Diseño de Bases de Datos Relacionales: Dependencia Funcional y Formas Normales

Juan F. Pérez

Departamento MACC
Matemáticas Aplicadas y Ciencias de la Computación
Universidad del Rosario

juanferna.perez@urosario.edu.co

Primer Semestre de 2019

Contenidos

Escogiendo un buen diseño

Alternativas de diseño

- Esquemas grandes: muchos atributos por esquema

Alternativas de diseño

- Esquemas grandes: muchos atributos por esquema
- *estudiantesCursos(estID, curso_cod, nombre, apellido, nombre_curso, nota)*

Alternativas de diseño

- Esquemas grandes: muchos atributos por esquema
- *estudiantesCursos(estID, curso_cod, nombre, apellido, nombre_curso, nota)*
- Ventaja: consultas se pueden escribir con menos joins

Alternativas de diseño

- Esquemas grandes: muchos atributos por esquema
- *estudiantesCursos(estID, curso_cod, nombre, apellido, nombre_curso, nota)*
- Ventaja: consultas se pueden escribir con menos joins
- Desventaja: los datos del estudiante y el curso (diferente a los códigos usados como llave primaria) deben ser consistentes en todos los registros (difícil de mantener)

Alternativas de diseño

- Esquemas grandes: muchos atributos por esquema
- *estudiantesCursos(estID, curso_cod, nombre, apellido, nombre_curso, nota)*
- Ventaja: consultas se pueden escribir con menos joins
- Desventaja: los datos del estudiante y el curso (diferente a los códigos usados como llave primaria) deben ser consistentes en todos los registros (difícil de mantener)
- Desventaja: imposible crear un estudiante sin inmediatamente asignarle un curso (y viceversa) a menos que se usen nulos

Alternativas de diseño

- Esquemas pequeños: pocos atributos por esquema

Alternativas de diseño

- Esquemas pequeños: pocos atributos por esquema
- *estudiantesCursos(estID, curso_cod, nombre, apellido, nombre_curso, nota)*

Alternativas de diseño

- Esquemas pequeños: pocos atributos por esquema
- *estudiantesCursos(estID, curso_cod, nombre, apellido, nombre_curso, nota)*
- *estudiantes(estID, nombre, apellido)*

Alternativas de diseño

- Esquemas pequeños: pocos atributos por esquema
- *estudiantesCursos(estID, curso_cod, nombre, apellido, nombre_curso, nota)*
- *estudiantes(estID, nombre, apellido)*
- *cursos(curso_cod, nombre_curso, nota)*

Alternativas de diseño

- Esquemas pequeños: pocos atributos por esquema
- *estudiantesCursos(estID, curso_cod, nombre, apellido, nombre_curso, nota)*
- *estudiantes(estID, nombre, apellido)*
- *cursos(curso_cod, nombre_curso, nota)*
- *estudiantesCursos(estID, curso_cod, nota)*

Alternativas de diseño

- ¿Cómo identificar división adecuada?

Alternativas de diseño

- ¿Cómo identificar división adecuada?
- ¿Qué campos tienen valores que se repiten y corresponden al mismo "objeto"?

Alternativas de diseño

- ¿Cómo identificar división adecuada?
- ¿Qué campos tienen valores que se repiten y corresponden al mismo "objeto"?
- Es necesario hacer explícitas que cada estudiante tiene exactamente un nombre y un apellido, o cada curso tiene un nombre

Alternativas de diseño

- ¿Cómo identificar división adecuada?
- ¿Qué campos tienen valores que se repiten y corresponden al mismo "objeto"?
- Es necesario hacer explícitas que cada estudiante tiene exactamente un nombre y un apellido, o cada curso tiene un nombre
- En el esquema *estudiantes(estID, nombre, apellido)* , estID es una llave primaria (no como en el caso inicial de *estudiantesCursos*)

Alternativas de diseño

- ¿Cómo identificar división adecuada?
- ¿Qué campos tienen valores que se repiten y corresponden al mismo "objeto"?
- Es necesario hacer explícitas que cada estudiante tiene exactamente un nombre y un apellido, o cada curso tiene un nombre
- En el esquema *estudiantes(estID, nombre, apellido)* , estID es una llave primaria (no como en el caso inicial de *estudiantesCursos*)
- Dependencia funcional: estID $\rightarrow$ nombre

Alternativas de diseño

- **Dependencia funcional**: estID $\rightarrow$ nombre

Alternativas de diseño

- **Dependencia funcional**: estID $\rightarrow$ nombre
- Para todo par de registros, si el valor del atributo a la izquierda es igual, el valor del atributo a la derecha debe ser igual

Alternativas de diseño

- **Dependencia funcional**: estID $\rightarrow$ nombre
- Para todo par de registros, si el valor del atributo a la izquierda es igual, el valor del atributo a la derecha debe ser igual
- Problema con el esquema inicial *estudiantesCursos*: como estID no puede ser llave primaria, es necesario repetir el nombre del estudiante

Alternativas de diseño

- **Dependencia funcional**: estID $\rightarrow$ nombre
- Para todo par de registros, si el valor del atributo a la izquierda es igual, el valor del atributo a la derecha debe ser igual
- Problema con el esquema inicial *estudiantesCursos*: como estID no puede ser llave primaria, es necesario repetir el nombre del estudiante
- ¿Cómo descomponer un esquema?

