R.lugar

R.dni → R.nombre

R.dni → R.apellidos

Matrícula-$_6$ ≡ *(dni, asignatura, nota, aula)*

FK

Imparte ≡ *(asignatura, curso)*

Ubicación ≡ *(aula, lugar)*

Alumno ≡ *(dni, nombre, apellidos)*

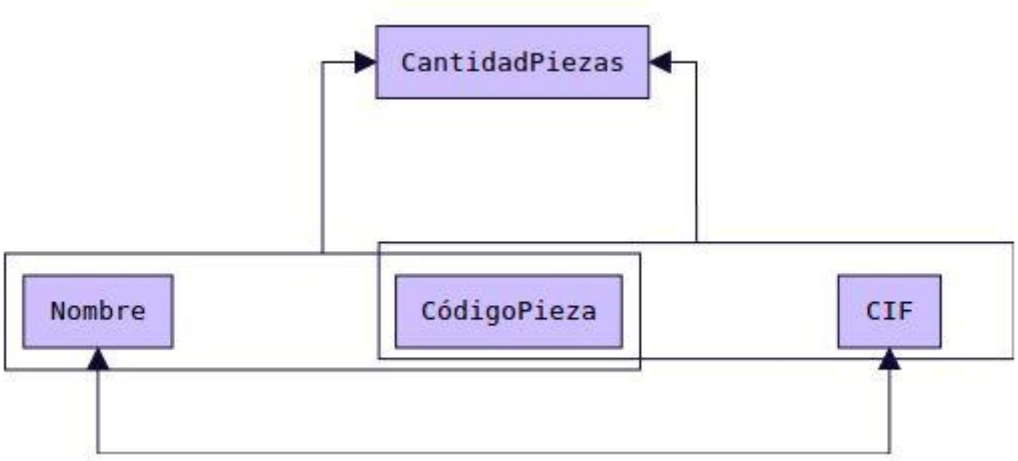
Ejemplo:

Supongamos una tabla con información de proveedores, códigos de piezas y cantidades de esa pieza que proporcionan los proveedores. Cada proveedor tiene un nombre único. La representación de la tabla sería como sigue:

Suministros

CIF	Nombre	Código_pieza	Cantidad
11XX	Ferroman	1	10
22XX	Ferrotex	1	7
33XX	Ferropet	3	4
11XX	Ferroman	2	20
22XX	Ferrotex	2	15
22XX	Ferrotex	3	8

El gráfico de dependencias funcionales es el siguiente:



El atributo *Cantidad* tiene dependencia funcional de dos claves candidatas compuestas, que son:

1- {*Nombre, Código_pieza*}.
2- {*CIF, Código_pieza*}.

Existe también una dependencia funcional en doble sentino que no nos afecta:

Nombre ↔ *CIF*.

Para esta tabla existe un solapamiento de 2 claves candidatas compuestas. Para evitar el solapamiento de claves candidatas dividimos la tabla:

Proveedores

CIF	Nombre
11XX	Ferroman
22XX	Ferrotex
33XX	Ferropet
11XX	Ferroman
22XX	Ferrotex
22XX	Ferrotex

Suministros

CIF	Código_pieza	Cantidad
11XX	1	10
22XX	1	7
33XX	3	4
11XX	2	20
22XX	2	15
22XX	3	8

Forma Normal 4 (FN4)

Dada una relación R, se dice que el atributo $R.y \in R$ depende de forma multivaluada de otro atributo $R.x \in R$, o que *R.x* multidetermina a *R.y*, y se expresa de la forma $R.x \rightarrow\rightarrow R.y$ si, y sólo si, cada valor de *R.x* tiene asignado un conjunto bien definido de valores de *R.y* y este conjunto es independiente de cualquier valor que tome otro atributo $R.z \in R$, el cual depende del valor de *R.x*.

Las Dependencias Multivaluadas representan la independencia existente entre dos conjuntos *R.y* y *R.z*, la cual está correlacionada por la dependencia que tiene cada uno de estos conjuntos con el conjunto *R.x* del cual dependen ambos de forma multivaluada.

R.x →→ R.y / R.z

Forma Normal 5 (FN5)

Una relación R satisface la FN5, también denominada Forma Normal de Proyección si, y sólo si, toda la dependencia de reunión en R está implicada por las claves candidatas entre sí y no por cualquier otro atributo de R, forme o no parte de las claves candidatas,

Dada una relación R de esquema $R(a_1, a_2, \ldots, a_n)$ se dice que existe una Dependencia de Reunión si, y sólo si, la relación R puede ser construida a partir de la reunión natural de las relaciones $R_1, R_2, \ldots, R_n$ obtenidas por la proyección de R sobre los atributos $a_1, a_2, \ldots, a_n$ respectivamente.

Operadores

R_1	a	b	c
	1	a	1
	1	b	2
	1	c	1
	2	a	2
	3	b	1

R_2	d	e	f
	1	a	1
	1	c	2
	1	b	1
	2	a	1
	2	b	2
	3	a	1
	3	b	1

R_3	g	h
	1	a
	2	b
	3	c
	1	c
	2	a

Suma / UNION

$R_4 = R_1 + R_2$ (Deben ser compatibles, homogéneos).

R_4	x	y	z
	1	a	1
	1	b	2
	1	c	1
	2	a	2
	3	b	1
	~~1~~	~~a~~	~~1~~
	1	c	2
	1	b	1
	2	a	1
	2	b	2
	3	a	1
	~~3~~	~~b~~	~~1~~

Tuplas de R_1 y R_2 y eliminar las que están repetidas.

En SQL:

```
1  SELECT *
2  FROM R1
3  UNION
4  SELECT *
5  FROM R2;
```

NOTA: $R_1 + R_2 = R_2 + R_1$

Resta / SUBSTRACT

$R_4 = R_1 - R_2$ (Deben ser compatibles, homogéneos).

R_4	x	y	z
	~~1~~	~~a~~	~~1~~
	1	b	2
	1	c	1
	2	a	2
	~~3~~	~~b~~	~~1~~

Tuplas de R_1 – las que se encuentran en R_2.

En SQL:

```
1 | SELECT *
2 | FROM R1
3 | SUBSTRACT
4 | SELECT *
5 | FROM R2;
```

```
1 | SELECT *
2 | FROM R1
3 | WHERE NOT IN
4 | (SELECT *
5 | FROM R2);
```

NOTA: $R_1 - R_2 \neq R_2 - R_1$

Selección / SELECT

R_4 = SELECT (R / Q) (Sólo opera sobre una relación, es decir, es un operador unario).

R ≡ Relación.

Q ≡ Expresión lógica.

R_4 = SELECT (R_1 / a >= 1 AND b = a)

R_4	a	b	c
	1	a	1
	~~1~~	~~b~~	~~2~~
	~~1~~	~~c~~	~~1~~
	2	a	2
	~~3~~	~~b~~	~~1~~

En SQL:

```
1 | SELECT *
2 | FROM R1
3 | WHERE R1.a >= 1 AND R1.b = a;
```

Proyección (Sólo opera sobre una relación, es decir, es un operador unario).

R_4 = PROJECT (R_1 / {a, b})

R_4	**a**	**b**
	1	a
	1	b
	1	c
	2	a
	3	b

R_4 = PROJECT (R_1 / {a, c})

R_4	**a**	**c**
	1	1
	1	2
	~~1~~	~~1~~
	2	2
	3	1

→ No repetición.

En SQL:

```
1  SELECT R1.a, R1.b
2  FROM R1;
```

Producto (No necesariamente compatibles).

$R_4 = R_1 \times R_3$

R_4	**a**	**b**	**c**	**g**	**h**
	1	a	1	1	a
	1	a	1	2	b
	1	a	1	3	c
	1	a	1	1	c
	1	a	1	2	a
	1	b	2	1	a
	1	b	2	2	b
	1	b	2	3	c
	1	b	2	1	c
	1	b	2	2	a
	1	c	1	1	a
	1	c	1	2	b
	1	c	1	3	c
	1	c	1	1	c
	1	c	1	2	a
	2	a	2	1	a
	2	a	2	2	b

Multiplicar cada componente de R_1 con todas las de R_3.

2	a	2	3	c
2	a	2	1	c
2	a	2	2	a
3	b	1	1	a
3	b	1	2	b
3	b	1	3	c
3	b	1	1	c
3	b	1	2	a

En SQL:

```
1  SELECT *
2  FROM R1, R2;
```

Intersección (Elementos compatibles entre dos relaciones).

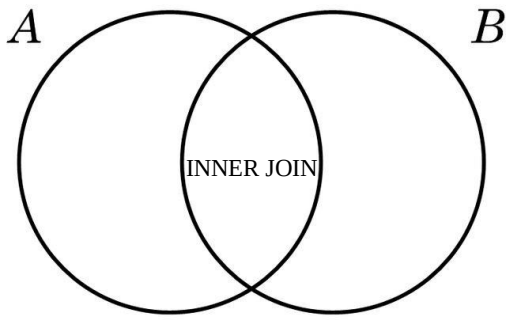
$R_4 = R_1 \cap R_2$

R_4	**x**	**y**	**z**
	1	a	1
	~~1~~	~~b~~	~~2~~
	~~1~~	~~c~~	~~1~~
	~~2~~	~~a~~	~~2~~
	3	b	1

Tuplas de R_1 que se encuentran en R_2.

En SQL:

```
1  SELECT *
2  FROM R1
3  INTERSECT
4  SELECT *
5  FROM R2;
```



Reunión (No necesariamente compatibles).

