Bases de Datos

Tema 3

Modelo Relacional

Francisco Ruiz

oct-2000

documentación preparada con ayuda de Esperanza Marcos (Universidad Rey Juan Carlos) y Mario Piattini (Universidad de Castilla-La Mancha)

Tema 3

Modelo Relacional

Complementar con:

** capítulo 5 del libro "Fundamentos y Modelos de Bases de Datos" (2ª edición). De Miguel, A.; Piattini, M.; Ra-Ma, 1999.*

** capítulo 3 del libro "Diseño de Bases de Datos Relacionales". De Miguel, A.; Piattini, M.; Marcos, E.; Ra-Ma, 1999.*

- *Presentar el modelo relacional de datos (MR), que es el utilizado en la actualidad en la gran mayoría de los sistemas y herramientas de bases de datos.*
- *En especial, profundizar en los conceptos que permiten modelar la estática y construir esquemas relacionales.*

- **Principales:**
  - [de Miguel y Piattini, 1999]
    - cap. 5
    - De Miguel, A.; Piattini, M.; Fundamentos y Modelos de Bases de Datos (2ª edición). Ra-Ma, 1999.
  - [de Miguel et al, 1999]
    - cap. 3
    - De Miguel, A.; Piattini, M.; Marcos, E.; Diseño de Bases de Datos Relacionales. Ra-Ma, 1999.
- **Otras:**
  - Elmasri, R.; Navathe, S.B.; Sistemas de Bases de Datos: Conceptos fundamentales (2ª edición). Addison-Wesley, 1997. Capítulos 3 y 21.
  - Connolly, T.; Begg, C.; Straghan, A.; Database Systems (2nd edition). Addison-Wesley, 1999. Capítulo 3.

Índice

En 1970 Codd publicó en ACM un trabajo proponiendo un nuevo MD que perseguía una serie de ***objetivos***:

- **Independencia física**: El modo cómo se almacenan los datos no debe influir en su manipulación lógica y, por tanto, los usuarios que acceden a esos datos no han de modificar sus programas por cambios en el almacenamiento físico.
- **Independencia lógica**: Añadir, eliminar o modificar cualquier elemento de la BD no debe repercutir en los programas y/o usuarios que están accediendo a subconjuntos parciales de los mismos (vistas).
- **Flexibilidad**: Ofrecer a cada usuario los datos de la forma más adecuada a la correspondiente aplicación.
- **Uniformidad**: Las estructuras lógicas de los datos presentan un aspecto uniforme (tablas), lo que facilita la concepción y manipulación de la BD por parte de los usuarios.
- **Sencillez**: Las características anteriores, así como unos lenguajes de usuario muy sencillos, producen como resultado que el modelo relacional (MR) sea fácil de comprender y de utilizar por parte del usuario final.

- Codd concedió mucha importancia al tema de la **independencia** de la representación lógica de los **datos** respecto a su almacenamiento interno, que concretó en tres tipos de independencia:
  - de **ordenación**,
  - de **indización**, y
  - de los **caminos de acceso**).
- Importancia que Codd manifiesta explícitamente:

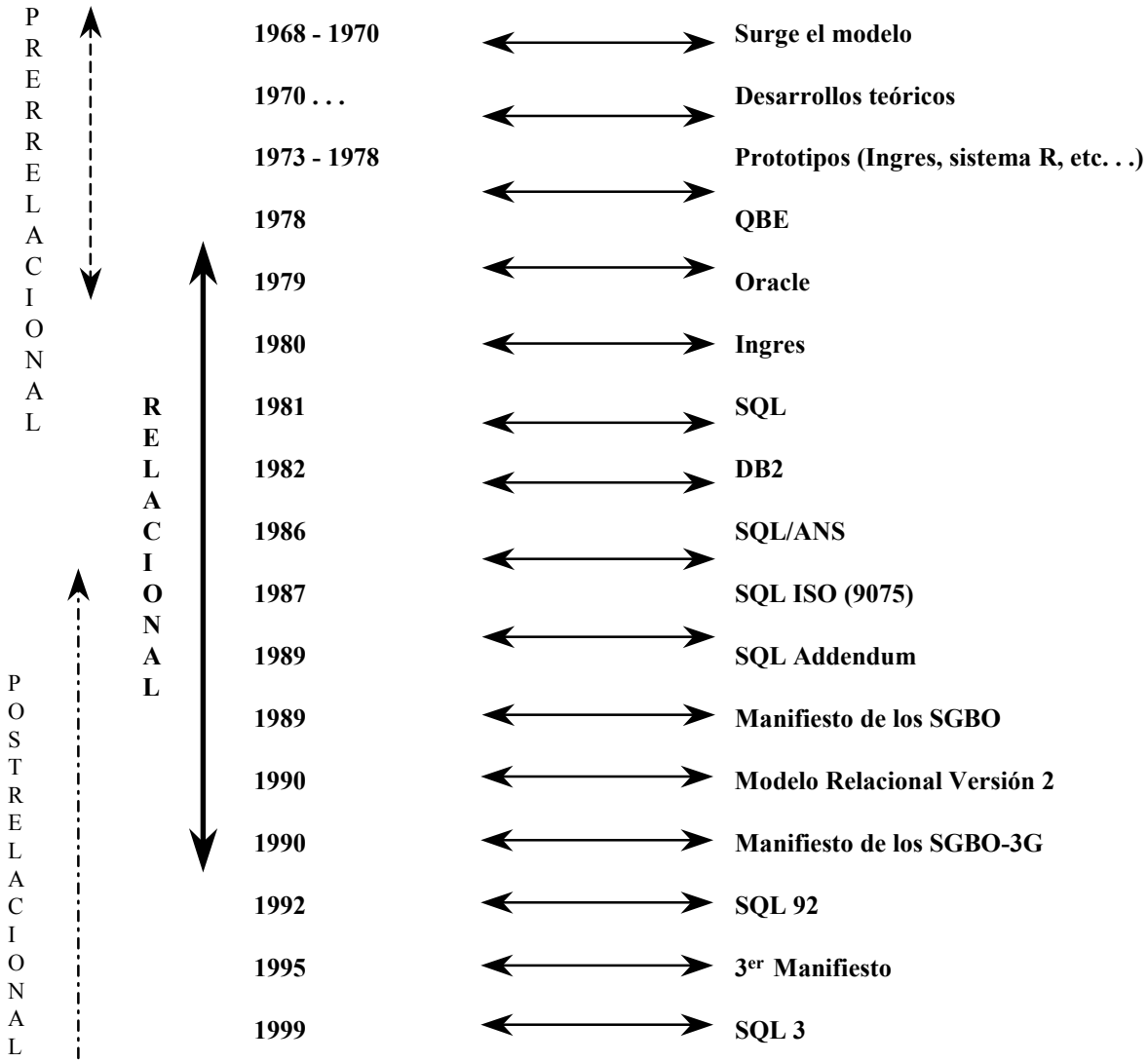
  "*... se propone un modelo relacional de datos como una base para proteger a los usuarios de sistemas de datos formateados de los cambios que potencialmente pueden alterar la representación de los datos, causados por el crecimiento del banco de datos y por los cambios en los caminos de acceso*".

- Los avances más importantes que el modelo de datos relacional incorpora respecto a los modelos de datos anteriores son:
  - **Sencillez y uniformidad**: Los usuarios ven la base de datos relacional como una colección de tablas, y al ser la tabla la estructura fundamental del modelo, éste goza de una gran uniformidad, lo que unido a unos lenguajes no navegacionales y muy orientados al usuario final, da como resultado la sencillez de los sistemas relacionales.
  - **Sólida fundamentación teórica**: Al estar el modelo definido con rigor matemático, el diseño y la evaluación del mismo puede realizarse por métodos sistemáticos basados en abstracciones.
  - **Independencia de la interfaz de usuario**: los lenguajes relacionales, al manipular conjuntos de registros, proporcionan una gran independencia respecto a la forma en la que los datos están almacenados.

