

# ***Introducción a las Bases de Datos.***

*Máster en Business Intelligence e  
Innovación Tecnológica*

## Índice de contenidos

1. ¿Qué es una Base de Datos?
2. Evolución histórica.
3. Concepto del SGDB.
4. Objetivos de un SGBD.
5. Modelos de SGBD
6. Principales Bases de Datos
7. Introducción al modelo relacional
8. Estructura y fundamentos del modelo relacional
9. Álgebra relacional
10. Cálculo relacional



*Se le llama **base de datos** a los bancos de información que contienen datos relativos a diversas temáticas y categorizados de distinta manera, pero que comparten entre sí algún tipo de vínculo o relación que busca ordenarlos y clasificarlos en conjunto.*

*Wikipedia*

El término bases de datos fue escuchado por primera vez en un simposio celebrado en California en 1963.

En una primera aproximación, se puede decir que una base de datos es un conjunto de información relacionada que se encuentra agrupada o estructurada.

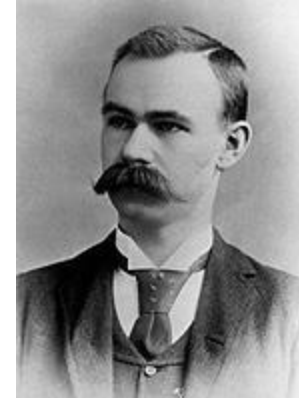


## Antigüedad



**1884**

Herman Hollerith creó la máquina automática de tarjetas perforadas. En esa época, los censos se realizaban de forma manual...Hollerith comenzó a trabajar en el diseño de una maquina tabuladora, basada en tarjetas perforadas.



## Década 50

- Aparición de las cintas magnéticas para automatizar la información y hacer respaldos.
- Sirvió para suplir las necesidades de información de las nuevas industrias.
- A través de este mecanismo se empezó a automatizar información.
- Desventaja: sólo se podía acceder de forma secuencial.





## **Década 60**

- Aparición de los primeros discos: proporcionan un acceso directo a la información, liberando de las limitaciones del acceso secuencial.
- Aparición de los primeros sistemas:
  - Bases de datos de red.
  - Bases de datos jerárquicas.
- Almacenamiento en estructuras de datos complejas: listas, árboles,...
- Explotación lógica de los datos mediante programas.

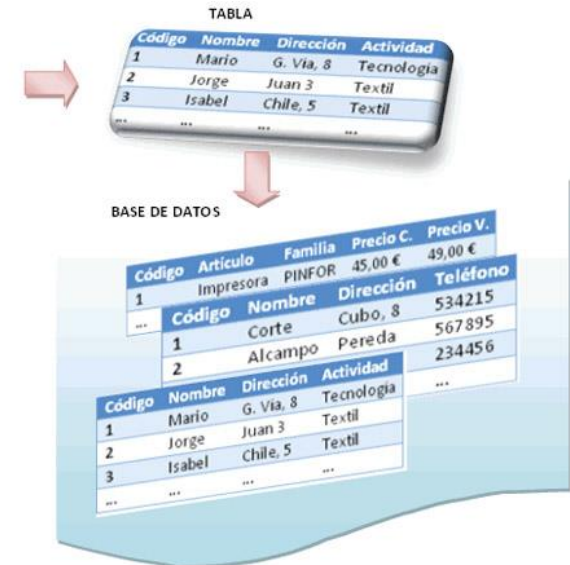
Alianza de IBM y American Airlines para desarrollar SABRE, un sistema operativo que manejaba las reservas de vuelos.

Aparece CODASYL (Conference on Data Systems Languages): un consorcio de industrias informáticas para la regularización de un lenguaje de programación estándar.



## Década 70

- E.F. Codd (IBM, 1970) definió el modelo relacional y formas no procedimentales de consultar los datos en el modelo relacional...
  - Los registros se relacionaban mediante punteros físicos. La modificación, inserción o movimiento de la información podía conllevar corrupción en los datos.
  - El modelo relacional proponía una relación lógica en tablas (columnas y filas)
- El modelo Relacional únicamente mantiene interés académico...no podía competir en rendimiento con los otros sistemas.
- IBM desarrolla System R.
- Larry Ellison), desarrolla Relational Software System...actualmente se conoce como Oracle Corporation.



## **Década 80**

- La evolución en el hardware posibilita un rendimiento adecuado.
- Se desarrolla el SQL (Structured Query Language), un lenguaje de consultas o lenguaje declarativo de acceso a bases de datos relacionales.
- El modelo Relacional se convierte en el estándar de la industria.
- Inicio de otras investigaciones, como las *Sistemas de Gestión de Bases de Datos Orientadas a Objetos* .