$R_4 = R_1 \,|x|\, R_2$

$R_4 = JOIN\ (R_1, R_2 \,/\, \underbrace{R_1.a = R_3.g}_{\text{Condición}})$

R_4	a	b	c	g	h
	1	a	1	1	a
	1	b	2	1	a
	1	c	1	1	a
	~~2~~	~~a~~	~~2~~	~~1~~	~~a~~
	~~3~~	~~b~~	~~1~~	~~1~~	~~a~~
	~~1~~	~~a~~	~~1~~	~~2~~	~~b~~
	~~1~~	~~b~~	~~2~~	~~2~~	~~b~~
	~~1~~	~~c~~	~~1~~	~~2~~	~~b~~
	2	a	2	2	b
	~~3~~	~~b~~	~~1~~	~~2~~	~~b~~
	~~1~~	~~a~~	~~1~~	~~3~~	~~c~~
	~~1~~	~~b~~	~~2~~	~~3~~	~~c~~
	~~1~~	~~c~~	~~1~~	~~3~~	~~c~~
	~~2~~	~~a~~	~~2~~	~~3~~	~~c~~
	3	b	1	3	c
	1	a	1	1	c
	1	b	2	1	c
	1	c	1	1	c
	~~2~~	~~a~~	~~2~~	~~1~~	~~c~~
	~~3~~	~~b~~	~~1~~	~~1~~	~~c~~
	~~1~~	~~a~~	~~1~~	~~2~~	~~a~~
	~~1~~	~~b~~	~~2~~	~~2~~	~~a~~
	~~1~~	~~c~~	~~1~~	~~2~~	~~a~~
	2	a	2	2	a
	~~3~~	~~b~~	~~1~~	~~2~~	~~a~~

En SQL:

```
1  SELECT *
2  FROM R1, R2
3  WHERE R1.a = R3.g;
```

División

$R_4 = R_1 \div R_5$

Campos de R_1 que no existen en R_5 (En este caso R_1.a). Una tupla se encuentra en la relación resultante si, y sólo si está asociado en R_1 con cada tupla de R_5.

Suponiendo:

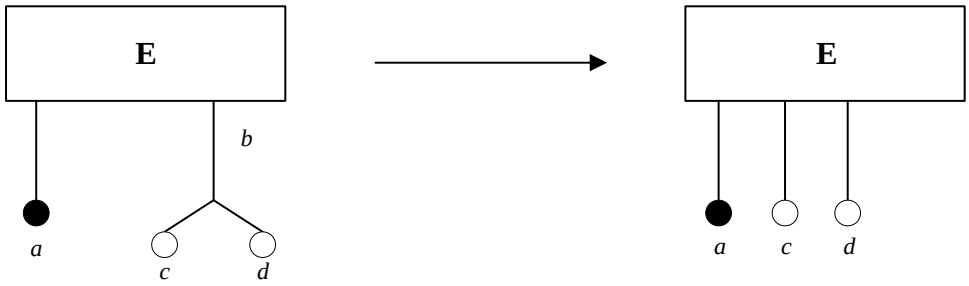
R_5	b	c
	a	1
	b	2

R_1	a	b	c
	1	a	1
	1	b	2
	1	c	1
	2	a	2
	3	b	1

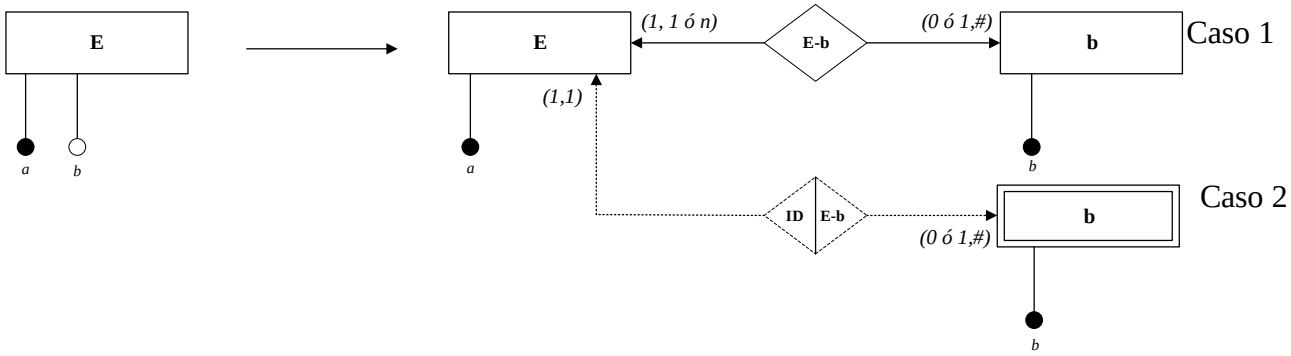
R_4	a
	1

Transformaciones

- **Tipos de entidad:**



E (a, c, d)



NOTA: todo atributo múltiple se convierte en un tipo de entidad.

Caso 1:

E *(a, ...)*

B *(b, ...)*

Caso 2:

E *(a, ...)*

B *(a, b, ...)*

- **Tipos de interrelaciones:**

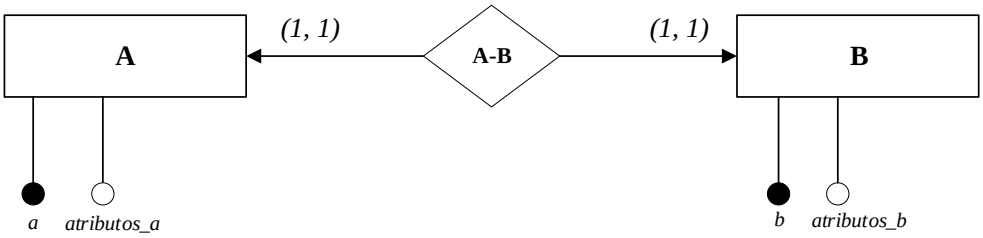
1:1	*(1, 1) – (1, 1)*	1 tabla por cada tipo de entidad.
	(1, 1) – (0, 1)	1 tabla por cada tipo de entidad.
	(0, 1) – (0, 1)	1 tabla por cada tipo de entidad + 1 tabla de la interrelación.

Nunca débiles por ID.

1:N	*(1, 1) – (1, n)*	1 tabla por cada tipo de entidad.
	(1, 1) – (0, n)	1 tabla por cada tipo de entidad.
	(0, 1) – (1, n)	1 tabla por cada tipo de entidad + 1 tabla de la interrelación.
	(0, 1) – (0, n)	1 tabla por cada tipo de entidad + 1 tabla de la interrelación.

N:N	*(?, n) – (?, n)*	1 tabla por cada tipo de entidad + 1 tabla de la interrelación.

(1, 1) – (1, 1)



1. **a ≡ b**

A *(a, atributos-a)*

B *(b, atributos-b)*

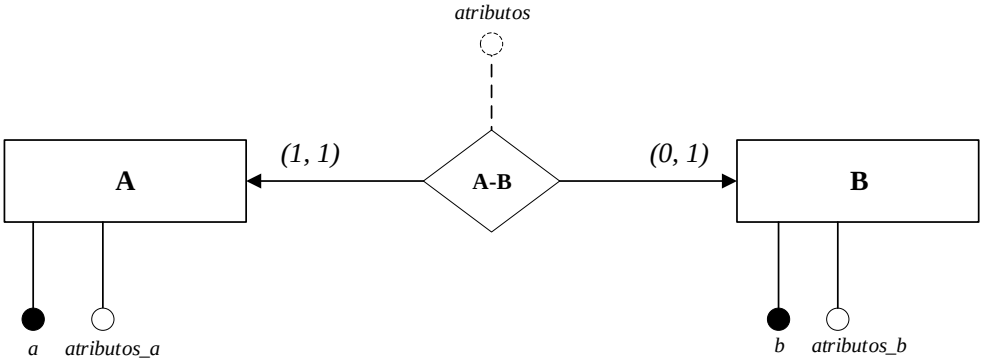
2. **a ≠ b**

FK FK

A *(a, b$^{NOT\ NULL}$, atributos-a)*

B *(b, a$^{NOT\ NULL}$, atributos-b)*

(1, 1) – (0, 1)

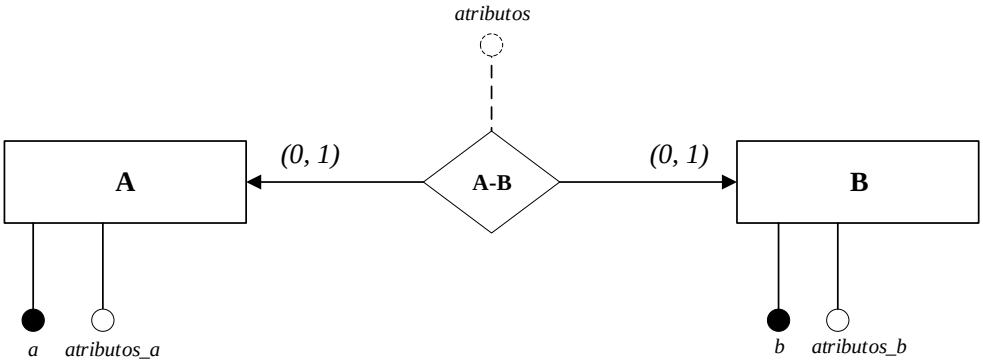


FK

A *(a, atributos-a)*

B *(b, a$^{NOT\ NULL}$, atributos-b, atributos)*

(0, 1) – (0, 1)



