TEMA 4: EL MODELO RELACIONAL. ESTÁTICA

1. Historia y Objetivos Del Modelo

En 1970, E.F. Codd[1] propone un modelo de datos basado en la teoría de las relaciones, donde los datos se estructuran lógicamente en forma de relaciones (representadas en forma de tablas), siendo un objetivo fundamental del modelo mantener la independencia de esta estructura lógica respecto al modo de almacenamiento y a otras características de tipo físico.

Objetivos:

- **Independencia física**: Que el modo en que se almacenan los datos no influya en su manipulación lógica y, por tanto, no sea necesario modificar los programas por cambios en el almacenamiento físico. Codd concede mucha importancia a la independencia de la representación lógica de los datos respecto de su almacenamiento interno (independencia de **ordenación**, independencia de **indexación** e independencia de los **caminos de acceso**).
- **Independencia lógica**: Que añadir, modificar o eliminar objetos de la base de datos no repercuta en los programas y/o usuarios que están accediendo a subconjuntos parciales de los mismos.
- **Flexibilidad**: Poder presentar a cada usuario los datos de la forma que éste prefiera.
- **Uniformidad**: Las estructuras lógicas de los datos presentan un aspecto uniforme, lo que facilita la concepción y manipulación de la base de datos por parte de los usuarios.
- **Sencillez**: Las características anteriores, así como lenguajes de datos sencillos, producen como resultado que el modelo de datos relacional sea fácil de comprender y de utilizar por parte del usuario final.

Para conseguir los objetivos mencionados, Codd introduce el concepto de relación —tabla— como estructura básica del modelo. Todos los datos de una base de datos relacional se representan en forma de relaciones —tablas— cuyo contenido varía en el tiempo. Formalmente, una relación es un conjunto de filas en la terminología relacional.

Con el fin de proporcionar estructuras más regulares en la representación de relaciones y eliminar dependencias entre atributos que originan anomalías de actualización de la base de datos surge la **teoría de la normalización** cuyas tres primeras formas normales fueron introducidas por Codd, y constituye el soporte para el diseño de bases de datos relacionales.

Con respecto a la componente dinámica del modelo, se propone un conjunto de operadores que se aplican a las relaciones, algunos de ellos son clásicos de la teoría de conjuntos, mientras que otros fueron introducidos específicamente para el modelo relacional. Todos ellos conforman el álgebra relacional, definida formalmente por Codd.

1 Codd, E.F. A Relational Model of Data for Large Shared Data Banks. CACM 13, 6 pp 377-387. 1970

2. ESTÁTICA DEL MODELO RELACIONAL

El elemento básico del modelo relacional es la relación, que puede representarse en forma de tabla:

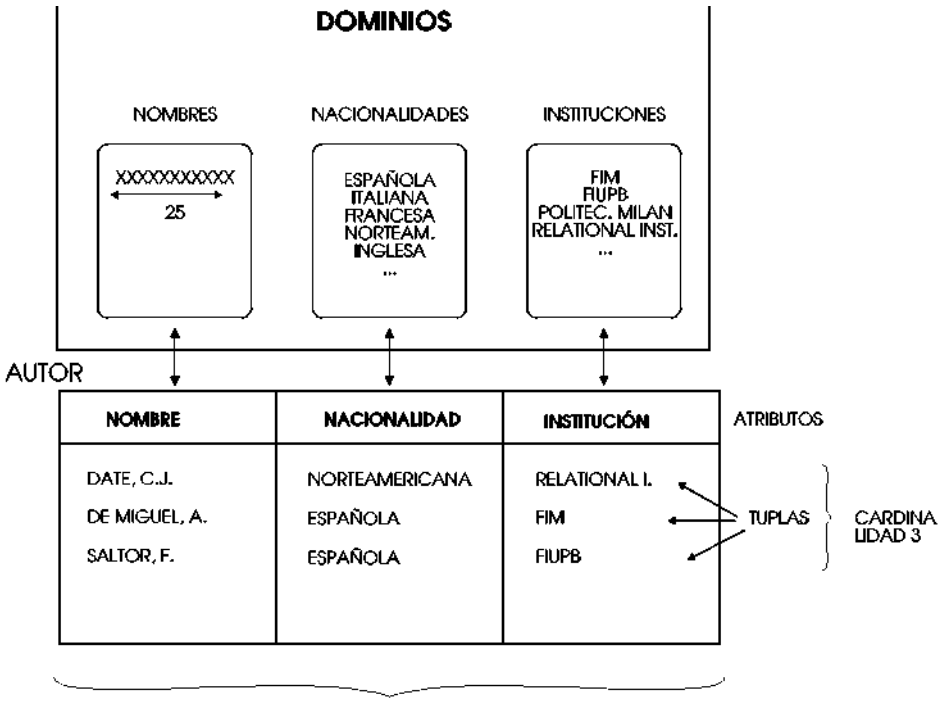
Nombre

atributo_1	*atributo_2*	*…*	*atributo_n*

En ella podemos distinguir un conjunto de columnas, denominadas ***atributos***, que representan propiedades de la misma y que están caracterizadas por un nombre, y un conjunto de filas llamadas ***tuplas***, que son las ocurrencias de la relación. El número de filas de una relación se llama **cardinalidad**, mientras que el número de columnas es el **grado**. Existen también **dominios** de donde los atributos toman sus valores.