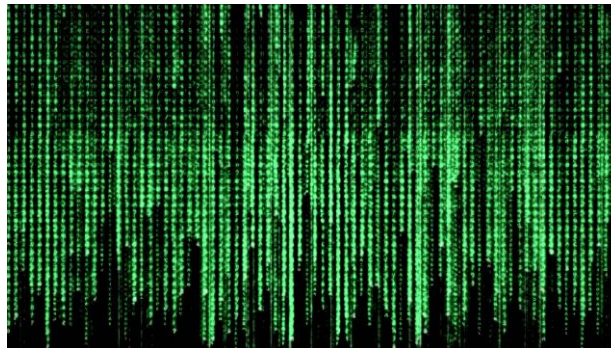


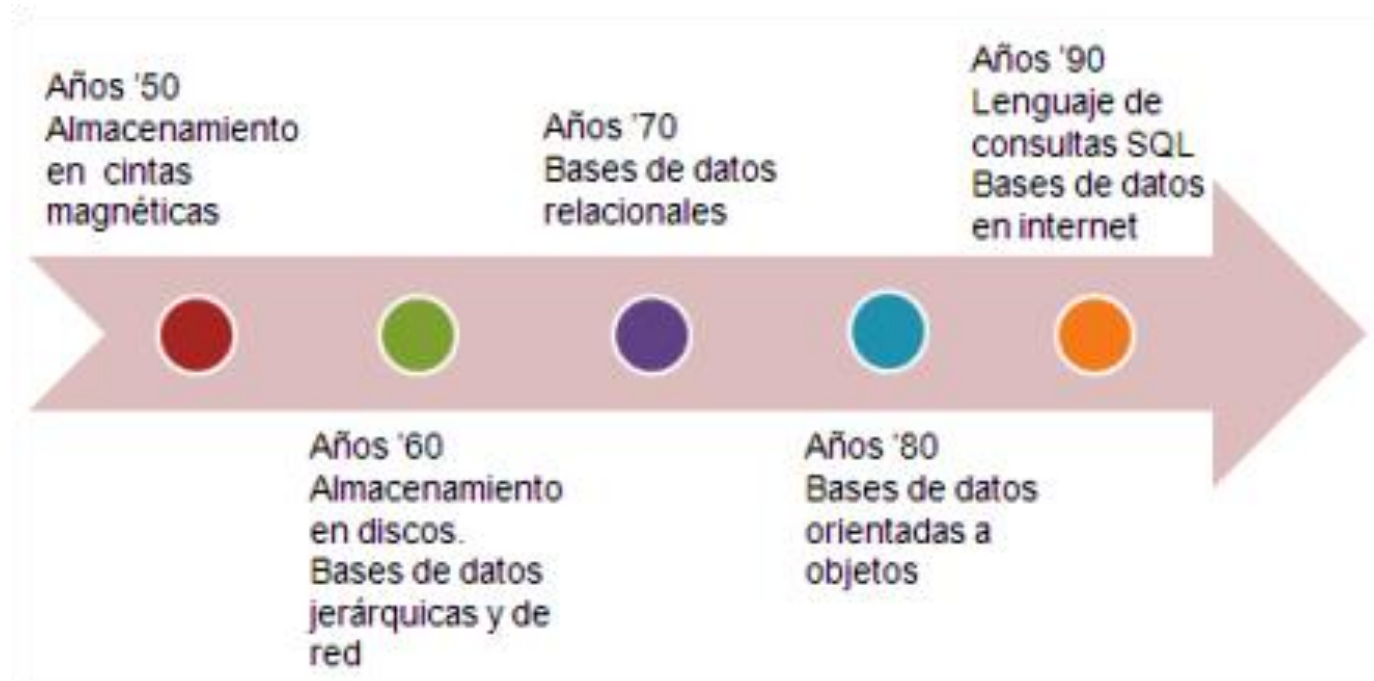
## **Década 90**

- SQL perfeccionado tras varias evoluciones.
- Aparición en el mercado de SGBDOO y sistemas paralelos.
- La explosión World Wide Web provoca:
  - Las bases de datos se implantaron mucho más extensivamente que nunca antes.
  - Tasas de transaccionalidad muy altas, muy alta fiabilidad y disponibilidad 24 × 7.

## Actualidad

- Sistemas Cloud
- Estandarización de Business Intelligence.
- Empresas dentro de un marco mucho más competitivo y cambio de paradigma: orientación al cliente:
  - Accesibilidad. Cambio de concepto...de la transacción a la interacción.
  - Múltiples generadores de información (IoT, smartphones,...).
- Nuevas metodologías (NOSQL,...).





[Ver evolución histórica](#)



*Un sistema gestor de bases de datos (SGBD) consiste en una colección de datos interrelacionados y un conjunto de programas para acceder a dichos datos. La colección de datos, normalmente denominada base de datos, contiene información relevante para una empresa. El objetivo principal de un SGBD es proporcionar una forma de almacenar y recuperar la información de una base de datos de manera que sea tanto práctica como eficiente.*

*Silberschatz, A. ; Korthz, H.F. ; Sudarshan, S*

- 1. Consultas no predefinidas y complejas**
- 2. Flexibilidad e independencia**
- 3. Problemas de la redundancia**
- 4. Integridad de los datos**
- 5. Concurrencia de usuarios**
- 6. Seguridad**

## 1. Consultas no predefinidas y complejas

- El objetivo fundamental de los SGBD es permitir que se hagan consultas no predefinidas (ad hoc) y complejas.
- Los usuarios podrán hacer consultas de cualquier tipo y complejidad directamente al SGBD. El SGBD tendrá que responder inmediatamente sin que estas consultas estén preestablecidas; es decir, sin que se tenga que escribir, compilar y ejecutar un programa específico para cada consulta.
- El usuario debe formular la consulta con un lenguaje sencillo (que se quede, obviamente, en el nivel lógico), que el sistema debe interpretar directamente. Sin embargo, esto no significa que no se puedan escribir programas con consultas incorporadas (por ejemplo, para procesos repetitivos).