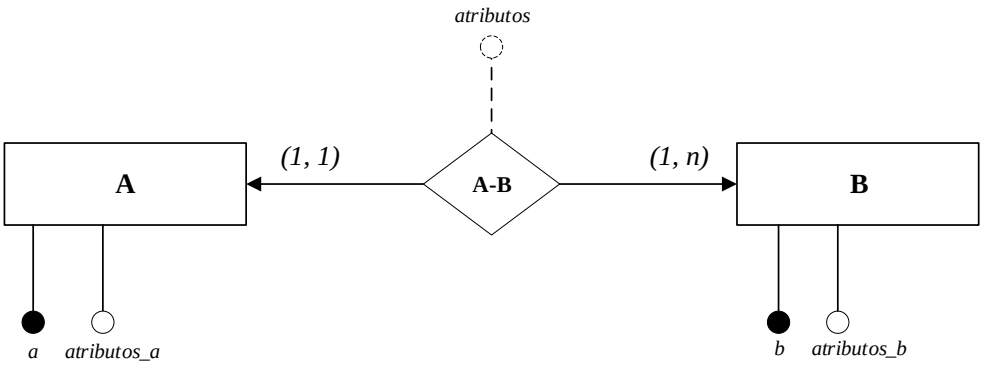
FK FK

A *(a, atributos-a)*

B *(b, atributos-b)*

AB *(a, b, atributos)*

(1, 1) – (0 ó 1, n)

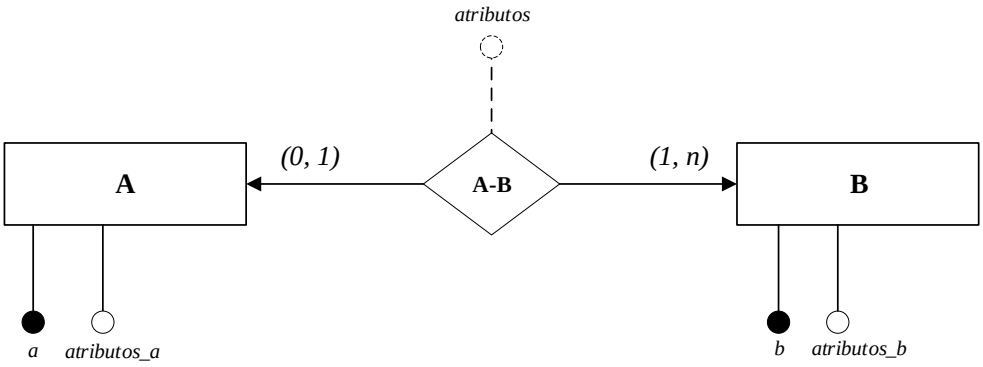


FK

A *(a, atributos-a)*

B *(b, $a^{NOT\ NULL}$, atributos-b, atributos)*

(0, 1) – (0 ó 1, n)



FK

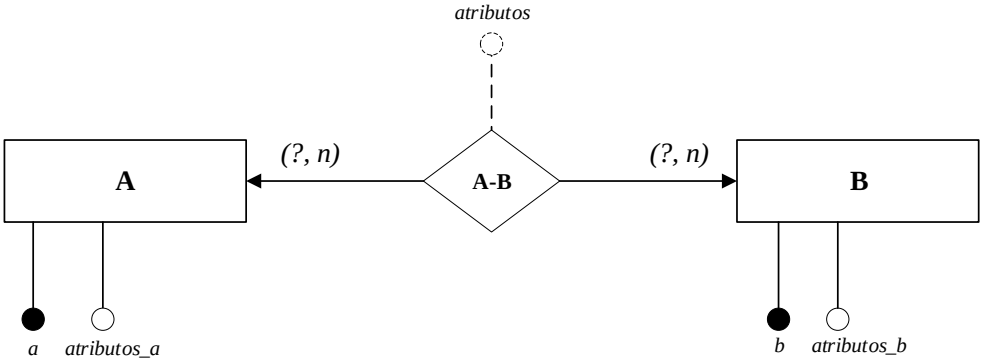
A *(a, atributos-a)*

B *(b, atributos-b)*

AB *($a^{NOT\ NULL}$, b, atributos)*

NOTA: Es clave primaria aquello que se puede repetir, en este caso *b*, por tener cardinalidad *n*.

(?, n) – (?, n)



FK FK

A (*a*, *atributos-a)*

B (*b*, *atributos-b)*

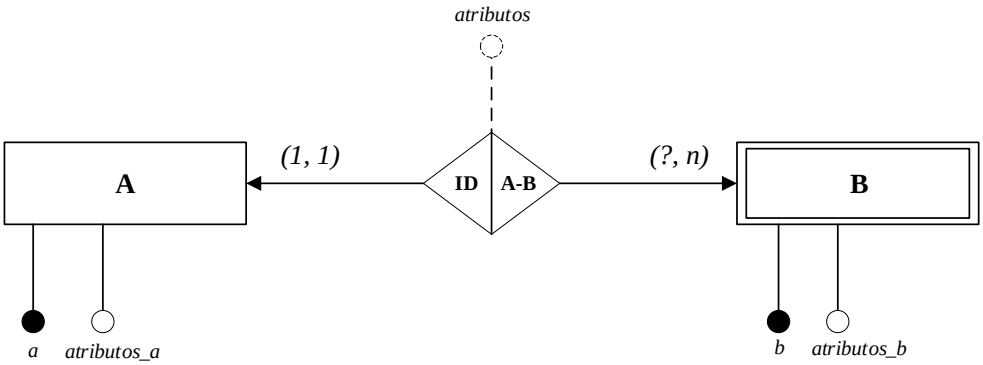
AB (*a*, *b*, *atributos)*

NOTA: Si sólo *a* fuera clave primaria, *b* no se podría repetir. De igual manera ocurriría si sólo lo fuera *b*.

Reglas de transformación para los tipos de entidad débiles por identificación

Únicamente se originará una nueva tabla en el siguiente caso:

(1, 1) – (1, n)



FK

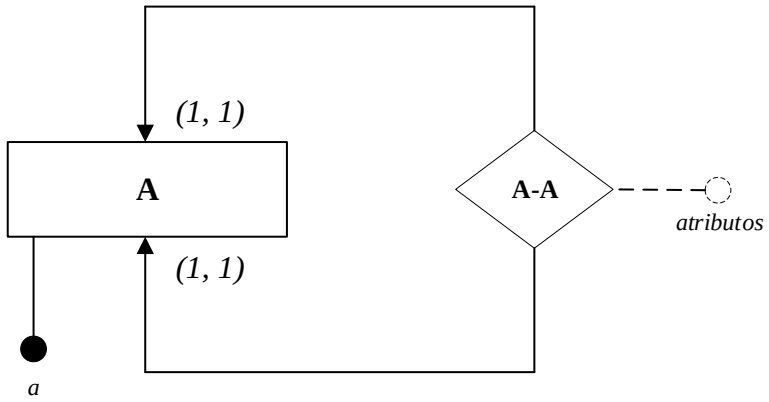
A (*a*, *atributos-a)*

B (*b*, *a*, *atributos-b*, *atributos)*

Reglas de transformación para los tipos de interrelaciones reflexivas

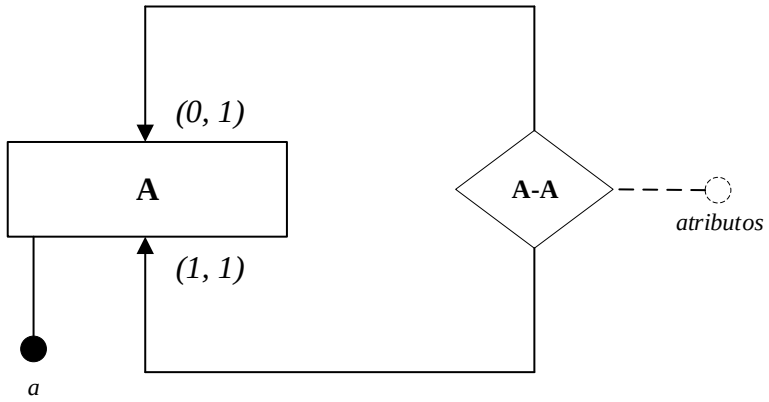
Las interrelaciones unarias se comportan de igual manera que las binarias.

(1, 1) – (1, 1)



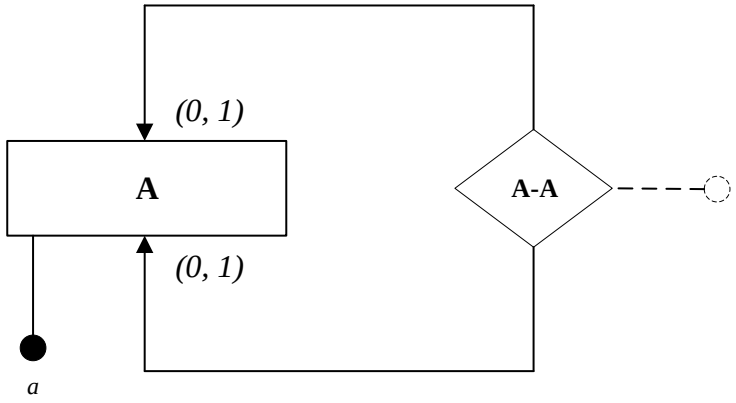
FK → **A** (*a, …, a', atributos)* **NOTA:** *a = a'*

(1, 1) – (0, 1)



A (*a, …, a', atributos)*

(0, 1) – (0, 1)

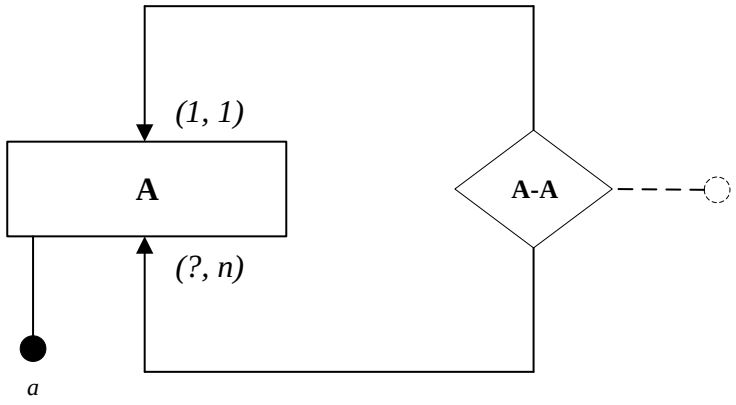


FK

A (*a, ...)*

AA' (*a, a', atributos)*

(1, 1) – (?, n)



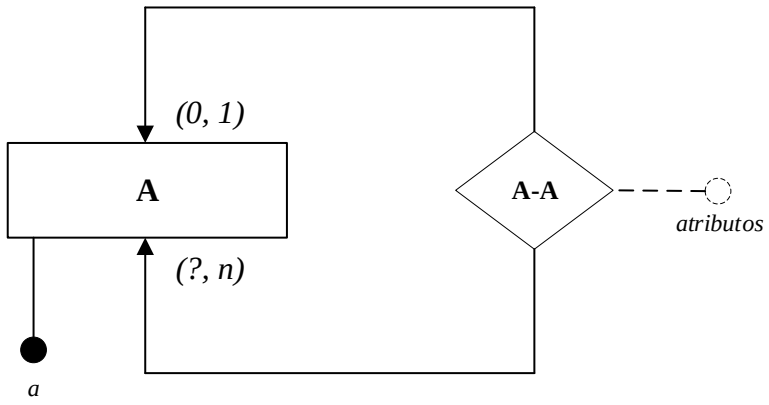
FK

A (*a, ..., a'* NOT NULL*, atributos)*

N 1

NOTA: Será clave primaria de la relación la que tenga cardinalidad n, y la otra será la que tenga cardinalidad 1.

