Capítulo 5

Normalización

Diseño de BD Relacionales

- **Meta**: aprender a hacer **diseños de calidad** de esquemas de BD relacionales, idealmente de manera automática.
- **Repaso**:
  - Un esquema relacional es un conjunto de esquemas de relaciones.
  - Un esquema de relación es una lista de nombres de atributos.

Diseño de BD Relacionales

- **Para tener un diseño de calidad:**
  - Evitar problemas de redundancia de información.
  - Evitar problemas de comprensibilidad.
  - Evitar problemas de incompletitud.
    - Restricciones de integridad incompletas.
    - Relaciones entre atributos no contempladas por esquemas de BD.
  - Evitar problemas de ineficiencia.
    - Chequeo ineficiente de restricciones de integridad.
    - Consultas ineficientes por tener un esquema inadecuado de BD.

Diseño de BD Relacionales

- **Enfoque ya visto para lograr esta meta**
  - Hacer buen *diseño de esquema-ER*.
  - Mapear esquema-ER a un esquema relacional.
  - **Dificultades**:
    - Es un trabajo **manual**
    - **Corregir** esquemas-ER con problemas de calidad.
    - **Tomar decisiones de diseño**:
      - Considerar alternativas suficientes de diseño
      - Elegir entre las alternativas la mejor basándose en **criterios de calidad.**

Diseño de BD Relacionales

- **Problema** ¿Cómo evitar tener que diseñar diagrama ER, tener que decidir entre alternativas de diseño ER, tener que corregir problemas de calidad en diseños ER?

Diseño de BD Relacionales

- **Solución**:
  - Identificar *K* el conjunto de todos los atributos (atómicos) del problema actual.
  - Luego a partir de *K* y para el problema actual definir un conjunto de ***restricciones de integridad*** *I*,
    - que servirán de guía para hacer un buen diseño.
  - Se aplica un algoritmo llamado de **normalización** que:
    - Calcula un esquema de BD relacional a partir de *K* y de *I.*

Diseño de BD Relacionales

Atributos Problema Actual y Restricciones Integridad

Atributos atómicos

Dependencias funcionales

Algoritmo de Normalización

Esquema de BD Relacional de Calidad

Esquema Universal

- **Situación**: para aplicar un algoritmo de normalización necesitamos conocer primero los **atributos** del problema actual.
- Dado el problema actual, el **esquema universal** consta de todos los atributos atómicos del mismo.

Esquema Universal

- **Ejemplo**: sistema de bibliotecas
  - Se tiene un *sistema de bibliotecas* de una ciudad, el sistema está formado por *bibliotecas* de las que se proveen su nombre, y su domicilio formado por calle y número; las bibliotecas tienen libros de los que se almacena su ISBN, su título y sus autores; las bibliotecas llevan un número de inventario para distinguir entre las distintas copias de libros; además las bibliotecas tienen *socios* para los que se almacena nombre, DNI y posición (la misma puede ser egresado, docente, estudiante, etc.); a los socios se les *prestan* ejemplares de libros.
  - **¿Cuál es el esquema universal para este problema?**

Esquema Universal

- **Ejemplo**: sistema de bibliotecas
  - Se tiene un *sistema de bibliotecas* de una ciudad, el sistema está formado por *bibliotecas* de las que se proveen su nombre, y su domicilio formado por calle y número; las bibliotecas tienen libros de los que se almacena su ISBN, su título y sus autores; las bibliotecas llevan un número de inventario para distinguir entre las distintas copias de libros; además las bibliotecas tienen *socios* para los que se almacena nombre, DNI y posición (la misma puede ser egresado, docente, estudiante, etc.); a los socios se les *prestan* ejemplares de libros.
  - SmaBibliotecas = (nombre, DNI, posición, numInv,
    nombreBib, calle, numero, ISBN, título, autores)
  - **¿Cuál es el significado de una tupla de una relación para el esquema universal?**

Esquema Universal

- **Ejemplo**: sistema de bibliotecas
  - Se tiene un *sistema de bibliotecas* de una ciudad, el sistema está formado por *bibliotecas* de las que se proveen su nombre, y su domicilio formado por calle y número; las bibliotecas tienen libros de los que se almacena su ISBN, su título y sus autores; las bibliotecas llevan un número de inventario para distinguir entre las distintas copias de libros; además las bibliotecas tienen *socios* para los que se almacena nombre, DNI y posición (la misma puede ser egresado, docente, estudiante, etc.); a los socios se les *prestan* ejemplares de libros.
  - SmaBibliotecas = (nombre, DNI, posición, numInv,
    nombreBib, calle, numero, ISBN, título, autores)
  - ***Significado:*** Una tupla de información para una relación de ese esquema para un sistema de bibliotecas consiste de: datos sobre una persona (correspondientes a *nombre*, *DNI*, *posición*) datos sobre un libro (correspondientes a *numInv*, *ISBN*, *título*, *autores*), datos sobre una biblioteca (correspondientes a *calle*, *número*, *nomBib*), tal que esa persona es socia de esa biblioteca, y se le ha prestado ese libro que es de esa biblioteca.

Esquema Universal

- Ya que el esquema universal es un esquema de BD relacional,
  - **¿conviene usarlo como diseño de una BD relacional?**

Esquema Universal

- Ya que el esquema universal es un esquema de BD relacional,
  - **¿conviene usarlo como diseño de una BD relacional?**
  - **Respuesta**: **no**, porque el esquema universal usualmente tiene problemas de calidad, como redundancia de información.

Esquema Universal

- Sea el siguiente esquema universal:
  - SmaAutomotor = (DNI, nombre, marca, modelo, patente, numSeguro, compañíaSeguro, direcciónCS)
- ***Problemas de diseño:***
  - **Redundancia de información**: *direcciónCS* aparece repetido para cada coche asegurado por esa compañía de seguros; *nombre* aparece repetido por cada coche que tiene esa persona.
  - **Manejo de valores nulos**: si una persona no tiene coche, aparecen los demás campos en nulo; si una compañía de seguro no tiene coches asegurados, aparecen los demás campos en nulo.
    - Antes de decidir si agregamos valores nulos, tenemos que hacer consultas.
    - P.ej. si a una persona le sacamos coche asegurado, hay que consultar si tiene otros coches asegurados, sino hay que poner nulos.

