Teoría Base de Datos

Ignacio Rimini

August 2025

Índice

1. Unidad 1 - Introducción.

1.1. Objetivos de los Sistemas de Bases de Datos.

1.1.1. Sistemas de Gestión de Bases de Datos.

Un **Sistema de Gestión de Bases de Datos (DBMS)** se compone de una colección de datos interrelacionados y de un conjunto de programas diseñados para acceder y manipular esos datos. Su objetivo principal es ofrecer un entorno conveniente y eficiente para el almacenamiento y la recuperación de información.

Los DBMS están pensados para administrar grandes volúmenes de datos y cumplen funciones esenciales como:

- Definición de estructuras para el almacenamiento de información.
- Provisión de mecanismos para la gestión de la información.
- Mantenimiento de la seguridad de la información almacenada (caídas del sistema, accesos no autorizados).
- Control de concurrencia para evitar resultados anómalos cuando múltiples usuarios acceden simultáneamente a los mismos datos.

Un problema frecuente que resuelven los DBMS es la **redundancia e inconsistena de datos,** ya que en sistemas tradicionales los archivos y programas de aplicación suelen ser desarrollados en distintos momentos por distintos programadores, lo que genera duplicaciones y formatos incompatibles.

1.1.2. Sistemas de Procesamiento de Archivos.

Previo al enfoque de bases de datos, era común trabajar con sistemas de procesamiento de archivos. Estos presentaban varias limitaciones que dificultaban el manejo de la información:

- **Dificultad de acceso a los datos:** si una consulta no había sido prevista en el diseño original, no existía un programa que pudiera resolverla. La única opción era extraer manualmente la información desde informes existentes o desarrollar un nuevo programa, lo que resultaba costoso e ineficiente.
- **Aislamiento de los datos:** al estar distribuidos en archivos separados con diferentes formatos, era complejo desarrollar nuevas aplicaciones que integraran la información.
- **Problemas de concurrencia:** múltiples programas podían acceder simultáneamente a los mismos datos sin coordinación, lo que generaba anomalías y dificultaba la supervisión.
- **Deficiencias en seguridad:** los programas de aplicación se incorporaban de manera puntual, dificultando la imposición de restricciones que aseguraran que cada usuario tuviero acceso solo a los datos autorizados.

Estas limitaciones motivaron el desarrollo de sistemas más robustos y centralizados, como los DBMS.

1.1.3. Ventajas del enfoque de bases de datos.

El enfoque basado en bases de datos, gestionados por un DBA (Administrador de Bases de Datos), ofrece múltiples beneficios en comparación con el procesamiento de archivos:

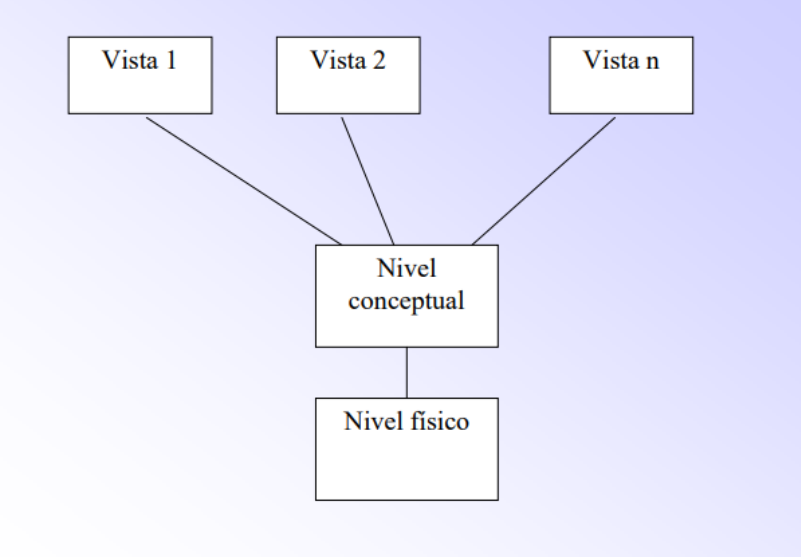
- **Reducción de la redundancia y la inconsistencia:** al centralizar los datos se evita duplicación y discrepancias.
- **Facilidad para compartir datos:** aplicaciones nuevas o existentes pueden acceder a la misma base de datos, fomentando la integración.

- **Cumplimiento de normas:** el DBA puede definir estándares para la representación de datos que se aplican de manera uniforme.
- **Seguridad centralizada:** se garantiza que el acceso sea solo a través de canales autorizados y con los controles adecuados.
- **Mantenimiento de la integridad:** el DBA puede establecer restricciones que aseguren la coherencia de los datos en cada operación de actualización.

1.2. Abstracción de datos: niveles externo, conceptual e interno.

La **abstracción de datos** permite separar la forma en que los datos se almacenan físicamente de la forma en que los perciben los usuarios y administradores. Para lograrlo, se distinguen tres niveles principales:

- **Nivel interno (o físico):** describe cómo se almacenan los datos en la base, detallando las estructuras de datos y las técnicas utilziadas para organizarlos de manera eficiente. Ejemplo: modelo relacional (tablas).
- **Nivel conceptual:** utilizado por los administradores de bases de datos, define qué datos se almacenan realmente y qué relaciones existen entre ellos, sin entrar en detalles de implementación física. Ejemplo: modelo entidad-relación (diagramas).
- **Nivel externo (o de visión):** corresponde a la perspectiva de los usuarios finales. Describe únicamente una parte de la base de datos completa, mostrando solo la información que resulta relevante para cada aplicación o usuario.



1.3. Modelos de datos.

Los **modelos de datos** proporcionan un marco conceptual para describir la organización de la información en una base de datos. Permiten representar qué datos se almacenan, cómo se relacionan entre sí y cómo se perciben desde distintos niveles de abstracción.

Existen dos grandes enfoques de modelos lógicos:

1. **Modelos basados en objetos:** se utilizan para describir los datos en los niveles **conceptual y externo (de visión).** Se centran en representar entidades, atributos y relaciones de manera cercana a la realidad.