(0, 1) – (?, n)



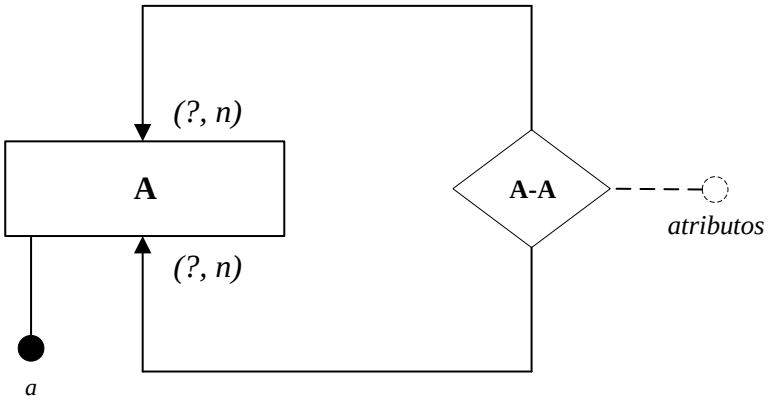
FK → **A** (*a, ...*)

AA' (*a, a', atributos*)

N 1

NOTA: Será clave primaria de la relación la que tenga cardinalidad n, y la otra será la que tenga cardinalidad 1.

(?, n) – (?, n)

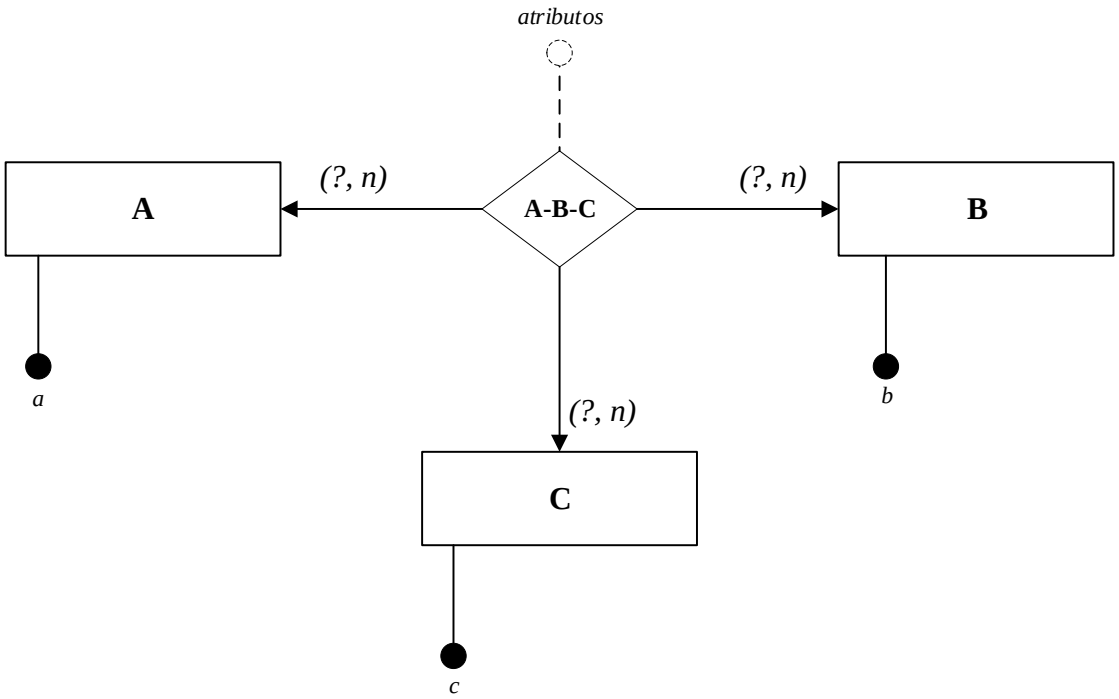


FK → **A** (*a, ...*)

AA' (*a, a', atributos*)

N – narias

1.



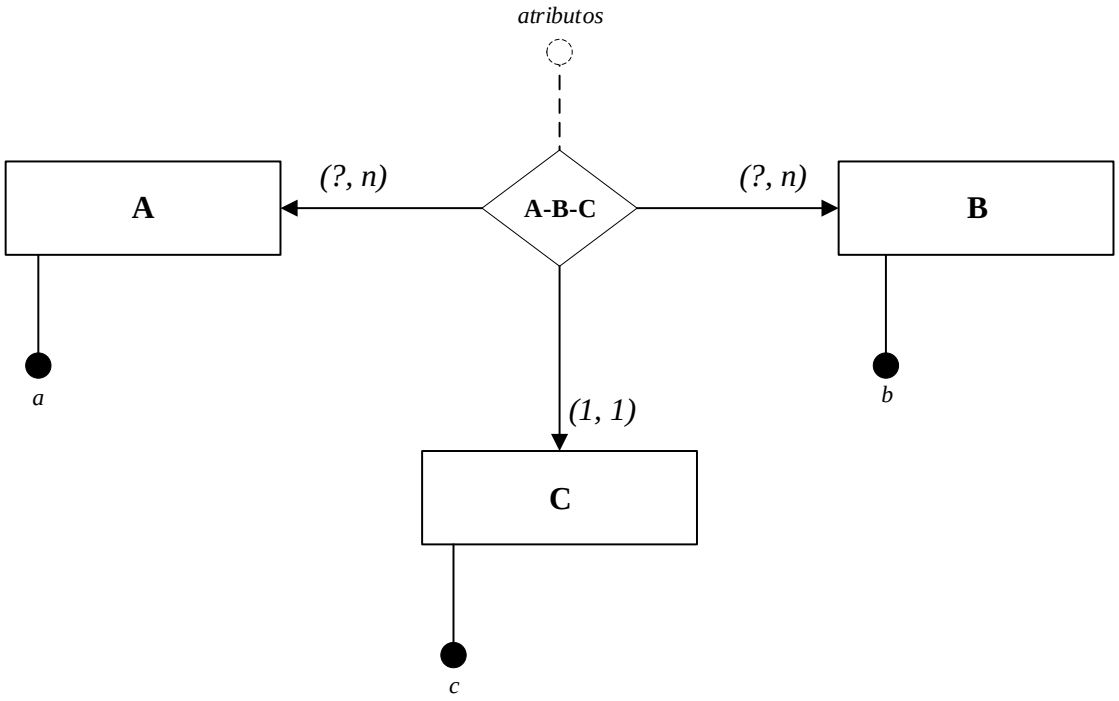
FK → **A** (*a, ...*)

FK → **B** (*b, ...*)

FK → **C** (*c, ...*)

ABC (*a, b, c, atributos*)

2.



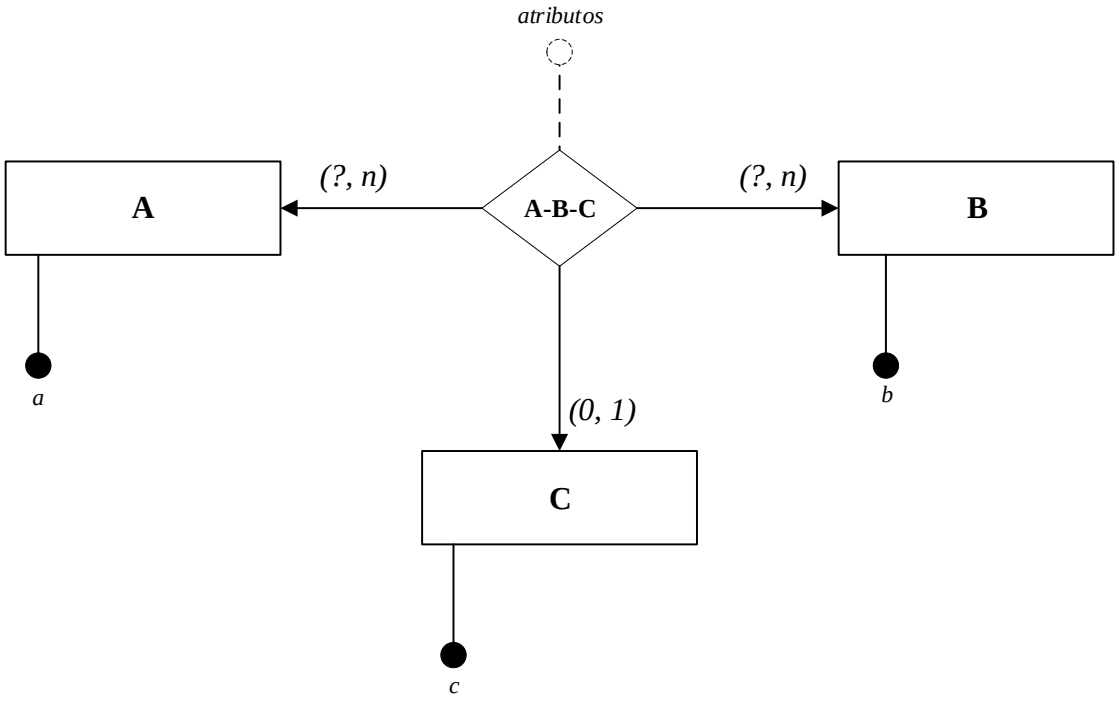
FK → **A** *(a, ...)*

FK → **B** *(b, ...)*

C *(c, ...)*

ABC *(a, b, c NOT NULL, atributos)*

3.