Alternativas de diseño

- **Dependencia funcional**: estID $\rightarrow$ nombre
- Para todo par de registros, si el valor del atributo a la izquierda es igual, el valor del atributo a la derecha debe ser igual
- Problema con el esquema inicial *estudiantesCursos*: como estID no puede ser llave primaria, es necesario repetir el nombre del estudiante
- ¿Cómo descomponer un esquema?
- Descomposición con pérdida de información (lossy): al descomponer se pierde información del esquema "grande" (e.g., no es posible obtener el esquema grande/original usando un join)

Alternativas de diseño

- **Dependencia funcional**: estID $\rightarrow$ nombre
- Para todo par de registros, si el valor del atributo a la izquierda es igual, el valor del atributo a la derecha debe ser igual
- Problema con el esquema inicial *estudiantesCursos*: como estID no puede ser llave primaria, es necesario repetir el nombre del estudiante
- ¿Cómo descomponer un esquema?
- Descomposición con pérdida de información (lossy): al descomponer se pierde información del esquema "grande" (e.g., no es posible obtener el esquema grande/original usando un join)
- Descomposición con pérdida de información (lossless): no se pierde

Primera Forma Normal

Primera Forma Normal

- El dominio de un atributo es atómico si los elementos del dominio son indivisibles

Primera Forma Normal

- El dominio de un atributo es atómico si los elementos del dominio son indivisibles
- Ej. atributos no atómicos: nombres compuestos, direcciones compuestas, mezcla de códigos en uno solo

Primera Forma Normal

- El dominio de un atributo es atómico si los elementos del dominio son indivisibles
- Ej. atributos no atómicos: nombres compuestos, direcciones compuestas, mezcla de códigos en uno solo

Primera forma normal

Un esquema relacional se considera en la primera forma normal (1NF) si los dominios de todos sus atributos son atómicos

Descomposición y Segunda Forma Normal (Boyce-Codd)

Relaciones y llaves

- Una relación $r(R)$ con conjunto de atributos R

Relaciones y llaves

- Una relación $r(R)$ con conjunto de atributos R
- Un subconjunto de atributos $K \subset R$ es una superllave si es posible identificar cada registro en la relación con los atributos en K

Relaciones y llaves

- Una relación $r(R)$ con conjunto de atributos R
- Un subconjunto de atributos $K \subset R$ es una superllave si es posible identificar cada registro en la relación con los atributos en K
- Dependencia funcional: $K \rightarrow R$

Relaciones y llaves

- Una relación $r(R)$ con conjunto de atributos R
- Un subconjunto de atributos $K \subset R$ es una superllave si es posible identificar cada registro en la relación con los atributos en K
- Dependencia funcional: $K \rightarrow R$
- Llave candidata: superllave que no tiene como subconjunto otra superllave (superllave mínima)

Relaciones y llaves

- Una relación $r(R)$ con conjunto de atributos R
- Un subconjunto de atributos $K \subset R$ es una superllave si es posible identificar cada registro en la relación con los atributos en K
- Dependencia funcional: $K \rightarrow R$
- Llave candidata: superllave que no tiene como subconjunto otra superllave (superllave mínima)
- Llave primaria: llave candidata seleccionada para identificar de manera única cada elemento en la relación

Relaciones y dependencias funcionales

- Una instancia de una relación $r(R)$ satisface la dependencia funcional $\alpha \rightarrow \beta$ si para todo par de tuplas t_1, t_2 se cumple que si $t_1[\alpha] = t_2[\alpha]$, entonces $t_1[\beta] = t_2[\beta]$

Relaciones y dependencias funcionales

- Una instancia de una relación $r(R)$ satisface la dependencia funcional $\alpha \rightarrow \beta$ si para todo par de tuplas t_1, t_2 se cumple que si $t_1[\alpha] = t_2[\alpha]$, entonces $t_1[\beta] = t_2[\beta]$
- La dependencia funcional $\alpha \rightarrow \beta$ se cumple si todo par de registros que tengan el mismo valor en los atributos α, tienen el mismo valor en los atributos β

Relaciones y dependencias funcionales

- Una instancia de una relación $r(R)$ satisface la dependencia funcional $\alpha \rightarrow \beta$ si para todo par de tuplas t_1, t_2 se cumple que si $t_1[\alpha] = t_2[\alpha]$, entonces $t_1[\beta] = t_2[\beta]$
- La dependencia funcional $\alpha \rightarrow \beta$ se cumple si todo par de registros que tengan el mismo valor en los atributos α, tienen el mismo valor en los atributos β
- Un esquema $r(R)$ satisface la dependencia funcional $\alpha \rightarrow \beta$ si toda instancia legal del mismo satisface la dependencia

Relaciones y dependencias funcionales

- *estudiantes(<u>estID</u>, nombre, apellido)*

Relaciones y dependencias funcionales

- *estudiantes(estID, nombre, apellido)*
- Se cumple la dependencia estID $\rightarrow$ nombre, apellido

Relaciones y dependencias funcionales

- *estudiantes(estID, nombre, apellido)*
- Se cumple la dependencia estID $\rightarrow$ nombre, apellido
- *cursos(curso_cod, nombre_curso, nota)*

Relaciones y dependencias funcionales

- *estudiantes(estID, nombre, apellido)*
- Se cumple la dependencia estID $\rightarrow$ nombre, apellido
- *cursos(curso_cod, nombre_curso, nota)*
- Se cumple la dependencia curso_cod $\rightarrow$ nombre_curso, nota

Relaciones y dependencias funcionales

- Una dependencia funciona $\alpha \rightarrow \beta$ se considera trivial si $\beta \subset \alpha$

Relaciones y dependencias funcionales

- Una dependencia funciona $\alpha \rightarrow \beta$ se considera trivial si $\beta \subset \alpha$
- Siempre se cumple

Relaciones y dependencias funcionales

- Una dependencia funciona $\alpha \rightarrow \beta$ se considera trivial si $\beta \subset \alpha$
- Siempre se cumple
- Si para un esquema $r(A, B, C)$ se cumple $A \rightarrow B$ y $B \rightarrow C$, entonces $A \rightarrow C$

Relaciones y dependencias funcionales

- Una dependencia funciona $\alpha \rightarrow \beta$ se considera trivial si $\beta \subset \alpha$
- Siempre se cumple
- Si para un esquema $r(A, B, C)$ se cumple $A \rightarrow B$ y $B \rightarrow C$, entonces $A \rightarrow C$
- Dado un conjunto de dependencia funcionales F, F^+ denota la clausura de F, es decir, el conjunto de todas las dependencias funcionales que se pueden inferir a partir de F

Segunda Forma Normal (Boyce-Codd)

- Elimina todas las redundancias que pueden descubrirse a partir de dependencias funcionales

Segunda Forma Normal (Boyce-Codd)

- Elimina todas las redundancias que pueden descubrirse a partir de dependencias funcionales

Segunda Forma Normal (Boyce-Codd)

Un esquema relacional $r(R)$ se considera en la segunda forma normal (BCNF) con respecto al conjunto de dependencias funcionales F si para todas las dependencias funcionales $\alpha \rightarrow \beta$ en F^+, se cumple una de las dos condiciones siguientes:

En toda dependencia $\alpha \rightarrow \beta$ no trivial α es una superllave

Segunda Forma Normal (Boyce-Codd)

- Elimina todas las redundancias que pueden descubrirse a partir de dependencias funcionales

Segunda Forma Normal (Boyce-Codd)

Un esquema relacional $r(R)$ se considera en la segunda forma normal (BCNF) con respecto al conjunto de dependencias funcionales F si para todas las dependencias funcionales $\alpha \rightarrow \beta$ en F^+, se cumple una de las dos condiciones siguientes:

En toda dependencia $\alpha \rightarrow \beta$ no trivial α es una superllave

Segunda Forma Normal (Boyce-Codd)

- Elimina todas las redundancias que pueden descubrirse a partir de dependencias funcionales

Segunda Forma Normal (Boyce-Codd)

Un esquema relacional $r(R)$ se considera en la segunda forma normal (BCNF) con respecto al conjunto de dependencias funcionales F si para todas las dependencias funcionales $\alpha \rightarrow \beta$ en F^+, se cumple una de las dos condiciones siguientes:

- $\alpha \rightarrow \beta$ es trivial
- α es una superllave de R

En toda dependencia $\alpha \rightarrow \beta$ no trivial α es una superllave

- *estudiantes(<u>estID</u>, nombre, apellido)*

- *estudiantes(estID, nombre, apellido)*
- Está en segunda forma normal

- *estudiantes(estID, nombre, apellido)*
- Está en segunda forma normal
- *cursos(curso_cod, nombre_curso, nota)*

- *estudiantes(estID, nombre, apellido)*
- Está en segunda forma normal
- *cursos(curso_cod, nombre_curso, nota)*
- Está en segunda forma normal

- *estudiantes(estID, nombre, apellido)*
- Está en segunda forma normal
- *cursos(curso_cod, nombre_curso, nota)*
- Está en segunda forma normal
- *estudiantesCursos(estID, curso_cod, nombre, apellido, nombre_curso, nota)*

- *estudiantes(estID, nombre, apellido)*
- Está en segunda forma normal
- *cursos(curso_cod, nombre_curso, nota)*
- Está en segunda forma normal
- *estudiantesCursos(estID, curso_cod, nombre, apellido, nombre_curso, nota)*
- estID $\rightarrow$ nombre, apellido

- *estudiantes(estID, nombre, apellido)*
- Está en segunda forma normal
- *cursos(curso_cod, nombre_curso, nota)*
- Está en segunda forma normal
- *estudiantesCursos(estID, curso_cod, nombre, apellido, nombre_curso, nota)*
- estID $\rightarrow$ nombre, apellido
- **NO** está en segunda forma normal

Sea $r(R)$ un esquema que no está en la Segunda Forma Normal:

- Existe al menos una dependencia $\alpha \rightarrow \beta$ tal que α no es una superllave

Sea $r(R)$ un esquema que no está en la Segunda Forma Normal:

- Existe al menos una dependencia $\alpha \rightarrow \beta$ tal que α no es una superllave
- Reemplace $r(R)$ con dos esquemas:

Sea $r(R)$ un esquema que no está en la Segunda Forma Normal:

- Existe al menos una dependencia $\alpha \rightarrow \beta$ tal que α no es una superllave
- Reemplace $r(R)$ con dos esquemas:
  - $\alpha \cup \beta$

Sea $r(R)$ un esquema que no está en la Segunda Forma Normal:

- Existe al menos una dependencia $\alpha \rightarrow \beta$ tal que α no es una superllave
- Reemplace $r(R)$ con dos esquemas:
  - $\alpha \cup \beta$
  - $R - (\beta - \alpha)$

Ejemplo:

- *estudiantesCursos(estID, curso_cod, nombre, apellido, nombre_curso, nota)*

Ejemplo:

- *estudiantesCursos(estID, curso_cod, nombre, apellido, nombre_curso, nota)*
- estID $\rightarrow$ nombre, apellido

Ejemplo:

- *estudiantesCursos(estID, curso_cod, nombre, apellido, nombre_curso, nota)*
- estID $\rightarrow$ nombre, apellido
  - α: estID

Ejemplo:

- *estudiantesCursos(estID, curso_cod, nombre, apellido, nombre_curso, nota)*
- estID $\rightarrow$ nombre, apellido
  - α: estID
  - β: nombre, apellido

Ejemplo:

- *estudiantesCursos(estID, curso_cod, nombre, apellido, nombre_curso, nota)*
- estID $\rightarrow$ nombre, apellido
  - α: estID
  - β: nombre, apellido
  - $\alpha \cup \beta$: estID, nombre, apellido

Ejemplo:

- *estudiantesCursos(estID, curso_cod, nombre, apellido, nombre_curso, nota)*
- estID $\rightarrow$ nombre, apellido
  - α: estID
  - β: nombre, apellido
  - $\alpha \cup \beta$: estID, nombre, apellido
  - $\beta - \alpha$: nombre, apellido

Ejemplo:

- *estudiantesCursos(estID, curso_cod, nombre, apellido, nombre_curso, nota)*
- estID $\rightarrow$ nombre, apellido
  - α: estID
  - β: nombre, apellido
  - $\alpha \cup \beta$: estID, nombre, apellido
  - $\beta - \alpha$: nombre, apellido
  - $R - (\beta - \alpha)$: curso_cod, estID, nombre_curso, nota

Ejemplo:

- *estudiantesCursos(estID, curso_cod, nombre, apellido, nombre_curso, nota)*
- estID $\rightarrow$ nombre, apellido
  - α: estID
  - β: nombre, apellido
  - $\alpha \cup \beta$: estID, nombre, apellido
  - $\beta - \alpha$: nombre, apellido
  - $R - (\beta - \alpha)$: curso_cod, estID, nombre_curso, nota
- curso_cod $\rightarrow$ nombre_curso

Ejemplo:

- *estudiantesCursos(estID, curso_cod, nombre, apellido, nombre_curso, nota)*
- estID $\rightarrow$ nombre, apellido
  - α: estID
  - β: nombre, apellido
  - $\alpha \cup \beta$: estID, nombre, apellido
  - $\beta - \alpha$: nombre, apellido
  - $R - (\beta - \alpha)$: curso_cod, estID, nombre_curso, nota
- curso_cod $\rightarrow$ nombre_curso
- Continuar la descomposición hasta que todos los esquemas estén en la segunda forma normal

Tercera Forma Normal

Tercera Forma Normal

- Similar a la segunda pero más flexible

Tercera Forma Normal

- Similar a la segunda pero más flexible

Tercera Forma Normal

Un esquema relacional $r(R)$ se considera en la tercera forma normal (3NF) con respecto al conjunto de dependencias funcionales F si para todas las dependencias funcionales $\alpha \rightarrow \beta$ en F^+, se cumple una de las tres condiciones siguientes:

Tercera Forma Normal

- Similar a la segunda pero más flexible

Tercera Forma Normal

Un esquema relacional $r(R)$ se considera en la tercera forma normal (3NF) con respecto al conjunto de dependencias funcionales F si para todas las dependencias funcionales $\alpha \rightarrow \beta$ en F^+, se cumple una de las tres condiciones siguientes:

- $\alpha \rightarrow \beta$ es trivial
- α es una superllave de R
- Cada atributo $A \in \beta - \alpha$ está contenido en una llave candidata para $r(R)$

Tercera Forma Normal

- Todo esquema en BCNF está también 3NF

Tercera Forma Normal

- Todo esquema en BCNF está también 3NF
- Comparada con BCNF, evita descomponer algunas relaciones

Tercera Forma Normal

- Todo esquema en BCNF está también 3NF
- Comparada con BCNF, evita descomponer algunas relaciones
- Sin esa descomposición, algunas dependencias funcionales pueden ser revisadas más eficientemente

Ejemplo:

- *asesorUnidad(estID, instID, nombre_unidad)*

Ejemplo:

- *asesorUnidad(estID, instID, nombre_unidad)*
- Todo instructor debe estar asociado a un único departamento

Ejemplo:

- *asesorUnidad(estID, instID, nombre_unidad)*
- Todo instructor debe estar asociado a un único departamento
  - instID $\rightarrow$ nombre_unidad

Ejemplo:

- *asesorUnidad(estID, instID, nombre_unidad)*
- Todo instructor debe estar asociado a un único departamento
  - instID $\rightarrow$ nombre_unidad
- Todo estudiante puede tener a lo sumo un asesor en cada unidad académica

Ejemplo:

- *asesorUnidad(estID, instID, nombre_unidad)*
- Todo instructor debe estar asociado a un único departamento
  - instID $\rightarrow$ nombre_unidad
- Todo estudiante puede tener a lo sumo un asesor en cada unidad académica
  - estID, nombre_unidad $\rightarrow$ instID

Ejemplo:

- *asesorUnidad(estID, instID, nombre_unidad)*
- Todo instructor debe estar asociado a un único departamento
  - instID $\rightarrow$ nombre_unidad
- Todo estudiante puede tener a lo sumo un asesor en cada unidad académica
  - estID, nombre_unidad $\rightarrow$ instID
- ¿Está en BCNF? No

Ejemplo:

- *asesorUnidad(estID, instID, nombre_unidad)*
- Todo instructor debe estar asociado a un único departamento
  - instID $\rightarrow$ nombre_unidad
- Todo estudiante puede tener a lo sumo un asesor en cada unidad académica
  - estID, nombre_unidad $\rightarrow$ instID
- ¿Está en BCNF? No
  - instID no es una súperllave
  - (estID, nombre_unidad) no es una súperllave

Ejemplo:

- *asesorUnidad(estID, instID, nombre_unidad)*
- Todo instructor debe estar asociado a un único departamento
  - instID $\rightarrow$ nombre_unidad
- Todo estudiante puede tener a lo sumo un asesor en cada unidad académica
  - estID, nombre_unidad $\rightarrow$ instID
- ¿Está en BCNF? No
  - instID no es una súperllave
  - (estID, nombre_unidad) no es una súperllave
- ¿Está en 3NF? Sí

Ejemplo:

- *asesorUnidad(estID, instID, nombre_unidad)*
- Todo instructor debe estar asociado a un único departamento
  - instID $\rightarrow$ nombre_unidad
- Todo estudiante puede tener a lo sumo un asesor en cada unidad académica
  - estID, nombre_unidad $\rightarrow$ instID
- ¿Está en BCNF? No
  - instID no es una súperllave
  - (estID, nombre_unidad) no es una súperllave
- ¿Está en 3NF? Sí
  - nombre_unidad - instID = nombre_unidad es parte de una llave candidata

Ejemplo:

- *asesorUnidad(estID, instID, nombre_unidad)*
- Todo instructor debe estar asociado a un único departamento
  - instID $\rightarrow$ nombre_unidad
- Todo estudiante puede tener a lo sumo un asesor en cada unidad académica
  - estID, nombre_unidad $\rightarrow$ instID
- ¿Está en BCNF? No
  - instID no es una súperllave
  - (estID, nombre_unidad) no es una súperllave
- ¿Está en 3NF? Sí
  - nombre_unidad - instID = nombre_unidad es parte de una llave candidata
  - instID - (estID, nombre_unidad) = instID es parte de una llave candidata

Ejemplo:

- *asesorUnidad(estID, instID, nombre_unidad)*

Ejemplo:

- *asesorUnidad(estID, instID, nombre_unidad)*
- Todo instructor debe estar asociado a un único departamento

Ejemplo:

- *asesorUnidad(estID, instID, nombre_unidad)*
- Todo instructor debe estar asociado a un único departamento
  - instID $\rightarrow$ nombre_unidad

Ejemplo:

- *asesorUnidad(estID, instID, nombre_unidad)*
- Todo instructor debe estar asociado a un único departamento
  - instID $\rightarrow$ nombre_unidad
- Todo estudiante puede tener a lo sumo un asesor en cada unidad académica

Ejemplo:

- *asesorUnidad(estID, instID, nombre_unidad)*
- Todo instructor debe estar asociado a un único departamento
  - instID $\rightarrow$ nombre_unidad
- Todo estudiante puede tener a lo sumo un asesor en cada unidad académica
  - estID, nombre_unidad $\rightarrow$ instID

Ejemplo:

- *asesorUnidad(estID, instID, nombre_unidad)*
- Todo instructor debe estar asociado a un único departamento
  - instID $\rightarrow$ nombre_unidad
- Todo estudiante puede tener a lo sumo un asesor en cada unidad académica
  - estID, nombre_unidad $\rightarrow$ instID
- Para garantizar BCNF necesitaríamos descomponer en dos esquemas

Ejemplo:

- *asesorUnidad(estID, instID, nombre_unidad)*
- Todo instructor debe estar asociado a un único departamento
  - instID $\rightarrow$ nombre_unidad
- Todo estudiante puede tener a lo sumo un asesor en cada unidad académica
  - estID, nombre_unidad $\rightarrow$ instID
- Para garantizar BCNF necesitaríamos descomponer en dos esquemas
  - (estID, instID)
  - (instID, nombre_unidad)

Ejemplo:

- *asesorUnidad(estID, instID, nombre_unidad)*
- Todo instructor debe estar asociado a un único departamento
  - instID $\rightarrow$ nombre_unidad
- Todo estudiante puede tener a lo sumo un asesor en cada unidad académica
  - estID, nombre_unidad $\rightarrow$ instID
- Para garantizar BCNF necesitaríamos descomponer en dos esquemas
  - (estID, instID)
  - (instID, nombre_unidad)
- Chequear la condición *estID, nombre_unidad* $\rightarrow$ *instID*

Ejemplo:

- *asesorUnidad(estID, instID, nombre_unidad)*
- Todo instructor debe estar asociado a un único departamento
  - instID $\rightarrow$ nombre_unidad
- Todo estudiante puede tener a lo sumo un asesor en cada unidad académica
  - estID, nombre_unidad $\rightarrow$ instID
- Para garantizar BCNF necesitaríamos descomponer en dos esquemas
  - (estID, instID)
  - (instID, nombre_unidad)
- Chequear la condición *estID, nombre_unidad* $\rightarrow$ *instID*
- Requiere combinar varias tablas: menos eficiente que mantener un solo esquema

Teoría de Dependencias Funcionales

Axiomas de Armstrong

- Sea $r(R)$ un esquema con atributos R

Axiomas de Armstrong

- Sea $r(R)$ un esquema con atributos R
- Denotamos subconjuntos de R con letras griegas α, β, γ

Axiomas de Armstrong

- Sea $r(R)$ un esquema con atributos R
- Denotamos subconjuntos de R con letras griegas α, β, γ

Axiomas de Armstrong

Axiomas de Armstrong

- Sea $r(R)$ un esquema con atributos R
- Denotamos subconjuntos de R con letras griegas α, β, γ

Axiomas de Armstrong

- **[Reflexividad]** Sean $\alpha, \beta \subset R$ con $\beta \subset \alpha$, entonces $\alpha \rightarrow \beta$

Axiomas de Armstrong

- Sea $r(R)$ un esquema con atributos R
- Denotamos subconjuntos de R con letras griegas α, β, γ

Axiomas de Armstrong

- **[Reflexividad]** Sean $\alpha, \beta \subset R$ con $\beta \subset \alpha$, entonces $\alpha \rightarrow \beta$
- **[Aumentación]** Sean $\alpha, \beta, \gamma \subset R$. Si $\alpha \rightarrow \beta$, entonces $\alpha \cup \gamma \rightarrow \beta \cup \gamma$

Axiomas de Armstrong

- Sea $r(R)$ un esquema con atributos R
- Denotamos subconjuntos de R con letras griegas α, β, γ

Axiomas de Armstrong

- **[Reflexividad]** Sean $\alpha, \beta \subset R$ con $\beta \subset \alpha$, entonces $\alpha \rightarrow \beta$
- **[Aumentación]** Sean $\alpha, \beta, \gamma \subset R$. Si $\alpha \rightarrow \beta$, entonces $\alpha \cup \gamma \rightarrow \beta \cup \gamma$
- **[Transitividad]** Sean $\alpha, \beta, \gamma \subset R$. Si $\alpha \rightarrow \beta$ y $\beta \rightarrow \gamma$, entonces $\alpha \rightarrow \gamma$

Axiomas de Armstrong

- Sea $r(R)$ un esquema con atributos R
- Denotamos subconjuntos de R con letras griegas α, β, γ

Axiomas de Armstrong

- **[Reflexividad]** Sean $\alpha, \beta \subset R$ con $\beta \subset \alpha$, entonces $\alpha \rightarrow \beta$
- **[Aumentación]** Sean $\alpha, \beta, \gamma \subset R$. Si $\alpha \rightarrow \beta$, entonces $\alpha \cup \gamma \rightarrow \beta \cup \gamma$
- **[Transitividad]** Sean $\alpha, \beta, \gamma \subset R$. Si $\alpha \rightarrow \beta$ y $\beta \rightarrow \gamma$, entonces $\alpha \rightarrow \gamma$
- Completos y sólidos

Axiomas de Armstrong

- Sea $r(R)$ un esquema con atributos R
- Denotamos subconjuntos de R con letras griegas α, β, γ

Axiomas de Armstrong

- **[Reflexividad]** Sean $\alpha, \beta \subset R$ con $\beta \subset \alpha$, entonces $\alpha \rightarrow \beta$
- **[Aumentación]** Sean $\alpha, \beta, \gamma \subset R$. Si $\alpha \rightarrow \beta$, entonces $\alpha \cup \gamma \rightarrow \beta \cup \gamma$
- **[Transitividad]** Sean $\alpha, \beta, \gamma \subset R$. Si $\alpha \rightarrow \beta$ y $\beta \rightarrow \gamma$, entonces $\alpha \rightarrow \gamma$
- Completos y sólidos
- A partir de F, generar su clausura F^+ aplicando estos axiomas repetidamente

Axiomas de Armstrong

- Otras reglas que se pueden derivar de los axiomas de Armstrong

Axiomas de Armstrong

- Otras reglas que se pueden derivar de los axiomas de Armstrong
- Simplifican el cálculo de F^+

Axiomas de Armstrong

- Otras reglas que se pueden derivar de los axiomas de Armstrong
- Simplifican el cálculo de F^+

