Universidad Nacional de la Patagonia San Juan Bosco
Facultad de Ingeniería

Cátedra: **Bases de datos I**

Diseño de las Bases de Datos Relacionales

título	fecha	socio
T1	FT	S1
T2	FU	S2
T3	FV	S1
T4	FG	S4
T1	FH	S3
T2	FT	S4
T3	FV	S3

título	fecha
T1	FT
T2	FU
T3	FV
T4	FG
T1	FH
T2	FT
T3	FV

título	socio
T1	S1
T2	S2
T3	S1
T4	S4
T1	S3
T2	S4
T3	S3

fecha	socio
FT	S1
FU	S2
FV	S1
FG	S4
FH	S3
FT	S4
FV	S3

- Objetivo de una BDR
  - Generar un conjunto de esquemas relacionales que permita:

    ✓ **Almacenar información sin redundancia innecesaria**
    ✓ **Recuperar información fácilmente**.

  - Una técnica es diseñar esquemas que tengan una **forma normal adecuada**.
  - Para determinar si un esquema de relaciones tiene una de las formas normales, necesitaremos información adicional sobre la empresa del "mundo real" que vamos a modelar con la BD.

- Peligros en el diseño de BDR
  - Entre las propiedades indeseables que un mal diseño puede tener se encuentran:

    ✓ Repetición de información
    ✓ Incapacidad para representar cierta información
    ✓ Pérdida de información

- Veremos un ejemplo con los siguientes esquemas de relaciones:

  PROVEEDOR=(id_proveedor, nombre_proveedor, calificación, ciudad_proveedor)
  ENVÍO=(id_proveedor , id_artículo, cantidad)
  ARTÍCULO=(id_artículo, nombre_artículo, color)

- Una tupla de la relación *ENVÍO* tiene el siguiente significado intuitivo:
- ✓ t[cantidad] es la cantidad del artículo t[id_artículo] que envía el proveedor t[id_proveedor]

- Una tupla de la relación ARTÍCULO tiene el siguiente significado intuitivo:
- ✓ t[id_artículo] es el número asignado a un artículo
- ✓ t[nombre_artículo] es la descripción del artículo cuya identificación es t[id_artículo]

- Ahora, un diseño alternativo para la BD anterior en el que sustituimos los esquemas por uno solo:

  VENTA=(id_proveedor, nombre_proveedor, calificación, ciudad_proveedor, id_artículo, cantidad)

- Supóngase que queremos añadir un nuevo envío a esta BD: el artículo A349 es enviado por el proveedor P273, el cual envía 30 unidades. En el diseño original añadiríamos a la relación *ENVÍO* la tupla:

  (“P273”, “A349”, 30)

- Bajo el diseño alternativo, necesitamos una tupla que tenga valores para todos los atributos de *VENTA*.

  (“P273”, “Pérez”, 77, “Trelew”, “A349”,30)

  → tuvimos que añadir todos los datos del proveedor.

- La repetición de información que requiere el uso del diseño alternativo no es conveniente y complica la actualización de la BD.
- Supóngase, p. e. que el proveedor “P129”, que vivía en Madryn se traslada a Comodoro Rivadavia.
- Bajo el diseño original, se necesita cambiar una tupla de la relación *PROVEEDOR*.
- Bajo el diseño alternativo, se necesita cambiar muchas tuplas de la relación *VENTAS*, todas las veces que aparece este proveedor en un envío.

  → Las actualizaciones son más costosas bajo el diseño alternativo que bajo el diseño original.
- Cuando realizamos la actualización en la BD alternativa, debemos asegurarnos de que se actualizan cada una de las tuplas pertenecientes al proveedor “P129”, de lo contrario la BD presentará dos ciudades para el mismo.

- Dependencias funcionales
  - Este concepto es muy importante en la teoría de diseño de BDR.
  - Las dependencias funcionales implican restricciones que se deben cumplir en la BD.
  - Algunas definiciones:
  - ✓ Una dependencia funcional es una relación 1:M
  - ✓ Sea r una relación y sean X e Y subconjuntos del conjunto de atributos de r. Se dice que Y es funcionalmente dependiente de X (**X → Y**) si cada valor de X en r está asociado con, precisamente, un valor de Y en r. En otras palabras, cuando dos tuplas de r coinciden sobre un mismo valor de X, también coinciden sobre un mismo valor de Y.

X	**T**	**Y**
X_1	T_1	Y_1
X_1	T_3	Y_1
X_2	T_1	Y_2
X_3	T_2	Y_3
X_1	T_3	Y_1

  - **X →Y**: a X se lo llama determinante, a Y dependiente.
  - Se lee **X determina a Y** o **Y es funcionalmente dependiente de X**

- Las definiciones anteriores explicitan que para una relación r es posible considerar un amplio conjunto de dependencias funcionales entre los atributos que la componen. Esto es obvio si consideramos para la segunda definición que X es una clave candidata -más aún si es una clave principal- e Y son todos los atributos de la relación.

- Las DF exigen que un conjunto de atributos determine de manera unívoca a otro conjunto de atributos.

- P. e. En el caso de la relación *ENVÍO,* Existe una dependencia funcional del conjunto de atributos {id_proveedor, id_artículo} al atributo {cantidad}:

✓ **id_proveedor, id_artículo → cantidad**, lo que significa que para cualquier valor dado del par id_proveedor e id_artículo, sólo existe un valor correspondiente del atributo cantidad. Además muchos valores distintos del par id_proveedor, id_artículo, pueden tener el mismo valor de cantidad.