Esquema Universal

- **Problema*: ¿Cómo se pueden eliminar los problemas citados?***
- **Solución**: ***descomponer*** el esquema universal para eliminar los problemas citados.

Esquema Universal

- **Ejemplo**: Descomponemos el esquema
  - SmaAutomotor = (DNI, nombre, marca, modelo,
    patente, numSeguro, compañíaSeguro, direcciónCS):

  en:
  - Persona = (DNI, nombre)
  - Auto = (marca, modelo, patente)
  - CompañíaAseguradora = (compañíaSeguro, direcciónCS)
  - Seguro = (patente, compañíaSeguro, numSeguro)
  - TieneAuto = (patente, DNI)

Problemas de usar esquema universal

- La **teoría de normalización** estudia cómo descomponer esquemas universales para eliminar los problemas citados.
  - **Implicaciones de usar normalización**:
    - no hace falta ser bueno en modelado o diseño;
    - *pero hay que ser bueno encontrando las restricciones de integridad necesarias (i.e. dependencias funcionales) y el esquema universal.*

Descomposiciones

- **Motivación**: Un algoritmo de normalización va descomponiendo un esquema universal
  - hasta obtener un esquema de BD relacional de calidad.
- Sea R un esquema de relación. Un conjunto de esquemas de relación $\{R_1,\ldots,R_n\}$ es una **descom-posición** de R sii

$$R = R_1 \cup \ldots \cup R_n.$$

Ejemplo

- Sea el siguiente esquema relacional:
  - SmaBibliotecas = (nombre, DNI, posición, numInv,
    nombreBib, calle, numero, ISBN, título, autores)
- **Señalar redundancias de información**

Ejemplo

- Sea el siguiente esquema relacional:
  - SmaBibliotecas = (nombre, DNI, posición, numInv,
    nombreBib, calle, numero, ISBN, título, autores)
- **Podemos identificar la siguientes redundancias de información:**
  - el mismo usuario saca varios libros: se repite nombre y posición varias veces
  - para cada persona que es socia de biblioteca se repite calle y número
  - el mismo libro se saca varias veces prestado y se repite titulo y autores
- **Indicar situaciones que obligan a trabajar con valores nulos**

Ejemplo

- **Situaciones que obligan a trabajar con valores nulos:**
  - si tengo una persona solamente (no es socia, no le prestaron libros, etc.) entonces los campos van ser nulos.
  - lo mismo si tengo libro solamente, los demas campos van a ser nulos.
- **Lo malo de trabajar con valores nulos:**
  - Si borro una tupla de tabla del esquema para eliminar una asociación entre entidades, hay que tener cuidado de no eliminar entidades que deben permanecer.
  - Esto obliga a hacer consulta extra y posible modificación de tupla agregando valores nulos.

Ejemplo

- **Problema**: ¿Cómo arreglar problemas de redundancia de información?
- **Idea**: puedo ir descomponiendo gradualmente el esquema universal de acuerdo a conceptos que sirven para sacar atributos que ocasionan redundancia de información afuera.
- SmaBibliotecas = (nombre, DNI, posición, numInv,
  nombreBib, calle, numero, ISBN, título, autores)
- **¿Cómo eliminar la redundancia: el mismo usuario saca varios libros: se repite nombre y posición varias veces?**

Ejemplo

- **Problema**: ¿Cómo arreglar problemas de redundancia de información?
- **Idea**: puedo ir descomponiendo gradualmente el esquema universal de acuerdo a conceptos que sirven para sacar atributos que ocasionan redundancia de información afuera.
- Consideremos el esquema anterior:

SmaBibliotecas = (nombre, DNI, posición, numInv,
nombreBib, calle, numero, ISBN, título, autores)

- Persona = (nombre, **DNI**, posición)
- SmaBibliotecas1 = (DNI, numInv, nombreBib, calle, numero, ISBN, titulo, autores)
- Saqué de SmaBibliotecas: nombre y posición eliminando la primera redundancia de información.

Ejemplo

- **¿Cómo eliminar la redundancia de información: para cada persona que es socia de biblioteca se repite calle y número?**

- SmaBibliotecas1 = (DNI, numInv, nombreBib, calle, numero, ISBN, titulo, autores)

Ejemplo

- **¿Cómo eliminar la redundancia de información: para cada persona que es socia de biblioteca se repite calle y número?**
- SmaBibliotecas1 = (DNI, numInv, nombreBib, calle, numero, ISBN, titulo, autores)
- Sea el esquema:
- Biblioteca = (**<u>nombreBib</u>**, calle, número)
- SmaBibliotecas2=(DNI, numInv, nombreBib, ISBN, título, autores)
- Saqué de SmaBibliotecas1: calle y número.

Ejemplo

- **¿Cómo eliminar la redundancia de información: el mismo libro se saca varias veces prestado y se repite titulo y autores?**
- SmaBibliotecas2=(DNI, numInv, nombreBib, ISBN, título, autores)

Ejemplo

- **¿Cómo eliminar la redundancia de información: el mismo libro se saca varias veces prestado y se repite titulo y autores?**
- SmaBibliotecas2=(DNI, numInv, nombreBib, ISBN, título, autores)
- Sea el esquema:
- Libro = (**ISBN**, título, autores)
- SmaBibliotecas3 = (DNI, numInv, nombreBib, ISBN)
- Saqué de SmaBibliotecas2: título y autores.