Axiomas de Armstrong

- Otras reglas que se pueden derivar de los axiomas de Armstrong
- Simplifican el cálculo de F^+
- **[Unión]** Sean $\alpha, \beta \subset R$. Si $\alpha \rightarrow \beta$ y $\alpha \rightarrow \gamma$, entonces $\alpha \rightarrow \beta \cup \gamma$

Axiomas de Armstrong

- Otras reglas que se pueden derivar de los axiomas de Armstrong
- Simplifican el cálculo de F^+

- **[Unión]** Sean $\alpha, \beta \subset R$. Si $\alpha \rightarrow \beta$ y $\alpha \rightarrow \gamma$, entonces $\alpha \rightarrow \beta \cup \gamma$
- **[Descomposición]** Sean $\alpha, \beta, \gamma \subset R$. Si $\alpha \rightarrow \beta \cup \gamma$, entonces $\alpha \rightarrow \beta$ y $\alpha \rightarrow \gamma$

Axiomas de Armstrong

- Otras reglas que se pueden derivar de los axiomas de Armstrong
- Simplifican el cálculo de F^+

- **[Unión]** Sean $\alpha, \beta \subset R$. Si $\alpha \to \beta$ y $\alpha \to \gamma$, entonces $\alpha \to \beta \cup \gamma$
- **[Descomposición]** Sean $\alpha, \beta, \gamma \subset R$. Si $\alpha \to \beta \cup \gamma$, entonces $\alpha \to \beta$ y $\alpha \to \gamma$
- **[Pseudo-transitividad]** Sean $\alpha, \beta, \gamma, \delta \subset R$. Si $\alpha \to \beta$ y $\gamma \cup \beta \to \delta$, entonces $\alpha \cup \gamma \to \delta$

Axiomas de Armstrong

Ejemplo:

- $R = (A, B, C, D, E, F)$

Axiomas de Armstrong

Ejemplo:

- $R = (A, B, C, D, E, F)$
- $F = \{A \rightarrow B, A \rightarrow C, C \cup D \rightarrow E, C \cup D \rightarrow F, B \rightarrow E\}$

Axiomas de Armstrong

Ejemplo:

- $R = (A, B, C, D, E, F)$
- $F = \{A \rightarrow B, A \rightarrow C, C \cup D \rightarrow E, C \cup D \rightarrow F, B \rightarrow E\}$
- $A \rightarrow E$ (transitividad, $A \rightarrow B, B \rightarrow E$)

Axiomas de Armstrong

Ejemplo:

- $R = (A, B, C, D, E, F)$
- $F = \{A \to B, A \to C, C \cup D \to E, C \cup D \to F, B \to E\}$
- $A \to E$ (transitividad, $A \to B, B \to E$)
- $C \cup D \to E \cup F$ (union, $C \cup D \to E, C \cup D \to F$)

Axiomas de Armstrong

Ejemplo:

- $R = (A, B, C, D, E, F)$
- $F = \{A \to B, A \to C, C \cup D \to E, C \cup D \to F, B \to E\}$
- $A \to E$ (transitividad, $A \to B, B \to E$)
- $C \cup D \to E \cup F$ (union, $C \cup D \to E, C \cup D \to F$)
- $A \cup D \to F$ (pseudo-transitividad, $A \to C, C \cup D \to F$)

Axiomas de Armstrong

function CALCULA $F^+(F)$
  $F^+ = F$
  repeat
    for $f \in F^+$ **do**
      Aplique reflexividad y aumentación a f
      Agregue las dependencias funcionales resultantes a F^+
    end for
    for pareja $(f_1, f_2) \in F^+$ **do**
      if f_1 y f_2 se pueden combinar con transitividad **then**
        Agregue la dependencia funcional resultante a F^+
      end if
    end for
  until F^+ no cambie
  return F^+
end function

Calculando la clausura de un conjunto de atributos

Calculando la clausura de un conjunto de atributos

- Un atributo B está funcionalmente determinado por α si $\alpha \rightarrow B$

Calculando la clausura de un conjunto de atributos

- Un atributo B está funcionalmente determinado por α si $\alpha \rightarrow B$
- El conjunto de atributos determinado por α es la clausura de α y se denota α^+

Calculando la clausura de un conjunto de atributos

- Un atributo B está funcionalmente determinado por α si $\alpha \rightarrow B$
- El conjunto de atributos determinado por α es la clausura de α y se denota α^+
- Podemos probar si α es una superllave, verificando que $\alpha^+ = R$

Calculando la clausura de un conjunto de atributos

function CLAUSURA DE α BAJO $F(\alpha, F)$
    $S = \alpha$
    repeat
        for $f : \beta \to \gamma \in F$ **do**
            if $\beta \in S$ **then**
                $S = S \cup \gamma$
            end if
        end for
    until S no cambie
    return S
end function

Calculando la clausura de un conjunto de atributos

También podemos usar este algoritmo para verificar si la dependencia funcional $\alpha \rightarrow \beta$ es válida bajo F (i.e., si $\alpha \rightarrow \beta \in F^+$):

- Calculamos α^+

Calculando la clausura de un conjunto de atributos

También podemos usar este algoritmo para verificar si la dependencia funcional $\alpha \rightarrow \beta$ es válida bajo F (i.e., si $\alpha \rightarrow \beta \in F^+$):

- Calculamos α^+
- Si $\beta \subset \alpha^+$, entonces $\alpha \rightarrow \beta$ es válida bajo F

Atributos extraños y cobertura mínima

Atributos extraños

Dado un conjunto F de dependencias funcionales y una dependencia funcional $\alpha \rightarrow \beta \in F$,

Atributos extraños

Dado un conjunto F de dependencias funcionales y una dependencia funcional $\alpha \rightarrow \beta \in F$,

- Un atributo $A \in \alpha$ es extraño si F implica $(F - \{\alpha \rightarrow \beta\}) \cup \{(\alpha - A) \rightarrow \beta\}$

Atributos extraños

Dado un conjunto F de dependencias funcionales y una dependencia funcional $\alpha \rightarrow \beta \in F$,

- Un atributo $A \in \alpha$ es extraño si F implica $(F - \{\alpha \rightarrow \beta\}) \cup \{(\alpha - A) \rightarrow \beta\}$
- Un atributo $A \in \beta$ es extraño si el conjunto $(F - \{\alpha \rightarrow \beta\}) \cup \{\alpha \rightarrow (\beta - A)\}$ implica a F