## 2. Flexibilidad e independencia

- La complejidad de las BD y la necesidad de ir las adaptando a la evolución del SI hacen que un objetivo básico de los SGBD sea dar flexibilidad a los cambios. Interesa obtener la máxima independencia posible entre los datos y los procesos usuarios para que se pueda llevar a cabo todo tipo de cambios tecnológicos.
- **Independencia física de los datos:** No necesitar saber nada sobre el soporte físico, ni estar al corriente de qué SO se utiliza, qué índices hay, la compresión o no compresión de datos, etc.
- **Independencia lógica de los datos:** No tengan que hacer cambios cuando se modifica la descripción lógica o el esquema de la BD (por ejemplo, cuando se añaden/suprimen entidades o interrelaciones, atributos, etc. Que diferentes procesos puedan tener diferentes visiones lógicas de una misma BD, y que estas visiones se puedan mantener lo más independientes posibles de la BD, y entre ellas mismas.

### 3. Problemas de la redundancia

- La redundancia de datos, en sus inicios, suponía en primera instancia un problema económico: el precio del *byte*.
- En segunda instancia la redundancia conlleva un problema derivado: la existencia de datos redundantes puede provocar un problema de integridad al producirse actualizaciones de los mismos.
- El SGBD debe **permitir** que el diseñador defina datos redundantes, pero entonces tendría que ser el mismo SGBD el que hiciese automáticamente la **actualización** de los datos en todos los lugares donde estuviesen repetidos.



## 4. Integridad de los datos

- Nos interesará que los SGBD aseguren **el mantenimiento de la calidad** de los datos en cualquier circunstancia.
- Cuando el SGBD detecte que un programa quiere hacer una operación que va contra las reglas establecidas al definir la BD, no se lo deberá permitir, y le tendrá que devolver un estado de error.
- Al diseñar una BD para un SI concreto y escribir su esquema, no sólo definiremos los datos, sino también las reglas de integridad que queremos que el SGBD haga cumplir.
- Ante una incidencia o pérdida de integridad, los procesos de restauración (restore o recovery) de los que todo SGBD dispone pueden reconstruir la BD y darle el estado consistente y correcto anterior al incidente.

## 5. Concurrencia de usuarios

- Un objetivo fundamental de los SGBD es permitir que varios usuarios puedan acceder concurrentemente a la misma BD.
- Denominamos **transacción de BD** o, simplemente, transacción un conjunto de operaciones simples que se ejecutan como una unidad. Los SGBD deben conseguir que el conjunto de operaciones de una transacción nunca se ejecute parcialmente. O se ejecutan todas, o no se ejecuta ninguna.
- Nos interesará que el SGBD ejecute las transacciones de forma que no se interfieran; es decir, que queden aisladas unas de otras. Para conseguir que las transacciones se ejecuten como si estuviesen aisladas, los SGBD utilizan distintas técnicas. La más conocida es el **bloqueo** (lock).

## 6. Seguridad

En el campo de los SGBD, el término seguridad se suele utilizar para hacer referencia a los temas relativos a la confidencialidad, las autorizaciones, los derechos de acceso, etc.

- Los SGBD permiten definir autorizaciones o derechos de acceso a diferentes niveles: al nivel global de toda la BD, al nivel entidad y al nivel atributo.
- Nos puede interesar almacenar la información con una codificación secreta; es decir, con **técnicas de encriptación** (como mínimo se deberían encriptar las contraseñas). Muchos de los SGBD actuales tienen prevista la encriptación.

En la actualidad, estos términos pueden verse requeridos por conceptos legales...LOPD, LSSI,...

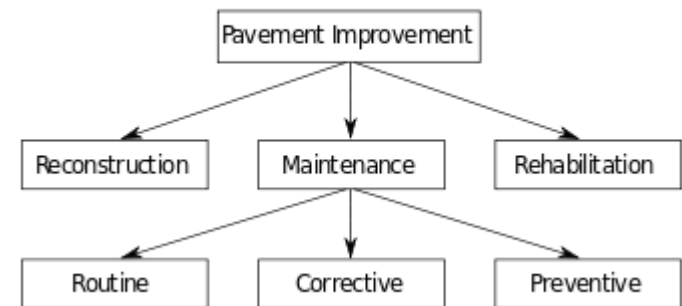
## Modelo Jerárquico

Usada en los primeros mainframe, las relaciones entre los registros forman una estructura de árbol.

Un *nodo padre* de información puede tener varios *hijos*. El nodo que no tiene padres es llamado *raíz*, y a los nodos que no tienen hijos se los conoce como *hojas*.

Actualmente las bases de datos jerárquicas más utilizadas son IMS de IBM, el Registro de Windows de Microsoft, el contenido de ficheros XML y bases de datos geográficas.