- Las ventajas citadas han contribuido a que desde mediados de los años 80, el MR sea utilizado por prácticamente la totalidad de los SGBD comerciales. Este éxito se refleja en:
  - Algunas de las principales empresas informáticas del mundo, son en origen, empresas de SGBD: ORACLE, Sybase, INFORMIX, ...
  - Los grandes fabricantes de software tienen "su" SGBD relacional: IBM DB2, Microsoft SQL Server, ...
  - Existen bastantes SGBD diseñados para PC's y usuarios no expertos: Microsoft Access, etc.
- El tremendo éxito real del MR ha supuesto que el **cambio tecnológico** a la siguiente generación esté siendo **evolutivo** y no revolucionario:
  - Triunfan los *SGBD Objeto-Relacionales*, y
  - Fracasan, en general, los SGBD de Objetos puros.

- La aparición del MR representa un verdadero hito en el desarrollo de las bases de datos, ya que ha marcado tres etapas diferentes, conocidas como **generaciones de los SGBD's**:
  - **La prerrelacional -primera generación de BD's-**, en la cual los SGBD se soportan en los modelos Codasyl (en red) y Jerárquico.
  - **La relacional -segunda generación de BD's-** donde los sistemas relacionales se van aproximando a su madurez y los productos basados en este modelo van desplazando poco a poco a los sistemas de primera generación, hasta conseguir una mayor cuota en el mercado de las bases de datos.
  - **La postrelacional -tercera generación de Bd's-** en la que aparecen otros modelos de datos, en especial los orientados al objeto, que están en estos momentos intentando abrirse un hueco en el mercado de las bases de datos e integrándose como extensiones en los SGBD's previos de la generación relacional.

EVOLUCIÓN DEL MODELO RELACIONAL



Relación

Es la estructura básica del modelo relacional. Se representa mediante una ***tabla***

Atributo

Representa las propiedades de la relación. Se representa mediante una ***columna***

Dominio

Es el conjunto válido de ***valores*** que toma un atributo

Tupla

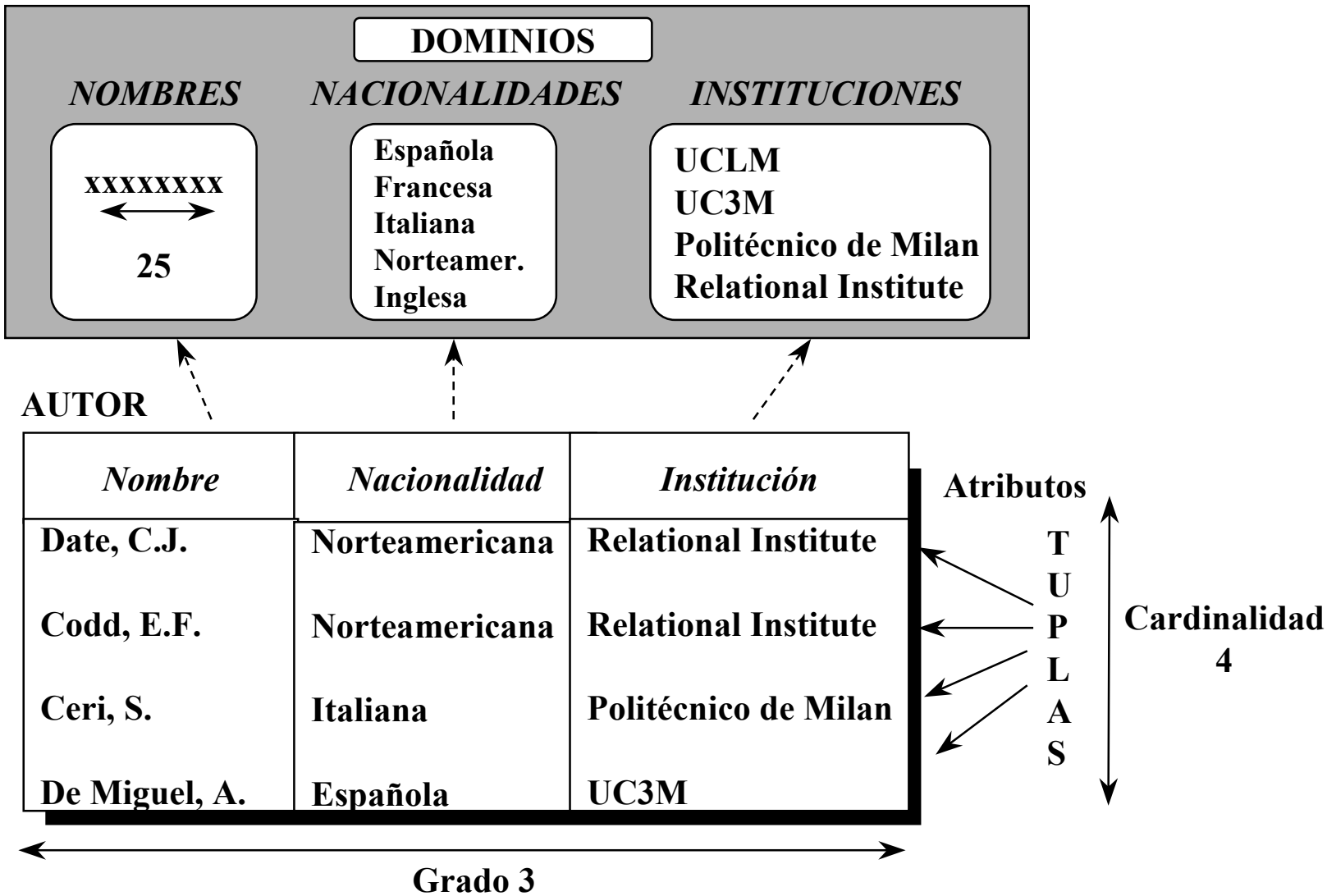
Es una ocurrencia de la relación. Se representa mediante una ***fila***

- La **relación** es el elemento fundamental del modelo relacional (de ahí el nombre del modelo), y se puede representar en forma de tabla:

NOMBRE

atributo 1	atributo 2	……	atributo n	
XXX	XXX	……	XXX	→ tupla 1
XXX	XXX	……	XXX	→ tupla 2
.....	.....	……	.....	.....
XXX	XXX	……	XXX	→ tupla m

- Pero, !CUIDADO!, una relación no es una tabla. Existen diferencias entre ambas estructuras.



Ejemplo de relación

Comparación de la terminología relación, tabla, fichero

RELACIÓN	~	TABLA	~	FICHERO
TUPLA		FILA		REGISTRO
ATRIBUTO		COLUMNA		CAMPO
GRADO		Nº DE COLUMNAS		Nº DE CAMPOS
CARDINALIDAD		Nº DE FILAS		Nº DE REGISTROS

Modelo Relacional (teoría)	SGBD Relacionales (implementación)	Sistemas de Ficheros Clásicos

- El UD de una BD relacional está compuesto por un conjunto de dominios $\{D_i\}$ y de relaciones $\{ R_i \}$ definidas sobre los dominios.
- Un **dominio** es un conjunto nominado, finito y homogéneo de valores atómicos =>
  - el dominio se identifica por un nombre,
  - tiene un número finito de valores,
  - todos los valores son del mismo tipo, y
  - los valores son atómicos respecto del MR, es decir, no pueden ser a su vez una relación, un grupo repetitivo, etc.
- Cada dominio puede definirse de dos maneras:
  - **Extensión** (dando sus posibles valores):
    - *días de la semana = {lunes, martes, miércoles, jueves, viernes, sábado, domingo}*
  - **Intensión** (mediante un tipo de datos):
    - edad = entero
  - A veces se asocia unidad de medida (kilos, metros, etc.) y/o ciertas restricciones (como un rango de valores).