Atributos extraños

Dado un conjunto F de dependencias funcionales y una dependencia funcional $\alpha \to \beta \in F$,

- Un atributo $A \in \alpha$ es extraño si F implica $(F - \{\alpha \to \beta\}) \cup \{(\alpha - A) \to \beta\}$
- Un atributo $A \in \beta$ es extraño si el conjunto $(F - \{\alpha \to \beta\}) \cup \{\alpha \to (\beta - A)\}$ implica a F
- Podemos quitar a A (de α o de β) sin afectar la clausura de F

Atributos extraños

Determinar si un atributo es extraño:

- Dado un conjunto F de dependencias funcionales, una dependencia funcional $\alpha \rightarrow \beta \in F$, y un atributo A,

Atributos extraños

Determinar si un atributo es extraño:

- Dado un conjunto F de dependencias funcionales, una dependencia funcional $\alpha \rightarrow \beta \in F$, y un atributo A,
- Si $A \in \beta$ defina

$$F' = (F - \{\alpha \rightarrow \beta\}) \cup \{\alpha \rightarrow (\beta - A)\},$$

calcule α^+ bajo F'. Si $A \in \alpha^+$, entonces A es extraño en β ($\alpha \rightarrow A$ se cumple).

Atributos extraños

Determinar si un atributo es extraño:

- Dado un conjunto F de dependencias funcionales, una dependencia funcional $\alpha \rightarrow \beta \in F$, y un atributo A,
- Si $A \in \beta$ defina

$$F' = (F - \{\alpha \rightarrow \beta\}) \cup \{\alpha \rightarrow (\beta - A)\},$$

  calcule α^+ bajo F'. Si $A \in \alpha^+$, entonces A es extraño en β ($\alpha \rightarrow A$ se cumple).
- Si $A \in \alpha$ defina $\gamma = \alpha - A$, calcule γ^+ bajo F. Si $\beta \subset \gamma^+$, entonces A es extraño en α ($\gamma \rightarrow \beta$ se puede inferir de F).

Cobertura mínima

Una cobertura mínima F_c para F es un conjunto de dependencias funcionales tal que:

Cobertura mínima

Una cobertura mínima F_c para F es un conjunto de dependencias funcionales tal que:

- F implica F_c

Cobertura mínima

Una cobertura mínima F_c para F es un conjunto de dependencias funcionales tal que:

- F implica F_c
- F_c implica F

Cobertura mínima

Una cobertura mínima F_c para F es un conjunto de dependencias funcionales tal que:

- F implica F_c
- F_c implica F
- Ninguna dependencia en F_c contiene atributos extraños

Cobertura mínima

Una cobertura mínima F_c para F es un conjunto de dependencias funcionales tal que:

- F implica F_c
- F_c implica F
- Ninguna dependencia en F_c contiene atributos extraños
- El lado izquierdo de las dependencias en F_c es único. Es decir, no existen dos dependencias $\alpha_1 \rightarrow \beta_1, \alpha_2 \rightarrow \beta_2 \in F_c$ tal que $\alpha_1 = \alpha_2$

Cobertura mínima

Una cobertura mínima F_c para F tiene las siguientes propiedades:

Cobertura mínima

Una cobertura mínima F_c para F tiene las siguientes propiedades:

- F_c tiene la misma clausura de F

Cobertura mínima

Una cobertura mínima F_c para F tiene las siguientes propiedades:

- F_c tiene la misma clausura de F
- F_c es mínima en el sentido de que no tiene atributos extraños y combina todas las dependencias con el mismo lado izquierdo

Cobertura mínima

Una cobertura mínima F_c para F tiene las siguientes propiedades:

- F_c tiene la misma clausura de F
- F_c es mínima en el sentido de que no tiene atributos extraños y combina todas las dependencias con el mismo lado izquierdo
- F_c es más conveniente para hacer pruebas que F

Calculando la cobertura mínima

function COBERTURA MÍNIMA(F)
    $F_c = F$
    repeat
        Reemplazar todo par de dependencias en F_c de la forma $\alpha_1 \to \beta_1$ y $\alpha_1 \to \beta_2$ con una dependencia $\alpha_1 \to \beta_1 \cup \beta_2$ (unión)
        Buscar dependencias funcionales $\alpha \to \beta \in F_c$ con atributos extraños en α o en β
        if $A \in \alpha$ es extraño en $\alpha \to \beta \in F_c$ **then**
            Reemplazar $\alpha \to \beta$ por $(\alpha - A) \to \beta$ en F_c
        else if $A \in \beta$ es extraño en $\alpha \to \beta \in F_c$ **then**
            Reemplazar $\alpha \to \beta$ por $\alpha \to (\beta - A)$ en F_c
        end if
    until F_c no cambie
    return F_c
end function

Descomposición sin pérdidas

Descomposición sin pérdidas

Una descomposición del conjunto ed atributos R en R_1 y R_2 no tiene pérdidas si al menos una de las siguientes dependencias se cumple:

Descomposición sin pérdidas

Una descomposición del conjunto ed atributos R en R_1 y R_2 no tiene pérdidas si al menos una de las siguientes dependencias se cumple:

- $R_1 \cap R_2 \rightarrow R_1$

Descomposición sin pérdidas

Una descomposición del conjunto ed atributos R en R_1 y R_2 no tiene pérdidas si al menos una de las siguientes dependencias se cumple:

- $R_1 \cap R_2 \rightarrow R_1$
- $R_1 \cap R_2 \rightarrow R_2$

Descomposición sin pérdidas

Una descomposición del conjunto ed atributos R en R_1 y R_2 no tiene pérdidas si al menos una de las siguientes dependencias se cumple:

- $R_1 \cap R_2 \rightarrow R_1$
- $R_1 \cap R_2 \rightarrow R_2$
- Es decir, si $R_1 \cap R_2$ es una superllave para R_1 o para R_2

Normalización de BDs usando la Formal Normal Boyce-Codd (BCNF)

Segunda Forma Normal (Boyce-Codd)

Segunda Forma Normal (Boyce-Codd)

Segunda Forma Normal (Boyce-Codd)