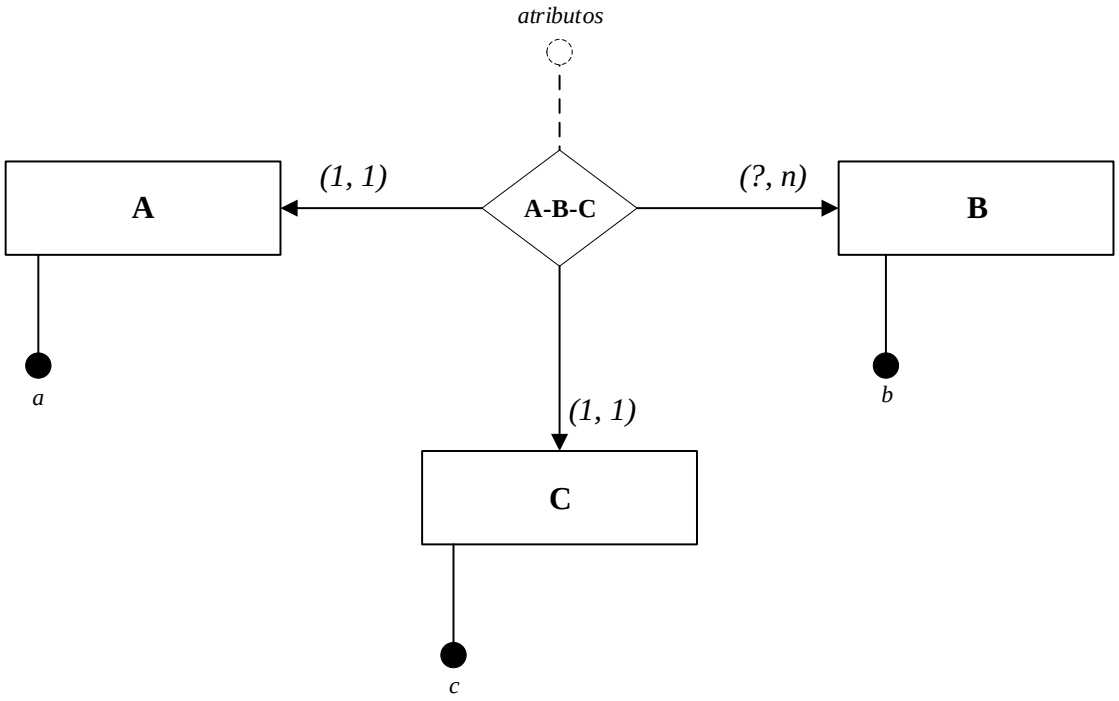
FK → **A** *(a, ...)*

FK → **B** *(b, ...)*

C *(c, ...)*

ABC *(a, b, c* ^{NULL}*, atributos)*

4.

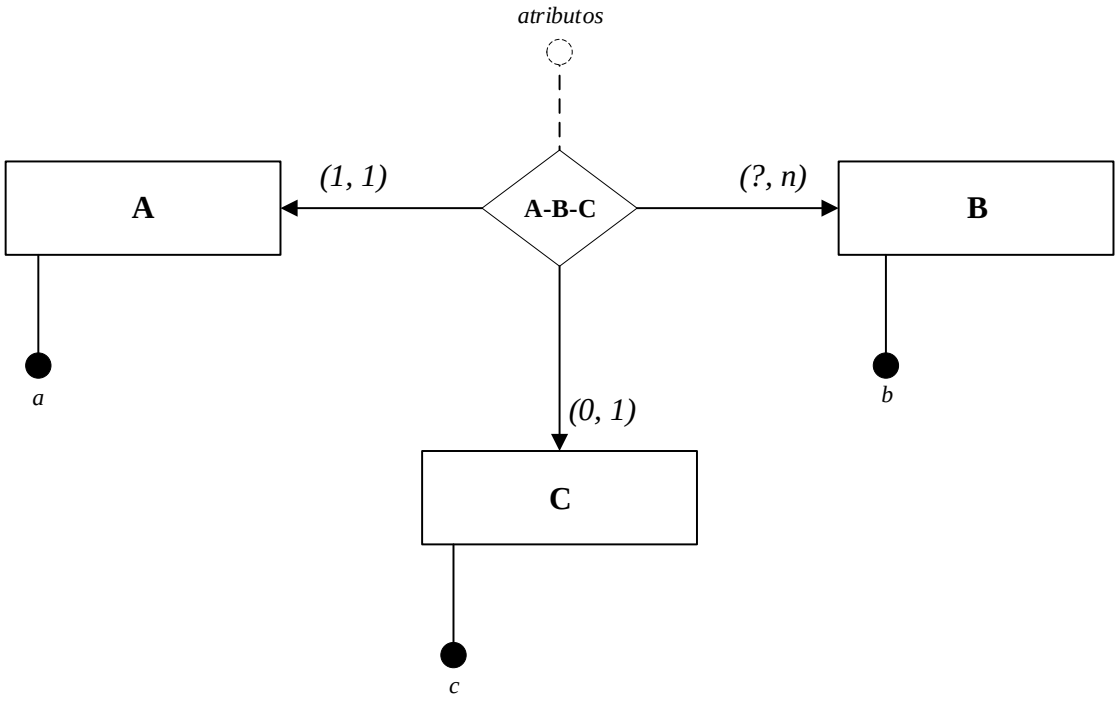


A *(a, ...)*

B *(b, ..., a NOT NULL, c NOT NULL, atributos)*

C *(c, ...)*

5.

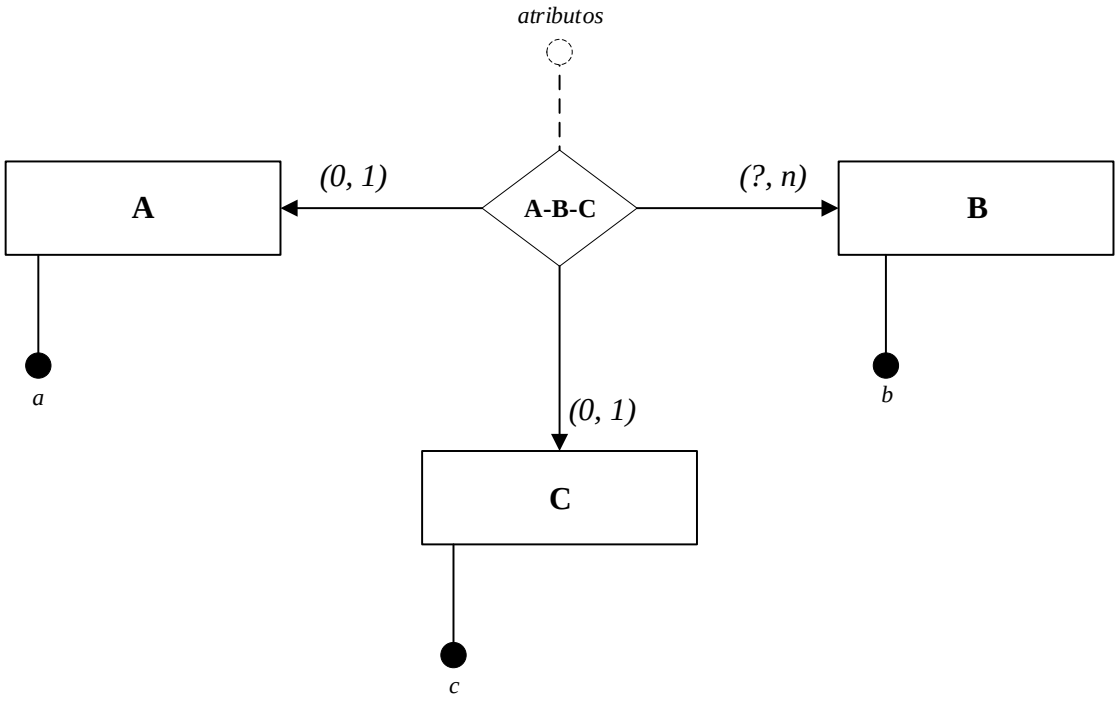


A *(<u>a</u>, ...)*

B *(<u>b</u>, ..., a* $^{NOT\ NULL}$*, c* ^{NULL}*, atributos)*

C *(<u>c</u>, ...)*

6.



