Normalización

Autor: Sergio D'Arrigo

¿Cualquier diseño es bueno?

- Al diseñar, se trata de lograr esquemas que:
  - Reflejen la información modelada y la realidad
  - Permitan un eficiente mantenimiento y recuperación de la información almacenada
  - No tengan riesgos de perder información ni recuperar información incorrecta.

- Algunos **lineamientos generales** para lograr buenos diseños son:
  - Utilizar **semánticas claras** para los atributos de las relaciones
  - **Reducir** (en lo posible evitar) la **información redundante** en las tuplas
  - **Reducir** la ocurrencia de **valores nulos** en las tuplas
  - **No permitir** la posibilidad de generar **tuplas espurias**

Utilizar semánticas claras para los atributos de las relaciones

- Mientras más clara sea la semántica de las relaciones y sus atributos, mejor será el diseño de la relación.
- Los nombres deben ser declarativos y representativos de la semántica. En caso de Claves Foráneas, debe quedar bien identificado el rol y la interrelación que representan.
- Al realizar el diseño lógico, **evitar combinar atributos de múltiples entidades e interrelaciones en una única relación**.
  - ✍ *En general, siguiendo los lineamientos vistos para Diseño Conceptual y Diseño Lógico, se generan buenos diseños de relaciones.*

Reducir / evitar la información redundante en las tuplas

- La información redundante en las tuplas genera inconvenientes:
  - **Aumento del espacio de almacenamiento**
- Ej

ALUMNO_CARRERA

DNI	Nombre	Apelido	Teléfono	Cod_Carrera	Carrera	Pabellón
43432	Juan	A		F	Física	1
5244	Maria	B	11 2233	M	Matematica	1
4524	Pedro	C	22 3344	C	Computación	0
123754	Alex	D	33 4455	B	Biologia	2
6534	Iris	E	44 5566	Q	Quimica	2
635634	Cris	F	66 7788	D	Datos	0
987	Marcel	G	77 8899	C	Computación	0
4544	Carle	H	99 0011	C	Computación	0

- ¿Estamos ocupando el espacio de manera eficiente?

Por cada tupla que tenga carreras ya existentes, ocuparemos espacio innecesario

Reducir / evitar la información redundante en las tuplas

- La información redundante en las tuplas genera inconvenientes:
  - Aumento del espacio de almacenamiento
  - **Anomalías de Inserción**
- Ej

ALUMNO_CARRERA

DNI	Nombre	Apelido	Teléfono	Cod_Carrera	Carrera	Pabellón
43432	Juan	A		F	Física	1
5244	Maria	B	11 2233	M	Matematica	1
4524	Pedro	C	22 3344	C	Computación	0
123754	Alex	D	33 4455	B	Biologia	2
6534	Iris	E	44 5566	Q	Quimica	2
635634	Cris	F	66 7788	D	Datos	0
987	Marcel	G	77 8899	C	Computación	0
4544	Carle	H	99 0011	C	Computación	0

- ¿Qué pasa si queremos insertar un nuevo alumno de la carrera de Datos?
- Deberíamos registrar los mismos valores de Carrera y Pabellón que en los registros existentes, si no la relación quedaría inconsistente (asumiendo que cada carrera está en un único Pabellón)

Reducir / evitar la información redundante en las tuplas

- La información redundante en las tuplas genera inconvenientes:
  - Aumento del espacio de almacenamiento
  - **Anomalías de Inserción**
- Ej

ALUMNO_CARRERA

DNI	Nombre	Apelido	Teléfono	Cod_Carrera	Carrera	Pabellón
43432	Juan	A		F	Física	1
5244	Maria	B	11 2233	M	Matematica	1
4524	Pedro	C	22 3344	C	Computación	0
123754	Alex	D	33 4455	B	Biologia	2
6534	Iris	E	44 5566	Q	Quimica	2
635634	Cris	F	66 7788	D	Datos	0
987	Marcel	G	77 8899	C	Computación	0
4544	Carle	H	99 0011	C	Computación	0

- ¿Qué pasa si queremos insertar una nueva carrera aún sin alumnos?
- No podríamos, porque la PK de esta relación es DNI, que es un dato de alumno, y no puede tomar valor nulo.