Hierarchical Model



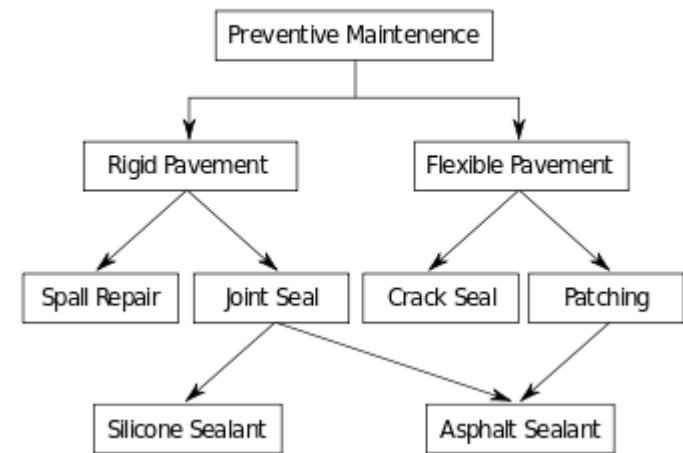
## Modelo de red

Éste es un modelo ligeramente distinto del jerárquico; su diferencia fundamental es la modificación del concepto de *nodo*: se permite que un mismo nodo tenga varios padres.

Fue una gran mejora con respecto al modelo jerárquico: ofrecía una solución eficiente al problema de redundancia de datos. Aún así, la dificultad que significa administrar la información en una base de datos de red ha significado que sea un modelo utilizado en su mayoría por programadores más que por usuarios finales.

Entre los SGBD más populares que tienen arquitectura en red se encuentran Total e IDMS. IDMS logró una importante base de usuarios; en 1980 adoptó el modelo relacional y SQL

Network Model





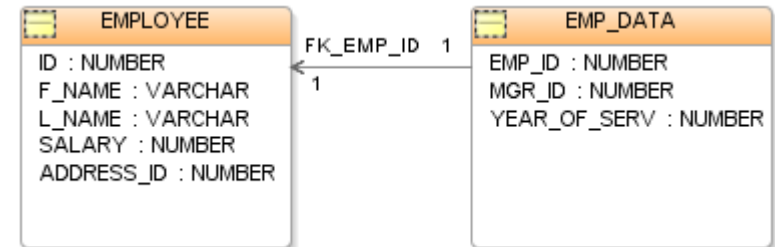
## Modelo Relacional

Tras ser postulados sus fundamentos en 1970 por Edgar Frank Codd no tardó en consolidarse como un nuevo paradigma en los modelos de base de datos.

Su idea fundamental es el uso de "relaciones". Las relaciones se consideran como conjuntos de datos llamados "tuplas". Cada relación se representa como si fuese una tabla que está compuesta por registros (las filas de una tabla), que representarían las tuplas, y campos (las columnas de una tabla).

En este modelo, el lugar y la forma en que se almacenen los datos no tienen relevancia (a diferencia de otros modelos como el jerárquico y el de red).

Principales sistemas: Oracle, DB2, MySQL, PostgreSQL, Microsoft SQL Server.



## Modelo Orientado a Objetos

Nacido a partir de los lenguajes de programación orientados a objetos, incorpora los principales conceptos de este paradigma:

- Encapsulación – Se oculta la información al resto de los objetos, impidiendo así accesos incorrectos o conflictos.
- Herencia – Los objetos pueden heredar comportamiento dentro de una jerarquía de clases.
- Polimorfismo – Una operación puede ser aplicada a distintos tipos de objetos.

### Object-Oriented Model

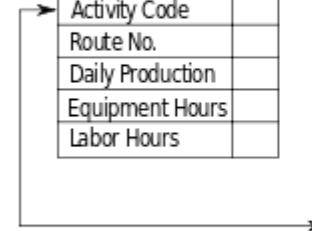
**Object 1:** Maintenance Report      Object 1 Instance

Date	
Activity Code	
Route No.	
Daily Production	
Equipment Hours	
Labor Hours	

01-12-01
24
I-95
2.5
6.0
6.0

**Object 2:** Maintenance Activity

Activity Code	
Activity Name	
Production Unit	
Average Daily Production Rate	



Los tipos de datos complejos (audio, video,...) adquieren un formato nativo.

Los principales productos en el mercado son: GEMSTONE/OPAL (ServicLogic), ONTOS (Ontologic), Objectivity (Objectivity Inc.), Versant (Versant Technologies), ObjecStore (Object Design), pero son productos muy específicos y con un mercado muy limitado.

# ORACLE®

Desarrollado por Oracle Corporation, se considera a Oracle como uno de los sistemas de bases de datos más completos, destacando soporte de transacciones, estabilidad, escalabilidad y soporte multiplataforma.



Propiedad de IBM, bajo la cual se comercializa el sistema de gestión de base de datos, cuenta con la excelente integración con sus servidores como una de sus principales bazas destacando su extensa integración en entornos mainframe.

Integra XML de manera nativa a partir de la versión 9 (pureXML), que permite almacenar documentos completos para realizar operaciones y búsquedas de manera jerárquica dentro de éste, e integrarlo con búsquedas relacionales.



Freeware, motor de base de datos gratuito, soporte multiusuario. Multithread. SQL. Versiones disponibles para Win95/Win98/NT, Linux, Solaris, FreeBSD, AIX, SunOS, etc. JDBC drivers. Freeware bajo licencia GPL.



Freeware. Avanzada base de datos, DBMS objetos-relacional, corre en varios sistemas operativos y contiene drivers para ODBC y JDBC.