- Definiciones básicas
  - Definamos una relación envío modificada, *ENVÍO2*, para ilustrar las definiciones de esta sección:

id_proveedor	ciudad	id_artículo	cantidad
P1	CRD	A1	100
P1	CRD	A2	100
P2	Madryn	A1	200
P2	Madryn	A2	200
P3	Madryn	A2	300
P4	Trelew	A2	400
P4	Trelew	A4	400
P4	Trelew	A5	400

  - Es **muy importante** distinguir entre: el conjunto de tuplas de una relación en un momento dado y el conjunto de todos los valores posibles que puede tener esa relación en distintos momentos.
  - Las dependencias se establecen para todos los posibles valores legales de los atributos, todos los posibles en la realidad.
  - En nuestro ejemplo las dependencias funcionales son:
  - Id_proveedor → ciudad
  - Id_proveedor, Id_artículo → cantidad
  - Id_proveedor → cantidad

- La dependencia funcional id_proveedor → cantidad, que en la relación en un momento dado se cumplía, ahora significaría que todo envío que haga un proveedor es por la misma cantidad. Y esa no es una restricción, es cierta en ese momento de la relación (para esas tuplas en particular).

- **Si X es una clave candidata de la relación r, entonces todos los atributos Y de la relación deben ser dependientes funcionalmente de X.**
- P. e. Para el esquema artículo, tenemos necesariamente:

  id_artículo → id_artículo, nombre_artículo, color

- Ya que id_artículo es clave de esa relación. De hecho si la relación r satisface la DF A → B, y A no es clave candidata, entonces r tendrá cierta redundancia.
- P. e. En el caso de *ENVÍO2*, la DF id_proveedor → ciudad, implica que un proveedor vive en una ciudad, ésta aparecerá tantas veces como nombremos al proveedor en la relación.

- El conjunto de DF de una relación es muy grande, aún cuando nos limitemos a las DF que son válidas todo el tiempo. Lo que interesa es reducir ese número a un tamaño manejable.

- ¿Por qué interesa este objetivo? Porque las **DF representan restricciones de integridad** y el SGBD necesita conocerlas para hacerlas cumplir. Entonces si podemos encontrar un subconjunto de DF manejable pero que interprete a las totales, el SGBD verificará que se cumpla este subconjunto.

- Dependencias triviales
  - Una forma de reducir el tamaño del conjunto de DF con las que necesitamos tratar es eliminando las dependencias triviales.
  - Una dependencia es trivial si es imposible que no pueda ser satisfecha. Esto ocurre si y sólo **si el conjunto a la derecha es un subconjunto del que está a la izquierda**. En la práctica estas dependencias no son interesantes, sin embargo cuando tratamos con la teoría formal debemos tomar en cuenta todas las dependencias.
  - id_artículo, nombre_artículo → id_artículo
  - id_artículo → id_artículo

- Cierre de conjuntos de dependencias
  - Algunas DF pueden implicar otras, por ejemplo la siguiente DF:
  - ✓ Id_proveedor, Id_artículo → cantidad, ciudad
  - Implica las dos siguientes
  - ✓ Id_proveedor, Id_artículo →ciudad
  - ✓ Id_proveedor, Id_artículo → cantidad

  - Al **conjunto de todas las DF implicadas por un conjunto dado S de DF se le llama cierre de S**, y se escribe **S^+**. Para encontrar S^+ necesitamos un algoritmo que nos permita calcularlo a partir de S. Existen un conjunto de reglas de inferencia, conocidas como **axiomas de Armstrong** mediante las cuales es posible inferir nuevas DF a partir de las dadas.

- Una de las formas mas sencillas de enunciar dichas reglas es la siguiente:

✓ Sean A, B, y C subconjuntos de atributos de la relación R y acordemos escribir AB para representar la unión de A y B. Entonces:

1. **Reflexividad**: Si B es un subconjunto de A entonces A → B
2. **Aumento**: Si A → B, entonces AC → BC
3. **Transitividad**: Si A → B y B → C entonces A → C

- Las reglas son **completas**, en el sentido de que dado un subconjunto S de DF, todas las dependencias implicadas por S pueden derivarse de S usando estas reglas.

- Estas reglas también son **correctas**, porque no generan dependencias funcionales incorrectas. Es decir las reglas pueden usarse para derivar S^+.

- Existen **reglas adicionales** (que se derivan de las anteriores) pero se usan para simplificar la tarea práctica de calcular S^+ a partir de S:

4. **Autodeterminación**: A → A.
5. **Descomposición**: si A → BC entonces A → B y A → C.
6. **Unión**: Si A → B y A → C, entonces A → BC.

- **Composición**: Si A → B y C → D, entonces AC → BD.

- P. e. Tenemos una relación con los atributos A, B , C, D, E y F y las DF:
- A → BC
- B → E
- CD → EF
- Vamos a demostrar que la DF AD → F es válida para r y por lo tanto, miembro del conjunto dado.
- **A → BC** dada
- **A → C** descomposición
- **AD → CD** aumento
- **CD → EF** dada
- **AD → EF** transitividad
- **AD → F** descomposición

- Cierre de un conjunto
  - En principio podemos calcular S^+ mediante un algoritmo que dice "aplicar repetidamente las reglas vistas hasta que dejen de producir nuevas DFs".
  - **En la práctica existe poca necesidad de calcular el cierre.**
  - Para identificar el conjunto de dependencias funcionales de una relación:
  - Se comienza identificando las dependencias que puedan deducirse a partir de la semántica de los atributos
  - Se aplican los axiomas de Armstrong para inferir dependencias funcionales que también sean ciertas para esa relación (calculando el cierre de A, A^+, para cada conjunto de atributos A que aparezca en el determinante de una dependencia funcional)

- Cierre de un conjunto de Atributos
  - Es interesante calcular el cierre de un conjunto de atributos A. Para comprobar si ese conjunto A es una superclave de una relación r, hay que utilizar un algoritmo para calcular el conjunto de atributos determinados funcionalmente por A.
  - Se denomina **cierre de A bajo S**, al conjunto de todos los atributos determinados funcionalmente por A bajo el conjunto de dependencias funcionales S, se denomina A^+.
  - A es superclave si A^+ contiene todos los atributos de r
  - P. e. Supongamos una relación con los atributos A, B, C, D, E y F y las DFs:
  - A → BC
  - E → CF
  - B → E
  - CD → EF
  - Queremos saber si A, B es superclave: debemos encontrar el cierre de $\{A, B\}^+$
  1. Comenzamos con {A, B}

     Vemos la primer DF, A → BC, entonces agregamos C: {A, B, C}
  2. En la segunda DF, como no encontramos en su parte izquierda ningún atributo que esté en {A, B, C} no hay cambios.
  3. B → E, agrega E al conjunto de cierre {A, B, C, E}
  4. CD → EF no cambia nada
  5. Volvemos a la primera, DF y no cambia nada, en la segunda E → CF, agrega F al cierre {A, B, C, E, F}, en las otras dos DF no cambia nada.
  - Repetimos el proceso desde la primer DF y no cambia nada. Entonces A^+ es el conjunto {A, B, C, E, F}. Por lo tanto {A, B} no es superclave.