Reducir / evitar la información redundante en las tuplas

- La información redundante en las tuplas genera inconvenientes:
  - Aumento del espacio de almacenamiento
  - Anomalías de Inserción
  - **Anomalías de Actualización**
- Ej

ALUMNO_CARRERA

DNI	Nombre	Apelido	Teléfono	Cod_Carrera	Carrera	Pabellón
43432	Juan	A		F	Física	1
5244	Maria	B	11 2233	M	Matematica	1
4524	Pedro	C	22 3344	C	Computación	0
123754	Alex	D	33 4455	B	Biologia	2
6534	Iris	E	44 5566	Q	Quimica	2
635634	Cris	F	66 7788	D	Datos	0
987	Marcel	G	77 8899	C	Computación	0
4544	Carle	H	99 0011	C	Computación	0

- ¿Qué pasa si quisiéramos cambiar el pabellón de la carrera de Computación?
- Deberíamos registrar ese cambio en todas las tuplas en las que aparece el Depto. Computación

Reducir / evitar la información redundante en las tuplas

- La información redundante en las tuplas genera inconvenientes:
  - Aumento del espacio de almacenamiento
  - Anomalías de Inserción
  - Anomalías de Actualización
  - **Anomalías de Eliminación**
- Ej

ALUMNO_CARRERA

DNI	Nombre	Apelido	Teléfono	Cod_Carrera	Carrera	Pabellón
43432	Juan	A		F	Física	1
5244	Maria	B	11 2233	M	Matematica	1
4524	Pedro	C	22 3344	C	Computación	0
123754	Alex	D	33 4455	B	Biologia	2
6534	Iris	E	44 5566	Q	Quimica	2
635634	Cris	F	66 7788	D	Datos	0
987	Marcel	G	77 8899	C	Computación	0
4544	Carle	H	99 0011	C	Computación	0

- ¿Qué pasa si quiero eliminar la tupla del alumno 43432?
- Dado que es el único que tiene carrera Física, perderíamos la información de la existencia de dicha carrera.

Reducir / evitar la información redundante en las tuplas

- La información redundante en las tuplas genera inconvenientes:
  - Aumento del espacio de almacenamiento
  - Anomalías de Inserción
  - Anomalías de Actualización
  - Anomalías de Eliminación

- **Es fundamental evitar las redundancias** y las anomalías mencionadas en la relaciones.
- **Si llegan a existir, explicitarlas** para que puedan tener un tratamiento correcto en las aplicaciones.

- ¿por qué podría necesitarse en algunas ocasiones tener redundancia en un sistema transaccional?

Reducir la ocurrencia de valores nulos en las tuplas

- Si una relación tiene muchos atributos que no aplican a todas las tuplas, encontraremos muchos valores nulos.
- La existencia de **valores nulos** en las tuplas ocasiona algunas dificultades:
  - Potencial espacio de **almacenamiento inutilizado**
  - Dificultad para la **interpretación** de su significado.
  - Afectación de las **condiciones de junta**, resultados esperados no siempre intuitivos.
  - Tratamiento particular de los valores nulos en los **conteos y agregaciones**
  - Lógica de **booleana de 3 valores** para las comparaciones

- **Evitar incluir en una relación atributos que frecuentemente toman valores nulos**. En muchos casos podría estar faltando generar una nueva relación (por ej , una especialización)

No permitir la posibilidad de generar tuplas espurias

- Las **tuplas espurias** se generan cuando, a partir de un diseño dado de los esquemas de relación, al realizar juntas se pueden obtener **combinaciones de valores que no corresponden** al dominio de problema.
- Ej

Diseño alternativo a ALUMNO_CARRERA

CARRERA

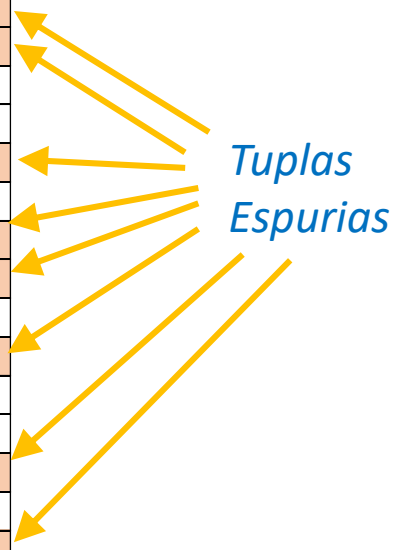
Cod_Carrera	Carrera	Pabellón
F	Física	1
M	Matematica	1
C	Computación	0
B	Biologia	2
Q	Quimica	2
D	Datos	0