- Un **atributo** (A) es la interpretación de un determinado dominio en una relación, es decir el "papel" que juega en la misma.
- Notación:

  D = Dom (A) => D es el dominio de A

- Un atributo y un dominio pueden llamarse igual, pero ..
  - Un atributo está siempre asociado a una relación, mientras que un dominio tiene existencia propia con independencia de las relaciones.
  - Un atributo representa una propiedad de una relación.
  - Un atributo toma valores de un dominio.
  - Varios atributos distintos (de la misma o de diferentes relaciones) pueden tomar sus valores del mismo dominio.

- Además de los dominios y atributos simples, que acabamos de definir, en ampliaciones posteriores del MR se ha introducido el concepto de dominio compuesto, que es muy útil en la práctica.
- Un **dominio compuesto** se puede definir como una combinación de dominios simples a la que se puede aplicar ciertas restricciones de integridad.
  - Ejemplo: el dominio compuesto denominado *Fecha* se construye por agregación de los dominios simples *Día*, *Mes* y *Año*, incorporando las adecuadas restricciones de integridad a fin de que no aparezcan valores no válidos para la fecha.
- Al igual que es posible definir dominios compuestos, existen también **atributos compuestos**.
- Tanto los atributos compuestos como los dominios compuestos pueden ser tratados, si así lo precisa el usuario, como "piezas únicas" de información, es decir, como valores atómicos.

- Matemáticamente, una **relación** definida sobre un conjunto de dominios $D_1 ... D_n$ (no necesariamente distintos) es un subconjunto del producto cartesiano de los n dominios, donde cada elemento de la relación (*tupla*) es una serie de n valores ordenados:
  - $R \subseteq D_1 \times D_2 \times ... \times D_n$ siendo n es el grado de la relación

- Esta definición no tiene en cuenta a los atributos, por eso en Bases de Datos se utiliza otra definición que incluye los siguientes elementos:
  - **nombre**,
  - **cabecera**,
  - **cuerpo**,
  - **esquema**, y
  - **estado**.

- **Nombre:** Las relaciones se identifican por un nombre.
  - ciertas relaciones que no necesitan identificarse (por ejemplo, resultados intermedios) pueden no tener nombre.
- **Cabecera de relación:** Conjunto de n pares atributo-dominio subyacente,

  $\{ (A_i : D_i) \}_{i=1}^{n}$ donde n es el **grado**;
  - Se corresponde con la primera fila cuando la relación se representa como tabla.
  - El conjunto A de atributos sobre los que se define la relación se llama **contexto** de la misma.
- **Cuerpo de la relación:** Conjunto de m tuplas,
  - $\{ t_1, t_2, ..., t_m \}$
  - siendo cada tupla un conjunto de n pares atributo-valor:

    $\{ (A_i : V_{ij}) \}$ siendo V_{ij} el valor j del dominio D_i asociado al atributo A_i.
  - el número de tuplas m es la **cardinalidad**.
- Mientras que la cabecera es invariante, el cuerpo varía en el transcurso del tiempo, al igual que la cardinalidad.

- El **esquema de una relación** está constituido por el nombre R -si existe- y la cabecera:

  $$R\ (\{\ A_i : D_i\ \}^{n}_{i=1}\ )$$

  – representa la parte definitoria y estática y también se denomina ***intensión***;

  – se corresponde con lo que hemos llamado *tipo* (de entidad) en el ME/R.

- El **estado r de una relación de esquema R**, al que también denominaremos simplemente **relación**, se representa como **r(R)** y está constituido por el esquema y el cuerpo de la relación:

  r(R) = <esquema, cuerpo>

  – siendo el cuerpo el conjunto de tuplas que, en un instante dado, satisface el correspondiente esquema de relación.

  – también se llama ***extensión***.

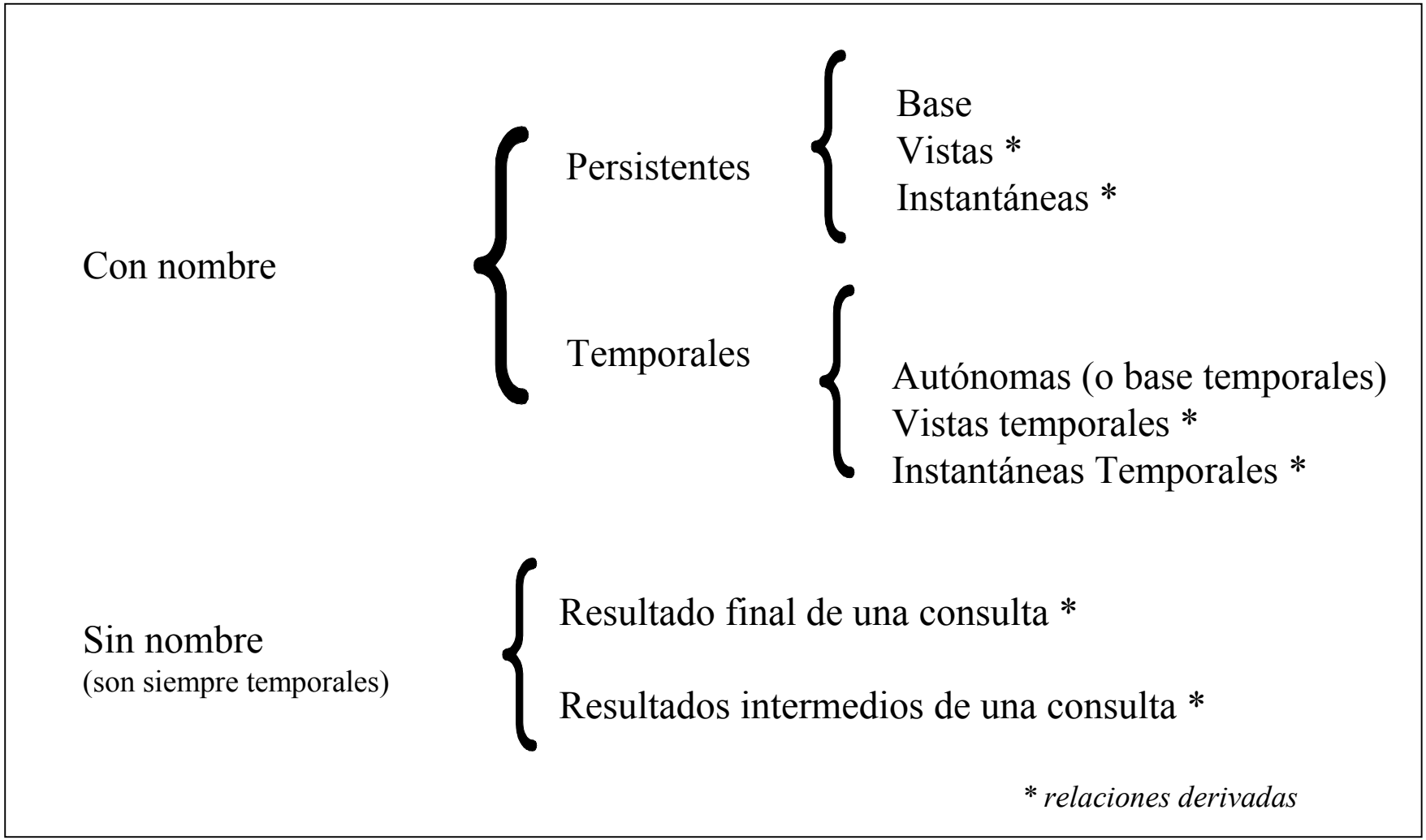
ESQUEMA DE RELACIÓN (INTENSIÓN):