Un esquema relacional $r(R)$ se considera en la segunda forma normal (BCNF) con respecto al conjunto de dependencias funcionales F si para todas las dependencias funcionales $\alpha \rightarrow \beta$ en F^+, se cumple una de las dos condiciones siguientes:

- $\alpha \rightarrow \beta$ es trivial
- α es una superllave de R

En toda dependencia $\alpha \rightarrow \beta$ no trivial α es una superllave

Determinar si un esquema está en BCNF (sin calcular F^+)

Sea F el conjunto de dependencias funcionales definidas para una BDs y $r(R)$ un esquema relacional en la BD

Determinar si un esquema está en BCNF (sin calcular F^+)

Sea F el conjunto de dependencias funcionales definidas para una BDs y $r(R)$ un esquema relacional en la BD

function PRUEBA BCNF($(r(R), F)$)
    for all $\alpha \subset R$ **do**
        Calcule α^+ bajo F
        if α^+ no incluye atributos en $R - \alpha$ **then**
            Toda dependencia del tipo $\alpha \rightarrow \beta$ es trivial
        else if $\alpha^+ = R$ **then**
            α es una superllave para R
        else
            El esquema no está en BCNF con respecto a la dependencia $\alpha \rightarrow (\alpha^+ - \alpha) \cap R$
        end if
    end for
end function

Normalizar una Base de Datos en BCNF

Sea F el conjunto de dependencias funcionales definidas para una BDs y $S = \{R_1, \ldots, R_n\}$ los esquemas relacionales presentes en la BD

Normalizar una Base de Datos en BCNF

Sea F el conjunto de dependencias funcionales definidas para una BDs y $S = \{R_1, \ldots, R_n\}$ los esquemas relacionales presentes en la BD

function NORMALIZAR BCNF($S = \{R_1, \ldots, R_n\}, F$))
  repeat
    for all $R_i \in S$ **do**
      if R_i no está en BCNF con respecto a $\alpha \rightarrow \beta$ y $\alpha \cap \beta = \emptyset$
then
        $S = (S - R_i) \cup (R_i - \beta) \cup (\alpha, \beta)$
      end if
    end for
  until S no cambie
  return S
end function

Normalización de BDs usando la Tercera Formal Normal (3NF)

Preservación de dependencias

- Sea F un conjunto dependencias funcionales en el esquema R y sea $\{R_1, \ldots, R_n\}$ una descomposición de R

Preservación de dependencias

- Sea F un conjunto dependencias funcionales en el esquema R y sea $\{R_1, \ldots, R_n\}$ una descomposición de R
- F_i: la restricción F a R_i (dependencias en F^+ que solo incluyen dependencias en R_i)

Preservación de dependencias

- Sea F un conjunto dependencias funcionales en el esquema R y sea $\{R_1, \ldots, R_n\}$ una descomposición de R
- F_i: la restricción F a R_i (dependencias en F^+ que solo incluyen dependencias en R_i)
- F_i: pueden revisarse eficientemente (un esquema)

Preservación de dependencias

- Sea F un conjunto dependencias funcionales en el esquema R y sea $\{R_1, \ldots, R_n\}$ una descomposición de R
- F_i: la restricción F a R_i (dependencias en F^+ que solo incluyen dependencias en R_i)
- F_i: pueden revisarse eficientemente (un esquema)
- $F' = F_1 \cup \cdots \cup F_n$: todas las dependencias que pueden revisarse eficientemente

Preservación de dependencias

- Sea F un conjunto dependencias funcionales en el esquema R y sea $\{R_1, \ldots, R_n\}$ una descomposición de R
- F_i: la restricción F a R_i (dependencias en F^+ que solo incluyen dependencias en R_i)
- F_i: pueden revisarse eficientemente (un esquema)
- $F' = F_1 \cup \cdots \cup F_n$: todas las dependencias que pueden revisarse eficientemente

Descomposición que preserva dependencias

$\{R_1, \ldots, R_n\}$ es una descomposición que preserva dependencias si

$$F'^+ = F^+$$

Preservación de dependencias

- Sea F un conjunto dependencias funcionales en el esquema R y sea $\{R_1, \ldots, R_n\}$ una descomposición de R

Preservación de dependencias

- Sea F un conjunto dependencias funcionales en el esquema R y sea $\{R_1, \ldots, R_n\}$ una descomposición de R
- Sea $\alpha \rightarrow \beta \in F$

Preservación de dependencias

- Sea F un conjunto dependencias funcionales en el esquema R y sea $\{R_1, \ldots, R_n\}$ una descomposición de R
- Sea $\alpha \rightarrow \beta \in F$
- Si $\alpha \rightarrow \beta \in F'^{+}$, la descomposición preserva la dependencia $\alpha \rightarrow \beta$

Preservación de dependencias

- Sea F un conjunto dependencias funcionales en el esquema R y sea $\{R_1, \ldots, R_n\}$ una descomposición de R
- Sea $\alpha \rightarrow \beta \in F$
- Si $\alpha \rightarrow \beta \in F'^{+}$, la descomposición preserva la dependencia $\alpha \rightarrow \beta$
- Para saber si $\alpha \rightarrow \beta \in F'^{+}$, calculamos la clausura de α bajo F'.

Preservación de dependencias

- Sea F un conjunto dependencias funcionales en el esquema R y sea $\{R_1, \ldots, R_n\}$ una descomposición de R
- Sea $\alpha \rightarrow \beta \in F$
- Si $\alpha \rightarrow \beta \in F'^{+}$, la descomposición preserva la dependencia $\alpha \rightarrow \beta$
- Para saber si $\alpha \rightarrow \beta \in F'^{+}$, calculamos la clausura de α bajo F'.
- Si β está en la clausura de α bajo F', $\alpha \rightarrow \beta \in F'^{+}$, y la descomposición preserva la dependencia $\alpha \rightarrow \beta$

Calcular la clausura de α bajo F'

function CLAUSURA DE α BAJO $F'((\alpha, S = \{R_1, \ldots, R_n\}))$
    result= α
    repeat
        for all $R_i \in S$ **do**
            $t = (\text{result} \cap R_i)^+$ $capR_i$
            result = result $\cap$ t
        end for
    until result no cambie
    return result
end function