ALUMNO_PABELLON

DNI	Nombre	Apelido	Teléfono	Pabellón
43432	Juan	A		1
5244	Maria	B	11 2233	1
4524	Pedro	C	22 3344	0
123754	Alex	D	33 4455	2
6534	Iris	E	44 5566	2
635634	Cris	F	66 7788	0
987	Marcel	G	77 8899	0
4544	Carle	H	99 0011	0

ALUMNO_PABELLON ⋈ CARRERA

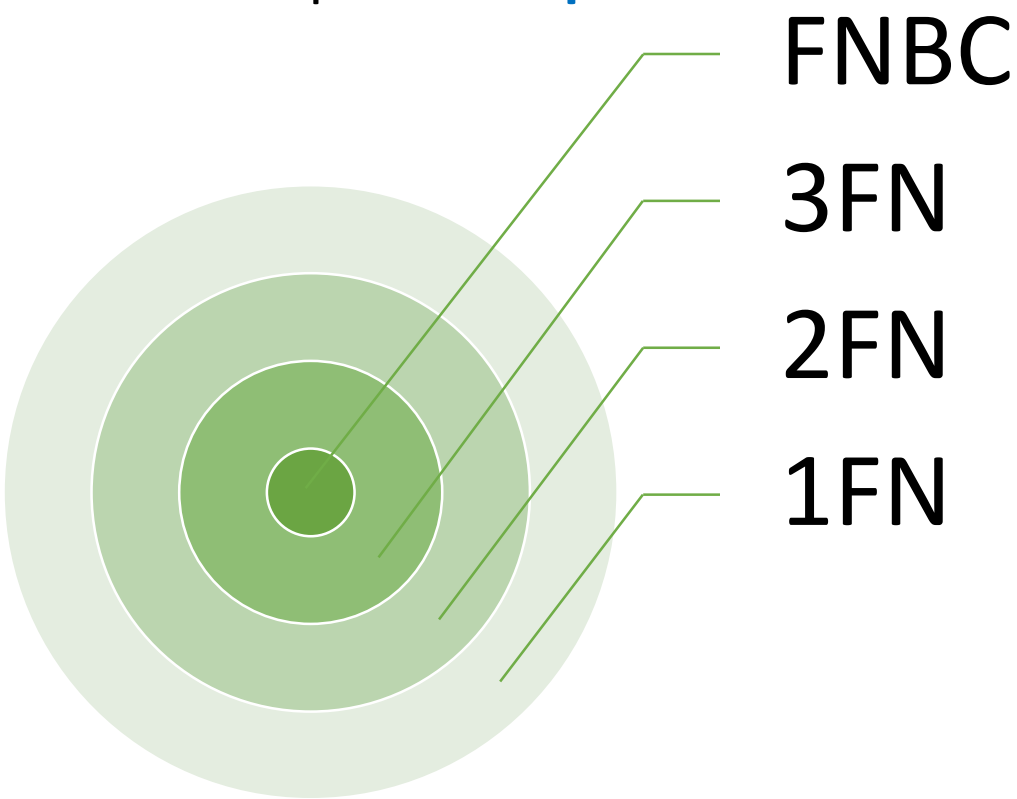
DNI	Nombre	Apelido	Teléfonos	Pabellón	Cod_Carrera	Carrera
43432	Juan	A		1	F	Física
43432	Juan	A		1	M	Matematica
5244	Maria	B	11 2233	1	F	Física
5244	Maria	B	11 2233	1	M	Matematica
4524	Pedro	C	22 3344	0	C	Computación
4524	Pedro	C	22 3344	0	D	Datos
123754	Alex	D	33 4455	2	B	Biologia
123754	Alex	D	33 4455	2	Q	Quimica
6534	Iris	E	44 5566	2	B	Biologia
6534	Iris	E	44 5566	2	Q	Quimica
635634	Cris	F	66 7788	0	C	Computación
635634	Cris	F	66 7788	0	D	Datos
987	Marcel	G	77 8899	0	C	Computación
987	Marcel	G	77 8899	0	D	Datos
4544	Carle	H	99 0011	0	C	Computación
4544	Carle	H	99 0011	0	D	Datos