Ejemplo

- Pero sigue el problema de los nulos:
  - Persona en biblioteca sin préstamo de libro: nulo en numInv.
  - Libro en biblioteca pero no hubo préstamo, DNI nulo.
- **Eliminar estos problemas introduciendo otros conceptos (que representan relaciones entre atributos) con sus respectivos esquemas.**
- Divido SmaBibliotecas3 = (DNI, numInv, nombreBib, ISBN) en:
- Socio = (<u>DNI, nombreBib</u>)
- Préstamo = (<u>DNI, nombreBib, numInv</u>)
- CopiaLibro = (<u>nombreBib, numInv</u>, ISBN)
- Así se resuelven los problemas anteriores.
- Pero hay que tener cuidado que al hacer esto no se pierdan relaciones entre atributos del problema.

Observaciones

- **Receta**: arreglar problemas de diseño introduciendo esquemas para conceptos varios que dependen de conocimiento del dominio y que son una descomposición del esquema universal
- Recordar que modelado ER consideraba la introducción de conceptos varios y por lo tanto si se hace bien se obtendrá un buen diseño relacional luego de pasar a tablas.
- La diferencia de modelado ER con lo que hicimos aquí es que hoy analizamos problemas de diseño y los fuimos eliminando uno a uno con conceptos que son introducidos.

Observaciones

- **Meta**: Queremos que la descomposición salga automáticamente y no que dependa de un conocedor del dominio que hace uso inteligente del mismo.
  - Esto se torna muy trabajoso si el esquema universal tiene cientos de atributos, cosa que suele pasar en la vida real.

Ejemplo

- **Ejemplo**: SmaBibliotecas = (nombre, DNI, posición, numInv, nombreBib, calle, numero, ISBN, título, autores)
- Tiene atributos redundantes *nombre* y *posición*
- **El valor de DNI determina unívocamente los valores de *nombre* y *posición*** y lo indicamos de la siguiente forma: DNI -> nombre, posición.
- Usamos DNI -> nombre, posición para descomponer SmaBibliotecas en:
  - SmaBibliotecas1 = (DNI, numInv, nombreBib, calle, numero, ISBN, titulo, autores)
  - (DNI, nombre posición)

Ejemplo

- SmaBibliotecas1 = (DNI, numInv, nombreBib, calle, numero, ISBN, titulo, autores)
- Tiene atributos redundantes *calle* y *número*.
- **¿Qué propiedad podemos usar con esos atributos a la derecha de ->?**

Ejemplo

- SmaBibliotecas1 = (DNI, numInv, nombreBib, calle, numero, ISBN, titulo, autores)
- Tiene atributos redundantes *calle* y *número*.
- nombreBib -> calle, número
- Usamos nombreBib -> calle, número para descomponer *SmaBibliotecas1* en:
  - (<u>nombreBib</u>, calle, número)
  - SmaBibliotecas2=(DNI, numInv, nombreBib, ISBN, título, autores)

Ejemplo

- SmaBibliotecas2=(DNI, numInv, nombreBib, ISBN, título, autores)
- Tiene atributos redundantes título y autores.
- **¿Qué propiedad podemos usar con esos atributos a la derecha de ->?**

Ejemplo

- SmaBibliotecas2=(DNI, numInv, nombreBib, ISBN, título, autores)
- Tiene atributos redundantes título y autores.
- ISBN -> título, autores
- Usamos ISBN -> título, autores para descomponer SmaBibliotecas2:
  - (<u>ISBN</u>, título, autores)
  - SmaBibliotecas3 = (DNI, numInv, nombreBib, ISBN)

Observaciones

- Generalizando, aplicamos los siguientes pasos:
  - El esquema *R* tiene atributos redundantes J.
  - Encontramos propiedad del tipo K -> J.
  - Usamos K -> J para descomponer *R* en: K U J y R1 = R – J.
- A las propiedades del tipo α -> β, con α y β conjuntos de atributos se las llama **dependencias funcionales**.

Observaciones

- **Fijarse que hemos seguido el siguiente procedimiento:**
  - los atributos de la derecha de la DF se sacan del esquema que descompongo y
  - agrego un esquema para los atributos de toda la DF.
- Usualmente puedo ir descomponiendo el esquema universal para eliminar mucha de la redundancia de información en el mismo por medio del uso de dependencias funcionales.
- Más aun, si tenemos las DF del problema, esto es automatizable (debido a la forma repetitiva de descomponer esquemas).

Dependencias Funcionales

- Hay que definir las restricciones de integridad para el *conjunto de tablas legales.*
  - **Tablas legales** son las tablas con las que la empresa quiere poder trabajar.
  - Son tablas donde las tuplas tienen un cierto significado y cumplen con ciertas propiedades obligatorias.
- Las **dependencias funcionales (DF)** requieren que para las tablas legales,
  - el valor de un cierto conjunto de atributos determine unívocamente el valor de otro conjunto de atributos.

Dependencias Funcionales

- **Formalización**:
  - Sea *R* un esquema relacional

$$\alpha \subseteq R \ \ y \ \beta \subseteq R$$

  - La dependencia funcional

$$\alpha \rightarrow \beta$$

se cumple en *R* si y solo si para todas las relaciones legales *r*(R), cada vez que dos tuplas t_1 y t_2 de *r* coinciden en los atributos α, también coinciden en los atributos β. Formalmente:

$$t_1[\alpha] = t_2[\alpha] \ \Rightarrow \ t_1[\beta] = t_2[\beta]$$

Dependencias Funcionales

- **¿Las DF solo pueden ser usadas para eliminar redundancia de información?**
- Ahora veremos con un ejemplo que la respuesta es no.

Dependencias Funcionales

- **Ejemplo**: Sea el esquema universal de un banco.

  Préstamo = (numSucursal, ciudad, activo, numCliente, numPréstamo, importe)

  – Lo descomponemos en:

  SucursalCliente = (numSucursal, ciudad, activo, numCliente)

  ClientePréstamo = (numCliente, numPréstamo, importe)

  – **Evaluación de la descomposición**: Si tenemos un cliente con varios *préstamos* en distintas sucursales:

    - no se puede decir el *préstamo* que pertenece a cada sucursal en la descomposición que tenemos.
    - Luego en la descomposición que tenemos se perdió información con relación al esquema universal.