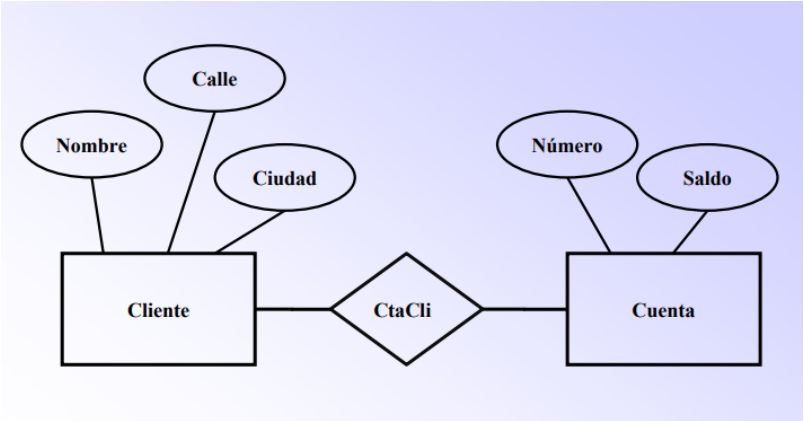
2. **Modelos basados en registros:** se emplean para describir datos tanto en el nivel **conceptual** como en el nivel **físico.** Organizan la información en estructuras de registros con campos de longitud fija o variable.

1.3.1. El modelo Entidad-Relación (E-R).

El **modelo E-R** parte de la idea de que el mundo real está compuesto por **entidades** y por **relaciones** entre ellas.

- **Entidad:** objeto distinguible de los demás gracias a un conjunto de atributos. Ejemplo: una cuenta bancaria puede describirse por medio de atributos como número y saldo.
- **Relación:** asociación que conecta dos o más entidades. Ejemplo: la relación `CtaCli` asocia a un cliente con cada una de sus cuentas.

Este modelo suele representarse mediante **diagramas E-R,** en los que las **entidades aparecen como rectángulos, las relaciones como rombos y los atributos como ovalos.**



1.3.2. El modelo relacional.

El **modelo relacional** organiza los datos y las relaciones entre ellos mediante un conjunto de **tablas.** Cada tabla (o relación) consta de filas y columnas:

- Las **filas** representan tuplas o registros.
- Las **columnas** representan atributos con nombres únicos.

Por ejemplo, una tabla de clientes puede contener los atributos `Nombre, Calle, Ciudad, Número` con varias filas que representan a diferentes personas.

Nombre	Calle	Ciudad	Número
Lowery	Maple	Queens	900
Shiver	North	Bronx	556
Shiver	North	Bronx	647
Hodges	Sidehill	Brooklyn	801
Hodges	Sidehill	Brooklyn	647

1.3.3. Instancias y esquemas.

En una base de datos se distinguen dos conceptos fundamentales:

- **Instancia (snapshot):** corresponde al conjunto de información efectivamente almacenada en un momento determinada. Es dinámica y cambia con el tiempo.
- **Esquema:** es el diseño global de la base de datos. Es más estable y define la estructura general de cómo se organizan los datos.

Los sistemas de bases de datos suelen manejar varios esquemas:

- **Esquema físico:** nivel más bajo, describe cómo se almacenan los datos.
- **Esquema conceptual:** nivel intermedio, define qué datos existen y cómo se relacionan.
- **Subesquemas (o vistas):** nivel más alto, muestran solo partes de la base de datos relevantes para ciertos usuarios o aplicaciones.

1.3.4. Independencia de datos.

Una de las ventajas clave de los sistemas de bases de datos es la **independencia de datos,** es decir, la capacidad de modificar un esquema en un nivel sin afectar los niveles superiores.

Se distinguen dos tipos:

- **Independencia física de datos:** posibilidad de modificar el esquema físico (cómo se almacenan los datos) sin necesidad de reescribir los programas de aplicación.
- **Independencia lógica de datos:** posibilidad de modificar el esquema conceptual (por ejemplo, añadir un camapo nuevo) sin alterar los programas de aplicación que acceden a la base.

1.4. Lenguaje de definición y manipulación de datos.

Los sistemas de bases de datos cuentan con lenguajes especializados para **definir** la estructura de la base y para **manipular** la información que contiene. Estos lenguajes garantizan que los usuarios y administradores puedan trabajar con los datos de manera organizada, eficiente y consistente.

1.4.1. Lenguaje de definición de datos (DDL).

El **Lenguaje de Definicion de Datos (DDL)** se utiliza para establecer el **esquema de la base de datos,** es decir, su estructura global.

Un componente central del DDL es el **diccionario de datos** (también llamado catálogo o directorio). Este es un archivo que almacena un conjunto de tablas y contiene **metadatos,** es decir, información acerca de la propia base de datos (como nombres de tablas, atributos, tipos de datos y relaciones). Antes de acceder o modificar los datos reales, el sistema consulta este diccionario para garantizar la coherencia.

Algunas palabras de este lenguaje son: `CREATE TABLE, DROP TABLE,` etc.

1.4.2. Lenguaje de manipulación de datos (DML).

El **Lenguaje de Manipulación de Datos (DML)** permite a los usuarios acceder y modificar los datos almacenados en la base de datos. Sus principales operaciones son:

- **Recuperación** de información.
- **Inserción** de nuevos datos.
- **Supresión** de datos existentes.

- **Modificación** de datos ya almacenados.

Existen dos tipos de DML:

- **Procedimentales:** el usuario debe especificar tanto qué datos necesita como la manera de obtenerlos.
- **No procedimentales:** el usuario indica qué datos desea, pero no cómo obtenerlos; el sistema se encarga de resolver la consulta.

Dentro de este contexto, una **consulta** es una sentencia que solicita la recuperación de información de la base de datos. El lenguaje de consultas (como SQL) es el ejemplo más representativo de DML no procedimental. Ejemplo: `SELECT, INSERT, DELETE,` etc.

1.5. Adminitración de bases de datos.

La administración de bases de datos comprende tanto los componentes de software encargados de gestionar la información como las funciones humanas de quienes diseñan, mantienen y utilizan el sistema.

1.5.1. Gestor de bases de datos (DBMS).

El **gestor de bases de datos (DBMS)** (o motor de base de datos) es un módulo de software que actúa como intermediario entre los datos almacenados en bajo nivel y los programas de aplicación o consultas realizadas por los usuarios. Sus responsabilidades incluyen:

- **Definición y manipulación de datos:** traduce sentencias DDL y DML a comandos de bajo nivel sobre el sistema de archivos, asegurando el almacenamiento, recuperación y actualización de la información.
- **Implantación de la integridad:** controla que los valores almacenados cumplan las restricciones de consistencia definidas por el DBA (por ejemplo, evitar saldos negativos en cuentas bancarias).
- **Implantación de la seguridad:** garantiza que los requisitos de acceso y protección de datos, establecidos por el DBA, se cumplan correctamente.
- **Copia de seguridad y recuperación:** detecta fallos (como cortes de energía o daños en disco) y restaura la base de datos al estado anterior al error.
- **Control de concurrencia:** coordina las operaciones cuando varios usuarios acceden simultáneamente a la base de datos, evitando inconsistencias.