AUTOR (*Nombre***: *Nombres*, ***Nacionalidad***: *Nacionalidades*, *Institución*: *Instituciones*)**

RELACIÓN (EXTENSIÓN, ESTADO* u *OCURRENCIA):

AUTOR

Nombre	*Nacionalidad*	*Institución*
Date, C.J.	**Norteamericana**	**Relational Institute**
De Miguel, A.	**Española**	**UC3M**
Ceri, S.	**Italiana**	**Politécnico de Milan**



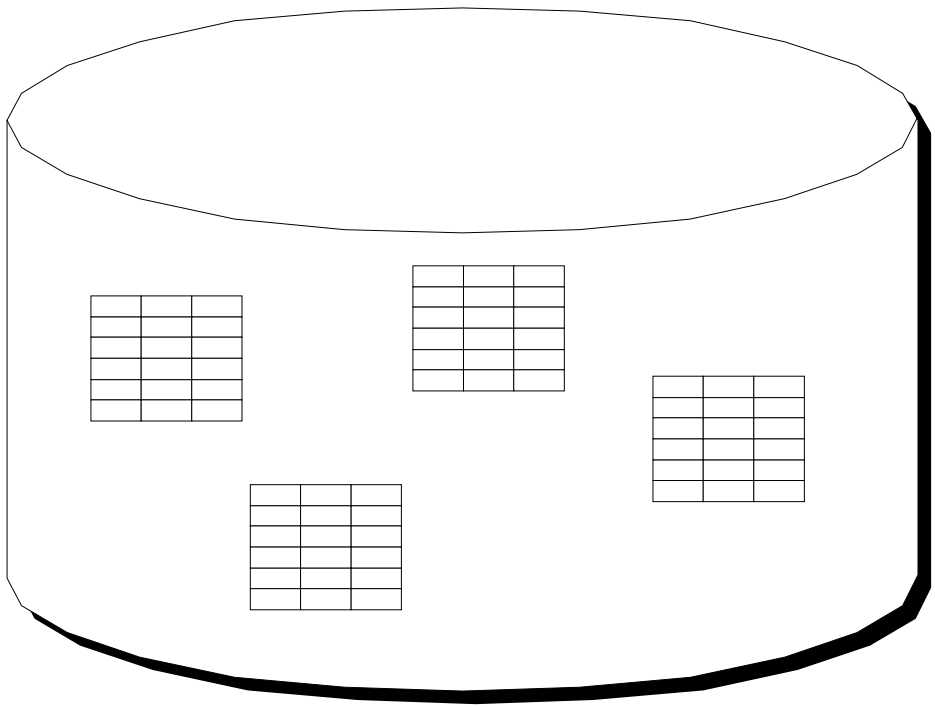
En el MR los datos siempre se manejan en forma de relaciones

- **Persistentes**: su definición (esquema) permanece en la base de datos, borrándose solamente mediante una acción explícita del usuario.
- **Base**: existen por sí mismas, no en función de otras relaciones.
  - Se crean especificando explícitamente su esquema de relación (nombre y conjunto de pares atributo/dominio).
  - Sus extensiones (ocurrencias de relación), al igual que su definición, también se encuentran almacenadas.
  - Se corresponden con el nivel conceptual de la arquitectura ANSI.
- **Vistas** (*View*): son relaciones derivadas que se definen dando un nombre a una expresión de consulta.
  - Se podría decir que son relaciones virtuales (como *ventanas* sobre otras relaciones), en el sentido de que no tienen datos almacenados, sino que lo único que se almacena es su definición en términos de otras relaciones con nombre, las cuales pueden ser relaciones base, otras vistas o instantáneas.
  - Se corresponden con el nivel externo de la arquitectura ANSI.

- **Instantáneas** (*Snapshot*): son relaciones derivadas al igual que las vistas, pero tienen datos propios almacenados, que son el resultado de ejecutar la consulta especificada.
  - También se llaman *vistas materializadas*.
  - Las instantáneas no se actualizan cuando cambian los datos de las relaciones sobre las que están definidas, pero se "refrescan" cada cierto tiempo, de acuerdo con lo indicado por el usuario en el momento de su creación.
  - Son sólo de lectura, no pudiendo ser actualizadas por el usuario, sino únicamente "refrescadas" por el sistema.
- **Temporales**: a diferencia de las persistentes, una relación temporal desaparece de la BD en un cierto momento sin necesidad de una acción de borrado específica del usuario; por ejemplo, al terminar una sesión o una transacción.

- **Clave Candidata** (*Candidate Key*): conjunto de atributos que identifican unívoca y mínimamente cada tupla de la relación.
  - De la propia definición de relación se deriva que siempre existe, al menos, una clave candidata (*al ser una relación un conjunto y no existir dos tuplas iguales, el conjunto de todos los atributos siempre tiene que identificar unívocamente a cada tupla*).
  - La propiedad de minimalidad implica que no se incluye ningún atributo innecesario: *KC cumple la propiedad de minimalidad si no existe un atributo X tal que {KC-X} sea clave candidata*.
- Una relación puede tener más de una clave candidata. En este caso se debe distinguir entre:
  - **Clave Primaria** (*Primary Key*):
    - Es la clave candidata que el usuario escoge para identificar las tuplas de la relación.
    - Cuando sólo existe una clave candidata, ésta es la clave primaria (siempre existe clave primaria).
  - **Claves Alternativas** (*Alternative Key*):
    - Las claves candidatas que no han sido escogidas como clave primaria.

- Una BD relacional es una colección de relaciones (o tablas en términos de implementación);
- Pero ... entonces, ¿Representa la figura una BD relacional?



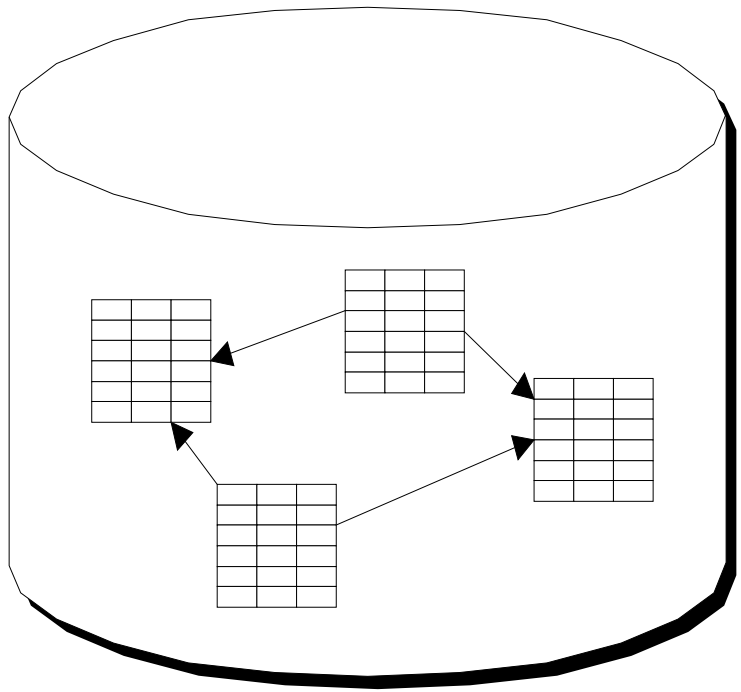
- Respuesta: NO, porqué una BD (relacional o de otro tipo) es una colección de datos **interrelacionados**.