Una relación siempre tiene un nombre y en ella es posible distinguir una cabecera (esquema de relación o intensión) que define la estructura de la tabla, es decir sus atributos con los dominios subyacentes, y un cuerpo, extensión, que está formado por un conjunto de tuplas que varían en el tiempo.

El hecho de que la relación se represente en forma de tabla es la causa de que los productos relacionales y los usuarios utilicen habitualmente el nombre de *tabla* para referirse a las relaciones, y como consecuencia de ello llame *filas* a las tuplas y *columnas* a los atributos.



2.1. DOMINIO Y ATRIBUTO

Dominio es un conjunto finito de valores homogéneos y atómicos, caracterizados por un nombre. **Homogéneos**, porque son todos del mismo tipo, y **atómicos** porque son indivisibles en lo que al modelo se refiere. Todo dominio debe tener un nombre por el cual nos referiremos a él y un tipo de datos. Los dominios pueden definirse por intensión (por ejemplo, edades) o por extensión (por ejemplo, nacionalidades).

Un atributo es el papel que tiene un determinado dominio en una relación.

El universo del discurso de una base de datos relacional, está compuesto por un conjunto finito y no vacío de atributos estructurados en relaciones. Cada atributo toma sus valores de un único dominio (dominio subyacente) y varios atributos pueden tener el mismo dominio subyacente.

Valores nulos: En ocasiones, pude ocurrir que el valor de un atributo para una fila sea desconocido.[2] En esos casos, se asocia a esa ocurrencia un valor *nulo* para ese atributo. No hay que confundir un valor nulo con una cadena de caracteres vacía o con un valor numérico igual a cero.

2.2. RELACIÓN

De la misma forma que hicimos con las entidades de modelo E/R, donde distinguíamos entre tipo y ocurrencia, conviene distinguir dos conceptos en la noción de relación, a los que ya nos hemos referido:

- **Intensión o Esquema de relación**, es la parte definitoria y estática de la relación (cabecera de la tabla).
- **Extensión, ocurrencia o instancia de relación**, el conjunto de tuplas que, en un instante determinado, satisface el esquema correspondiente.

2.2.1. INTENSIÓN DE UNA RELACIÓN:

AUTOR (NOMBRE: NOMBRES, NACIONALIDAD: NACIONALIDADES,
INSTITUCIÓN: INSTITUCIONES)

2.2.2. EXTENSIÓN DE LA RELACIÓN:

AUTOR

NOMBRE	NACIONALIDAD	INSTITUCIÓN
DATE, C.J.	NORTEAMERICANA	RELATIONAL INS.
DE MIGUEL, A.	ESPAÑOLA	FIM
SALTOR	ESPAÑOLA	FIUPB
CERI	ITALIANA	POLITÉCNICO MILÁN

2.3. CLAVES

Clave candidata: es un conjunto no vacío de atributos que identifican unívoca y mínimamente cada tupla.
Clave primaria: la que el usuario escoge de entre las claves candidatas.

[2] En la versión 2 del Modelo Relacional, Codd habla de valores desconocidos o no aplicables y de *marcas* en lugar de nulos.

Claves alternativas: claves candidatas que no han sido escogidas como clave primaria.
Clave ajena: conjunto de atributos de una relación cuyos valores han de coincidir con los valores de la clave primaria de otra relación (no necesariamente distinta). La clave ajena y la clave primaria han de estar definidas necesariamente sobre los mismos dominios.

3. RESTRICCIONES

3.1. RESTRICCIONES INHERENTES

- No puede haber dos tuplas iguales.
- El orden de las tuplas no es significativo.
- El orden de los atributos (columnas) no es significativo.
- Cada atributo sólo puede tomar un único valor del dominio, no admitiéndose por tanto los grupos repetitivos.
- Se debe cumplir la regla de *integridad de entidad*: "Ningún atributo que forme parte de la clave primaria de una relación puede tomar un valor desconocido o inexistente".

3.2. RESTRICCIONES DE USUARIO

Integridad referencial: "Si una relación R_2 (relación que referencia) tiene un descriptor que es la clave primaria de la relación R_1 (relación referenciada), todo valor de dicho descriptor debe concordar con un valor de la clave primaria de R1 o ser nulo". R_1 y R_2 son relaciones no necesariamente distintas. Además, la clave ajena puede ser también parte de la clave principal de R_2.
Además de definir las claves ajenas, hay que determinar las acciones que deben tomarse como consecuencia de operaciones de modificación y borrado realizadas sobre filas de la tabla referenciada. Podemos distinguir:

- ***Operación RESTRINGIDA***: Sólo se puede borrar una fila de la tabla que tiene la clave primaria referenciada si no existen filas con esa clave ajena en la tabla que referencia.
- ***Operación con transmisión en cascada***: El borrado o la modificación de una fila de la tabla que contiene la clave primaria lleva consigo el borrado o la modificación en cascada de las filas de la tabla que referencia cuya clave ajena coincide con el valor de la clave primaria de la tabla referenciada.
- ***Operación con puesta a nulos***: El borrado o la modificación de una fila de la tabla que contiene la clave primaria lleva consigo la puesta a nulos de los valores de la clave ajena de las filas de la tabla que referencia cuya clave ajena coincide con el valor de la clave primaria de la tabla referenciada.