Dependencias Funcionales

- Aquí se pierde relación entre atributos del problema al descomponer. Esto es algo grave y hay que evitarlo.
- El problema del ejemplo anterior se resuelve así:
- numSucursal -> ciudad, activo
- la uso para descomponer préstamo en:
- Sucursal = (numSucursal, ciudad, activo)
- prest = (numSucursal, numCliente, numPrestamo, importe)
- Nuevamente resolvimos el problema usando una DF para descomponer el universal.

Dependencias Funcionales

- **Observación:** Dado esquema R, hay dependencias funcionales de cierto tipo que implican redundancia de información en R y que se pueden usar para hacer paso automático de descomposición de R.
  - Son dependencias donde los atributos a la izquierda no determinan todos los atributos del esquema R (Los atributos a la derecha son información redundante - por definición de dependencia funcional).
- **Consecuencias**:
  - Las DF ayudan a identificar redundancia de información.
  - Se puede usar una DF para descomponer un esquema con redundancia de información.

Derivando Dependencias Funcionales

- **Problema**: ¿Cómo encontrar las dependencias funcionales (DF) de un problema del mundo real?
- No hace falta listar todas las DF de un problema del mundo real.
- **Ejemplo**: persona = (DNI, nombre, posición)
  - DNI -> nombre, posición
  - Resulta bastante obvio que : DNI-> nombre y DNI -> posición.
  - Estas últimas se pueden derivar de la primera.

Derivando Dependencias Funcionales

- **Ejemplo**: Sea *R* = (A, B, C, G, H, I),
  - *F* = { A→ B, A→ C, B → H }
  - Resulta bastante obvio que también se cumplen:

    A→ H y A → B, C.
  - Estas últimas se pueden derivar o deducir de las de F. Luego no hace falta listarlas.

Derivando Dependencias Funcionales

- **Conclusión 1:** Si F conjunto de DF, hay otras DF fuera de F que también se cumplen y que se pueden derivar de las de F.
- **Conclusión 2:** no necesito dar todas las DF que se cumplen en el problema del mundo real sino un subconjunto de ellas lo menor posible tal que todas las demás DF se puedan derivar de ese subconjunto.

Derivando Dependencias Funcionales

- Necesitamos formalizar qué significa derivar. Para esto consideramos un conjunto de reglas de inferencia.
- Las reglas consideradas se llaman **axiomas de Armstrong:**
  - if $\beta \subseteq \alpha$, then $\alpha \rightarrow \beta$ **(reflexividad)**
  - if $\alpha \rightarrow \beta$, then $\gamma\,\alpha \rightarrow \gamma\,\beta$ **(aumentatividad)**
  - if $\alpha \rightarrow \beta$, and $\beta \rightarrow \gamma$, then $\alpha \rightarrow \gamma$ **(transitividad)**

Derivando Dependencias Funcionales

- **Ejemplo**: queremos deducir $AC \rightarrow D$ a partir de las DF $\{A \rightarrow B; CB \rightarrow D\}$ usando las reglas de Armstromg.
  - **Bosquejar una deducción.**

Derivando Dependencias Funcionales

- **Ejemplo**: queremos deducir $AC \rightarrow D$ a partir de las DF $\{A \rightarrow B; CB \rightarrow D\}$ usando las reglas de Armstromg.
  - Bosquejar una deducción.

1) A->B
2) CB ->D
3) AC -> CB (aumentatividad a 1))
4) AC -> D (transitividad a 3) y 2))

Derivando Dependencias Funcionales

- **Generalizando**: Dado un esquema relacional R una DF f con atributos en R **se deduce** de un conjunto de DFs F con atributos en R si existe una lista de DFs $f_1, \ldots, f_n$ tales que $f_n = f$ y para todo $1 \leq i \leq n$:
  1. $f_i \in F$ ó
  2. f_i se obtiene por aplicar la regla de reflexividad ó
  3. f_i se obtiene por aplicar aumentatividad ó transitivi-dad a pasos anteriores en la lista.
- **Notación**: Usaremos $F \vdash f$ para decir que f se deduce de F.

Derivando Dependencias Funcionales

- Usar solo los axiomas de Armstrong es un poco pesado para hacer derivaciones. Y con algunos axiomas adicionales, muchas derivaciones se tornan más cortas y sencillas.
- Podemos simplificar más las derivaciones a partir de *F,* usando las siguientes reglas adicionales:
  - If $\alpha \rightarrow \beta$ holds *and* $\alpha \rightarrow \gamma$ holds, then $\alpha \rightarrow \beta\gamma$ holds (**union**)
  - If $\alpha \rightarrow \beta\gamma$ holds, then $\alpha \rightarrow \beta$ holds and $\alpha \rightarrow \gamma$ holds (**decomposición**)
  - If $\alpha \rightarrow \beta$ holds *and* $\gamma\beta \rightarrow \delta$ holds, then $\alpha\gamma \rightarrow \delta$ holds (**pseudotransitividad**)

Las reglas anteriores se pueden inferir a partir de los axiomas de Armstrong.

Cierre de conjunto de DF

- El cierre de un conjunto de DF *F* (*F*+) son todas las dependencias que se deducen de *F*.
- **Problema**: tenemos *F* conjunto de DF para problema de mundo real y queremos saber si vale la pena agregar DF *f* a *F*.
- **Idea 1**: calcular *F*+ y ver si *f* está en *F*+
  - Si *f* está en *F*+ entonces no se agrega a *F*; en caso contrario hay que agregarla.
- **¿Es esta idea viable?**

Cierre de conjunto de DF

- **¿Es viable calcular F+ si tenemos muchos atributos en el problema del mundo real?**

- En la práctica hay cientos de atributos en esquema universal y no es viable calcular *F*+.

- Una razón es que para $\boldsymbol{\alpha}$ hay $2^{|\alpha|}$ dependencias triviales.