  => Necesitamos asociar unas relaciones con otras.

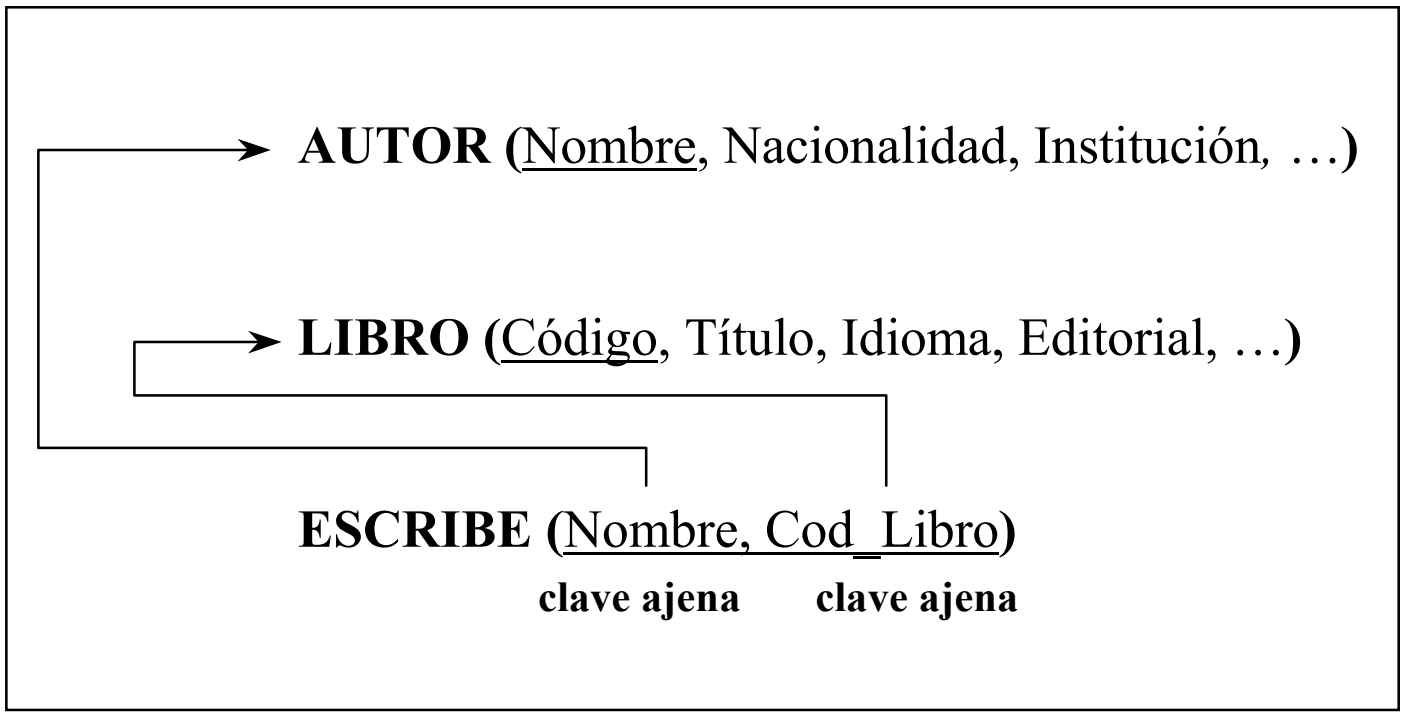
Se denomina **clave ajena** de una relación R2 a un conjunto no vacío de atributos cuyos valores han de coincidir con los valores de la clave candidata de una relación R1.

- R1 y R2 pueden ser la misma relación.

- La clave ajena y la correspondiente clave candidata han de estar definidas sobre el mismo dominio.



Ejemplo de claves primarias y ajenas



Los atributos principales (forman la clave primaria) se subrayan

- Las **restricciones inherentes** vienen impuestas por el propio MD.
- En el caso del MR, una **relación** tiene unas propiedades intrínsecas que no tiene una tabla, y que se derivan de la misma definición matemática de relación, ya que, al ser un **conjunto**, en una relación:
    - No puede haber dos tuplas iguales.
        => obligatoriedad de la clave primaria
    - El orden de las tuplas no es significativo.
    - El orden de los atributos no es significativo.
    - Cada atributo sólo puede tomar un único valor del dominio subyacente; no se admiten grupos repetitivos (ni otro tipo de estructuras) como valores de los atributos de una tupla.
        - Se dice que la relación está **normalizada** (en Primera Forma Norma).
- Existe otra restricción inherente que es la **regla de integridad de entidades**: "*Ningún atributo que forme parte de la clave primaria de una relación puede tomar un valor nulo*"; es decir, un valor desconocido o inexistente.

- No se deben confundir los conceptos de tabla y de relación, puesto que:
  - Una tabla es una forma de representar una relación (una estructura de datos).
  - Una tabla no tiene las restricciones inherentes de una relación -como conjunto- :
    - Puede haber dos filas iguales.
    - Las filas están ordenadas en el orden de grabación física por defecto o según el valor de la clave primaria.
    - Los atributos tienen un orden según se han definido en la tabla.
    - En cada celda de una tabla puede haber uno o varios valores. Si bien en el segundo caso se puede obtener una tabla equivalente que cumple la regla de normalización.

AUTOR1

Nombre	*Nacionalidad*	*Institucion*	*Idiomas*
Date, C.J.	**Norteamericana**	**Relational Institute**	***Inglés, Español***
Codd, E.F.	**Norteamericana**	**Relational Institute**	**Inglés**
Ceri, S.	**Italiana**	**Politécnico de Milan**	***Italiano, Inglés***
De Miguel, A.	**Española**	**UC3M**	**Español**

Transformación de una tabla para normalizarla (1FN)

AUTOR2

Nombre	*Nacionalidad*	*Institucion*	*Idioma*
Date, C.J.	**Norteamericana**	**Relational Institute**	**Inglés**
Date, C.J.	**Norteamericana**	**Relational Institute**	**Español**
Codd, E.F.	**Norteamericana**	**Relational Institute**	**Inglés**
Ceri, S.	**Italiana**	**Politécnico de Milan**	**Italiano**
Ceri, S.	**Italiana**	**Politécnico de Milan**	**Inglés**
Saltor, F.	**Española**	**UC3M**	**Español**

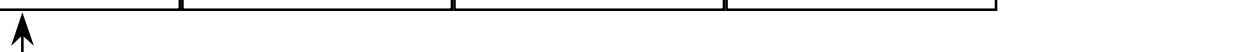
- Son facilidades que el modelo ofrece a los usuarios para que puedan reflejar en el esquema, lo más fielmente posible, la semántica del mundo real.
- Los tipos de restricciones semánticas permitidos en el MR (incorporados a SQL 92) son:
  - **Clave Primaria** (PRIMARY KEY),
  - **Unicidad** (UNIQUE),
  - **Obligatoriedad** (NOT NULL),
  - **Integridad Referencial** (FOREIGN KEY),
  - Restricciones de Rechazo:
    - **Verificación** (CHECK), y
    - **Aserción** (ASSERTION).
  - **Disparador** (*trigger*), incluido en SQL3 pero no en SQL92.
  - **Dependencia** (se estudiará en otro tema).