Otras restricciones de usuario son las **dependencias** entre atributos, estudiadas en el siguiente tema, Teoría de la Normalización.

4. MODELO LOGICO DE DATOS. Obtención Modelo Lógico de Datos (MLD) a partir del Modelo Conceptual de Datos (MCD).

4.1. ETAPAS DEL DISEÑO LÓGICO

En el diseño lógico se deben coordinar exigencias casi siempre encontradas, como son eliminar redundancias, conseguir la máxima simplicidad y evitar cargas suplementarias de programación, obteniendo una estructura lógica adecuada que venga a establecer el debido equilibrio entre las exigencias de los usuarios y la eficiencia.

Habrá que conseguir, por tanto, equilibrar los distintos requisitos exigidos al sistema: flexibilidad, confidencialidad, integridad, tiempo de respuesta, etc., estableciendo unas prioridades y adoptando una solución de compromiso.

Etapas dentro del diseño lógico :

a) Diseño lógico estándar: A partir del esquema conceptual de la etapa anterior, se elabora un esquema lógico estándar.
   El esquema lógico estándar se describirá utilizando el lenguaje estándar del modelo de datos (SQL) en nuestro caso, por ser el lenguaje estándar de las SGBD Relacionales.

b) Diseño lógico específico : Con el esquema lógico estándar, y teniendo en cuenta el modelo lógico específico propio del SGBDR (ORACLE, INFORMIX, SQL Server, ...) se elabora el esquema lógico específico, que será descrito en el Lenguaje de Definición de Datos (LDD) del producto comercial que estemos utilizando.

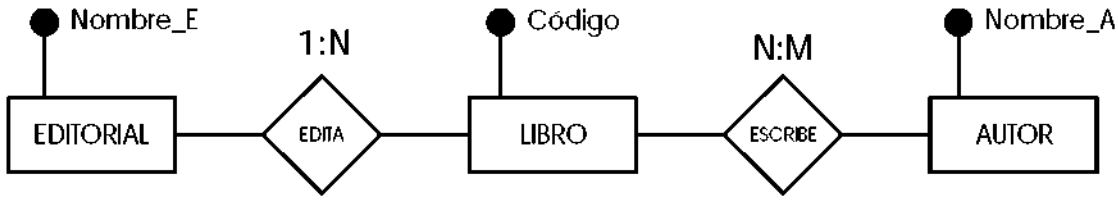
4.2. TRANSFORMACIÓN DEL ESQUEMA CONCEPTUAL AL LÓGICO ESTÁNDAR.

El paso de un esquema en el modelo E/R al relacional esta basado en los tres principios siguientes:

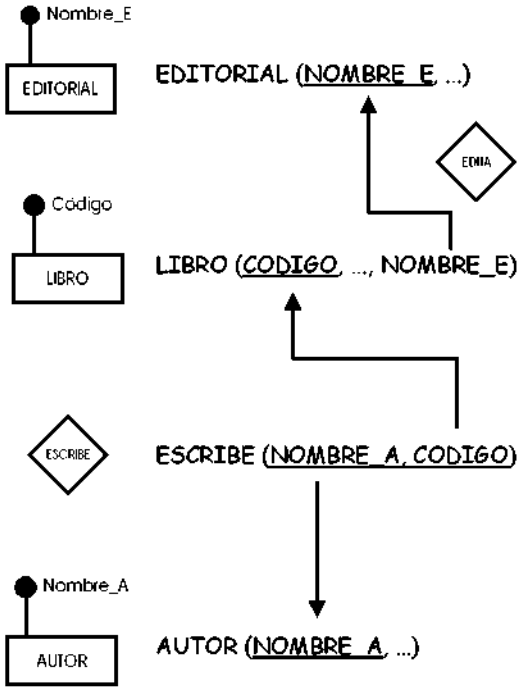
1. Todo tipo de entidad se convierte en una relación.
2. Todo tipo de interrelacion N:M se transforma en una relación.
3. Todo tipo de interrelacion 1:N se transforma mediante un fenómeno de *propagación de clave* o se crea una nueva relación.

A primera vista puede observarse que con la transformación de un esquema E/R a un esquema relacional se pierde semántica puesto que tanto las entidades como las interrelaciones se transforman en relaciones. También hay perdida de semántica en la propagación de clave, donde desaparece incluso la relación 1:N.
Esta pérdida de semántica no implica un peligro para la integridad de la BD, ya que pueden definirse restricciones de integridad referencial que aseguren la conservación de la misma.