1.5.2. Administrador de bases de datos (DBA).

El **administrador de bases de datos (DBA)** es la persona responsable de la gestión técnica y operativa del sistema, implementando las políticas definidas por el administrador de datos (DA). Entre sus funciones principales se encuentran:

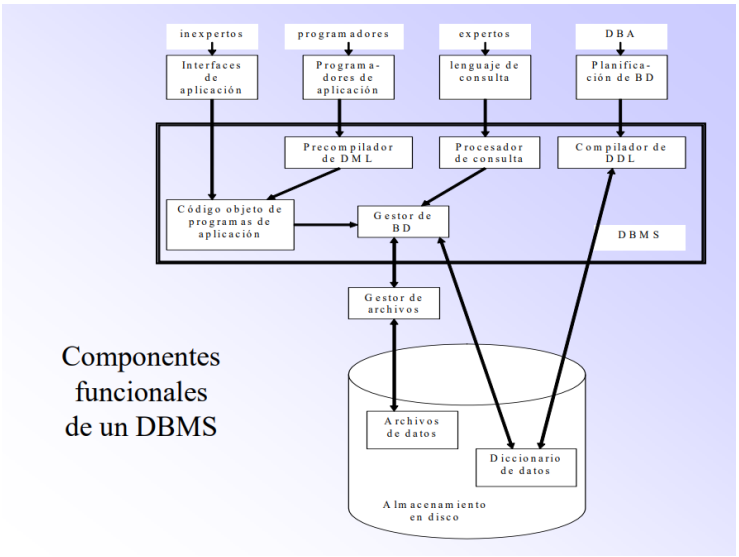
- **Definición del esquema conceptual:** creación del diseño lógico dela base de datos mediante sentencias DDL, que luego son traducidas por el DBMS en tablas.
- **Definición del esquema interno:** determinación de la estructura de almacenamiento y métodos de acceso (diseño físico), como la asignación en disco o el uso de índices.
- **Modificación del esquema y ajustes físicos:** supervisión del rendimiento del sistema y realización de cambios según nuevas necesidades.
- **Gestión de autorizaciones:** concesión de permisos de acceso para controlar qué usuarios pueden visualizar o modificar distintas partes de la base de datos.

- **Definición de restricciones de integridad:** establecimiento de reglas que garanticen la consistencia de los datos (por ejemplo, impedir que se registre un saldo negativo).
- **Definición de procedimientos de respaldo y recuperación:** planificación y supervisión de mecanismos que protejan la información frente a pérdidas o fallos.

1.5.3. Usuarios de bases de datos.

Los usuarios de una base de datos interactúan con el sistema en diferentes niveles, según sus conocimientos y necesidades:

1. **Programadores de aplicaciones:** integran sentencias DML dentro de programas escritos en un lenguaje principal. Un precompilador se encarga de traducir estas sentencias en llamadas normales a procedimientos del lenguaje.
2. **Usuarios de consutas:** emplean lenuajes de consulta para recuperar la información directamente de la base de datos, sin necesidad de programar.
3. **Usuarios finales de aplicaciones:** acceden al sistema a través de programas ya desarrollados, interactuan con interfaces diseñadas para tareas específicas.



2. Unidad 2 - El modelo Entidad-Relación.

2.1. Modelado de datos con el modelo Entidad-Relación.

El **Modelo Entidad-Relación (E-R)** es una técnica de modelado de datos propuesta por Peter Chen en 1976. Permite representar de manera gráfica y conceptual los objetos del mundo real (entidades), sus propiedades (atributos) y las asociaciones entre ellos (relaciones). Es ampliamente utilizado en la etapa de **diseño conceptual de bases de datos.**

2.1.1. Entidades.

Una **entidad** es un objeto que existe y es distinguible de otros. Puede ser:

- **Concreta:** persona, empleado, casa, automóvil.
- **Abstracta:** cuenta bancaria, empresa, curso.

Cada entidad se describe mediante un **conjunto de atributos.**

2.1.2. Atributos.

Los **atributos** son propiedades que describen una entidad. Ejemplo: una entidad `Persona` puede tener atributos como `nombre, edad, dirección.`

El **dominio** de un atributo es el conjunto de valores permitidos para él. Formalmente, un atributo puede considerarse como una función que asigna a cada entidad un valor en su dominio.

Una entidad puede representarse por un conjunto de pares (atributo, valor). Por ejemplo, `empleado` se puede describir mediante el conjunto:

```
empleado = {(DNI, 67.789.901), (nombre, López), (calle, Mayor), (ciudad,
                                Rosario)}
```

2.1.3. Tipos de atributos.

1. **Simples y compuestos.**
   - Simples: indivisibles (ej: `nombre, calle`).
   - Compuestos: pueden subdividirse (ej: `nombre-persona` → `nombre, primer-apellido, segundo-apellido`).
2. **Monovalorados y multivalorados.**
   - Monovalorados: poseen un único valor (ej: `fecha-nacimiento`).
   - Multivalorados: permiten múltiples valores (ej: `número-telefono` de un empleado).
3. **Derivados.** Su valor se calcula a partir de otros atributos ya existentes en la entidad. Por ejemplo, el atributo **edad** se puede derivar a partir de la fecha de nacimiento.

   El valor de estos atributos no se almacena, sino que se calcula cuando es necesario.
4. **Valores nulos.** Un atributo toma un valor **nulo** cuando una entidad no tiene un valor para ese atributo.

Ejemplo de entidades y atributos.

- **Sucursal:** { `nombre-sucursal, ciudad-sucursal, activo`}. El conjunto de todas las sucursales de un banco determinado.
- **Cliente:** {`nombre-cliente, seguridad-social, calle, ciudad-cliente`}. El conjunto de todas las personas que tienen una cuenta en el banco.

- **Empleado:** {`nombre-empleado, numero-telefono`}. El conjunto de todas las personas que trabajan en el banco.
- **Cuenta:** {`numero-cuenta, saldo`}. El conjunto de todas las cuentas que mantiene el banco.
- **Transacción:** {`numero-transaccion, fecha, cantidad.`}. El conjunto de todas las transacciones realizadas en cuentas del banco.