Pequeña librería en C que implementa un SBGD self-contained. En lugar de establecer una comunicación cliente-servidor, se integra en una aplicación como parte de la misma.



Uno de los proyectos NoSQL más conocidos del mercado. Se trata de una base de datos distribuida de segunda generación con alta escalabilidad que está siendo usada por gigantes como Facebook (que es quien la ha desarrollado), Digg, Twitter, Cisco y más empresas. El objetivo es ofrecer un entorno consistente, tolerante a fallos y de alta disponibilidad a la hora de almacenar datos.



*En este modelo todos los datos son almacenados en relaciones, y como cada relación es un conjunto de datos, el orden en el que estos se almacenen no tiene relevancia (a diferencia de otros modelos como el jerárquico y el de red). Esto tiene la considerable ventaja de que es más fácil de entender y de utilizar por un usuario no experto. La información puede ser recuperada o almacenada por medio de consultas que ofrecen una amplia flexibilidad y poder para administrar la información.*

*Este modelo considera la base de datos como una colección de relaciones. De manera simple, una relación representa una tabla que no es más que un conjunto de filas, cada fila es un conjunto de campos y cada campo representa un valor que interpretado describe el mundo real. Cada fila también se puede denominar tupla o registro y a cada columna también se le puede llamar campo o atributo.*

Wikipedia

- El artículo que inició todo:  
Edgar Frank Codd, “A **Relational Model** of Data for Large Shared Data Banks”, 1970.
- Modelo lógico basado en teoría de conjuntos: operaciones sobre conjuntos de datos.
- Es sencillo comparado con otros modelos lógicos (red, jerárquico).
- SQL

Base de Datos = Conjunto de Relaciones

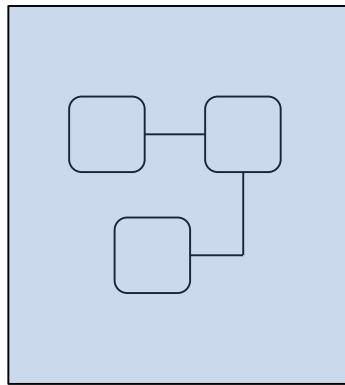
- **Relación**

- Estructura de datos fundamental del modelo.
- Tiene un nombre y representa una entidad genérica
- Conjunto de tuplas
  - Cada tupla representa una entidad concreta
- Compuesta de atributos con nombre (y dominio)
  - Cada atributo representa un atributo de la entidad
- Representada mediante una tabla con filas y columnas

- **Modelo basado en Teoría matemática**

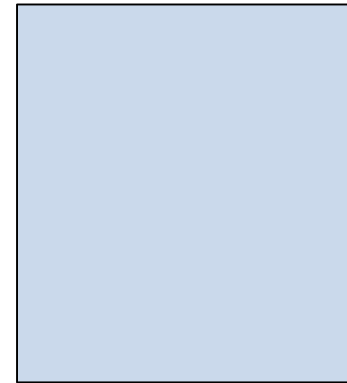
- Analogía entre “Relación” (concepto matemático) y “Tabla”
- Teoría de Conjuntos y Lógica de Predicados de 1er orden Sólida Base Formal

Modelo  
conceptual (E-R)



Transfor-  
mación

Modelo lógico  
(Relacional)



No se deben confundir los conceptos del modelo E-R con los del modelo relacional.

Ej: el concepto de clave primaria no hace parte del modelo E-R

## Ventajas:

- ✓ Separación clara del nivel lógico y el físico.
- ✓ Sencillo y maduro.
- ✓ Fácil modificación de datos y del esquema.
- ✓ Operadores con gran poder de manipulación de los datos → álgebra y cálculo relacional
- ✓ Fundamentación teórica sólida (teoría de conjuntos).
- ✓ Compatibilidad y estandarización.
- ✓ Garantiza la independencia de los datos.
- ✓ Soportado por numerosos sistemas comerciales que garantizan conectividad con los lenguajes de programación más usados (Java, C#, PHP, Visual Basic, etc.)
- ✓ Muy difundido: se consigue fácilmente apoyo técnico.

## Desventajas:

- ✖ No incluye comportamiento a diferencia de los modelos objetual y objeto relacional.
- ✖ Dificultad o imposibilidad para representar :
  - ✖ reglas complejas de negocio
  - ✖ reglas de conocimiento (inferencia)<sup>1</sup>.
- ✖ Dificultad para manejar herencia (se trata de simular).
- ✖ Descompone un elemento de interés en varias tablas<sup>2</sup>.
- ✖ Dificultad ante datos complejos:
  - ✖ Su manejo (p. ej. atributos grupales y multivaluados del modelo conceptual semántico) podría llevar a diseños complejos.
  - ✖ No representa de forma nativa otros elementos comunes: multimedia (video, audio, imagen), datos geográficos, etc.