Ejemplo:
Esquema Entidad/Interrelación:



Esquema Relacional:

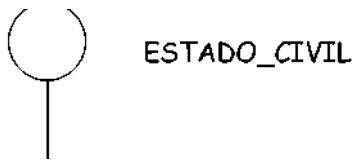


4.3. REGLAS DE TRANSFORMACIÓN

4.3.1. Transformación de dominios

En el modelo relacional estándar un dominio es un objeto más, propio de la estructura del modelo y tendrá su definición concreta en el LDD.

Ejemplo:
Esquema Entdad/Interrelación:



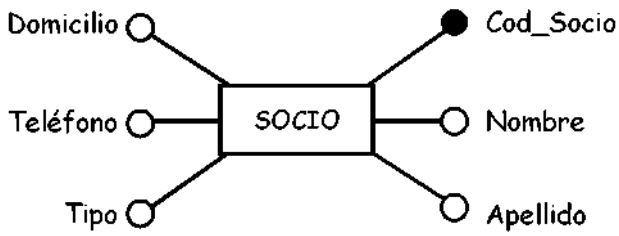
Esquema Relacional:

DOMINIO E_CIVIL CHAR(1) IN ('S','C','V','D')

4.3.2. Transformación de entidades

Cada tipo de entidad se convierte en una relación.
El modelo lógico estándar posee el objeto RELACION o TABLA mediante el cual representaremos las entidades. La tabla se llamará igual que el tipo de entidad de donde proviene. Para su definición dispondremos en SQL de la sentencia CREATE TABLE.
En este caso la transformación es directa y no hay pérdida de semántica.

Ejemplo:
Esquema Entidad/Interrelación :



Esquema Relacional:
SOCIO (COD_SOCIO, NOMBRE, APELLIDO, DOMICILIO, TELEFONO, TIPO)

SOCIO

COD *SOCIO*	NOMBRE	APELLIDO	DOMICILIO	TELEFONO	TIPO
00001	SUSANA	HIDALGO	RIOS ROSAS 2	914138060	I
00002	ADOLFO	SANCHEZ	S. BERNAR 44	914131419	P

4.3.3. Transformación de Atributos de Entidades

Cada atributo de una entidad se transforma en una columna de la relación a la que ha dado lugar la entidad. Pero teniendo en cuenta que tenemos atributos identificadores principales, alternativos y el resto, cada uno de los diferentes tipos sufrirá un tipo de transformación diferente:

Atributos Identificadores Principales: El atributo(s) identificador principal de cada tipo de entidad pasan a ser la Clave Primaria de la relación.

El lenguaje lógico estándar recoge directamente este concepto por medio de la cláusula PRIMARY KEY en la descripción de la tabla, luego la transformación es directa y no hay perdida de semántica.

Atributos Identificadores Alternativos: El modelo lógico estándar recoge por medio de la cláusula UNIQUE estos objetos, ya que son soportados directamente por el modelo relacional. Al ser la transformación directa no hay perdida de semántica.

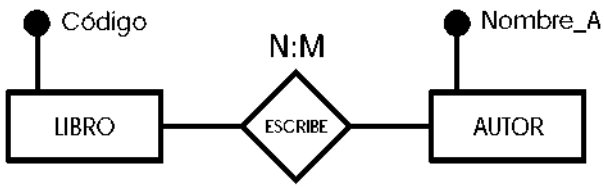
Atributos No Identificadores: Los atributos no principales pasan a ser columnas de la tabla, las cuales tienen permitido tomar valores nulos, a no ser que se indique lo contrario.

4.3.4. Transformaciones de Relaciones

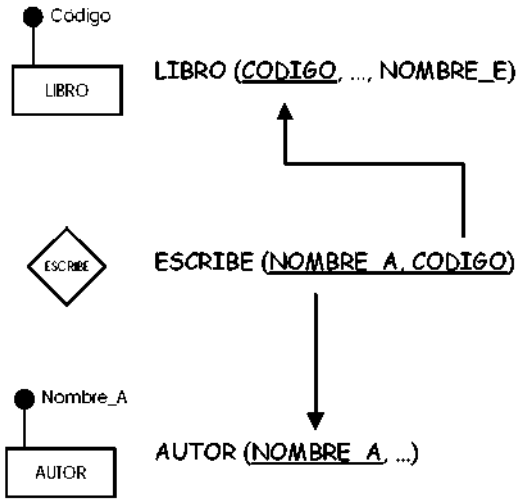
Relaciones N:M: Un tipo de relación N:M se transforma en una relación que tendrá como clave primaria la concatenación de los AIP de los tipos de entidad que asocia. Hay perdida

de semántica, ya que no hay manera de diferenciar dentro de un esquema relacional qué relaciones provienen de una entidad y cuáles proceden de la transformación de relaciones.

Ejemplo:
Esquema Entidad/Interrelación:



Esquema Relacional:



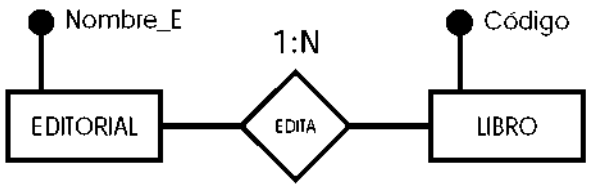
Cada uno de los atributos que forman la clave primaria de esta relación son una CLAVE AJENA respecto a cada una de las tablas donde este atributo es clave primaria, lo que se especifica en el lenguaje lógico estándar a través de la clausura FOREIGN KEY dentro de la sentencia de creación de la tabla. Habrá que estudiar qué ocurre en el caso de borrado o modificación de la clave primaria referenciada.