FK

A *(a, ...)*

B *(b, ...)*

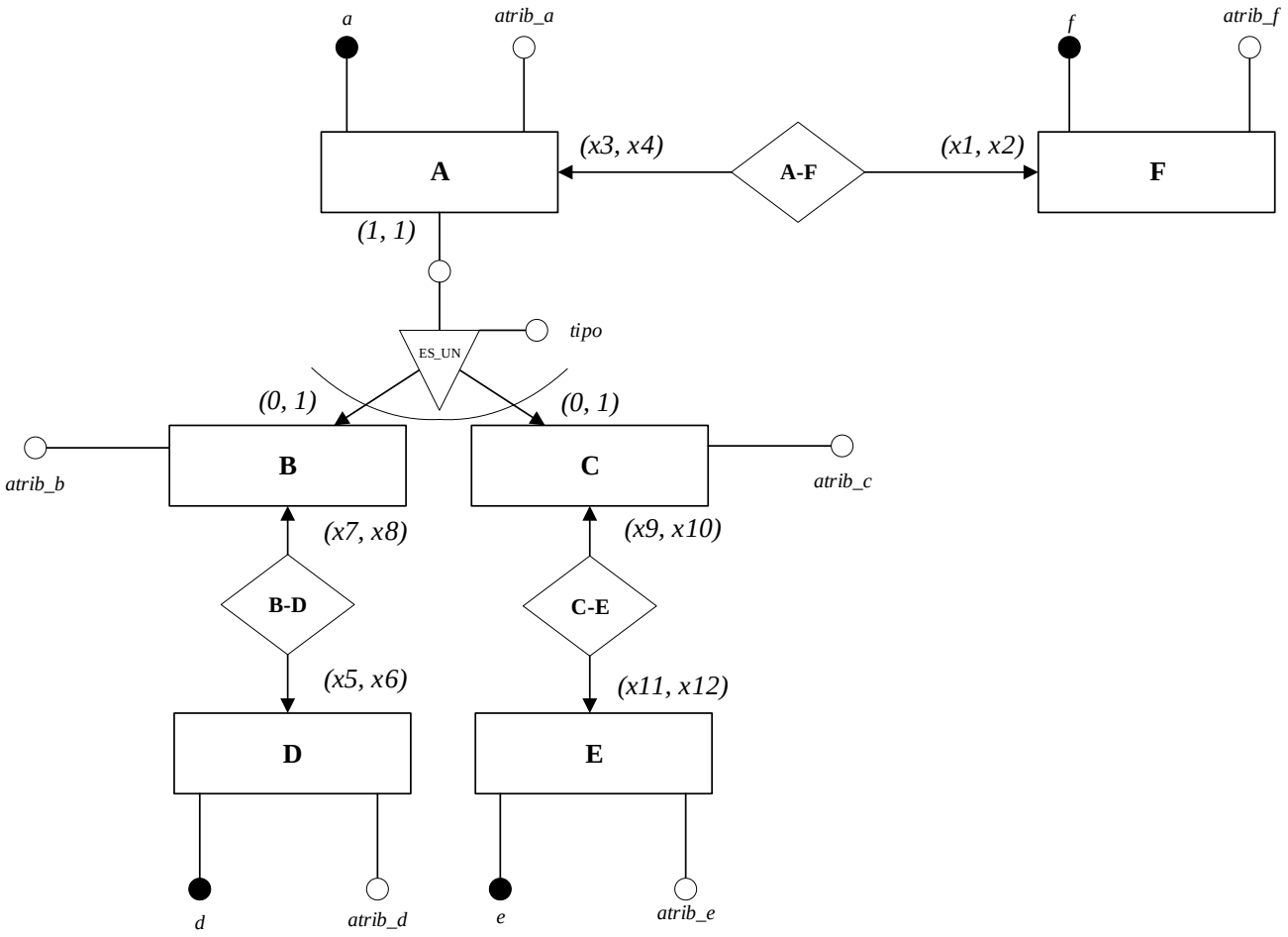
C *(c, ...)*

ABC *(b, a NOT NULL, c NOT NULL, atributos)*

NOTA: *a* y *c* no pueden tomar valor nulo ya que representa la relación.

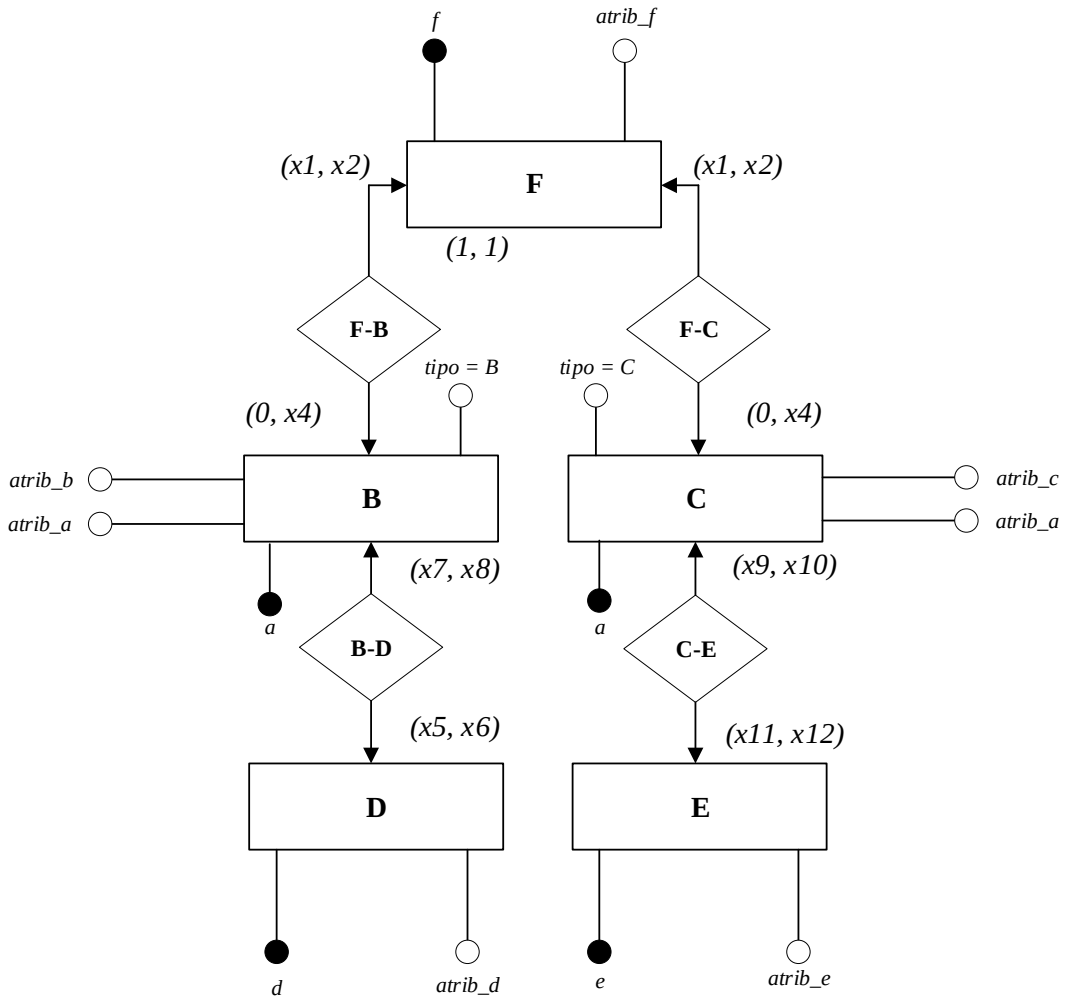
Reglas de transformación para los tipos de interrelaciones jerárquicas

Ejemplo:



1. **Eliminación del supertipo de entidad**

NOTA: Si no es total no se puede eliminar el supertipo.



F *(f, atrib-f)*

B *(a, atrib-a, atrib-b, tipo = B)*

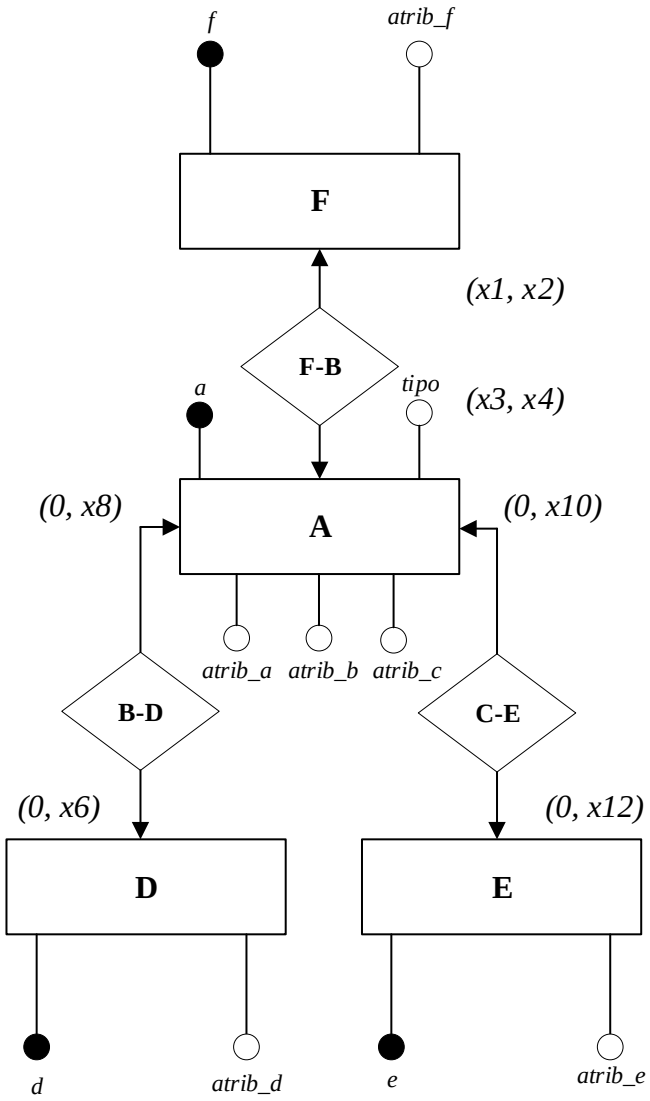
C *(a, atrib-a, atrib-c, tipo = C)*

D *(d, atrib-d)*

E *(e, atrib-e)*

NOTA: b no puede tener valor si ya hay un valor existente en c; de igual manera ocurre con c. Esto se debe a la exclusividad de la relación jerárquica.