- Dependencias Funcionales Útiles
  - La tarea de especificar todas las dependencias funcionales posibles en los proyectos reales de BD suele ser muy a menudo imposible de abordar. Dado que el conjunto de todas las dependencias funcionales para una relación dada puede tener un gran tamaño, **¿cómo se puede identificar cuáles son las dependencias funcionales útiles dentro de una relación?**

  - Es necesaria una técnica que permita **reducir el conjunto de dependencias funcionales a un tamaño manejable**: Un conjunto de dependencias funcionales Y está cubierto por un conjunto de dependencias funcionales X, si toda dependencia funcional de Y está también en X^+, es decir, si toda dependencia funcional contenida en Y puede inferirse a partir de X. A esto se denomina recubrimientos canónicos o mínimos. Puede haber varios recubrimientos mínimos.

- Recubrimiento canónico
  - **Si se dispone de un conjunto de DF S** en el esquema de una relación, siempre que se lleve a cabo una actualización de la relación, el SGBD debe asegurar que **en el nuevo estado de la BD se satisfagan todas las DF de S**, y debe retroceder la actualización si esto no ocurre.

  - Es posible reducir el cierre de S a un conjunto llamado conjunto de dependencias funcionales simplificado. Cualquier BD que satisfaga el conjunto de dependencias simplificado también satisfará el conjunto original y viceversa.

- **Un recubrimiento canónico S_c de S es un conjunto de dependencias funcionales simplificado** (o mínimo) tal que S implica lógicamente todas las dependencias de S^+ y S_c implica lógicamente todas las dependencias de S. Además S_c debe cumplir las siguientes propiedades:

✓ La parte derecha de toda DF en S involucra solo un atributo.

✓ Cada lado izquierdo de una DF es irreducible. No es posible descartar ningún atributo

- P. e. Sean las DFs
- legajo → dni, apellido, dirección
- dni → apellido
- legajo, dni → apellido
- legajo → dni
- Legajo, apellido→ salario

- Encontremos el recubrimiento canónico de S:

1. El primer paso es escribir las DFs de modo que a la derecha haya solo un elemento

- legajo → dni
- legajo → apellido
- legajo → dirección
- dni → apellido
- legajo, dni → apellido
- legajo → dni (ya existe)
- legajo, apellido→ salario

2. Podemos eliminar **apellido** de **legajo, apellido → salario** , ya que:
   - tenemos **legajo → apellido**,
   - por lo tanto es verdad **legajo → legajo, apellido**
   - y tenemos **legajo, apellido → salario**
   - por lo tanto **legajo → salario** por transitividad. (el lado izquierdo era reducible)

2. **legajo, dni → apellido**, puede ser eliminada, ya que:
   - tenemos **legajo → apellido**
   - por lo tanto **legajo, dni → apellido, dni**, por aumento
   - y por descomposición **legajo, dni → apellido**

2. **legajo → apellido** está implicada en **legajo → dni** y **dni → apellido**

– Por lo tanto el recubrimiento canónico es:

✓ **legajo → dni**

✓ **legajo → dirección**

2. **dni → apellido**

✓ **legajo → salario**

- Descomposición sin pérdida
  - El procedimiento de descomposición implica **separar** (descomponer) una relación en otras y además requiere que esta descomposición sea **reversible**, de tal forma que no se pierda información en el proceso.

  - Por eso en las única forma de descomposición en que estamos interesados es en la descomposición sin pérdida. Veremos que una **descomposición sin pérdida está ligada con el concepto de dependencia funcional**.

  - A continuación tenemos un esquema que podemos llamar *EMPLEADO*, que pertenece a los empleados de una empresa:

    EMPLEADO=(legajo, departamento, dirección)

  - La siguiente figura representa una relación de este esquema, y dos posibles descomposiciones:

legajo	**departamento**	**dirección**
1234	Estadística	San Martín 23
3452	Estadística	Rivadavia 210
5432	Informática	Alem 234

- Descomposición sin pérdida (cont.)

legajo	departamento	dirección
1234	Estadística	San Martín 23
3452	Estadística	Rivadavia 210
5432	Informática	Alem 234

  - Descomposición 1:

legajo	departamento
1234	Estadística
3452	Estadística
5432	Informática

legajo	dirección
1234	San Martín 23
3452	Rivadavia 210
5432	Alem 234

  - Descomposición 2:

legajo	departamento
1234	Estadística
3452	Estadística
5432	Informática

departamento	dirección
Estadística	San Martín 23
Estadística	Rivadavia 210
Informática	Alem 234

- Si observamos las dos descomposiciones vemos que en la primera no se pierde información, podemos encontrar que el empleado 1234 trabaja en el departamento estadística y vive en San Martín 23. Por lo tanto es **sin pérdida**.

- En la segunda descomposición podemos ver que el empleado 3452 trabaja en el departamento Estadística pero no se cual es su dirección. Esta descomposición es **con pérdida**.