- **Clave Primaria** (PRIMARY KEY):
  - Permite declarar un atributo o un conjunto de atributos como *clave primaria* de una relación.
    - => sus valores no se podrán repetir ni se admitirán los nulos (o valores "ausentes").
    - Ni el SQL92 ni los SGBD's relacionales obligan a la declaración de una clave primaria para cada tabla (el modelo teórico sí la impone), aunque permiten la definición de la misma.
    - Debemos distinguir entre la restricción inherente de obligatoriedad de la clave primaria y la restricción semántica que le permite al usuario indicar qué atributos forman parte de la clave primaria.
- **Unicidad** (UNIQUE):
  - Los valores de un conjunto de atributos (uno o más) no pueden repetirse en una relación. Esta restricción permite la definición de claves alternativas.
- **Obligatoriedad** (NOT NULL):
  - El conjunto de atributos no admite valores nulos.

- **Integridad Referencial** (FOREING KEY):
  - Si una relación R2 (relación que referencia) tiene un descriptor (subconjunto de atributos) CA que referencia a una clave candidata CC de la relación R1 (relación referenciada), todo valor de dicho descriptor CA debe coincidir con un valor de CC o ser nulo.
    - La condición puede expresarse como R2.CA = R1.CC
    - El descriptor CA es, por tanto, una clave ajena de la relación R2.
    - Las relaciones R1 y R2 no son necesariamente distintas.
    - La clave ajena puede ser también parte (o la totalidad) de la clave primaria de R2.
    - CA puede admitir nulos o tener restricción de obligatoriedad (NOT NULL).
  - Todo atributo de una clave primaria compuesta de una relación R2 que no está definido sobre un dominio compuesto, debe ser clave ajena de R2 referenciando a una relación R1, cuya clave primaria sea simple.

EDITORIAL

Nombre_e	*Dirección*	*País*	*Ciudad*
Universal Books	Brown Sq. 23	EEUU	Los Angeles
Rama	Canillas, 144	España	Madrid
Mc Graw-Hill	Basauri 17	España	Madrid
Paraninfo	Virtudes 7	España	Madrid



LIBRO

Código	*Título*	...	*Editorial*
00345D7	Int. Artificial		Paraninfo
1022305	Concep. y Dis.		Rama
4939H2	Turbo C++		Mc Graw-Hill
0045307	Virus Informát.		Nulo
01123J3	Sist. Informac.		Rama

Ejemplo de Integridad Referencial

```
CREATE TABLE editorial (
        nombre_e  CHAR(20) PRIMARY KEY
        dirección CHAR(50) NOT NULL,
        ciudad CHAR (15),
        país CHAR(15));
```

Ejemplo SQL

Varias restricciones semánticas

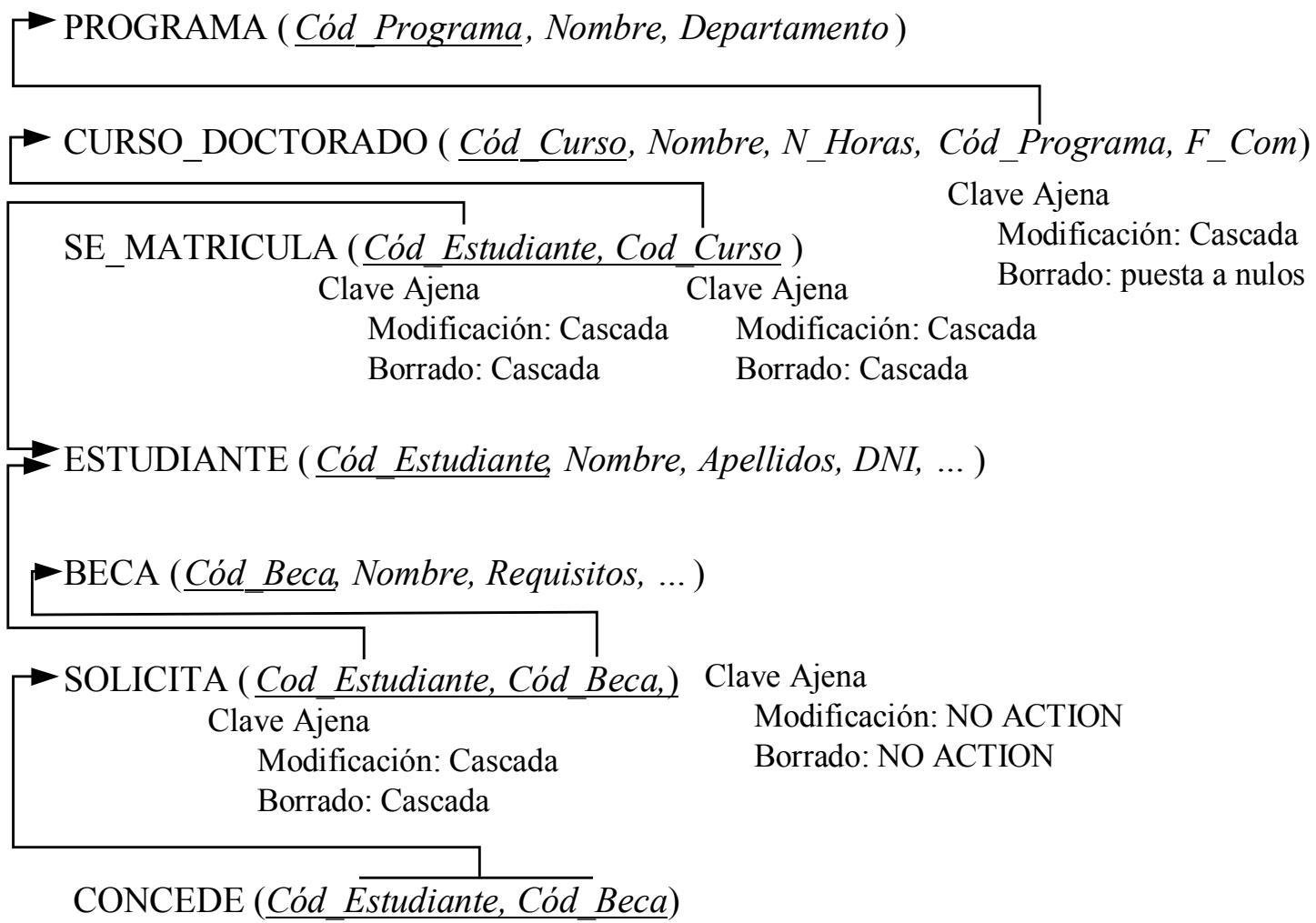
```
CREATE TABLE libro (
        código CHAR(3),
        titulo CHAR (50) UNIQUE,
        idioma CHAR(25),
        nombre_e CHAR(20),

        PRIMARY KEY (codigo),
        FOREING KEY (nombre_e) REFERENCES editorial
                ON DELETE SET NULL
                ON UPDATE CASCADE);
```

- **Integridad Referencial** (FOREING KEY):
  - Además de definir las claves ajenas, hay que determinar las consecuencias que pueden tener ciertas operaciones (**borrado** y **modificación**) realizadas sobre tuplas de la relación referenciada; pudiéndose distinguir, según el estándar SQL92, las siguientes ...

Opciones de Borrado y Modificación en Claves Ajenas

- NO ACTION: rechazar la operación de borrado o modificación.
- CASCADE: propagar la modificación (o borrado) de las tuplas de la tabla que referencia.
- SET NULL: poner valor nulo en la CA de la tabla que referencia.
- SET DEFAULT: poner un valor por defecto en la CA de la tabla que referencia.

NOTA: Ambos modos (de borrado y de modificación) son independientes, es decir, cada uno tomará una de las cuatro opciones por separado.