Otra característica que debemos recoger en esta transformación son las cardinalidades máxima y mínima de cada una de las entidades que participan en la relación. La cardinalidad mínima se transforma mediante la admisión o no de valores nulos en la entidad que propaga su AIP y la máxima mediante la especificación de restricciones o aserciones.

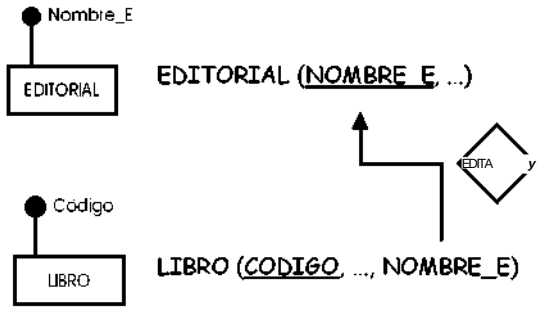
Relaciones 1:N: Existen dos soluciones para la transformación de una relación 1:N.

(a) Propagar el AIP del tipo de entidad que tiene cardinalidad máxima 1 a la que tiene cardinalidad máxima *n*, desapareciendo el nombre de la relación, con la consiguiente pérdida de semántica.

Ejemplo:
Esquema Entidad/Interrelación:



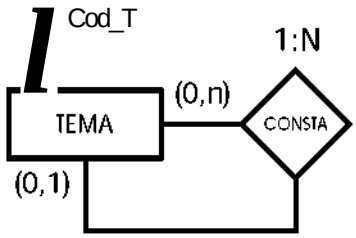
Esquema Relacional:



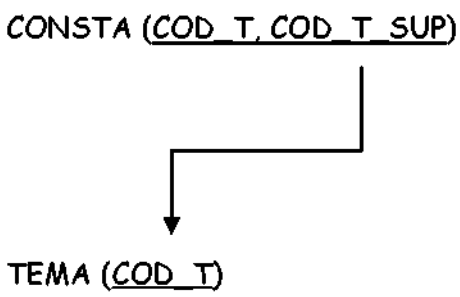
(b) *Transformarlo en una relación, como si se tratara de una relación N:M*
Los casos en los que es mejor transformar la relación en una relación son los siguientes:

1. Cuando el número de ocurrencias relacionadas de la entidad que propaga su clave es muy pequeño y cabe, por tanto, la posibilidad de que existan muchos valores nulos.
2. Cuando la relación tiene atributos propios.

Ejemplo :
Esquema Entidad/Interrelación:



Esquema Relacional:
Solución (a):



Solución (b):

TEMA (COD_TEMA,...., COD_TEMA_SUP)

Relaciones 1:1

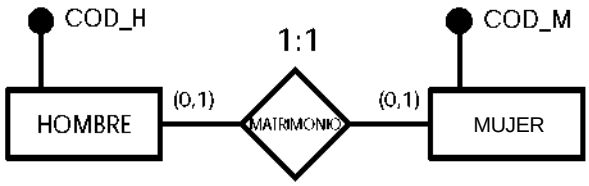
Este es un caso particular de las interrelaciones con tipo de correspondencia 1:N. No hay una regla fija para su transformación, pudiéndose crear una nueva tabla o transformarla mediante una propagación de clave.

Casos:

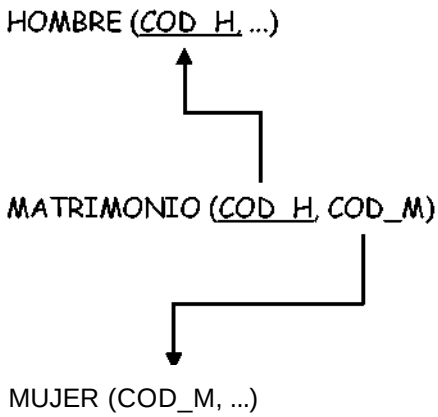
(a) Si las entidades que se asocian poseen cardinalidades (0,1), entonces la interrelación se transforma en una relación, además de las dos relaciones a las que se transforman cada una de las entidades.

Ejemplo :

Esquema Entidad/Interrelación:



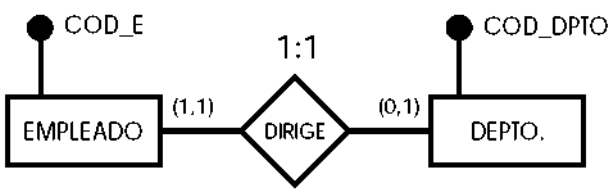
Esquema Relacional:



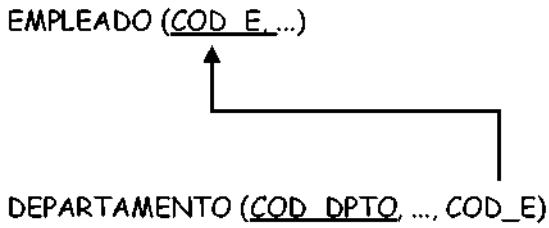
(b) Si una de las entidades que participa en la relación posee cardinalidad (0,1), mientras que en la otra es (1,1), conviene propagar la clave de la entidad con cardinalidad (1,1) a la tabla resultante de la entidad de cardinalidades (0,1) con el fin de evitar que aparezcan valores nulos.