- Veamos que el proceso al que hemos nombrado como descomposición es en realidad una proyección de la relación original, *EMPLEADO*. En el caso 1 no se perdió información, es decir que si juntamos nuevamente las dos relaciones obtenemos la relación original. Sin embargo en el caso 2 si juntamos las dos relaciones no podemos armar nuevamente empleado. Si el operador para descomponer una relación es la **proyección**, el operador para "recomponer" la relación original es el **join natural**.

- ¿Cómo sabemos cual es la descomposición sin pérdida? Aquí es donde entran las dependencias funcionales. Observemos que *EMPLEADO* satisface el conjunto mínimo de DF (recubrimiento canónico)

  legajo → departamento y legajo → dirección

- Por eso es que la primera descomposición es sin pérdida mientras que la segunda es con pérdida, ya que en ésta última se pierde una de las DF de *EMPLEADO*, es claro que no está la dependencia legajo → dirección.

- Teoría de la Normalización
  - Una estructura de datos puede ser modelada de formas muy distintas. El modelo relacional sería un camino a seguir. Un modelo de datos debería ser manejable y garantizar al mismo tiempo un acceso rápido a los datos. Si las características se modificasen, debería poderse cambiar la estructura sin grandes dificultades, por ejemplo, cuando se requiere un nuevo atributo.
  - La normalización es una **técnica para obtener estructuras de datos eficientes**. El concepto de normalización fue introducido por Codd y fue pensado para sistemas relacionales pero tiene aplicaciones más amplias. La normalización es la expresión formal de una manera de proceder según la cual trabajan muchos creadores de sistemas. Esta ofrece posibilidades para poder describir la estructura lógica de los datos en un sistema de información.

  - Las ventajas de la normalización son:
  - ✓ Evitar ciertas anomalías de actualización
  - ✓ La reestructuración de datos al dar de alta nuevos datos es mínima. La independencia de datos queda mejorada, pudiendo ejecutar ampliaciones en la base de datos con muy poca o casi ninguna repercusión en los programas de aplicación existentes, que acceden a la base de datos
  - ✓ No se practica ninguna fuerza artificial sobre las estructuras de datos
  - ✓ Eliminar cierta clase de redundancia
  - ✓ Producir un diseño que sea una buena representación de la realidad, intuitivamente fácil de entender
  - ✓ Simplificar el cumplimiento de ciertas restricciones de integridad

- Los principios fundamentales de la normalización sirven de herramienta en el desarrollo de sistemas de información.
- Estos principios pueden utilizarse tanto para BD como en cualquier sistema de gestión de archivos.

- La normalización involucra fases que deben realizarse en orden:

✓ Primera Forma Normal –1FN
✓ Segunda Forma Normal – 2FN
✓ Tercera Forma Normal – 3FN
✓ Forma Normal de Boyce Codd – FNBC
✓ Cuarta Forma Normal – 4FN
✓ Quinta Forma Normal – 5 FN

- Primera, Segunda y Tercera Formas Normales
  - El proceso de Codd consiste de tres reglas o etapas que transforman una relación no normalizada en otra expresada en 3FN.

  - **Primera forma normal**: Una relación está en 1FN si no incluye grupos repetitivos, es decir, si cumple con la propiedad de toda tupla contiene exactamente un valor para cada atributo, es decir no tienen atributos que a su vez sean conjuntos.

  - **Segunda Forma Normal**: Una relación está en 2FN si todos sus campos (atributos) dependen de la clave primaria completa y no sólo de una parte de ésta. Es decir, una relación está en 2FN si, además de estar en 1FN, cualquiera de sus atributos no primarios (no forma parte de la clave) tienen una dependencia funcional con cada una de las claves candidatas. Esta forma aboga por relaciones en que solamente existan dependencias funcionales completas, eliminando, mediante proyección, las que estando en 1FN no cumplan esa condición.

  - **Tercera Forma Normal**: Una relación está en 3FN, si está en 2FN y además ninguno de sus campos no primarios tienen dependencia transitiva respecto de las claves candidatas.

- P. e. Un conjunto de proveedores distribuyen artículos escolares en ciudades especificas. Un proveedor solo puede cubrir una ciudad. La siguiente tabla muestra la distribución de los artículos que son distribuidos por cada proveedor y las cantidad vendida.

id_proveedor	**departamento**	**ciudad**	**id_artículo**	**descripción**	**Cantidad**
P1	Escalante	CRD	A1	Lapicera	300
			A2	Regla	200
			A3	Goma	400
			A4	Lápiz negro	200
			A5	Marcadores	100
			A6	Cuaderno	100
P2	Rawson	Trelew	A1	Lapicera	300
			A2	Regla	400
P3	Rawson	Trelew	A2	Regla	200
P4	Escalante	CRD	A2	Regla	200
			A5	Marcadores	300
			A6	Cuaderno	400

- Primera Forma Normal

Clave: id_proveedor

id_proveedor	departamento	ciudad
P1	Escalante	CRD
P2	Rawson	Trelew
P3	Rawson	Trelew
P4	Escalante	CRD

Clave: id_proveedor, id_artículo

id_proveedor	id_artículo	descripción	Cantidad
P1	A1	Lapicera	300
P1	A2	Regla	200
P1	A3	Goma	400
P1	A4	Lápiz negro	200
P1	A5	Marcadores	100
P1	A6	Cuaderno	100
P2	A1	Lapicera	300
P2	A2	Regla	400
P3	A2	Regla	200
P4	A2	Regla	200
P4	A5	Marcadores	300
P4	A6	Cuaderno	400

- Segunda Forma Normal

Clave: id_proveedor

id_proveedor	departamento	ciudad
P1	Escalante	CRD
P2	Rawson	Trelew
P3	Rawson	Trelew
P4	Escalante	CRD

Clave: id_artículo

id_artículo	descripción
A1	Lapicera
A2	Regla
A3	Goma
A4	Lápiz negro
A5	Marcadores
A6	Cuaderno

id_proveedor	id_artículo	Cantidad
P1	A1	300
P1	A2	200
P1	A3	400
P1	A4	200
P1	A5	100
P1	A6	100
P2	A1	300
P2	A2	400
P3	A2	200
P4	A2	200
P4	A5	300
P4	A6	400