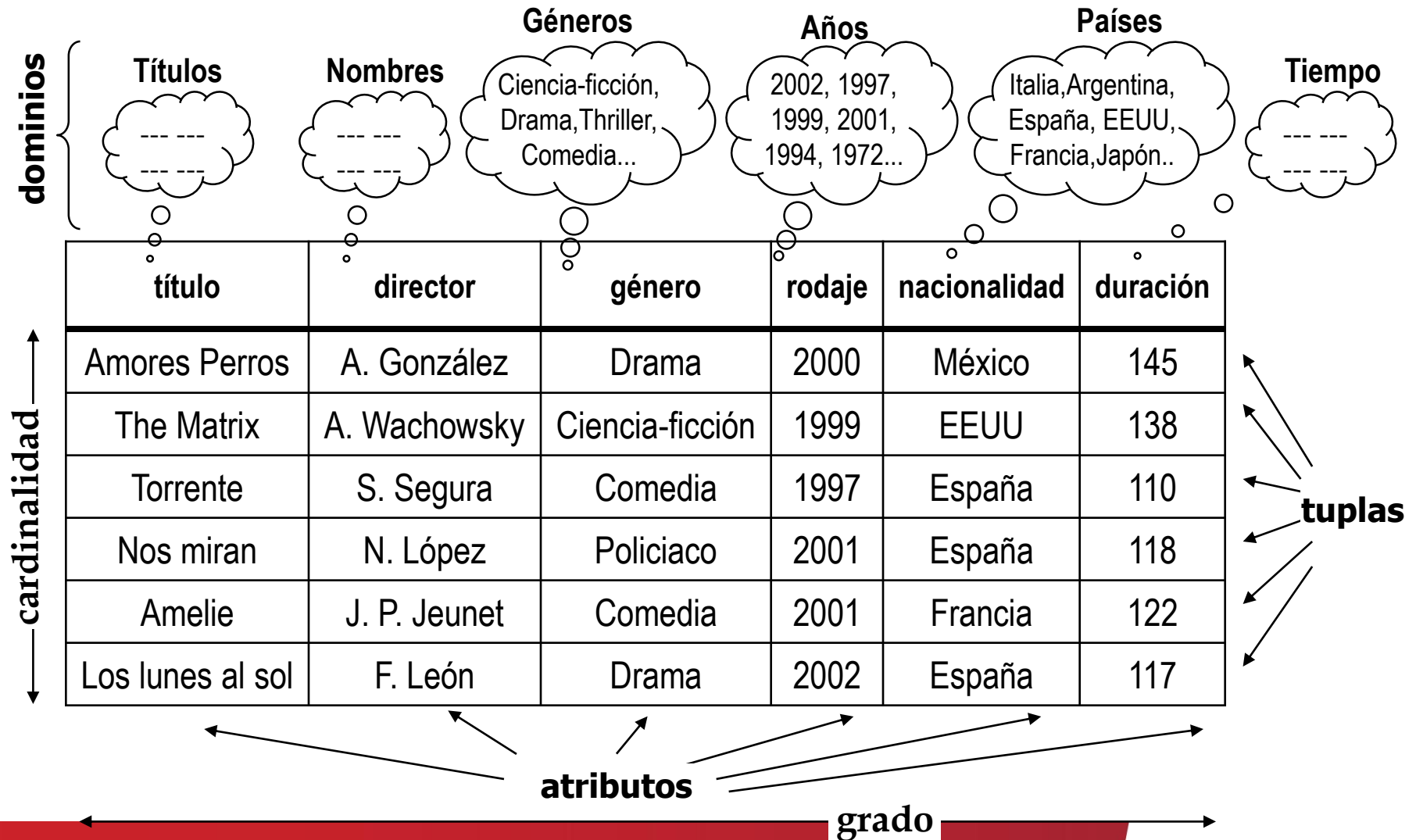
1 - Por medio de vistas se puede suplir en parte este aspecto.

2 - Esto implica que hay que **reconstruir** el elemento de interés.

# Términos básicos:

Modelo Relacional			Procesamiento de Ficheros
Formal		SQL-92	
Relación		Tabla	Fichero
Tupla	Si la tupla $t$ está en la relación $R$ , entonces $t \in R$	Fila	Registro concreto
Atributo	Debe tener un nombre único dentro de cada relación	cabecera de Columna	Nombre de Campo de registro
Cardinalidad	nº de tuplas en una relación	=	
Grado	nº atributos en una relación	=	
Dominio	colección de valores permitidos para ciertos atributos	=	





## Definición Dominios:

- Conjunto de valores atómicos del mismo tipo, donde toman su valor los atributos
  - La definición de dominios forma parte de la definición de la BD
  - Cada atributo definido sobre un ÚNICO dominio  $\Leftrightarrow$  **OBLIGATORIO**
  - Si A, B representan un mismo concepto, A y B con mismo dominio
  - Dominio D puede contener valores no tomados por ningún atributo

$$\{\text{valores de A}\} \subseteq \text{Dominio(A)}$$

- Comparaciones Restringidas a Dominio
  - La comparación de dos atributos sólo tiene sentido si ambos toman valores del mismo dominio
  - Si el SGBD soporta dominios, podrá detectar este tipo de errores

## Definición Relaciones:

Una relación R, sobre conjunto de dominios D1, D2 ... Dn se compone de dos partes:

- **Esquema o Cabecera**

Conjunto de **pares Atributo:Dominio**

$$\{ (A_1:D_1), (A_2:D_2) \dots (A_n:D_n) \}$$

- Cada  $A_j$  tiene asociado sólo un  $D_j$
- Los  $D_i$  no tienen por qué ser distintos entre sí