Ejemplo:

Esquema Entidad/Interrelación:



Esquema Relacional:



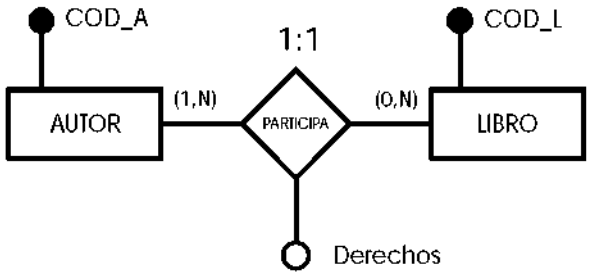
(c) En el caso de que ambas entidades presenten cardinalidades (1,1), se puede propagar la clave de cualquiera de ellas a la tabla resultante de la otra, teniendo en cuenta en este caso los accesos mas frecuentes y prioritarios a los datos de las tablas.

4.3.5. Transformación de Atributos de Relaciones

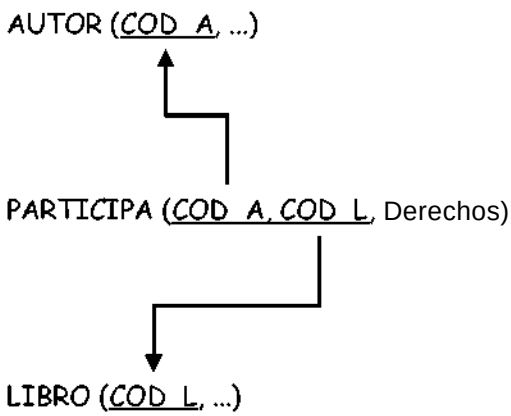
Si la relación se transforma en una relación, todos sus atributos pasan a ser columnas de la relación.

Ejemplo: La relación “escribe” entre AUTORES y LIBROS tiene un atributo “derechos” que representa los derechos del autor que éste recibe por el libro.
La transformación es directa y no hay perdida de semántica.

Ejemplo:
Esquema Entidad/Interrelación:



Esquema Relacional:



4.3.6. Transformación de Jerarquías de Tipos y Subipos

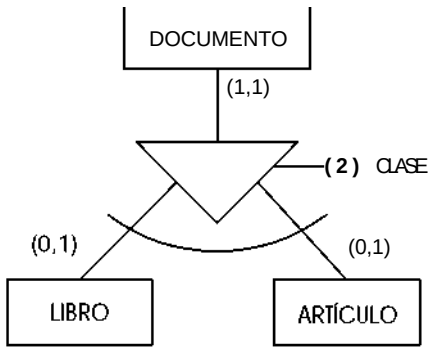
Casos:

Opción a: Englobar todos los atributos de la entidad y sus subtipos en una sola relación. Adoptaremos esta solución, cuando los subtipos se diferencien en muy pocos atributos y las relaciones que los asocian con el resto de entidades del esquema sean las mismas para todos los subtipos.

Opción b: Crear una relación para el supertipo y tantas relaciones como subtipos haya, con sus atributos correspondientes. Esta es la solución cuando existen muchos atributos distintos entre los subtipos y se quieren mantener de todas las maneras los atributos comunes a todos ellos en una relación.

Opción c: Considerar las relaciones distintas para cada subtipo, que contengan además los atributos comunes. Se elegirá esta opción cuando se dieran las mismas condiciones que en el caso anterior (muchos atributos distintos) y los *accesos* realizados sobre los datos de los distintos subtipos siempre afectan a atributos comunes.

Ejemplo :
Esquema Entidad/Interrelación:



Esquema Relacional:

a) DOCUMENTO(CODIGO, TITULO, IDIOMA, ... TIPO)

b) DOCUMENTO(CODIGO, TITULO, IDIOMA, ...)
A ^

ARTICULO(CODIGO,...) ****

LIBRO(CODIGO,....)

c) LIBRO(CODIGO, TITULO, IDIOMA,...)

ARTICULO(CODIGO, TITULO, IDIOMA,...)

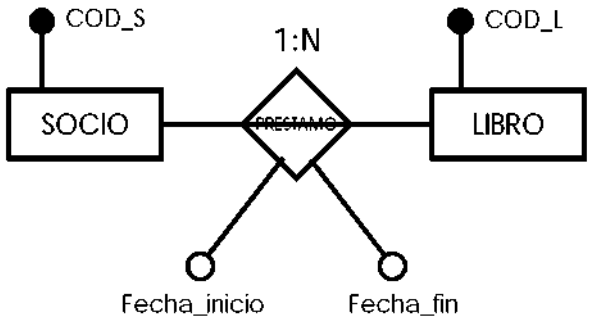
Podemos elegir entre tres estrategias distintas para la transformación de un tipo y sus subtipos al modelo relacional. Sin embargo, desde el punto de vista exclusivamente semántico la opción (b) es la mejor.