- **Diseñar esquemas que puedan ser joineados con condiciones de igualdad por atributos que están apropiadamente relacionados** (Claves Foráneas, Claves Primarias). Evitar asociaciones entre relaciones que no sean (Claves Foráneas, Claves Primarias)

Cómo medimos esto?

- **Teoría de la normalización**.
- Permite medir la **calidad de diseño** de relaciones individuales.
- **Formas Normales**: estándares de calidad de diseño relacional.
- Se basan en el concepto de **Dependencias Funcionales**



Mientras más alta la forma normal, menos redundancia y mejor diseño.

✍ *Existen también 4FN y 5FN, no las veremos en la materia*

Dependencias funcionales

- Son restricciones sobre dos conjuntos de atributos de una base de datos.
- Dados dos conjuntos de atributos X e Y de R

$$X \rightarrow Y \text{ si } (t_1[X]=t_2[X] \Rightarrow t_1[Y]=t_2[Y]) \forall t_1, \forall t_2, \forall r(R)$$

Se dice que **X determina funcionalmente a Y**. O lo que es lo mismo, que Y está determinado (o depende) funcionalmente de X

La dependencia funcional **es una propiedad de la semántica** de los atributos

Y también lo es del esquema de relación R, no de un estado particular. Vale para todos los estados r(R)

Mirando un estado particular, sin saber semántica no se puede afirmar que una cierta dependencia funcional se cumple, aunque sí se podría afirmar lo contrario (encontrando un contraejemplo)

Reglas de Inferencia

Axiomas de Armstrong

- Dados R y un conjunto de dependencias funcionales F sobre R, $X \subseteq R$, $Y \subseteq R$, , $Z \subseteq R$
- **Reflexividad**

  Si $Y \subseteq X$, entonces $X \rightarrow Y$

- **Aumentación**

  Si $X \rightarrow Y$, entonces $XZ \rightarrow YZ$

- **Transitividad**

  Si $X \rightarrow Y$ y $Y \rightarrow Z$, entonces $X \rightarrow Z$

Reglas de Inferencia

Reglas derivadas de los Axiomas de Armstrong

- Dados R y un conjunto de dependencias funcionales F sobre R, $X \subseteq R$, $Y \subseteq R$, $Z \subseteq R$, $W \subseteq R$
- **Descomposición**

  Si $X \rightarrow YZ$, entonces $X \rightarrow Y$

- **Unión**

  Si $X \rightarrow Y$ y $X \rightarrow Z$, entonces $X \rightarrow YZ$

- **PseudoTransitividad**

  Si $X \rightarrow Y$ y $WY \rightarrow Z$, entonces $WX \rightarrow Z$

Reglas de Inferencia

ATENCIÓN

Estas que siguen NO SON reglas de inferencia válidas

- Dados R y un conjunto de dependencias funcionales F sobre R, $X \subseteq R$, $Y \subseteq R$, $Z \subseteq R$
- **Simetría**

Si $X \rightarrow Y$, **NO ES VALIDO INFERIR QUE** $Y \rightarrow X$

- **Descomposición de antecesor**

Si $XY \rightarrow Z$, **NO ES VALIDO INFERIR QUE** $X \rightarrow Z$

Claves

- Basándonos en el concepto de dependencias funcionales, podemos dar una definición más formal de clave:
- Un conjunto de atributos $\{A_1, \ldots A_n\}$ es **clave** de una relación R si
  - $\{A_1, \ldots A_n\}$ **determina funcionalmente al resto** de los atributos de R
  - **Ningún subconjunto estricto** de $\{A_1, \ldots A_n\}$ **determina a todos** los atributos de R
- Cualquier superconjunto de la relación R que contiene un conjunto de atributos clave, es una superclave
  - Recordemos que las superclaves no necesariamente son minimales