- **Estado, Cuerpo o Instancia**

- **Conjunto de tuplas** que contiene en un instante concreto
- tupla = conjunto de pares Atributo:Valor

$$\{ \{ (A_1:v_{i1}), (A_2:v_{i2}) \dots (A_n:v_{in}) \} \}, \text{ donde } i=1..m$$


- **Relación vs. Tabla**

- Relación: Representación **abstracta** de un elemento de datos
- Tabla: Representación **concreta** de tal elemento abstracto
- Ventajas
  - Representación muy **sencilla** (tabla) del elemento abstracto básico (relación) del Modelo Relacional
  - Fácil de utilizar, entender, razonar...
- Inconveniente
  - **Aparente orden** entre filas y entre columnas de la tabla

- **Relación vs. Tabla**

- Relación: Representación **abstracta** de un elemento de datos
- Tabla: Representación **concreta** de tal elemento abstracto
- Ventajas
  - Representación muy **sencilla** (tabla) del elemento abstracto básico (relación) del Modelo Relacional
  - Fácil de utilizar, entender, razonar...
- Inconveniente
  - **Aparente orden** entre filas y entre columnas de la tabla

- Definida por Codd, 1972
- Colección de **operadores** que toman **relaciones como operandos** y devuelven **relaciones como resultado**
  - **Operadores tradicionales sobre conjuntos**
    - unión
    - intersección
    - diferencia
    - producto cartesiano

 Los operandos son relaciones, y NO conjuntos arbitrarios

⇒ operaciones adaptadas a relaciones (tipo especial de conjuntos)
  - **Operadores relacionales especiales**
    - restricción
    - proyección
    - reunión ( join )
    - división

- **Clausura relacional**

El resultado de cualquier operación del álgebra relacional es otra relación

La salida de una operación puede ser entrada (operando) de otra.

⇒ **Expresiones Anidadas**

Sus operandos son otras expresiones del álgebra (en lugar de nombres de relación)



## Compatibilidad de tipos:

Sean  $R ( r_1, r_2, \dots, r_n )$ ,  $S ( s_1, s_2, \dots, s_n )$

- **Relaciones R y S compatibles en tipo** si tienen el “mismo” esquema, es decir:
  1. Igual número de atributos:  
 $\text{grado}(R) = \text{grado}(S) = n$
  2. Atributos correspondientes definidos sobre el mismo dominio:  
 $\text{dom}(r_i) = \text{dom}(s_i) \text{ , , } i = 1, 2, \dots, n$

Ejemplo: DIRECTOR y DIR\_FOTOG son de tipos compatibles

- **UNIÓN, INTERSECCIÓN, DIFERENCIA** necesitan operandos compatibles en tipo
- **PRODUCTO CARTESIANO** no necesita compatibilidad de tipo en sus operandos

- **Unión de relaciones**

$R \cup S$ , con  $R$  y  $S$  compatibles en tipo, es una **relación** tal que:

Esquema: **el de  $R$  (o  $S$ )**

Estado: **conjunto de tuplas que están en  $R$ , en  $S$  o en ambas**

👉 Las tuplas repetidas se eliminan (por definición)

Ejemplo:  $DIRECTOR \cup DIR\_FOTOG$

- **Intersección de relaciones**

$R \cap S$ , con  $R$  y  $S$  compatibles en tipo, es una **relación** tal que:

Esquema: **el de  $R$  (o  $S$ )**

Estado: **conjunto de tuplas que están a la vez en  $R$  y en  $S$**

Ejemplo:  $DIRECTOR \cap DIR\_FOTOG$

- **Diferencia entre relaciones**

$R - S$ , con  $R$  y  $S$  compatibles en tipo, es una **relación** tal que:

Esquema: **el de  $R$  (o  $S$ )**

Estado: **conjunto de tuplas que están en  $R$ , pero NO en  $S$**

👉 operación con «cierta direccionalidad», como la resta aritmética

Ejemplo: DIRECTOR — DIR\_FOTOG

- **Secuencias de operaciones**

La propiedad de clausura relacional permite aplicar una operación tras otra

Sean  $R$ ,  $S$ ,  $T$  relaciones de tipos compatibles,

– Única expresión: **expresiones anidadas**  $R \cap (S \cup T)$

– Varias expresiones: **relaciones intermedias con nombre**

$A \leftarrow S \cup T$   
 $B \leftarrow R \cap A$

- **Producto Cartesiano:**

$R \times S$ , con  $R$  y  $S$  cualesquiera, es una **relación** tal que:

Esquema: **combinación (unión) de los esquemas de  $R$  y  $S$**

Estado: **conjunto de todas las tuplas formadas por las posibles combinaciones de cada tupla de  $R$  con cada tupla de  $S$**

Ejemplo:  $PELICULA \times DIRECTOR$

Obtiene un conjunto de tuplas tales que cada una es la combinación de una tupla de  $PELICULA$  y otra de  $DIRECTOR$

El **esquema** de la relación resultante de  $R \times S$  debe estar **bien formado** (nombres de atributos únicos)

Si  $R$  y  $S$  tienen atributos con igual nombre,  $R \times S$  tendría dos atributos nombrados igual  $\rightarrow$  Error!