2. Eliminación del subtipo de entidad



F *(f, atrib-f)*

A *(a, atrib-a, atrib-b, atrib-c, tipo)* (Atributos del subtipo)

D *(d, atrib-d)*

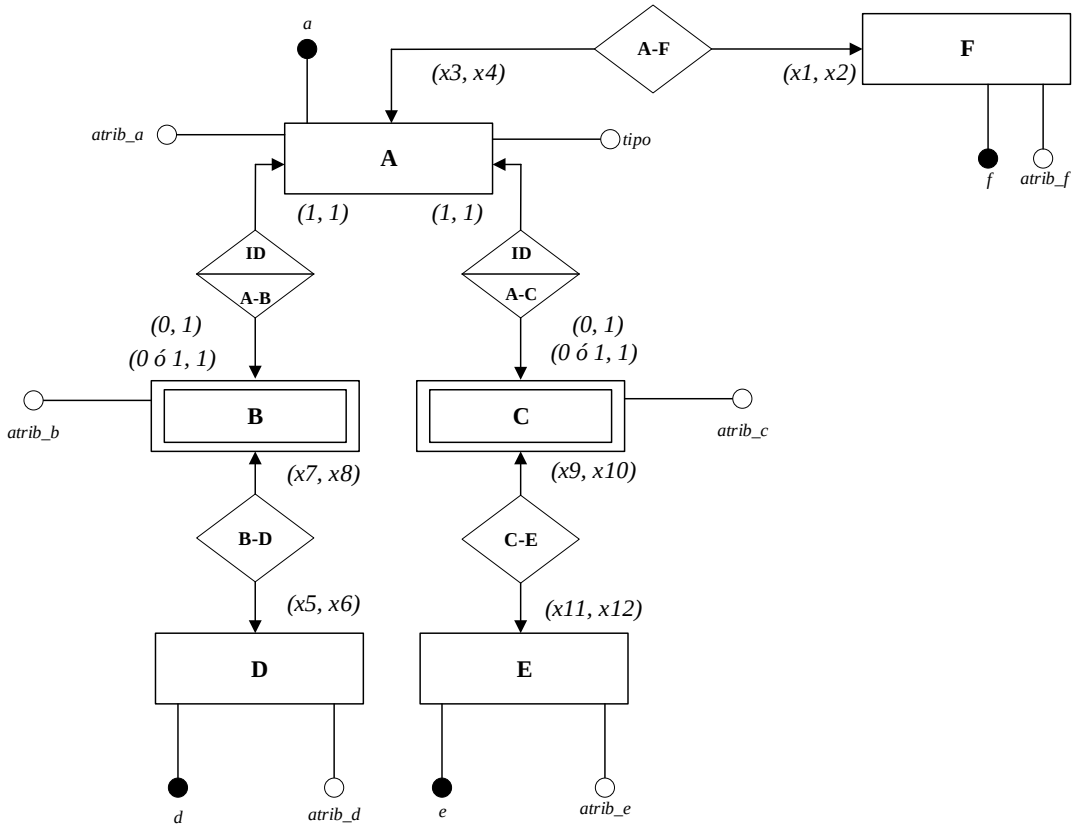
E *(e, atrib-e)*

INCLUSIVA	EXCLUSIVA
tipo = B, C, BC, NULL	tipo = B, C, NULL

NOTA-1: La cardinalidad *(0, x8)* tiene un valor de 0 ya que *D* no puede estar relacionada con *A* mediante ningún *A.tipo-c*, sino de *A.tipo-b*. De igual manera ocurre en el lado opuesto.

NOTA-2: Si la relación es exclusiva, y *A.tipo-b*, los atributos de *c* tienen que ser nulos y viceversa.

3. Eliminación del tipo de interrelación



F *(f, atrib-f)*

FK FK

A *(a, atrib-a, tipo)*

B *(a, atrib-b)*

C *(a, atrib-c)*

D *(d, atrib-d)*

E *(e, atrib-e)*

Integridad, Seguridad y Privacidad de una Base de Datos

Integridad de las Bases de Datos

Para que una base de datos sea íntegra, debe hacer uso de todos aquellos mecanismos disponibles en el modelo de datos y en el SGBD que permiten que las bases de datos se encuentren en un estado consistente con la representación del problema por la que están siendo usadas. Son reglas que se activan en las operaciones de inserción, modificación y borrado de elementos de la base de datos.

El objetivo del concepto de recuperación es el de proteger la base de datos contra fallos lógicos y físicos que destruyan los datos parcial o totalmente. Independientemente de la naturaleza de los fallos, estos pueden afectar a dos aspectos del almacenamiento de la base de datos, como son:

- Fallos que provocan la pérdida de memoria volátil.
- Fallos que provocan la pérdida del contenido de memoria secundaria.

1. Assertions

Son predicados que expresan una condición que la base de datos debe satisfacer siempre.

En SQL:

```
1  CREATE ASSERTION <nombre_aserto>
2  AS CHECK <predicado>;
```

2. Triggers

Son acciones que el sistema ejecuta de forma automática antes, en lugar de, o después de que ocurra un evento en la base de datos que pueda ocasionar una modificación de la misma.

Son mecanismos útiles para realizar de manera automática ciertas tareas cuando se cumplen determinadas condiciones.

En SQL:

```
1  SET serveroutput ON;
2  CREATE OR REPLACE TRIGGER <nombre_trigger>
3     BEFORE/AFTER UPDATE OR INSERT ON <nombre_tabla>
4     FOR EACH ROW
5  BEGIN
6     <procedimiento>
7  END;
```

3. Transactions

Secuencia de operaciones que han de ejecutarse de forma atómica. Pueden terminar con éxito y quedar grabadas en la base de datos; o por el contrario, pueden fracasar y se debe restaurar el estado anterior de la base de datos.

El componente del sistema encargado de lograr atomicidad se conoce como Administrador de Transacciones y las operaciones *COMMIT* (comprometer) y *ROLLBACK* (retroceder) son la clave de su funcionamiento.

En SQL:

```
1  SET TRANSACTION <nombre_transacción>
2     <procedimiento>
3  END;
```

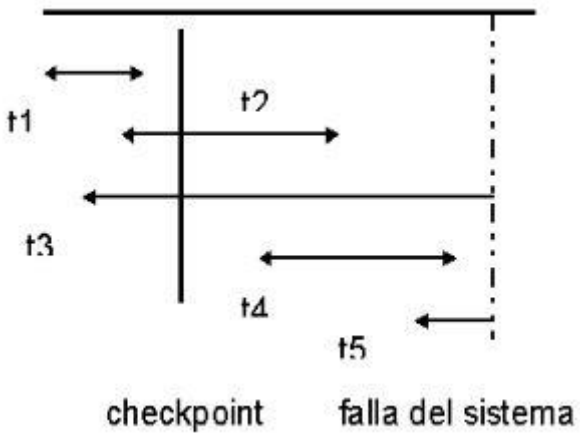
Las características de una transacción son:

- **Atomicidad *(Atomicity)*:** se ejecutan todas las sentencias o ninguna.
- **Consistencia *(Consistency)*:** la ejecución de una transacción deja la base de datos en un estado consistente.
- **Aislamiento *(Isolation)*:** esta propiedad asegura que una operación no pueda afectar a otras, es decir, que la realización de dos transacciones sobre la misma información sean independientes y no generen ningún tipo de error. Esta propiedad define cómo y cuándo los cambios producidos por una operación se hacen visibles para las demás operaciones concurrentes.
- **Persistencia *(Persistence)*:** cuando la transacción acaba exitosamente, sus efectos perduran en la base de datos.

Para anular y recuperar transacciones, el método más usado consiste en utilizar un *Log-File* en el que se guarda toda la información necesaria para deshacer o rehacer las transacciones. Este archivo contiene:

- Identificador de transacción.
- Hora de modificación.
- Identificador de registro afectado.
- Tipo de acción.
- Valor anterior/posterior de registro.

Para manejar de forma eficiente el contenido de un *Log-File* se hace uso de Puntos de Control o *CHECKPOINTS*, que permite no recorrer todo el fichero cada vez que se quiera realizar una de las acciones anteriores, si no que van marcando puntos del fichero como posibles inicios de lectura.



4. Constraints

Los *constraints* o restricciones validan que los datos introducidos en la base de datos cumplan con un criterio establecido.

```
1  ALTER TABLE <nombre_tabla>
2  ADD CONSTRAINT <nombre_constraint> CHECK <condición>;
```

Seguridad y Privacidad de las Bases de Datos

La seguridad de las bases de datos se basa en la protección de las mismas frente a accesos malintencionados.

En relación al SGBD, debe mantener información de los usuarios, su tipo y los accesos y operaciones permitidas a éstos.

1. Tipos de usuarios

- DBA *(Data Base Administrator)*. (Página 6)
- Usuarios con derecho a crear, borrar y modificar objetos y que además puede conceder privilegios a otros usuarios sobre los objetos que ha creado.
- Usuarios con derecho a consultar, actualizar y/o sin derecho a crear o borrar objetos.

2. Privilegios

EXECUTE, INSERT, DELETE, UPDATE, SELECT.

Otro mecanismo de seguridad que ofrecen los SGBD en entregar información a los usuarios es a través de vistas *(CREATE VIEW)*, que son subesquemas de una base de datos.

En SQL:

```
1  CREATE VIEW <nombre_vista> AS <expresión_SQL>
2  GRANT/REVOKE <lista_privilegios>
3  ON <lista_objetos>
4  TO <lista_usuarios>;
```

3. Creación de roles

Un rol define un tipo de usuario de la base de datos que tiene concedidos una serie de autorizaciones sobre la misma.