Ejemplos de claves ajenas con opciones de borrado y modificación

- **Restricciones de Rechazo**: el usuario formula una condición mediante un predicado definido sobre un conjunto de atributos, tuplas o dominios, que debe ser verificado en toda operación de actualización para que el nuevo estado constituya una ocurrencia válida del esquema. En SQL92 existen dos clases:
  - **Verificación** (CHECK): Comprueba, en toda operación de actualización, si el predicado es cierto o falso y, en el segundo caso, rechaza la operación. La restricción de verificación se define sobre un único elemento (dentro de un CREATE TABLE) y puede o no tener nombre.

    CHECK N_HORAS > 30 en CURSO_DOCTORADO

  - **Aserción** (ASSERTION): Actúa de forma idéntica a la anterior, pero se diferencia de ella en que puede afectar a varios elementos (por ejemplo, a dos tablas distintas). Por tanto, su definición no va unida a la de un determinado elemento del esquema y siempre ha de tener un nombre.

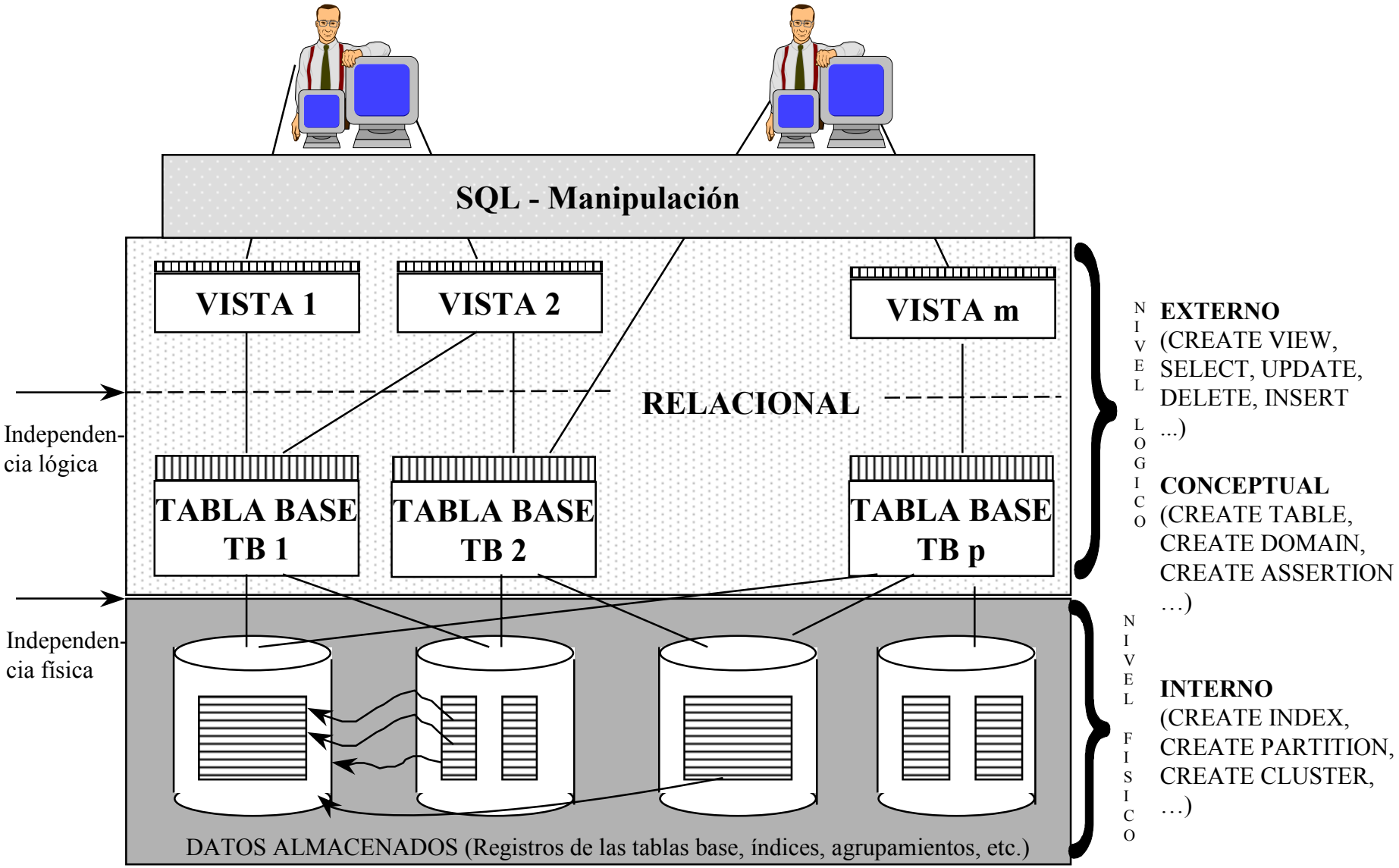
```
CREATE ASSERTION CONCEDE_SOLICITA AS
    CHECK (SELECT Cod_Estudiante, Cod_Beca FROM CONCEDE) IN
        (SELECT Cod_Estudiante, Cod_Beca FROM SOLICITA));
```

- **Disparador** (*trigger*): Restricción en la que el usuario pueda especificar libremente la respuesta (acción) ante una determinada condición.
  - Así como las anteriores reglas de integridad son *declarativas*, los disparadores son *procedimentales*, siendo preciso que el usuario escriba el procedimiento que ha de aplicarse en caso de que se cumpla la condición.
  - Ejemplo: si una beca es solicitada por más de 50 alumnos, se introduce un texto en una tabla de mensajes para que la persona que gestiona las becas considere si es necesario ofrecer más becas;

```
CREATE TRIGGER Comprobar_Matriculados
    AFTER INSERT ON SOLICITA
DECLARE
    NUM_SOLICITUDES Number;
BEGIN
  SELECT COUNT(*) INTO NUM_SOLICITUDES FROM SOLICITA;
  IF NUM_SOLICITUDES > 50 THEN
            INSERT INTO MENSAJES VALUES ('Hay más de 50 solicitudes');
  END IF;
END Comprobar_Matriculados;
```

- Ahora podemos dar una definición más completa de **esquema de una relación**:
  - R<A:D, S>
  - siendo
    - R el nombre de la relación,
    - A la lista de atributos,
    - D los dominios sobre los que están definidos los atributos, y
    - S las restricciones de integridad intraelementos (afectan a atributos y/o tuplas de una única relación).
- Y el **esquema de una base de datos relacional** será:
  - $E < \{R_i\}, \{I_i\} >$
  - siendo
    - E el nombre del esquema relacional,
    - $\{R_i\}$ el conjunto de esquemas de relación, y
    - $\{I_i\}$ el conjunto de restricciones de integridad interelementos (afectan a más de una relación y/o dominio).

- En términos de implementación en SQL92, un esquema E tendrá la siguiente forma:
  - E <R, D, T, V>
  - siendo
    - R el conjunto de esquemas de relación (CREATE TABLE),
    - D el conjunto de definiciones de dominios (CREATE DOMAIN),
    - T el conjunto de restricciones interrelación y sobre dominios (CREATE ASSERTION, CREATE TRIGGER, ...), y
    - V el conjunto de vistas (CREATE VIEW).