- Además, si $\boldsymbol{\alpha} \rightarrow \boldsymbol{\beta}$ en F entonces hay $2^{|R|}$ maneras de aplicar aumentatividad a $\boldsymbol{\alpha} \rightarrow \boldsymbol{\beta}$ (R esquema universal).

- Como ven *F*+ es demasiado grande como para poder calcularlo todo.

Cierre de un conjunto de atributos

- **Idea 2**: intentar deducir *f* de *F* y si lo logramos: no se agrega f a F.
- ¿Y si no lo logramos? puede que no sepamos como derivar *f* de *F* o que *f* no sea derivable de *F*.
- Necesitamos saber que f no es deducible de F, pero no sabemos hacer ese tipo de pruebas.
- **Solución** (viable): Quiero saber si $\alpha \rightarrow \beta$ es deducible de F. Si calculamos las dependencias de F+ con lado izquierdo α, este conjunto es mucho más chico que F+ (porque hay 2^n tales α).

Cierre de un conjunto de atributos

- **Por lo tanto**: para responder si $F \vdash \alpha \rightarrow \beta$, bastaría contestar

  $\alpha \rightarrow \beta \in \{\alpha \rightarrow \varphi \mid F \vdash \alpha \rightarrow \varphi\}$ o mejor contestar:

- Pero se puede hacer mejor aún: obtener solo el conjunto de dependencias funcionales α -> A (derivables de F) donde A atributo.

- Esto funciona porque si:

  $$\beta \subseteq \{A \in R \mid F \vdash \alpha \rightarrow A\}$$

  Entonces por regla de unión aplicada finitas veces a todos los atributos de beta sale $F \vdash \alpha \rightarrow \beta$.

- Llegamos así a un conjunto conocido como **cierre de un conjunto de atributos** – es el conjunto a la derecha de la inclusión.

Cierre de un conjunto de atributos

- Sea R el esquema universal, F conjunto de DF del problema del mundo real (con atributos en el universal), sea $\boldsymbol{\alpha} \subseteq$ R. El **cierre de $\boldsymbol{\alpha}$ bajo F** (denotado por α_F^+) se define:

$$\alpha^+_F = \{A \in R : F \vdash \alpha \rightarrow A\}.$$

Cierre de un conjunto de atributos

- **Proposición:** $F \vdash \alpha \rightarrow \alpha^+_F$
- **Prueba**: sale aplicando unión finitas veces .
- **Proposición:** $F \vdash \alpha \rightarrow \beta$ si y solo si $\beta \subseteq \alpha^+_F$
- **Problema**: tenemos F conjunto de DF para problema de mundo real y queremos saber si vale la pena agregar DF $\alpha \rightarrow \beta$ a F.
- **Solución**: chequear si se cumple $F \vdash \alpha \rightarrow \beta$ usando la proposición anterior.
  - Si $\beta \subseteq \alpha^+_F$: la respuesta es sí, por lo tanto no agregar $\alpha \rightarrow \beta$ a F.
  - Sino: $\alpha \rightarrow \beta$ no se deduce de F, por lo tanto agregamos $\alpha \rightarrow \beta$ a F.
  - Por lo tanto, necesitamos un algoritmo para computar α^+_F

Cierre de un conjunto de atributos

- Algoritmo para computar α^+_F (la clausura de α bajo F)

  result := α;
  while (changes to *result*) **do**
      for each $\beta \rightarrow \gamma$ **in** F **do**
      begin
          if $\beta \subseteq$ *result* **then** *result* := *result* $\cup \gamma$
      end

  – Luego de cada asignación a la variable *result* se cumple *result* $\subseteq \alpha^+$.
  – Por lo tanto cuando el algoritmo termina se cumple

  $$result \subseteq \alpha^+ .$$

  – La prueba de que en este momento $\alpha^+ \subseteq$ *result* escapa al alcance de la materia.

Cierre de un conjunto de atributos

- **Ejercicio**: Dados R = (A, B, C, G, H, I),

  $F = \{ A \rightarrow B, A \rightarrow C, CG \rightarrow H, CG \rightarrow I, B \rightarrow H$ Calcular A^{+}_{F} y $(AG)^{+}_{F}$

Cierre de un conjunto de atributos

- Sea R esquema relacional y F es conjunto de DF, entonces α **superclave** de R si y solo si α -> R está en F+.

- **α** clave candidata de R si y solo si:
  - **α** superclave de R
  - para todo A en **α**: **α** - {A} no es superclave de R

- **Para chequear que α superclave de *R, ¿qué podemos hacer?***
  - computar α^{+}, y chequear si α^{+} contiene todos los atributos de *R*.

Cierre de un conjunto de atributos

- Sea R esquema relacional y F es conjunto de DF, entonces α **superclave** de R si y solo si α -> R está en F+.

- **α** clave candidata de R si y solo si:
  - **α** superclave de R
  - para todo A en **α**: **α** - {A} no es superclave de R

- **Para chequear que α superclave de *R:***
  - computar α^+, y chequear si α^+ contiene todos los atributos de *R*.

Cierre de un conjunto de atributos

- **Ejercicio**: Dados R = (A, B, C, G, H, I),

  F = { A → B, A → C, CG → H, CG → I, B → H }.