$ACTOR \times AGENCIA \Rightarrow$  “colisión” de nombres en atributo “nombre”

- **Restricción de una Operación**

Obtener un subconjunto de las tuplas de una relación para las cuales se satisface una condición de selección

$$\sigma_{\langle \text{condición} \rangle} (\langle \text{relación} \rangle)$$

Mecanismo de selección del sistema

- Aplica  $\langle \text{condición} \rangle$  a cada tupla individual de  $\langle \text{relación} \rangle$ , sustituyendo cada atributo por su valor en la tupla
- Si  $\langle \text{condición} \rangle$  es TRUE, la tupla se selecciona para el resultado

Operador Restricción: Unario

- Sólo se aplica a UNA relación
  - Nunca** puede **seleccionar** tuplas **de más de una relación**
- Se aplica a UNA sola tupla a la vez
  - $\langle \text{condición} \rangle$  **nunca** se **refiere** a **más de una tupla**

- **Proyección de una relación**

Sólo interesan algunos atributos de una relación

Se **proyecta** la relación sobre esos atributos

Restricción vs. Proyección :

- $\sigma$  selecciona algunas **tuplas** de la relación y desecha otras
- $\pi$  selecciona ciertos **atributos** y desecha los demás

$$\pi_{\langle \text{listAtrib} \rangle}(\langle \text{relación} \rangle)$$

Resultado: **Relación** (conjunto de tuplas) cuyos **atributos** son **sólo** los de **<listAtrib>** y **en** ese **orden**

- **<listAtrib>** lista de nombres de atributos de <relación>

\* Obtener el código, nombre y el caché de todos los actores

$$\pi_{\text{codA, nombre, cache}}(\text{ACTOR})$$

- **Reunión o Join entre dos relaciones**

**Combina las tuplas relacionadas de dos relaciones en una sola tupla**

Permite procesar **vínculos entre relaciones**

- \* Datos de **películas** junto con los de **su director** correspondiente
  - Es necesario combinar cada tupla de PELÍCULA, p, con la tupla DIRECTOR, d, tal que el valor de codDir en d coincida con el de director en p
  - Se consigue aplicando la operación REUNIÓN a las dos relaciones

$R1 \leftarrow \text{PELICULA} \quad \text{director=codDir} \quad \text{DIRECTOR}$



- **Funciones de agregados**

Funciones **aplicadas a un conjunto de tuplas**

- SUMA
- PROMEDIO
- MÁXIMO
- MÍNIMO
- CUENTA (número de tuplas en una relación)

**Agrupación de tuplas** según valor de ciertos atributos

- Puede aplicarse una función agregada a cada grupo por separado

\* Media del caché de los actores agrupados por agencias      ¿Solución?

- Agrupar actores según su agencia representante (valor de atributo *agencia*)
  - » Cada grupo incluye tuplas de actores representados por la misma agencia
- Cálculo del caché medio de cada grupo (función PROMEDIO)

El resultado es una **relación**      R(agencia, PROMEDIO\_caché)

- Lenguaje formal para BD Relacionales
- Basado en Cálculo de Predicados de Primer Orden (rama de Lógica Matemática)

Formas de adaptar el Cálculo de Predicados de 1er Orden para crear un Lenguaje de Consultas para BDR:

- Cálculo Relacional de Tuplas (CRT) (Codd, 1972)

*Una forma de la lógica en donde las variables son tuplas.  $\{t / P(t)\}$  el conjunto de tuplas tales que el predicado  $P$  es cierto.*

- Cálculo Relacional de Dominios (Lacroix y Pirotte, 1977)

*Está constituido con los mismos operadores que el CRT pero no hay tuplas sino variables dominio. Las expresiones del cálculo relacional de dominios son de la forma  $\{ (x, y, z, ...) / P(x, y, z, ...) \}$ , donde  $x, y, z$  representan las variables de dominio,  $P$  representa una fórmula compuesta de átomos (igual que en el CRT).*

## Lógica relacional:

- Podemos definir una formula con base a combinaciones de fórmulas atómicas.
- Una formula atómica es una combinación de variables (tipo tupla o tipo dominio, según corresponda) y atributos o constantes, gracias al uso de operadores como  $<$ ,  $>$ ,  $=$ ,  $\neq$ ,  $\leq$ ,  $\geq$ .
- También es una formula atómica variable  $\in$  Relación.
- Las combinaciones de fórmulas atómicas se generan a partir del uso de operadores como NOT ( $\neg$ ), AND ( $\wedge$ ), OR ( $\vee$ ),  $\rightarrow$ .
- Los cuantificadores  $\exists$ ,  $\forall$  limitan una variable:
  - $\exists$ , cuantificador existencial: elementos que existen dentro de un conjunto que cumplen una proposición.
  - $\forall$ , cuantificador universal: todos los elementos que cumplen una proposición.

<b>Cálculo Relacional vs. Álgebra Relacional</b>	
- Expresiones <b>Declarativas</b> (lenguaje no procedimental)	- <b>Secuencias</b> de Operaciones
» No se indica <b>CÓMO</b> evaluar la consulta, sino <b>QUÉ</b> se desea obtener » <b>Describe</b> la información deseada sin dar un procedimiento específico para obtenerla	» Aunque se anidan para formar una sola expresión, <b>siempre</b> se indica explícitamente cierto <b>orden</b> de las operaciones » <b>Estrategia parcial de evaluación</b> de la consulta (≈ lenguaje procedimental de alto nivel )