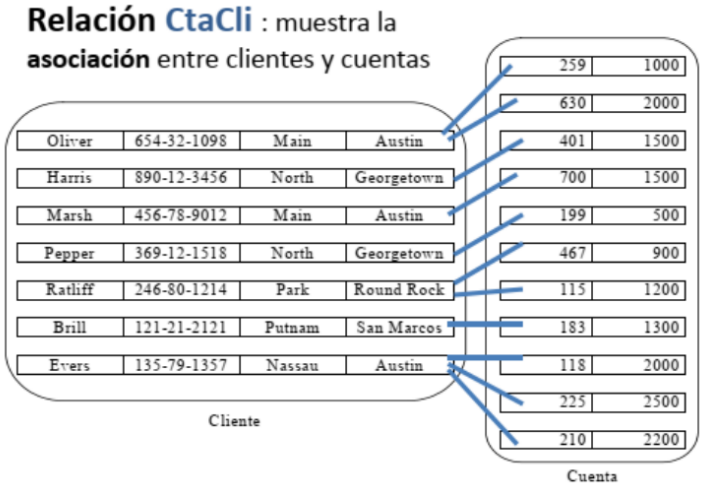
2.1.4. Relaciones.

Una **relación** es una asociación entre dos o más entidades.

Sean $E_1, E_2, ...E_n$ entidades distintas. Formalmente, una relación de grado n (o conjunto de relaciones R) es un subconjunto de:

$$\{(e_1, e_2, ...e_n) \mid e_1 \in E_1, e_2 \in E_2, ...e_n \in E_n\}$$

donde $(e_1, e_2, ...e_n)$ es una instancia de la relación.



Observaciones.

- Las relaciones **binarias** involucran 2 entidades (grado 2).
- Puede haber relaciones de una entidad en la misma entidad. Por ejemplo `trabaja-para` podría modelarse por pares ordenados de entidades `Empleo`, donde el primero es el jefe y el segundo es el subordinado.
- Una relación puede tener atributos descriptivos. Por ejemplo `fecha_prestamo` de un libro a un lector o `fecha` en `CtaCli`, que especifica la última fecha en la que un cliente tuvo acceso a su cuenta.

¿Atributo o entidad?

En algunos casos, un dato puede modelarse como un atributo o como una entidad.

- Caso 1: `Empleado (nombre-empleado, numero-telefono)`.
- Caso 2: `Empleado (nombre-empleado), Telefono (numero-telefono, oficina), Relación EmpTel (Empleado ↔ Telefono)`

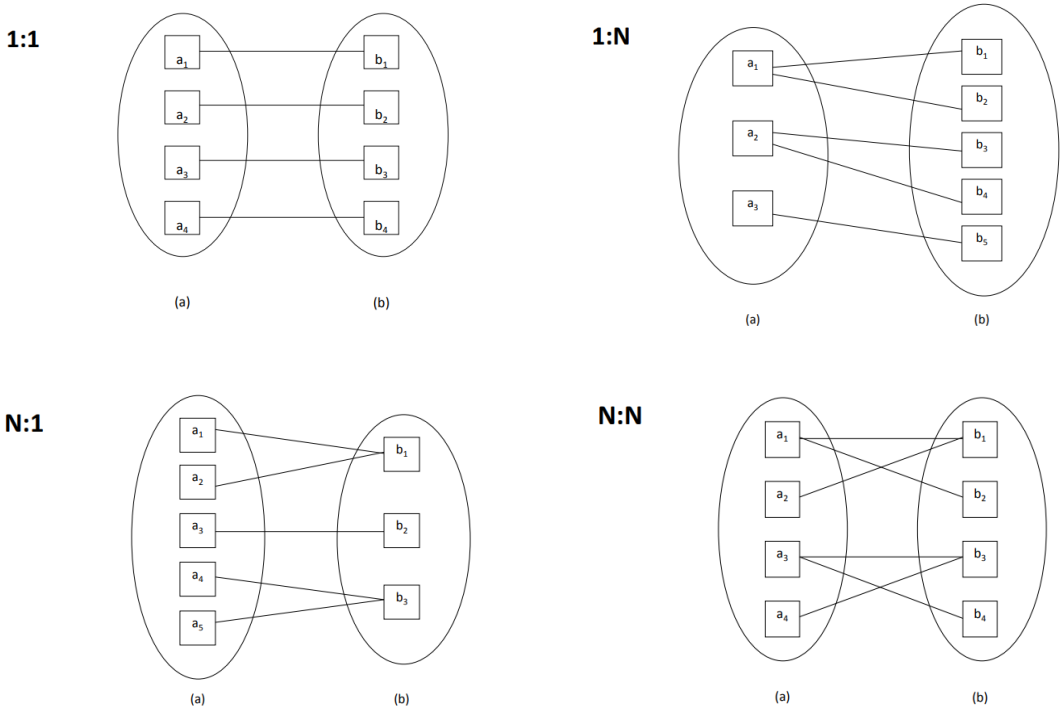
En el primer caso cada empleado tiene exactamente un número de teléfono. Mientras que en el segundo, los empleados pueden tener varios números de teléfono o que varios empleados compartan un número. La decisión depende de la estructura del dominio y de la semántica del atributo.

2.1.5. Restricciones de asignación (mapping).

La **cardinalidad de asignación** expresa el número de entidades con las que se puede asociar otra entidad a través de un conjunto de relaciones.

En una relación binaria entre las entidades `A` y `B`, la cardinalidad debe ser una de las siguientes:

- **1:1**: un registro de la entidad `A` se asocia con un solo registro de la entidad `B`, y viceversa.
- **1:N**: un registro de la entidad `A` se asocia con varios registros de la entidad `B`.
- **N:1**: varios registros de la entidad `A` se asocian con un mismo registro de la entidad `B`.
- **N:N**: varios registros de la entidad `A` se asocian con varios registros de la entidad `B`.



La **cardinalidad** depende del mundo real que se está modelando. Por ejemplo, para la relación `CtaCli`:

- Si una cuenta pertenece únicamente a un cliente, y un cliente puede tener varias cuentas → **1:N**.
- Si una cuenta puede pertenecer a varios clientes, y un cliente puede tener varias cuentas → **N:N**.

2.1.6. Dependencias de existencia.

Las **dependencias de existencia** son restricciones que establecen que la presencia de una entidad en la base de datos depende de la existencia previa de otra. Es decir, ciertos registros no pueden crearse, mantenerse o tener sentido sin estar asociados a una entidad *padre* o principal.