  1. Probar que de F no se deduce A → I
  2. Probar que AG es clave candidata.

Otros conceptos

- **Definición**: $\alpha \rightarrow \beta$ se cumple en r(R) si para todo $t1 \neq t2$ en r: $t1[\alpha] = t2[\alpha] \Rightarrow t1[\beta] = t2[\beta]$.
- **Ejercicio:** Listar algunas DF que no se cumplen y algunas DF que se cumplen en la siguiente tabla:

A	B	C
a1	b1	c1
a2	a1	c1
a1	b2	c2

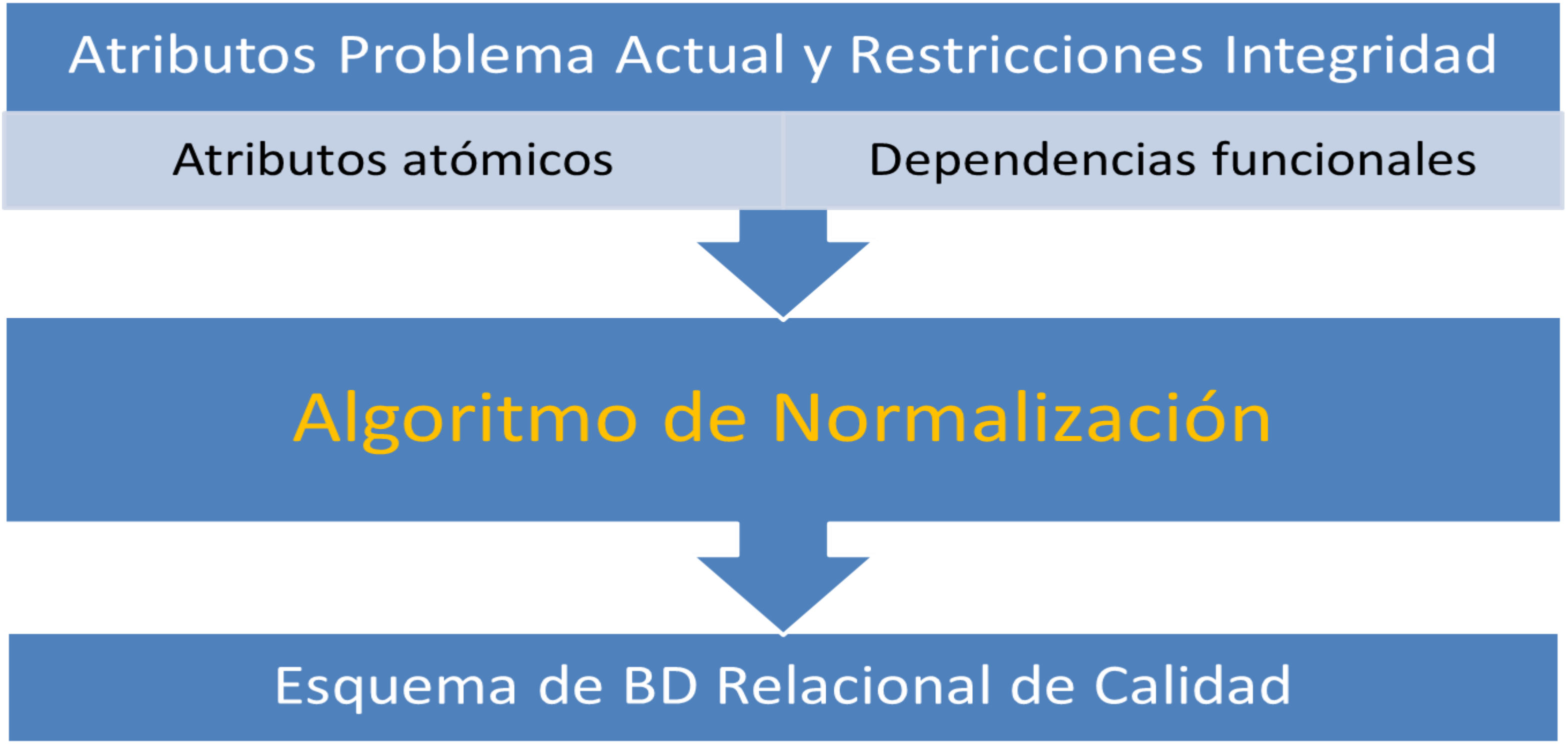
- No confundir las DF que se cumplen en una tabla con las DF de un problema del mundo real para el cual la tabla es legal.
  - El problema suele cumplir menos DF que la tabla.

Otros conceptos

- Una DF **trivial** se cumple en todas las tablas de un esquema.
  - **Ejemplo**: En *SmaBibliotecas*
    - *nombreBib, calle* $\to$ *nombreBib*
    - *calle* $\to$ *calle*
- Después veremos que las DF triviales juegan su papel en los algoritmos de normalización.
- **Ejercicio** (de la práctica): probar la siguiente
  - **Proposición**: $\alpha \to \beta$ es **trivial** si y solo si $\beta \subseteq \alpha$.

Normalización

- **Meta**: aprender a hacer **diseños de calidad** de esquemas de BD relacionales.
- **Solución**: algoritmos de normalización.



Normalización

- Si proveo inputs deficientes, la calidad de la solución calculada se va a ver perjudicada.
- **Aun cuando no aplico algoritmos de normalización necesito tener los inputs.**
    1. Debo considerar los atributos del problema en el diseño.
        - Sino el cliente va a quedar descontento.
    2. Las DF son restricciones de integridad que necesitan ser capturadas para mantener la integridad de la BD.
- **Conclusión**: El aplicar normalización es un bonus por tener los inputs adecuados que **son obligatorios**.

Normalización

- **Repaso**: si *R* esquema con redundancia de información en un conjunto de atributos β y la DF α -> β (α y β son disjuntos) se cumple en *R*:
  - Si α no determina todos los atributos de *R*,
  - la dependencia α -> β puede ser usada para eliminar redundancia de información por medio de la descomposición de *R*:
  - Para eliminar la redundancia de información se saca β de *R* y se crea un esquema con los atributos de α U β.
  - Al hacer esto desaparece la redundancia de información para los atributos de β.

Normalización

- Si hay 2 tuplas distintas de R que coinciden en α y vale $\alpha \rightarrow \beta$ y α no determina todos los atributos de R,
  - entonces para esas tuplas se van a repetir los valores de β; o sea, tenemos redundancia de información para los atributos de β.
- Sea R, F (conjunto de DF). Que α no determina todos los atributos de R es lo mismo que decir:
  - que $\alpha \rightarrow R$ no se deduce de F;
  - o equivalentemente, que α no es superclave de R

Normalización

- **Problema**: Queremos caracterizar un esquema *R* que no tiene redundancia de información proveniente de dependencias funcionales por medio de una propiedad.
  - Un esquema que cumple esa propiedad diremos que está en la forma normal Boyce-Codd (FNBC).