← Clave: id_proveedor, id_artículo

- Tercera Forma Normal

Clave: id_proveedor

id_proveedor	ciudad
P1	CRD
P2	Trelew
P3	Trelew
P4	CRD

Clave: ciudad

ciudad	departamento
CRD	Escalante
Trelew	Rawson

Clave: id_artículo

id_artículo	descripción
A1	Lapicera
A2	Regla
A3	Goma
A4	Lápiz negro
A5	Marcadores
A6	Cuaderno

id_proveedor	id_artículo	Cantidad
P1	A1	300
P1	A2	200
P1	A3	400
P1	A4	200
P1	A5	100
P1	A6	100
P2	A1	300
P2	A2	400
P3	A2	200
P4	A2	200
P4	A5	300
P4	A6	400

← Clave: id_proveedor, id_artículo

- Conservación de la dependencia
  - Durante el proceso de reducción, es frecuente el caso de que una relación pueda descomponerse en varias formas diferentes, sin pérdida. Pero existen descomposiciones independientes, en el siguiente sentido: es posible hacer actualizaciones a cualquiera de las proyecciones obtenidas sin considerar la otra. Esta descomposición en proyecciones independientes se denomina conservación de la dependencia. Veamos este concepto en el ejemplo anterior.
  - Suponga la descomposición dada
  - Id_proveedor + ciudad
  - Ciudad + departamento
  - E imaginemos otra posible descomposición, que conserva las DF:
  - Id_proveedor + ciudad
  - Id_proveedor + departamento
  - Veamos que en la primera situación las proyecciones son independientes, en el segundo caso no porque hay que vigilar las actualizaciones en las dos proyecciones de modo que no deje de cumplirse la DF ciudad → departamento

- Teoría de la Normalización
  - Una estructura de datos puede ser modelada de formas muy distintas. El modelo relacional sería un camino a seguir. Un modelo de datos debería ser manejable y garantizar al mismo tiempo un acceso rápido a los datos. Si las características se modificasen, debería poderse cambiar la estructura sin grandes dificultades, por ejemplo, cuando se requiere un nuevo atributo.
  - La normalización es una **técnica para obtener estructuras de datos eficientes**. El concepto de normalización fue introducido por Codd y fue pensado para sistemas relacionales pero tiene aplicaciones más amplias. La normalización es la expresión formal de una manera de proceder según la cual trabajan muchos creadores de sistemas. Esta ofrece posibilidades para poder describir la estructura lógica de los datos en un sistema de información.

  - Los principios fundamentales de la normalización sirven de herramienta en el desarrollo de sistemas de información.
  - Estos principios pueden utilizarse tanto para BD como en cualquier sistema de gestión de archivos.

  - La normalización involucra fases que deben realizarse en orden:
  - ✓ Primera Forma Normal –1FN
  - ✓ Segunda Forma Normal – 2FN
  - ✓ Tercera Forma Normal – 3FN
  - ✓ Forma Normal de Boyce Codd – FNBC
  - ✓ Cuarta Forma Normal – 4FN
  - ✓ Quinta Forma Normal – 5 FN

- Normalización Avanzada del Modelo Relacional
  - Las relaciones en 3FN están normalmente lo suficientemente bien estructuradas como para evitar los problemas asociados a la redundancia de datos. Sin embargo, se han creado una serie de formas normales más avanzadas para identificar una serie de problemas relativamente raros que afectan a las relaciones y que, si no se corrigen, pueden dar como resultado una redundancia de datos indeseable

- Forma Normal de Boyce-Codd
  - En lo anterior siempre consideramos que toda relación tenía una sola clave candidata. Pero la 3 FN no trata adecuadamente los casos de relaciones que tienen dos o más claves candidatas, y en los que podía suceder que estén compuestas y/ o que se traslapen, es decir tengan al menos un atributo en común.
  - Por eso apareció otra FN llamada FNBC, es necesario aclarar que la relación que describimos en el párrafo anterior puede no ser muy frecuente, por eso en relaciones donde no se encuentran esas condiciones la 3FN y la FNBC son equivalentes.
  - La FNBC se define de la siguiente forma: Una relación está en FNBC si y solo si cada determinante es una clave candidata. Teóricamente esta FN es más simple ya que no requiere FN previas.

- Veamos un ejemplo que involucra claves candidatas que no se traslapan. En el esquema de proveedor que definimos antes:

  PROVEEDOR=(id_proveedor, dni_proveedor, calificación, ciudad_proveedor)

  tenemos las siguientes claves candidatas: {id_proveedor}, {dni_proveedor} y las siguientes dependencias funcionales:
- Id_proveedor → dni_proveedor, calificación, ciudad
- dni_proveedor → Id_proveedor, calificación, ciudad
- Entonces el esquema *PROVEEDOR* está en FNBC. En el caso de tener más de una clave candidata el administrador de BD elije una como clave primaria. Pero es necesario especificar ambas claves al SGBD para que éste pueda hacer cumplir con las restricciones de integridad requeridas.

- Veamos ahora un ejemplo donde las claves candidatas se traslapan. Consideremos el siguiente esquema:

  ENVIO3= (id_proveedor, dni_proveedor, id_artículo, cantidad)

  las claves candidatas son {id_proveedor, id_articulo}, {dni_proveedor, id_artículo}
- ¿Este esquema está en FNBC? Veamos las DF:
- Id_proveedor → dni_proveedor
- dni_proveedor → Id_proveedor
- Id_proveedor, id_articulo → cantidad
- dni_proveedor, id-artículo → cantidad
- La respuesta es no, porque? Veamos un ejemplo para este esquema, sea la relación (parcial):

id_proveedor	dni_proveedor	id_artículo	Cantidad
P1	20.134.230	A1	300
P1	20.134.230	A2	200
P1	20.134.230	A3	400
P1	20.134.230	A4	200
P1	20.134.230	A5	100
P1	20.134.230	A6	100
P2	28.987.105	A1	300
…	…	…	…

- Como podemos ver esta relación involucra redundancia, por eso tiene problemas de actualización. Este problema no se hubiera detectado con la 3FN.
- La solución sería dividir esta relación en dos:

id_proveedor	dni_proveedor
P1	20.134.230
P2	28.987.105
…	…

id_proveedor	id_artículo	Cantidad
P1	A1	300
P1	A2	200
P1	A3	400
P1	A4	200
P1	A5	100
P1	A6	100
P2	A1	300
…	…	…

- Y si dejamos el DNI, no el ID?