Formalmente, si la existencia de la entidad `X` (entidad subordinada) depende de la existencia de la entidad **Y** (entidad dominante), entonces se dice que `X` **es dependiente por existencia de Y.** Luego, si se suprime la entidad `Y`, también se suprime `X`.

Ejemplo: la entidad `Cliente (id-cliente, nombre, direccion)` es independiente, y la entidad `Pedido (id-pedido, fecha, id-cliente)` es dependiente de la entidad `Cliente`, pues ningún pedido puede existir sin que antes exista el cliente que lo realizó.

2.2. Claves.

En el modelo Entidad-Relación (E-R), las **claves** son un concepto fundamental para garantizar la **identificación única** de entidades y relaciones. Su correcta definición asegura la consistencia de los datos, evita duplicidades y permite establece dependencias entre distintos elementos del modelo.

2.2.1. Superclaves.

Una **superclave** es un conjunto de uno o más atributos capaces de **identificar de manera única** cada entidad dentro de un conjunto.

Por ejemplo, para la entidad `Cliente (nombre-cliente, seguridad-social, calle, ciudad-cliente)`, los siguientes conjuntos son superclaves:

{`nombre-cliente, seguridad-social`}, {`seguridad-social`}

Además, si un conjunto K de atributos es superclave, entonces cualquier superconjunto de K también lo será.

2.2.2. Claves candidatas.

Una **clave candidata** es una **superclave mínima,** es decir, un conjunto de atributos que identifica de forma única a la entidad, pero tal que ningún subconjunto propio de esos atributos puede seguir cumpliendo esta función.

Una entidad puede tener una o varias claves candidatas. Por ejemplo, si `seguridad-social` ya identifica de manera única a un cliente, no se requiere **nombre-cliente** para la clave.

2.2.3. Clave primaria.

La **clave primaria** es aquella clave candidata que es seleccionada por el diseñador de la base de datos para ser utilizada de forma oficial dentro del modelo de datos.

La **clave primaria** es la que representa a la entidad en relaciones y asociaciones. Debe ser **única, estable, irreductible y no nula.**

2.2.4. Entidades fuertes y débiles.

- Una **entidad fuerte** posee al menos una clave candidata que la identifica sin necesidad de depender de otra entidad.
- Una **entidad débil** no cuenta con atributos suficientes para formar una clave candidata por sí sola y requiere de una entidad fuerte para poder identificarse.

Ejemplo: la entidad `Transaccion (numero-transaccion, fecha, cantidad)` es una entidad débil, pues si dos cuentas distintas tienen el mismo número de transacción, este atributo no basta para identificar de manera única la transacción.

Discriminador en entidades débiles.

El **discriminador** es el conjunto de atributos que distingue a las entidades débiles asociadas a una entidad fuerte específica.

En el ejemplo anterior: fijada la entidad fuerte `Cuenta`, el atributo **numero-transaccion** actúa como discriminador de la entidad **Transaccion**. Por lo tanto, la **clave primaria de una entidad débil** se construye como:

Clave primaria de la entidad fuerte + Discriminador de la entidad débil

En el ejemplo anterior, el conjunto {`numero-cuenta, numero-transaccion`} identifica de forma única a cada transacción.

2.2.5. Claves en relaciones.

Sean **R** una **relación** que involucra a las entidades $E_1, E_2, ...E_n$, y sea (E_i) la **clave primaria** de la entidad E_i.

Definimos el valor **atributo(R)** de la siguiente manera:

- Si **R** no tiene atributos, definimos **atributo(R)** como la unión de las claves de las entidades que relaciona:

$$\textbf{atributo(R)} = (E_1) \cup (E_2) \cup \cdots \cup (E_n)$$

- Si **R** tiene atributos descriptivos $\{a_1, a_2, ...a_m\}$, entonces definimos **atributo(R)** como la unión de las claves de las entidades que relaciona, unión sus propios atributos:

$$\textbf{atributo(R)} = (E_1) \cup (E_2) \cup \cdots \cup (E_n) \cup \{a_1, a_2, ...a_m\}$$

Por ejemplo, sea la relación `CtaCli` con un solo atributo descriptivo propio `fecha` y relaciona las entidades `Cliente` con clave primaria `seguridad-social` y la entidad `Cuenta` con clave primaria `número-cuenta` entonces el valor **atributo(R)** resulta:

$$\textbf{atributo(CtaCli)} = \{\texttt{seguridad\_social, numero\_cuenta, fecha}\}$$

Y para determinar la clave de la relación seguimos las siguientes instrucciones.

- Si **R** no tiene atributos, el conjunto **atributo(R)** (que en este caso será la unión de las claves primarias de las entidades que relaciona) forma una **superclave.**
  - Si **R** no tiene atributos y la cardinalidad es **N:N,** esta superclave es la **clave primaria.**
  - Si **R** no tiene atributos y la cardinalidad es **N:1** o **1:N**, entonces su **clave primaria** será un subconjunto de esta superclave, en particular, la clave primaria de la entidad de cardinal 1.

  Ejemplo: si `CtaCli` es **N:N** y no tiene atributos, entonces el conjunto **atributo(R)** = {`seguridad_social, numero_cuenta`} es la clave primaria.

  Si en cambio `CtaCli` es **N:1**, es decir, una persona puede tener una sola cuenta asociada pero una cuenta puede estar a nombre de varias personas, y además la relación no tiene atributos, entonces la clave primaria será {`seguridad_social`} ya que con este campo solo podemos determinar el número de cuenta.

- Si **R** tiene atributos asociados, entonces una superclave de la relación está formada igual que antes pero con el posible agregado de uno o más de sus propios atributos de relación.