Forma normal de Boyce Codd

- **Solución**: Un esquema R está en **forma normal de Boyce-Codd (FNBC)** con respecto a un conjunto F de DFs si para todas las DFs en F^+ de la forma $\alpha \to \beta$, donde $\alpha \subseteq R$ y $\beta \subseteq R$, al menos una de las siguientes propiedades se cumple:
  - $\alpha \to \beta$ es trivial (i.e., $\beta \subseteq \alpha$)
  - α es una superclave de R (i.e. $\alpha \to R \in F^+$).
- **Definición**: Sea R esquema universal, F conjunto de DFs. Una descomposición $\{R_1,\ldots,R_n\}$ de R está en Forma normal de Boyce-Codd (FNBC) con respecto a F si y solo si cada R_i está en FNBC con respecto a F.

Forma normal de Boyce Codd

- ***¿Cómo comprobar que un esquema R con respecto a F no está en FNBC?***
- Una DF de F^+ que no cumple la condición de FNBC se llama **violación** o **DF testigo**.
  - Es una DF $\alpha \rightarrow \beta$ no trivial en F^+ tal que $\alpha \rightarrow R \notin F^+$
- Para probar que R no está en FNBC con respecto a F basta con encontrar una DF testigo en F^+.
  - A veces (pero no siempre) la DF testigo está en F.

Forma normal de Boyce Codd

- **Ejemplo**: Sea $R = (A, B, C)$ esquema con DFs: $F = \{A \rightarrow B, B \rightarrow C\}$.
  - R no está en FNBC. **¿Por qué?**

Forma normal de Boyce Codd

- **Ejemplo**: Sea $R = (A, B, C)$ esquema con DFs:
  $F = \{A \rightarrow B, B \rightarrow C\}$.
  - R no está en FNBC.
  - $\{A\}$ es clave candidata de R
  - $B \rightarrow C$ es testigo:
    - A no está en B+ = {B, C},
    - B no es superclave de R
  - Sea la descomposición de R: $R_1 = (A, B)$, $R_2 = (B, C)$
  - Esta descomposición está en FNBC

Forma normal de *Boyce* Codd

- **Ejemplo**: Sea el esquema relacional R = (A, C, D) con DFs: F = {A→B, B→C}.
- **¿Está *R* en FNBC?**

Forma normal de *Boyce* Codd

- **Ejemplo**: Sea el esquema relacional *R* = (A, C, D) con DFs: *F* = {A→B, B→C}.
- **¿Está *R* en FNBC?**
- A->B: no se puede usar porque B no está en R. ¡Cuidado con este tipo de error común!
- A->C está en F+ (por transitividad de las DF de F)
- A no superclave de R porque D no está en A+
- Luego R no está en FNBC.

Forma normal de Boyce Codd

- **Proposición**: Para comprobar si *R*, *F* está en FNBC, donde los atributos de *F* están contenidos en *R*, basta con chequear la condición de FNBC para las DF de *F*.
  - Hay que ver que ninguna DF es testigo.
- **Consecuencia**: para descomponer el esquema universal basta con buscar testigos en *F*.
- Si los atributos de *F* no están contenidos en *R* necesitamos otra manera de probar que un esquema está en FNBC.

Forma normal de Boyce Codd

- **Ejemplo**: Sea $R = (A, B, C)$ esquema con DFs:

$$F = \{A \rightarrow B,\ B \rightarrow C,\ C \rightarrow A\}.$$

  - **¿Está *R* en FNBC?**

Forma normal de Boyce Codd

- **Ejemplo**: Sea $R = (A, B, C)$ esquema con DFs:

$$F = \{A \rightarrow B,\ B \rightarrow C,\ C \rightarrow A\}.$$

  - **¿Está *R* en FNBC?**
- Atributos(F) contenidos en R.
- Por proposición anterior basta con chequear dependencias del F.
- A, B y C son superclaves (justificarlo).
- Entonces ningún miembro de F es testigo.
- Por lo tanto R está en FNBC.

Comprobación de Forma normal de Boyce Codd

- **Situación:** lo intentamos y no encontramos una DF testigo.
  - En ese caso intentar probar que tenemos un esquema en FNBC.
  - Chequear todos los $\alpha \rightarrow \beta$ de F+ con atributos en R es demasiado costoso.
- **Comprobación de FNBC**: Sea R_U universal, con DFs F y sea R_i que forma parte de descomposición de R_U; para probar que R_i está en FNBC se puede hacer la siguiente comprobación:
  - $\forall \alpha \subseteq R_i : \alpha^+ \cap (R_i - \alpha) = \phi \lor R_i \subseteq \alpha^+$
- La primera condición del V significa: todas las DF con lado izquierdo α son triviales.

Comprobación de Forma normal de Boyce Codd

- **Prueba**: Supongamos que R_i está en FNBC y $\neg R_i \subseteq \alpha^+$:
  - toda $\alpha \rightarrow \beta$ en F^+ con atributos en R_i es trivial.
  - Esto equivale a $\beta \cap (R_i - \alpha) = \phi$
  - Luego: $\alpha^+ \cap (R_i - \alpha) = \phi$ (tomo $\beta = \alpha^+$)

Comprobación de Forma normal de Boyce Codd

- **Ejercicio**: Sea *F* dado por:
  1. nomBib → calle, numero
  2. calle, numero → nomBib
  3. ISBN → título, editorial, autores, edición
  4. nomBib, numInv → ISBN
- Sean los esquemas:
  - R = (nomBib, numInv, ISBN)
  - Biblioteca = (nomBib, calle, número)
  - Libro = (ISBN, título, editorial, autores, edición)
- Comprobar que *Biblioteca*, *Libro* están en FNBC.