En SQL:

```
1  CREATE ROLE <nombre_rol>
2  GRANT <lista_privilegios>
3  ON <lista_objetos>
4  TO <lista_roles>;
```

```
1  GRANT <lista_privilegios>
2  TO <lista_roles>;
```

Bases de Datos Distribuidas y Replicadas

1- Bases de Datos Distribuidas

Una base de datos distribuida es una base de datos almacenada y gestionada por varios nodos comunicados entre sí, de forma que funcionan como una sola base de datos. Destaca su gran autonomía.

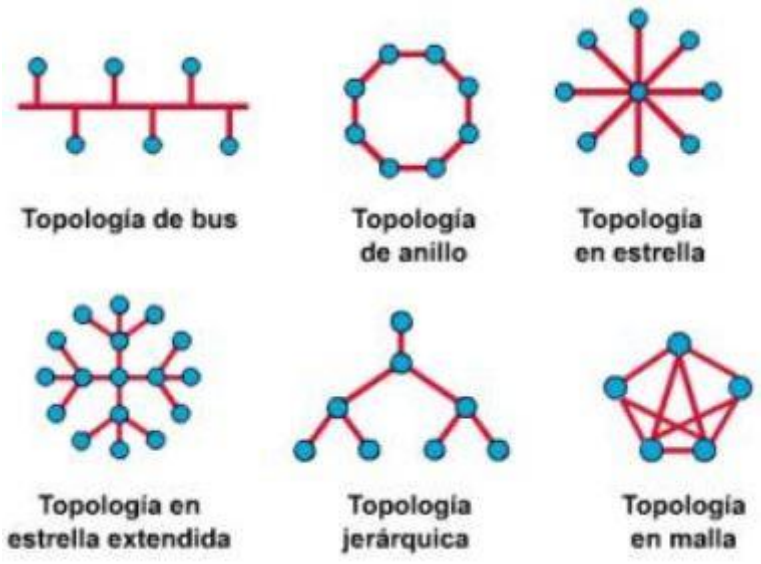
Para la comunicación entre los distintos nodos se hace uso de las transacciones, procesos por los cuales se interactúa con el SGBD para llevar a cabo las peticiones que realicen los usuarios.

- **Transacciones locales:** transacción que requiere un único agente[1] para poder llevarla a cabo.
- **Transacciones globales:** transacción que requiere varios agentes para poder satisfacer la transacción solicitada.

El **Sistema de Gestión de Bases de Datos Distribuida (SGBDD)** es el software que gestiona este tipo de bases de datos. Estos pueden ser:

- **Homogéneos:** si todos los SGBD locales hacen uso del mismo *Dictionay Data Language* o un subconjuto del mismo.
- **Heterogéneos:** si todos los SGBD son diferente.

Tipos de Bases de Datos Distribuidas (Topologías)



[1] Agente: proceso que coopera para ejecutar una transacción en un sitio concreto.

Fragmentación de Datos

- **Horizontal:**

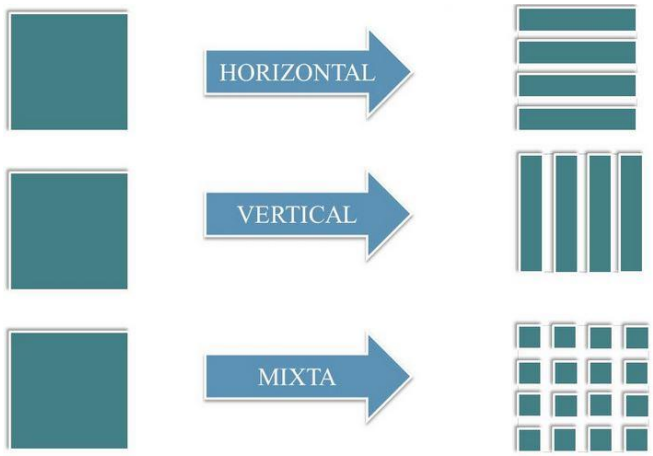
```
1  SELECT *
2  FROM <nombre_tabla>
3  WHERE <condición>;
```

- **Vertical:**

```
1  SELECT <lista_atributos>
2  FROM <nombre_tabla>;
```

- **Mixta:**

```
1  SELECT <lista_atributos>
2  FROM <nombre_tabla>
3  WHERE <condición>;
```



Catálogo de la Base de Datos

Elemento del sistema que contiene la información de la base de datos y los datos necesarios para el control distribuido de los mismos en los diferentes nodos. Encontramos distintas formas de almacenar esta información:

- **Centralizada:** se almacena una copia del catálogo en un nodo central. Depende de dicho nodo.
- **Réplicas completas:** se almacena una copia en cada nodo. Falta de autonomía.
- **Dividido:** cada nodo almacena el catálogo en su contenido local. El catálogo se obtiene de la unión de todos los catálogos. Operaciones no locales muy costosas.
- **Centralizado-Dividido:** cada nodo mantiene su propio catálogo local y además un nodo central único mantiene una copia centralizada.

Administrador de transacciones

- Transforma una consulta de alto nivel en otra equivalente que se ejecute con una estrategia más eficiente.
- Intercambia datos entre los diferentes nodos.
- Calibra la elección del orden de las operaciones relacionales y el mejor sitio para procesar los datos.

2- Bases de Datos Replicadas (Replicación de Bases de Datos)

Proceso consistente en la copia y mantenimiento de objetos de la base de datos en múltiples bases de datos que forman un sistema de bases de datos distribuido. Permite proporcionar a los usuarios acceso a los datos en todo lugar y en todo momento. Los cambios aplicados a un nodo se capturan y almacenan localmente antes de reenviarlos y aplicarlos a cada una de las ubicaciones remotas, es decir, son asíncronas.

Los nodos no colaboran en las transacciones, a diferencia que las distribuidas.

Ventajas	Inconvenientes
- Disponibilidad	- Complejidad
- Fiabilidad	- Sobrecarga de operaciones
- Rendimiento	- Gestión de concurrencia y recuperación de la base de datos
- Reducción de la carga	
- Procesamiento Offline	
- Soporte multiusuario	
- Soporta aplicaciones avanzadas	

Objeto de replicación

Objeto de la base de datos que existen en múltiples servidores de un sistema de bases de datos distribuido.

Grupos de replicación

Colección de objetos de replicación que están lógicamente relacionados. Facilita la administración de los objetos y pueden existir en múltiples sitios de replicación.

Sitios de replicación

- **Maestros:** controla uno o más grupos de replicación. Mantiene y propaga copias del grupo de replicación a los sitios esclavos.
- **Esclavos:** pueden contener la copia del grupo de replicación o parte de ella. Contienen la imagen del grupo de replicación.

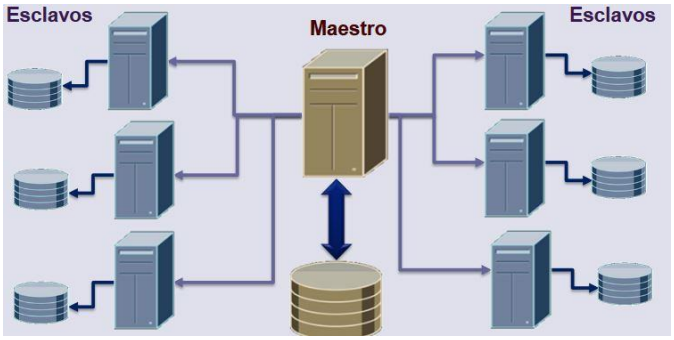
Entornos de la replicación

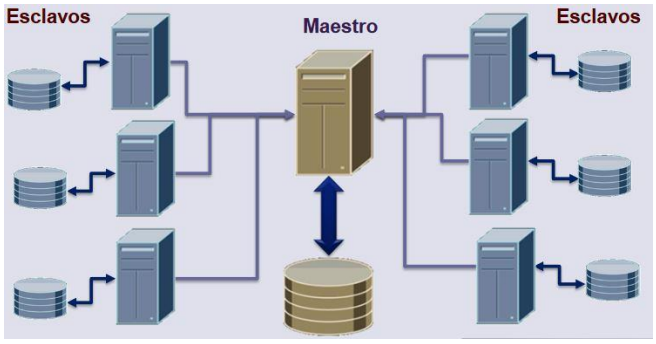
- **Replicación síncrona:** los datos replicados se actualizan de forma inmediata al actualizar los datos de origen.
- **Replicación asíncrona:** hay un retardo entre la actualización de datos origen y destino.

Propiedad Maestro/Esclavo

Los datos son asíncronamente replicados por el sitio maestro que tiene privilegios de actualización de los mismo.

- **Publicación/Suscripción:** el sitio maestro publica los datos y los sitios esclavos se suscriben, recibiendo copias de sólo lectura. No existen copias de datos iguales en diferentes sitios maestros.





Servidor de replicación

Técnica alternativa y potencialmente más eficiente que permite la actualización síncrona y asíncrona.

- **Características:**
  1. **Escalabilidad:** debe gestionar la replicación tanto de pequeños como de grandes volúmenes de datos.
  2. **Mapeado y transformación:** debe ser capaz de gestionar la replicación entre distintos SGBD y plataformas.
  3. **Replicación de objetos:** debe ser posible replicar objetos además de los datos (índices, *triggers*, etc).
  4. **Especificación del esquema de replicación:** debe proporcionar mecanismos para permitir que un usuario con privilegios especifique los objetos y datos que se desea replicar.
  5. **Mecanismo de suscripción:** permitir la inicialización de una réplica destino.
  6. **Fácil admisión:** simplicidad para el DBA de administrar el sistema, comprobar el estado y monitorizar el rendimiento del mismo.

- **Elementos:**
  1. **Actualizaciones transaccionales:** misma estructura de origen y destino.
  2. **Instantáneas:** permiten la replicación asíncrona del estado de la base de datos origen de acuerdo a un calendario mediante una gestión de colas para la actualización de réplicas.
  3. ***Triggers:*** es responsabilidad del usuario generar código dentro de un *trigger* que se ejecute cada vez que se produzca un evento. (Gran consumo y complejidad. No pueden planificarse.)