  Ejemplo. Sea la relación `BanqueroCli` que relaciona las entidades `Cliente` y `Banquero`, y tiene el atributo asociado `tipo`, con valores `prestamista` o `banquero personal`. Entonces:

  1. Si un banquero puede representar dos papeles distintos (puede ser `prestamista` o `banquero personal`) en una relación con un cliente, entonces la clave primaria de la relación `BanqueroCli` será:

     `clave-primaria(cliente)` ∪ `clave-primaria(banquero)` ∪ `tipo`

  2. Si en cambio un banquero puede tener un solo tipo de papel con un cliente, entonces la clave primaria de la relación `BanqueroCli` será:

     `clave-primaria(cliente)` ∪ `clave-primaria(banquero)`

Es decir, si el atributo `tipo` queda determinado por uno de los dos elementos de la clave, entonces **no forma parte de la clave.**

2.3. Diagrama Entidad-Relación.

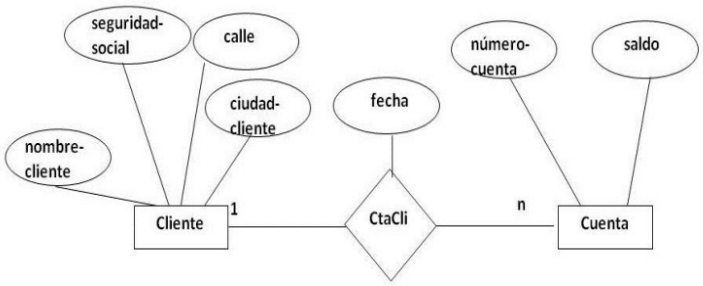
El **diagrama entidad-relación (E-R)** es una herramienta gráfica utilizada para representar los elementos fundamentales de una base de datos y las relaciones entre ellos. Consta de los siguientes componentes básicos:

- **Rectángulos:** representan conjuntos de entidades.
- **Elipses:** representan atributos de las entidades.
- **Rombos:** representan las relaciones entre conjuntos de entidades.
- **Líneas:** conectan atributos con conjuntos de entidades y conjuntos de entidades con relaciones.
- **Cardinalidad:** se nota la cardinalidad con 1 o n junto a la entidad correspondiente, cuando hay una relación entre conjuntos de entidades.

Cada componente debe ser etiquetado con el nombre de la entidad, atributo o relación que representa.

Ejemplo. En un sistema bancario, se podrían identificar:

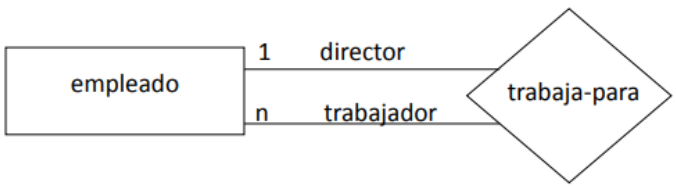
- **Entidades:** clientes y cuentas.
- **Relaciones:** un cliente posee una o varias cuentas.



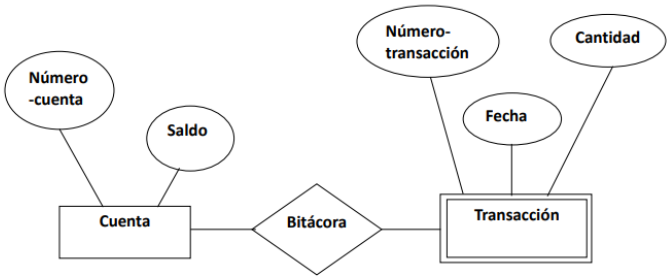
Observaciones.

- **Relaciones recursivas.** Cuando una entidad se relaciona consigo misma, se especifican los *papeles* de las entidades en la relación.

  Por ejemplo, un `Empleado` *trabaja-para* otro `Empleado`. En este caso, se distingue entre los roles: Director (1), Trabajador (n).



- **Entidades débiles.** Una entidad débil se representa con un **rectángulo de doble contorno.** Estas entidades dependen de una entidad fuerte para existir.



2.4. Reducción de diagramas Entidad-Relación a tablas.

2.4.1. Representación en tablas de entidades fuertes.

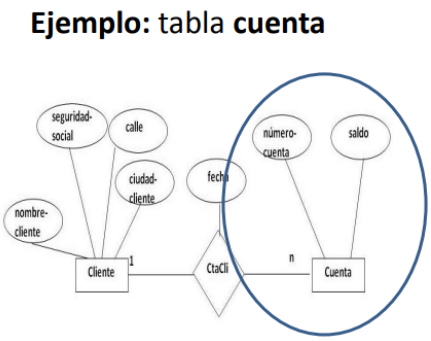
Sea E una **entidad fuerte** con atributos descriptivos $a_1, a_2, ...a_n$, entonces:

- Representamos esta entidad por medio de una **tabla** llamada E con n columnas.
- Cada **columna** corresponde a un atributo de E.
- Cada **fila** corresponde a una entidad.

Ejemplo. Tabla cuenta.

Sean D_1 el conjunto de todos los **números de cuenta** y D_2 el conjunto de todos los **saldos.**

- Cualquier fila de la tabla `Cuenta` consiste en una tupla binaria (v_1, v_2) con $v_1 \in D_1$ y $v_2 \in D_2$.
- El conjunto de todas las filas posibles de `Cuenta` es el producto cartesiano $D_1 \times D_2$.
- La tabla `Cuenta` contendrá por filas un subconjunto de $D_1 \times D_2$.



Número-cuenta	Saldo
259	1000
630	2000
401	1500
700	1500
199	500
467	900
115	1200
183	1300
118	2000
225	2500
210	2200

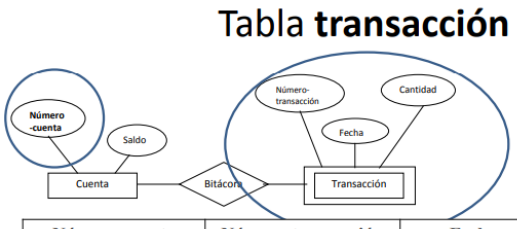
2.4.2. Representación en tablas de entidades débiles.

- Sea A una **entidad débil** con atributos descriptivos $a_1, a_2, ...a_r$.
- Sea B la **entidad fuerte** de la cual depende A con clave primaria $\{b_1, b_2, ...b_s\}$.

Entonces se representa la entidad A por medio de una tabla llamada A con columnas:

$$\{b_1, b_2, ...b_s\} \cup \{a_1, a_2, ...a_r\}$$

Es decir, las entidades débiles tienen por columnas las claves primarias de la entidad fuerte de la cual dependen, y columnas para sus propios atributos.