- Veamos un nuevo ejemplo: Un esquema con los atributos alumno, profesor y materia. Con las siguientes consideraciones, un profesor enseña una sola materia, pero cada materia pueden enseñarla varios profesores. Un alumno que cursa una materia tiene en esa materia un solo profesor.

A	**M**	**P**	**nota**
Jones	Matemática	Soler	8
Jones	Física	Pérez	6
Gómez	Matemática	Soler	4
Gómez	Física	Suárez	9

- Busquemos las DF de esta relación:
- A, M → P, nota
- P → M
- A, P → nota, M
- Las claves candidatas son: {A, M} y {A, P}, por lo que vemos se traslapan. Acá nuevamente tenemos problemas de actualización, si queremos eliminar la información de que Jones está estudiando Física, perderemos también la información de que el profesor Pérez enseña Física. Esto sucede porque entre las DF existe una cuyo determinante no es clave candidata. La relación debe descomponerse en

A	**P**	**nota**
Jones	Soler	8
Jones	Pérez	6
Gómez	Soler	4
Gómez	Suárez	9

P	**M**
Soler	Matemática
Pérez	Física
Suárez	Física

- Estas nuevas relaciones evitan las anomalías que existían pero introducen otras: no son independientes.
- Por ejemplo si queremos introducir una tupla para Jones y el profesor Suárez, en la primer relación, el sistema debería rechazarlo, ya que el profesor Suárez enseña física y a Jones le enseña física el profesor Pérez. Pero el sistema no detecta esto si no analiza la segunda relación. A veces los objetivos de descomponer una relación en FNBC y descomponerla en componentes independientes pueden estar en conflicto.

- Atributos multivaluados
  - Se dice que un atributo es multivaluado cuando para una misma entidad puede tomar varios valores diferentes, con independencia de los valores que puedan tomar el resto de los atributos. Se representa como X →→ Y, y se lee como X multidetermina Y.
  - P. e. sea la relación donde almacenemos contactos y números de teléfono:

    AGENDA(id_contacto, nombre, apellido, teléfono)
  - Para cada id_contacto de la agenda tendremos, en general, varios números de teléfono, id_contacto multidetermina teléfono: id_contacto →→ teléfono.
  - Además, id_contacto determina funcionalmente otros atributos, como nombre y apellido
  - En esta relación tenemos las siguientes dependencias:
  - ✓ id_contacto → apellido
  - ✓ id_contacto → nombre
  - ✓ id_contacto →→ teléfono

– Este tipo de atributos implica redundancia, ya que el resto de los atributos se repiten tantas veces como valores diferentes tenga el atributo multivaluado:

id_contacto	nombre	apellido	teléfono
1	Juan	Pérez	12322132
2	María	Mónaco	13321232
2	María	Mónaco	25565445
2	María	Mónaco	36635363
3	Pedro	Suárez	45665456

– Siempre podemos evitar el problema de los atributos multivaluados separándolos en relaciones distintas. En el ejemplo anterior, podemos crear una segunda relación que contenga el nombre y el teléfono:

✓ AGENDA(Id_contacto, nombre, apellido)

✓ TELÉFONO(Id_contacto, teléfono)

id_contacto	nombre	apellido
1	Juan	Pérez
2	María	Mónaco
3	Pedro	Suárez

id_contacto	teléfono
1	12322132
2	13321232
2	25565445
2	36635363
3	45665456

– Si existe más de un atributo multivaluado es cuando se presentan dependencias multivaluadas.

- Dependencias Multivaluadas
  - Las dependencias multivaluadas son una generalización de las dependencias funcionales. Dada una relación r con atributos A, B y C, decimos que existe una dependencia multivaluada entre A y B en r, si el conjunto de valores de B que concuerdan con el par (A, C) depende solo del valor A pero no del C. Una dependencia multivaluada existe cuando un atributo puede determinar mas de un valor del otro atributo (un conjunto de valores del implicado en lugar de uno solo)
  - Supongamos que en nuestra relación anterior de Agenda añadimos otro atributo para guardar direcciones de correo electrónico. Se trata, por supuesto, de otro atributo multivaluado, ya que cada persona puede tener más de una dirección de correo, y este atributo es independiente del número de teléfono:

    AGENDA(id_contacto, teléfono, correo, nombre, apellido)
  - Ahora surgen los problemas, supongamos que nuestra amiga "María", además de los tres números de teléfono, dispone de dos direcciones de correo. ¿Cómo almacenaremos la información relativa a estos datos?
  - Si optamos por crear tres tuplas, ya que hay tres teléfonos, y emparejar cada dirección con un teléfono, en la tercera tupla estaremos obligados a repetir una dirección. Otra opción sería usar un NULL en esa tercera tupla.

id_contacto	nombre	apellido	teléfono	Correo
2	María	Mónaco	13321232	fulano@sucasa.eko
2	María	Mónaco	25565445	fulano@sutra.aka
2	María	Mónaco	36635363	NULL
…	…	…	…	…

- Pero estas opciones ocultan el hecho de que ambos atributos son multivaluados e independientes entre si. Podría parecer que existe una relación entre los números y las direcciones de correo.