Por otra parte, desde el punto de vista de la eficiencia, tenemos que tener en cuenta que:

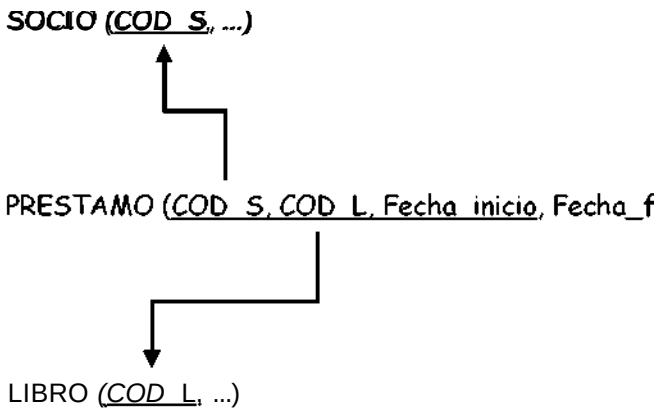
- Opción a: El acceso a una fila que refleje toda la información de una determinada entidad en mucho más rápido.
- Opción b: La menos eficiente, aunque es la mejor desde un punto de vista semántico.
- Opción c: Aumentamos la eficiencia ante determinadas consultas. Pierde mucha semántica.

4.3.7. Transformación de la dimensión temporal

Ejemplo:
Esquema Entidad/Interrelación:



Esquema Relacional:



5. DEFINICIÓN DE OBJETOS EN EL MODELO RELACIONAL. INTRODUCCIÓN AL DDL DEL SQL.

5.1. CREACIÓN DE DOMINIOS

5.2. TIPOS DE DATOS

Alfanuméricos
Numéricos
Fechas y Horas
Tipos de datos de usuario

5.2.1. Tipos de datos alfanuméricos

CHAR (n)
VARCHAR (n)

5.2.2. Tipos de datos numéricos

DECIMAL (precision [, escala])
INT
REAL

5.2.3. Fechas y horas

DATE
TIME
TIMESTAMP

5.3. Tipos de datos de usuario: Sentencia CREATE DATATYPE

CREATE DATATYPE tipo_usuario tipo_dato
... [[NOT] NULL]
... [DEFAULT valor_por_defecto]
... [CHECK (condicion)]

Los tipos de datos de usuario pueden tener cláusulas CHECK y valores por defecto asociados. También puede indicarse si admiten o no valores nulos. Estas condiciones son heredadas por cualquier columna definida sobre ese tipo de dato. Si sobre la columna se especifica cualquier otra condición, ésta prevalece sobre las condiciones del tipo de datos.

Ejemplos:

- La siguiente sentencia crea un nuevo tipo de dato llamado “dirección” que consiste en una cadena de 50 caracteres y que admite valores nulos.

  CREATE DATATYPE direccion CHAR (50) NULL

- La siguiente sentencia crea un tipo de dato llamado “id”, que consiste en un número entero positivo que no admite valores nulos.

  CREATE DATATYPE id INT NOT NULL CHECK (id >= 0)

5.4. Creación de relaciones (Tablas): CREATE TABLE

```
CREATE TABLE nombre_tabla (
definición_de_columna
[[CONSTRAINT nombre] restricción_de_columna ] | , ...
[[CONSTRAINT nombre] restricción_de_tabla ]
)
```

- definición de columna:
  ```
  nombre_columna tipo_dato [ NOT NULL ]
  [ DEFAULT valor_por_defecto ]
  ```

- restricción de columna:
  ```
  UNIQUE
  PRIMARY KEY
  REFERENCES nombre_tabla [( nombre_columna )] [acciones]
  CHECK ( condición )
  ```

- valor por defecto:

```
cadena_de_caracteres
número
CURRENT DATE
CURRENT TIME
CURRENT TIMESTAMP
NULL
```

- restricción de tabla:

```
UNIQUE ( nombre_columna, ... )
PRIMARY KEY ( nombre_columna, ... )
CHECK ( condición )
restricción_de_clave_ajena
```

- Restricción de clave ajena:

```
FOREIGN KEY [ alias ] [ ( nombre_columna, ... ) ]
REFERENCES nombre_tabla [ ( nombre_columna, ... ) ]
[ acciones ]
```

- Acciones:

```
[ ON UPDATE acción ] [ ON DELETE acción ]
Acción:
CASCADE
SET NULL
SET DEFAULT
RESTRICT
```

5.5. Desripción:

nombre_columna tipo_dato [NOT NULL] [valor por defecto]

Define una columna de la tabla. Los tipos de datos permitidos son los descritos anteriormente. Dos columnas en la misma tabla no pueden tener el mismo nombre.

Si se especifica NOT NULL, o si se le aplica una restricción UNIQUE o PRIMARY KEY, la columna no puede tomar valores nulos.