SQL - Manipulación
VISTA 1
VISTA 2
VISTA m
RELACIONAL
TABLA BASE TB 1
TABLA BASE TB 2
TABLA BASE TB p
Independencia lógica
Independencia física
DATOS ALMACENADOS (Registros de las tablas base, índices, agrupamientos, etc.)
NIVEL LOGICO
EXTERNO (CREATE VIEW, SELECT, UPDATE, DELETE, INSERT ...)
CONCEPTUAL (CREATE TABLE, CREATE DOMAIN, CREATE ASSERTION …)
NIVEL FISICO
INTERNO (CREATE INDEX, CREATE PARTITION, CREATE CLUSTER, …)

ANSI		RELACIONAL	
LOGICO	Nivel Externo	SQL	Vistas Relaciones Base
	Nivel Conceptual		Relaciones Base
FISICO	Nivel Interno	PRODUCTOS	Datos Almacenados - Relaciones base almacenadas - Indices - Punteros - Direcciones de página - ...

Resumen Relacional vs ANSI

- El MR se adapta a la arquitectura ANSI, pero con las siguientes excepciones importantes:
  - Al usuario se le permite *ver*, si tiene las correspondientes autorizaciones, tanto las relaciones base como las vistas, mientras que en la arquitectura ANSI, para un usuario, la BD está limitada al esquema externo -vistas-.
  - Aunque las vistas se corresponden con los esquemas externos de ANSI y éstos pueden actualizarse, en el MR no todas las vistas son actualizables.
  - Además, en la práctica muchos SGBD relacionales no responden a la arquitectura a tres niveles, ya que las definiciones del esquema conceptual y del esquema interno no están claramente diferenciadas.

- Cuando el MR triunfó comercialmente, muchos fabricantes que tenían productos "antiguos" no relacionales optaron por retocarlos o "camuflarlos" añadiéndoles la etiqueta *relacional*.
- Esto supuso una confusión que Codd intentó arreglar publicando sus 12+1 reglas, que indican las características que debe tener un SGBD para ser auténticamente relacional.
- Regla 0: *Un SGBD relacional debe emplear para gestionar la BD exclusivamente sus facilidades relacionales*.
- De esta regla genérica se derivan las 12 restantes:

  1) **Regla de información**: Toda la información en la Base de datos es representada en una y solo una forma: valores en columnas de filas de tablas.

  2) **Regla de acceso garantizado**: Cada valor escalar individual puede ser direccionado indicando los nombres de la tabla, columna y valor de la clave primaria de la fila correspondiente.

  3) **Tratamiento sistemático de valores nulos**: El SGBD debe soportar la representación y manipulación de información desconocida y/o no aplicable.

4) **Catálogo en línea** (diccionario de datos) basado en el modelo **relacional**.
5) **Sublenguaje de datos completo**: El SGBD debe soportar al menos un lenguaje relacional:
    a) con sintaxis lineal.
    b) que pueda ser usado interactivamente o en programas (embebido).
    c) con soporte para operaciones de:
        - definición de datos (p.e. declaración de vistas).
        - manipulación de datos (p.e. recuperación y modificación de tuplas).
        - restricciones de seguridad e integridad.
        - gestión de transacciones.
6) **Actualización de vistas**: todas las vistas teóricamente actualizables deben poder serlo en la práctica.
7) **Inserción, modificación y borrado de tuplas de alto nivel**: todas las operaciones de manipulación de datos deben operar sobre conjuntos de filas (lenguaje de especificación en vez de navegacional).
8) **Independencia física de los datos**: cambios en los métodos de acceso físico o la forma de almacenamiento no deben afectar al acceso lógico a los datos.
9) **Independencia lógica de los datos**: los programas de aplicación no deben ser afectados por cambios en las tablas que preservan la integridad.

- 10) **Independencia de la integridad**: Las restricciones de integridad deben estar separadas de los programas, almacenadas en el catálogo de la BD para ser editadas mediante un sublenguaje de datos.
- 11) **Independencia de la distribución**: Las aplicaciones no deben verse afectadas al distribuir (dividir entre varias máquinas), o al cambiar la distribución ya existente de la Base de Datos.
- 12) **Regla de no subversión**: Si el sistema posee un interface de bajo nivel (p.e. Mediante llamadas en C), éste no puede subvertir el sistema pudiendo evitar restricciones de seguridad o integridad.

Valor **nulo**

Señal utilizada para representar información desconocida, inaplicable, inexistente, no válida, no proporcionada, indefinida, etc.

Necesidad de los *valores nulos* en BD:

- Crear tuplas (filas) con ciertos **atributos cuyo valor es desconocido** en ese momento, v.g. el año de edición de un libro.
- **Añadir un nuevo atributo** a una relación existente; atributo que, en el momento de añadirse, no tendría ningún valor para las tuplas de la relación.
- **Atributos inaplicables** a ciertas tuplas, por ejemplo, la editorial para un artículo (ya que un artículo no tiene editorial) o la profesión de un menor.

El tratamiento de valores nulos exige redefinir:

- operaciones de comparación
- operaciones aritméticas
- operaciones algebraicas
- funciones de agregación

de forma específica para el caso en que un operando tome valor nulo. También obliga a introducir nuevos operadores especiales

En las operaciones de comparación se hace necesario definir una **lógica trivaluada** incorporando el valor **quizás**.

AND	C	Q	F
C	C	Q	F
Q	Q	Q	F
F	F	F	F

OR	C	Q	F
C	C	C	C
Q	C	Q	Q
F	C	Q	F

NOT	
C	F
Q	Q
F	C

- En la **lógica trivaluada** se utilizan tres valores: C (cierto), F (falso) y Q (quizás).
- Además, se incluyen **nuevos operadores**:
  - **MAYBE** (ES_POSIBLE): devuelve *cierto* si el argumento vale *quizás* y *falso* en otro caso.

MAYBE	
C	F
Q	C
F	F

  - **IS_NULL** (ES_NULO), que toma el valor *cierto* si el operando es *nulo* y *falso* en caso contrario.
- Se redefinen las operaciones aritméticas: se considera *nulo* el resultado de una suma, resta, multiplicación o división si alguno de los operandos toma valor nulo.
- . . . .

- La lógica trivaluada puede causar **problemas** *psicológicos* a los **usuarios**, porque *atenta contra la intuición*. Ejemplos:
  - Si un usuario realiza las consultas
    - libros editados en 1995, y
    - libros editados antes de 1995, y
    - libros editados después de 1995;

    tendría que recuperar todos los libros de la BD; pero esto no es así ya que no habría recuperado los libros cuyo año de edición es nulo (se desconoce).
  - Algunas expresiones siempre ciertas en la realidad (**tautologías**), pueden tomar valor nulo en la lógica trivaluada:
    - X = X, siendo X una variable
    - p OR NOT (p), siendo p una expresión condicional
- Algunos autores proponen como solución el uso de **valores por defecto**, pero esta opción, ya utilizada en lenguajes de programación "viejos" es bastante criticada.

- Otra solución propuesta es el uso de una **lógica cuatrivaluada**, distinguiendo entre un valor "A" que representa información desconocida pero aplicable y otro "I" que representa información inaplicable:
  - número de teléfono desconocido => A
  - profesión de un niño => I
  - las operaciones aritméticas quedan modificadas:
    - X & A = A & X = A X & I = I & X = I
    - A & I = I & A = I A & A = A
    - I & I = I siendo & un operador aritmético y X un valor no nulo de la BD.
  - Pero es muy <u>difícil de manejar</u>. Ni siquiera existe acuerdo entre los autores sobre las tablas de verdad de los operadores lógicos. Las propuestas por Codd son:

AND	C	A	F	I
C	C	A	F	I
A	A	A	F	I
F	F	F	F	F
I	I	I	F	I

OR	C	A	F	I
C	C	C	C	C
A	C	A	A	A
F	C	A	F	F
I	C	A	F	I

NOT	
C	F
A	A
F	C
I	I