- Ejemplo: En la siguiente relación, un curso puede dictarse por varios profesores y para un curso se utilizan varios textos, los cuales no son elegidos por el profesor sino vienen propuestos por el plan de estudios (es decir los textos son independientes de los profesores). También un texto puede usarse en varios cursos y un profesor puede dictar varios cursos. Veamos una relación con esta información:

Curso	**Profesor**	**Texto**
Física	Pérez	Mecánica Básica
Física	Pérez	Ppios de Óptica
Física	Rosas	Mecánica Básica
Física	Rosas	Ppios de Óptica
Matemática	Pérez	Mecánica Básica
Matemática	Pérez	Análisis Vectorial
Matemática	Pérez	Trigonometría

- Ahora la clave es el conjunto, curso, profesor, texto. En este caso vemos que:
- Curso →→ Profesor y Curso →→ Texto

– A pesar de que la relación está en FNBC ya que no existen dependencias funcionales y los tres atributos son claves, la relación implica gran cantidad de redundancia dando lugar a anomalías de actualización. P. e. para agregar un curso de física que va a estar dictado por un profesor nuevo, hay que agregar dos tuplas una por cada libro del curso de física. Por eso la solución sería dos tablas como las siguientes:

Curso	**Profesor**
Física	Pérez
Física	Rosas
Matemática	Pérez

Curso	**Texto**
Física	Mecánica Básica
Física	Ppios de Óptica
Matemática	Mecánica Básica
Matemática	Análisis Vectorial
Matemática	Trigonometría

- Definición formal de dependencia multivaluada

– Sea r una relación y A y B atributos (o subconjuntos de atributos) de r, decimos que A multidetermina a B, A →→ B, si y sólo si en todo valor válido de r, A y B son atributos tales que un cierto valor de A implica un conjunto bien definido de valores de B, con independencia de los demás atributos de la relación.

- Cuarta Forma Normal
  - El **objetivo** de la cuarta forma normal es eliminar las dependencias multivaluadas.
  - **Definición**: Una relación está en 4NF si y sólo si, en cada dependencia multivaluada X →→ Y no trivial, X es clave candidata. Es decir toda dependencia multivaluada viene determinada por una clave candidata.

  - Una dependencia multivaluada A →→ B es trivial cuando B es parte de A. Esto sucede cuando A es un conjunto de atributos, y B es un subconjunto de A.
  - Tomemos por ejemplo la relación *AGENDA*, pero dejando sólo los atributos multivaluados:

    AGENDA(id_contacto, teléfono, correo)
  - Lo primero que debemos hacer es buscar las claves y las dependencias. Recordemos que las claves candidatas deben identificar de forma unívoca cada tupla.
  - De modo que estamos obligados a usar los tres atributos para formar la clave candidata. Pero las dependencias que tenemos son:
  - ✓ id_contacto →→ teléfono
  - ✓ id_contacto →→ correo
  - Pero id_contacto no es clave candidata de esta relación, debemos separar esta relación en varias (tantas como atributos multivaluados tenga):

    TELÉFONO(id_contacto, teléfono)

    CORREO(id_contacto, correo)
  - Ahora en las dos relaciones se cumple la 4FN.

- En el ejemplo que vimos (curso, profesor, texto), con las dependencias multivaluadas:
- curso →→ profesor
- curso →→ texto
- La relación no se encuentra en 4FN, ya que estas dependencias multivaluadas están implicadas por curso, que no es una clave candidata. (La clave candidata es el conjunto de los tres atributos). Por eso se descompone en las dos proyecciones que vimos:

✓ CURSO1(curso, profesor)

✓ CURSO2(curso, texto)

- Veamos otro ejemplo:

✓ GEOMETRÍA(figura, color, tamaño)

figura	**color**	**tamaño**
Cuadrado	Rojo	Grande
Cuadrado	Azul	Grande
Cuadrado	Azul	Mediano
Círculo	Blanco	Mediano
Círculo	Azul	Pequeño
Círculo	Azul	Mediano

- En este ejemplo la clave *figura* multidetermina el color y el tamaño. Pero tamaño es independiente de color. Es decir que el multideterminante es clave candidata, por lo tanto está en 4FN. Sin embargo implica redundancia, por lo que se puede dividir en dos: (figura, color) y (figura, tamaño)

– Otro ejemplo: Supongamos la situación del primer ejemplo más un atributo adicional: duración que es la cantidad de días que dura el curso

curso	**profesor**	**texto**	**duración**
Física	Pérez	Mecánica Básica	5
Física	Pérez	Ppios de Óptica	5
Física	Rosas	Mecánica Básica	6
Física	Rosas	Ppios de Óptica	4
Matemática	Pérez	Mecánica Básica	4
Matemática	Pérez	Análisis Vectorial	3
Matemática	Pérez	Trigonometría	4

– El conjunto (curso, profesor, texto) es clave candidata y además duración depende funcionalmente de esa clave candidata, pero no está en 4FN ya que curso multidetermina a profesor y a texto, y curso no es clave candidata. La relación mostrada no se puede descomponer.

- Dependencias de Combinación (Unión)
  - Una dependencia de combinación es un tipo de dependencia en la que, p. e. para una relación R compuesta por una serie de subconjuntos de los atributos de R denominados A, B, …, Z, la relación R exhibirá una dependencia de combinación si y sólo si todo valor legal de R es igual a la combinación de sus proyecciones sobre A, B, …, Z
  - Las dependencias funcionales y multivaluadas permiten la descomposición sin pérdida de una relación en dos de sus proyecciones; sin embargo, existen relaciones en las que no se puede llevar a cabo tal descomposición binaria sin pérdida de información. Las dependencias de combinación son para descomposiciones en más de dos relaciones.
  - Ejemplo: Considérese el caso de vendedores que venden ciertos productos de distintas compañías. Los vendedores representan compañías que fabrican productos y venden productos. Supongamos que existe la siguiente restricción: si un vendedor vende un determinado producto y representa a una compañía que produce dicho producto, entonces el vendedor vende el producto de la compañía.