Número-cuenta	Número-transacción	Fecha	Cantidad
259	5	11 mayo 1990	+ 50
630	11	17 mayo 1990	+ 70
401	22	23 mayo 1990	- 300
700	69	28 mayo 1990	- 500
199	103	3 junio 1990	+ 900
259	6	7 junio 1990	- 44
115	53	7 junio 1990	+ 120
259	7	17 junio 1990	- 79

2.4.3. Representación de relaciones.

Resumen general:

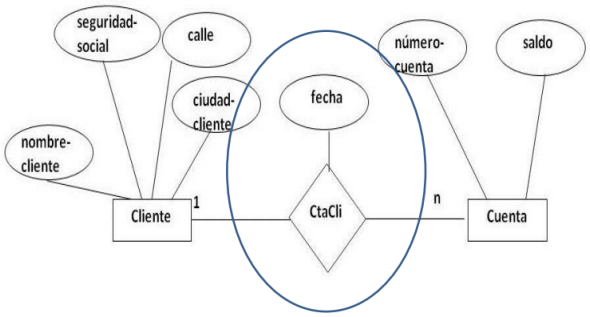
- Relaciones **N:N** se necesita crear una tabla intermedia.
- Relaciones **1:N** entre entidades fuertes: la entidad del lado N recibe una clave foránea que referencia a la entidad del lado 1 (si la relación no tiene atributos). Si la relación tiene atributos se debe crear una tabla intermedia.
- Relaciones **1:1** se coloca la clave de una entidad como clave foránea en la tabla de la otra entidad.

Explicación de los apuntes:

- **Relación entre entidades fuertes.** Sea **R** una relación que involucra a las entidades $E_1, E_2, ...E_m$. Supongamos que **atributo(R)** consta de n atributos. Entonces, representamos esta relación mediante una **tabla** llamada **R** con n **columnas distintas,** donde cada columna corresponde a un atributo de **atributo(R).**

  Ejemplo. Tabla `CtaCli`.

  La tabla de la relación `CtaCli` tendrá por columnas las claves primarias de las entidades que relaciona más su propio atributo, es decir, columnas de R: {`seguridad-social, numero-cuenta, fecha`}.



- **Relación entre entidades fuertes y débiles.** Las relaciones que conectan una entidad fuerte con una débil son un caso especial:
  - Son relaciones **muchas a una.**
  - **No tienen atributos descriptivos.**
  - La clave primaria de la entidad débil incluye la **clave primaria de la entidad fuerte** de la cual depende.

  Por esto, la tabla de una relación de este tipo resulta ser una **tabla redundante** y no necesita crearse.

  Ejemplo. Tabla `Bitácora`.

  Tenemos las entidades `Cuenta` como entidad fuerte con clave primaria `número-cuenta` y `Transacción` que es entidad débil con clave primaria {`numero-cuenta, numero-transaccion`}.

  Como la relación `Bitácora` no tiene atributos descriptivos, tendrá solo 2 columnas, equivalentes a las claves primarias de las entidades que relaciona: `numero-cuenta, numero-transaccion.`

  Pero la tabla de la entidad débil `Transacción` tiene 4 columnas: `numero-cuenta, numero-transaccion, fecha, cantidad`. Es decir, si existiese la tabla para la relación `Bitácora`, habría dos columnas con los mismos datos repetidos que la tabla `Transacción.`

2.5. Extensiones al Modelo Entidad-Relación: Generalización y Especialización.

En el **Modelo Entidad-Relación (DER)** existen extensiones que permiten representar de manera más precisa ciertas situaciones. Entre ellas se destacan la **generalización** y la **especialización,** que permiten organizar entidades en jerarquías.

2.5.1. Concepto de generalización y especialización.

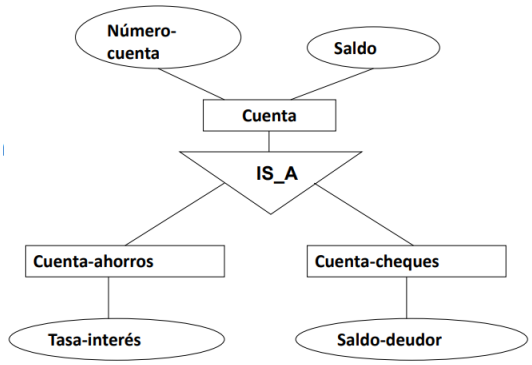
- **Generalización:** es una relación de inclusión en la que una **entidad de nivel más alto (superentidad)** agrupa a una o más **entidades de nivel más bajo.**
- **Especialización:** es la relación inversa, en la que una entidad general se divide en subtipos más específicos.

En el DER, estas relaciones se representan mediante un **triángulo etiqueta con 'IS_A',** que indica la pertenencia de una subentidad a su superentidad.

Ejemplo. Supongamos que la entidad `Cuenta` se clasifica en dos tipos:

1. **Cuenta de ahorros** (con atributo `tasa de interés`).
2. **Cuenta de cheques** (con atributo `saldo deudor`).

Ambas subentidades heredan los atributos comunes de `Cuenta` (número de cuenta, saldo), y además poseen atributos propios.



2.5.2. Representación en tablas de subentidades y superentidades.

La representación de generalización/especialización en una base de datos relacional puede realizarse de dos formas principales:

1. **Método 1: Una tabla para la superentidad y una tabla para cada subentidad.**
   - Se crea una tabla para la entidad del nivel más alto.
   - Se crea una tabla separada para cada subentidad que incluye: una columna por cada atributo propio y una columna por la clave primaria de la entidad superior.

   En nuestro ejemplo de `Cuenta` resultarían las siguientes tablas: `Cuenta (número-cuenta, saldo)`, `Cuenta-ahorros (número-cuenta, tasa-interés)`, `Cuenta-cheques (número-cuenta, saldo-deudor)`.

   Este método garantiza consistencia de datos, pero requiere realizar uniones (*joins*) para consultar información completa de las cuentas.

2. **Método 2: Solo tablas para las subentidades.**
   - No se crea una tabla para la entidad general.
   - Se crea una tabla para cada subentidad que incluye: sus atributos propios y los atributos heredados de la entidad de nivel superior.

   En nuestro ejemplo de `Cuenta` resultarían las siguientes tablas: `Cuenta-ahorros (número-cuenta, saldo, tasa-interés)`, `Cuenta-cheques (número-cuenta, saldo, saldo-deudor)`.

   Este método evita la necesidad de *joins,* pero puede generar redundancia de datos, ya que los atributos comunes (como `saldo`) se repiten en cada subentidad.

2.6. Resumen: Mapa canónico.

Habiendo descripto como se pasa a tablas cada componente de un diagrama entidad-relación, se presenta a continuación un resumen de reglas a seguir. En bases de datos, el **mapa canónico** es el proceso sistemático de transformar el Diagrama Entidad–Relación (DER) en un esquema relacional.