5.6. Restricciones de tabla:

Ayudan a mantener la integridad de los datos en la base de datos. Hay cuatro tipos de restricciones de integridad:

- **UNIQUE**: Identifica una o más columnas que identifican unívocamente cada fila de la tabla.
- **PRIMARY KEY**: Lo mismo que UNIQUE, salvo que una tabla sólo puede tener una PRIMARY KEY.
- **FOREIGN KEY**: restringe los valores para un conjunto de columnas que deben coincidir con los valores de la clave principal o una restricción UNIQUE de otra tabla.
- **CHECK**: impone condiciones de verificación (reglas) que deben cumplir los valores de una columna. Por ejemplo, podría utilizarse una restricción de este tipo para asegurarse de que la columna “sexo” sólo toma los valores “varón” o “mujer”.

5.7. Restricciones de columna:

Lo mismo que las restricciones de tabla. Por ejemplo, las siguientes sentencias son equivalentes:

```
CREATE TABLE MUNICIPIO (
Cod_Municipio int UNIQUE
)

CREATE TABLE MUNICIPIO (
Cod_Municipio int,
UNIQUE ( Cod_Municipio )
)
```

Normalmente se usan las restricciones de columna, salvo que la restricción afecte a más de una columna, en cuyo caso deben emplearse las restricciones de tabla.

5.8. RESTRICCIONES DE INTEGRIDAD

def_columna UNIQUE o UNIQUE (nombre_columna, ...)
Dos filas en la misma tabla no pueden tener valores idénticos en las columnas definidas como únicas. Una tabla puede tener más de una restricción UNIQUE.

Diferencias entre UNIQUE e índices únicos:
Las columnas sobre las que se define un índice único permiten nulos, mientras que si se define una restricción UNIQUE no. Por otro lado, una clave ajena puede referenciar a columnas sobre las que se ha definido una restricción UNIQUE pero no a columnas sobre las que se ha creado un índice único.

def_columna PRIMARY KEY o **PRIMARY KEY (nombre_columna, ...)**
La clave primaria de la tabla consiste en las columnas declaradas, y ninguna de esas columnas admite valores nulos. Si se usa la segunda forma, la clave primaria se crea con las columnas declaradas en el orden en el que se definieron, no en el orden en el que figuran en la cláusula.

def_columna REFERENCES nombre_tabla_padre [(nombre_columna_padre)]
La columna es una clave ajena de la clave principal o de la restricción UNIQUE de la tabla padre. (Normalmente será una clave ajena de una clave principal)
Si se especifica nombre de columna, ésta debe referirse a una columna de la tabla padre sobre la que se ha definido una restricción UNIQUE o PRIMARY KEY y esa restricción afecta sólo a esa columna. En otro caso, la clave ajena referencia a la clave principal de la tabla padre.

Las acciones de integridad referencial definen las acciones a tomar para mantener las relaciones de claves ajenas en la base de datos. Cuando se modifica o se borra una clave principal en una tabla, los valores correspondientes de claves ajenas en otras tablas deben modificarse igualmante. Pueden especificarse las cláusulas ON UPDATE , ON DELETE o ambas seguidas de una de las siguientes acciones:

FOREIGN KEY [alias] [(...)] REFERENCES nombre_tabla_padre [(...)]
La tabla contiene una clave ajena que referencia a la clave primaria o a una estricción UNIQUE de otra tabla. Normalmente, una clave ajena lo es de una clave principal.

CASCADE
Usado con ON UPDATE, modifica las claves ajenas al nuevo valor de la clave principal. Usado con ON DELETE, borra las filas cuyo valor de clave ajena coincide con el valor de la clave principal borrado.

SET NULL
Pone a NULL todos los valores de clave ajena que se correspondan con el valor de la clave principal borrada o modificada.

SET DEFAULT
Asigna el valor especificado por defecto por la cláusula DEFAULT para todos los valores de clave ajena que se correspondan con el valor de la clave principal borrada o modificada.

RESTRICT
Impide la modificación o el borrado de la clave principal si existen valores de claves ajenas que la referencian.

def_columna CHECK (condicion) o **CHECK (condicion)**
No se permiten filas que no cumplan la condición.

5.9. Creación de índices: CREATE INDEX

CREATE [UNIQUE][CLUSTERED|NONCLUSTERED] INDEX nombre_indice
ON nombre_tabla (nombre_columna [ASC | DESC], ...)

Crea un índice ordenado sobre las columnas especificadas de la tabla. Los índices son utlizados automáticamente por el gestor para aumentar el rendimiento de las consultas. Una vez que se ha creado un índice, no vuelve a referenciarse, salvo para borarlo mediante una sentencia DROP INDEX.

La restricción UNIQUE asegura que no habrá dos filas en la tabla con valores idénticos para las columnas del índice.

Por defecto se crea en orden ascendente.

El gestor crea automáticamente índices para las claves primarias y para las restricciones UNIQUE.

5.10. Modificación de la estructura de una tabla: ALTER TABLE

```
ALTER TABLE nombre_tabla
ADD def_columna [ restricción_de_columna ] |
ADD restricción de tabla
```

5.11. Borrado de objetos: DROP

```
DROP DATATYPE | INDEX | TABLE | VIEW
```