vendedor	**compañía**	**producto**
V1	C1	P1
V1	C2	P1
V2	C1	P1
V2	C1	P2
V2	C2	P1
V2	C2	P2

– En esta relación los tres campos son claves y no tiene dependencias funcionales o multivaluadas por lo que está en 4FN. Se puede descomponer en estas tres tablas:

vendedor	**compañía**
V1	C1
V1	C2
V2	C1
V2	C2

compañía	**producto**
C1	P1
C1	P2
C2	P1
C2	P2

vendedor	**producto**
V1	P1
V2	P1
V2	P2

– Vemos que la relación inicial se obtiene si realizamos un join natural entre las tres tablas menores pero no de dos cualesquiera de ellas. Si realizamos el join entre las dos primeras obtenemos la siguiente tabla:

vendedor	compañía	producto
V1	C1	P1
V1	C1	P2
V1	C2	P1
V1	C2	P2
V2	C1	P1
V2	C1	P2
V2	C2	P1
V2	C2	P2

- Las tuplas sombreadas son falsas, es decir que no corresponden a la relación original. Esto se denomina **pérdida de información**, no se puede reconstruir la relación original con dos relaciones. Pero al realizar un join natural con la tercer tabla si encontramos la relación original.

- Las dependencias de combinación son, al igual que las otras dependencias, una restricción sobre una relación y constituyen una generalización de las anteriores, de forma que siempre una dependencia funcional es también multivaluada y de combinación, y una multivaluada es también de combinación, pero la afirmación inversa no es cierta.

- Surge otra forma normal, la quinta forma normal - **5FN**.

- Quinta Forma Normal
  - Una tabla está en Quinta Forma Normal (**5FN**) o Forma Normal de Proyección-Unión si está en 4FN y las únicas dependencias que existen son las dependencias de unión.
  - Una relación está en 5FN si y sólo si está en 4FN y el contenido de su información no puede ser reconstruido con varias relaciones menores.

  - La 5FN se emplea cuando en una misma tabla tenemos mucha información redundante, con pocos atributos o cuando una tabla posee una gran cantidad de atributos y se hace por ello inmanejable.
  - Para conseguir que una tabla en 4FN con gran cantidad de atributos esté en 5FN, se parte la tabla original en tantas tablas como se desee, teniendo cada una de ellas en común con las demás, los campos que forman la clave primaria en la tabla original.
  - Ejemplo para el caso de una tabla que posee una gran cantidad de atributos:

id	**Datos_familiares**			**Datos_profesionales**			**Datos_personales**			**Datos_clínicos**		
1	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12

  - Si esta relación (*EMPLEADO*) ya está en 4FN, se puede partir en las relaciones *EMPLEADOS_PERSONAL*, *EMPLEADOS_FAMILIA*, *EMPLEADOS_PROFESIONAL*, *EMPLEADOS_CLÍNICOS*. De este modo, la velocidad de acceso y la gestión de datos por cada departamento de la empresa se simplifica, al no tenerse que crear ningún tipo de restricción sobre determinados atributos que no han de ser vistos por el personal que no los necesite.

– El resultado sería:

id	Datos_familiares		
1	D1	D2	D3

id	Datos_profesionales		
1	D4	D5	D6

id	Datos_personales		
1	D7	D8	D9

id	Datos_clínicos		
1	D10	D11	D12

– Ejemplo para el caso de una relación *BIBLIOTECA* que posee mucha información redundante, con pocos atributos:

título	fecha	socio
T1	FT	S1
T2	FU	S2
T3	FV	S1
T4	FG	S4
T1	FH	S3
T2	FT	S4
T3	FV	S3

– Se tiene la relación de préstamos de libros de una biblioteca y existen multitud de registros que se crean diariamente, pero para cada libro o para cada socio habrá pocos registros, con lo que una consulta como: ¿Cuáles son los libros leídos por un determinado socio?, puede tener una velocidad de respuesta elevada.

- Si esta relación se parte en las relaciones *TÍTULO_FECHA*, *TÍTULO_SOCIO* y *SOCIO_FECHA*, cualquier consulta similar a la anterior tendrá un tiempo de respuesta tolerable, y cuando sea necesario, se podrán realizar consultas que impliquen los datos de las tres relaciones.
- El resultado sería:

título	**fecha**
T1	FT
T2	FU
T3	FV
T4	FG
T1	FH
T2	FT
T3	FV

título	**socio**
T1	S1
T2	S2
T3	S1
T4	S4
T1	S3
T2	S4
T3	S3

fecha	**socio**
FT	S1
FU	S2
FV	S1
FG	S4
FH	S3
FT	S4
FV	S3

- La Quinta Forma Normal trata con casos donde la información puede ser reconstruida desde pequeñas piezas de información, las cuales pueden ser mantenidas con menor redundancia. La 2NF, 3NF y 4NF también sirven a ese propósito, pero la 5FN generaliza los casos no contemplados por las anteriores.

- Desnormalización

  - Hasta ahora hemos dado por hecho que es deseable llegar hasta la normalización completa, es decir lograr hasta la 5FN. Sin embargo, en la práctica a menudo se afirma que es necesaria la **desnormalización** para lograr un buen desempeño.

  - La **normalización total** significa muchas veces **relaciones separadas lógicamente**, muchas relaciones separadas significan muchos archivos almacenados que están separados físicamente, y esto implica **muchas operaciones de E/S**.

  - Por eso muchas veces es **suficiente con alcanzar la 3FN** o la **FNBC**.

  - Las ideas de normalización son útiles en el diseño de BD, pero no son la bala de plata.
  - Veamos algunas razones:
  - ✓ La normalización ayuda a hacer cumplir restricciones de integridad de forma sencilla, pero sabemos que las dependencias no son las únicas restricciones que tiene una BD.
  - ✓ La descomposición puede no ser única y hay pocos criterios objetivos para elegir entre descomposiciones alternativas.
  - ✓ En ocasiones pueden haber buenas razones para no normalizar todo.