Es decir: cómo traducir cada entidad, relación, atributo y restricción de cardinalidad en tablas. Se llama *canónico* porque hay reglas estándar que siempre se aplican de la misma forma:

- Entidad fuerte → tabla con su clave primaria.

- Entidad débil → tabla con la clave primaria de la entidad fuerte + su discriminante como clave primaria compuesta.
- Relación 1:N → la entidad del lado N recibe una clave foránea que referencia a la entidad del lado 1 (si la relación no tiene atributos). Si la relación tiene atributos se debe crear una tabla intermedia.
- Relación N:N → se crea una tabla intermedia con las claves primarias de ambas entidades como clave primaria compuesta (y posibles atributos propios de la relación).
- Relación 1:1 → se implementa con una clave foránea en cualquiera de las dos entidades (generalmente en la que depende).
- Atributos multivalorados → se crea una tabla aparte con:
  1. Clave primaria de la entidad original (como clave foránea).
  2. El atributo multivalorado (que pasa a ser atributo simple en la nueva tabla).
  3. (Opcional) Un identificador si se necesitan varios registros únicos.
- Atributos compuestos → se descomponen en atributos simples.
- Relaciones entre más de dos entidades →
  - Se crean tablas intermedias que incluyen las claves primarias de todas las entidades participantes como clave primaria compuesta.
  - Si la relación tiene atributos propios, se agregan en la tabla intermedia.
  - La cardinalidad **1** significa que por cada ocurrencia de la relación, la entidad participa con una sola tupla. En la práctica, esto permite que algunas claves foráneas no formen parte de la clave primaria, dependiendo del caso.

Ese conjunto de reglas es lo que llamamos mapa canónico: un puente único y directo entre DER ↔ Modelo Relacional.

3. Unidad 3 - El modelo Relacional.

3.1. Modelos de datos.

- **Modelos basados en objetos.**
  - Se utilizan para describir datos a **nivel conceptual.**
  - El ejemplo más importante es el **Modelo Entidad-Relación (MER),** una técnica de diseño de bases de datos que permite representar entidades, atributos y relaciones.
- **Modelos basados en registros.**
  - Se emplean para describir datos a **nivel físico.**
  - El ejemplo más representativo es el **Modelo Relacional,** que constituye una formalización teórica de las bases de datos relacionales (BDR).

3.2. Estructuras de las Bases de Datos Relacionales (BDR).

Una **BDR** es una **colección de tablas,** cada una de las cuales posee un nombre único. A través de ellas se organizan los datos y se definen sus relaciones.

3.2.1. Tablas = Relación matemática.

En el modelo relacional, una tabla se denomina **relación.** Cada **fila (tupla)** representa una relación entre un conjunto de valores.

Ejemplo. En la siguiente relación `Depósito`, la fila `(Downtown, 101, Johnson, 500)` indica que el cliente `Johnson` tiene una cuenta número `101` en la sucursal `Downtown` con saldo `500`.

nombre-sucursal	número-cuenta	nombre-cliente	saldo
Downtown	101	Johnsohn	500
Mianus	215	Smith	700
Perryridge	102	Hayes	400
Round Hill	305	Turner	350

3.2.2. Atributo.

- Cada columna de una tabla representa un **atributo.**
- Para cada atributo hay un conjunto de valores permitidos, llamado **dominio** de ese atributo.

Ejemplo. Para la tabla del ejemplo anterior, tenemos que los atributos son `nombre-sucursal, numero-cuenta, nombre-cliente, saldo`, donde para el atributo `nombre-sucursal`, el dominio es el conjunto de todos los nombres de sucursales existentes.

3.2.3. Tablas/Relación formalizado.

Formalmente, sean:

- $D_1 = \{$nombres de sucursales$\}$ (el dominio del atributo `nombre-sucursal`.
- $D_2 = \{$números de cuenta$\}$ (el dominio del atributo `numero-cuenta`.
- $D_3 = \{$nombres de clientes$\}$ (el dominio del atributo `nombre-cliente`.

- $D_4 = \{\text{saldos}\}$ (el dominio del atributo `saldo`.

Cada fila de `Depósito` debe constar de 4-tuplas de la forma:

$$(v_1, v_2, v_3, v_4)$$

donde $v_1 \in D_1, v_2 \in D_2, v_3 \in D_3, v_4 \in D_4$.

En general, `Depósito` contendrá unicamente un subconjunto del conjunto de todas las filas posibles. Es decir, `Depósito` $\subset D_1 \times D_2 \times D_3 \times D_4$.

Matemáticamente, una relación es un subconjunto de un producto cartesiano de una lista de n dominios. Esto se corresponde con la definición de tabla. Por lo tanto, usaremos los términos **tabla = relación** y **fila = tupla.**

Notación.

- Sea t una tupla variable de la relación `Depósito`, entonces `t[nombre-sucursal]` indica el valor de t en el atributo `nombre-sucursal`.
- La notación $t \in r$ indica que la tupla t está en (satisface) la relación r (la r minúscula = relación/tabla).

3.2.4. Esquemas e instancias.

- **Esquema:** es el diseño lógico de la base de datos, definido por la lista de atributos.
- **Instancia:** son los datos almacenadaos en un momento específico (snapshot).

Ejemplo de notación. `esquema-depósito = (nombre-sucursal, número-cuenta, nombre-cliente, saldo)`.

3.2.5. Claves.

- **Superclave:** conjunto de uno o más atributos que identifican de forma única a una tupla. Todo superconjunto de una superclave también es superclave.
- **Clave candidata:** superclave mínima (ningún subconjunto propio también es superclave).
- **Clave primaria:** clave candidata elegida por el diseñador de la base de datos.

Formalmente, sean R un esquema de relación y r una relación sobre R (denotado como $r(R)$). Un subconjunto K de R es superclave si:

$$\forall t_1, t_2 \in r / t_1 \neq t_2 \Rightarrow t_1[K] \neq t_2[K]$$

3.2.6. Propiedades de las relaciones.

- **No existen tuplas repetidas:** los conjuntos no incluyen elementos repetidos.
- **Las tuplas no están ordenadas:** no se puede hablar de la 5ta tupla.
- **Los atributos no están ordenados:** aunque en la tabla se muestran de izquierda a derecha, en el modelo relacional los atributos son un conjunto, no una secuencia.
- **Valores atómicos:** cada posición de fila y columna contiene un único valor, no listas ni estructuras complejas.

Si una relación cumple con estas propiedades, se considera que está **normalizada.**