Forma normal de Boyce Codd

- El método de comprobación de FNBC anterior va a ser usada por el algoritmo de normalización.
- **Situación:** la comprobación de FNBC falla para un α.
  - Eso nos permite definir una DF testigo.
  - ***¿Cuál es una dependencia testigo?***
  - Considerar la negación de la comprobación de FNBC

Forma normal de Boyce Codd

- **Observación**: Si $\alpha \subseteq R_i$ viola la condición:

$$\forall\, \alpha \subseteq R_i : \alpha^+ \cap (R_i - \alpha) = \phi \vee R_i \subseteq \alpha^+$$

entonces la siguiente DF es testigo:

$$\alpha \rightarrow \alpha^+ \cap (R_i - \alpha)\ .$$

- Usamos esta dependencia para descomponer Ri.
- Ahora estamos en condiciones para presentar el algoritmo de normalización.

Algoritmo de normalización en FNBC

- **Problema:** Sea R, F conjunto de DFs. ¿Cómo hallar una descomposicion de R que está en FNBC?
- **Solución**: Algoritmo de normalización en FNBC.
- *result* := $\{R\}$;

  while (there is a schema R_i in *result* that is not in BCNF) **do**

  begin

  let $\alpha \rightarrow \beta$ DF testigo de R_i and $\alpha \cap \beta = \emptyset$;

  result := $(result - R_i) \cup (R_i - \beta) \cup (\alpha, \beta)$;

  end

Algoritmo de normalización en FNBC

- Algunas aclaraciones sobre el algoritmo anterior si se **implementa automáticamente**:
  - Para buscar esquema que no está en FNBC se puede usar el algoritmo de comprobación de que esquema está en FNBC.

$$\forall\, \alpha \subseteq R_i : \alpha^+ \cap (R_i - \alpha) = \phi \vee R_i \subseteq \alpha^+$$

  - Ese algoritmo va a encontrar un α que no cumple la condición. Y a partir del mismo se puede obtener la DF testigo:

$$\alpha \rightarrow \alpha^+ \cap (R_i - \alpha)\ .$$

Algoritmo de normalización en FNBC

- **Ejercicio**: Aplicar el algoritmo de normalización en FNBC a:
  - R = (A, B, C, D, E, F)
  - F = {A → CB, E → FA}

Algoritmo de normalización en FNBC

- **Ejercicio**: Sea el esquema universal:

  BibLibs = (nomBib, calle, número, numInv, ISBN, título, editorial, autores, edición)

  Sea *F* dado por:

  – nomBib → calle, número

  – calle, número → nomBib

  – ISBN → título, editorial, autores, edición

  – nomBib, numInv → ISBN

  Aplicar el algoritmo de normalización en FNBC.

Algoritmo de normalización en FNBC

- **¿nomBib -> calle, numero es testigo?**

Algoritmo de normalización en FNBC

- **¿nomBib -> calle, numero es testigo?**
  - Sí porque no es trivial y
  - nomBib+ = {nomBib, calle, numero} <> BibLibs
  - Luego nomBib no es superclave de BibLibs.
- **¿Cómo queda la descomposición de BibLibs?**

Algoritmo de normalización en FNBC

- **¿nomBib -> calle, numero es testigo?**
  - Sí porque no es trivial y
  - nomBib+ = {nomBib, calle, numero} <> R
  - Luego nomBib no es superclave de R.
- **¿Cómo queda la descomposición de BibLibs?**
  - BibLibs2= (nomBib, numInv, ISBN, titulo, editoral, autores, edicion)
  - R1 = (nomBib, calle, numero)
- **¿ISBN → título, editorial, autores, edición es testigo?**

Algoritmo de normalización en FNBC

- **¿nomBib -> calle, numero es testigo?**
  - Sí porque no es trivial y
  - nomBib+ = {nomBib, calle, numero} <> R
  - Luego nomBib no es superclave de R.
- **¿Cómo queda la descomposición de BibLibs?**
  - BibLibs2= (nomBib, numInv, ISBN, titulo, editoral, autores, edicion)
  - R1 = (nomBib, calle, numero)
- **¿ISBN → título, editorial, autores, edición es testigo?**
  - Sí porque no es trivial y
  - ISBN+ = {ISBN, titulo, editoral, edición, autores} que es menor que BibLib2
  - Luego ISBN no es superclave de BibLib2.
- **¿Cómo queda la descomposición de BibLibs2?**

Algoritmo de normalización en FNBC

- BibLibs3 = (nomBib, numInv, ISBN)
- R2 = (ISBN, titulo, editorial, edicion, autores)

- En un ejercicio anterior sale que R1 y R2 están en FNBC.
- Falta ver que BibLib3 está en FNBC.
- Usamos la comprobación: $\forall \alpha \subseteq R_i : \alpha^+ \cap (R_i - \alpha) = \emptyset \vee R_i \subseteq \alpha^+$
- {numInv, nomBib}+ = BibLibs ⊃ BibLibs3
- Luego {numInv, nomBib} superclave y no hace falta chequear superconjuntos.
- {ISBN, numInv} + = R2 U {numInv}, luego no contiene BibLibs3
- {ISBN, numInv}+ ∩ BibLibs3 - {ISBN, numInv} = R2 ∩ {nomBib} = Ø
- {ISBN, nomBib} + = R2 U R1, luego no contiene BibLibs3
- {ISBN, nomBib} + ∩ BibLibs3 - {ISBN, nombib} = (R2 U R1) ∩ {numInv} = Ø

Algoritmo de normalización en FNBC

- nomBib + = {nomBib, calle, numero}
- nomBib + ∩ BibLibs3 - nomBib = {nomBib, calle, numero} ∩ {numInv, ISBN} = ∅.
- numinv + = numinv
- ISBN + = R2   luego ISBN no es superclave de BibLib3
- ISBN + ∩ {nomBib, numInv} = ∅
- Hemos chequeado todos los casos, por lo tanto BibLibs3 está en FNBC.