DF Triviales y atributos primos

- **DF Triviales**
- Una restricción sobre R se considera trivial si se mantiene para cada estado de la relación independientemente de cualquier otra restricción que se establezca.
- **Una DF $X \rightarrow Y$ es trivial si $Y \subseteq X$**

- **Atributos primos**
- Un atributo de una relación R:
  - se llama **primo** si es miembro de alguna clave candidata de R
  - Se llama **no primo** si no es miembro de ninguna clave candidata de R (si no es primo)

Primera Forma Normal (1FN)

- Está fuertemente vinculada con la definición formal de modelo relacional.
- Una relación está en **1FN** si todos sus atributos son **atómicos**. Es decir, no hay atributos multivaluados ni compuestos.

- Por la definición que vimos de modelo relacional, no se permiten atributos multivaluados ni compuestos.
  - Si hubiésemos modelado multivaluados, se debe generar una nueva relación para el atributo multivaluado
  - Si hubiésemos modelado atributos compuestos, se deben representar las hojas de ese atributo compuesto como atributos simples

- ¿qué pasa con los elementos repetitivos (teléfono1, teléfono2, teléfono3)?
  - Es un tema controversial. Para la definición formal, son 3 atributos atómicos, no invalida 1FN.
  - Sin embargo, para algunos autores sí la invalida ya que se trata de un multivaluado encubierto.
  - En cualquiera de los casos, es un mal diseño y no es recomendable… ¿por qué?

Segunda Forma Normal (2FN)

- Se basa en el concepto de dependencia funcional total
- Una relación R está en **2FN** si:
  - Está en **1FN**
  - Todo atributo no primo de R **depende completamente de todas** las claves de R
- Una relación R cuyas claves son simples (1 atributo) está en 2FN.
- Ej.

Empleado_Proyecto

Legajo	**Nombre_Emp**	***Cod_proyecto***
1	María	P1
1	María	P3
2	José	P7
3	Ana	P3
3	Ana	P7

DFs:

- Legajo → Nombre_Emp

PK: {{Legajo, Cod_Proyecto}}

Análisis

- Está en 1FN? Sí
- ¿Cuáles son los atributos no primos? Nombre_Emp
- Nombre_Emp depende funcionalmente de Legajo
- Legajo es una parte de la clave
- Por lo tanto Nombre_Emp depende parcialmente de la clave
- **No está en 2FN**

Segunda Forma Normal (2FN)

- Se basa en el concepto de dependencia funcional total
- Una relación R está en **2FN** si:
  - Está en **1FN**
  - Todo atributo no primo de R **depende completamente de todas** las claves de R
- Una relación R cuyas claves son simples (1 atributo) está en 2FN.
- Ej.

Empleado			
Legajo	**Nombre_Emp**	**Ciudad**	**Provincia**
1	María	Lomas de Zamora	Buenos Aires
2	José	Vicente López	Buenos Aires
3	Ana	Vicente López	Buenos Aires

DFs:
- Legajo → Nombre_Emp, Ciudad
- Ciudad → Provincia

PK: {Legajo}

Análisis
- Está en 1FN? Sí
- ¿Cuáles son los atributos no primos? Nombre_Emp, Ciudad, Provincia
- Nombre_Emp y Ciudad dependen funcionalmente de Legajo
- Por DF, Provincia depende funcionalmente de Ciudad, que no es clave
- Como Legajo → Ciudad y Ciudad → Provincia
- Entonces Legajo → Provincia (por transitividad)
- Por lo tanto Provincia depende (indirectamente) de Legajo, que es clave.
- **Está en 2FN**

Tercera Forma Normal (3FN)

- Se basa en el concepto de dependencia funcional transitiva
- Una relación R está en **3FN** si:
  - Para cada dependencia **no trivial** DF:X → Y, **X es superclave, o Y es primo**
- Dicho de otra forma:
  - Está en 3FN si está en 2FN y todo atributo no primo de R **depende directamente** de todas las claves de R (no hay dependencias funcionales transitivas).
- Ej

Empleado			
Legajo	**Nombre_Emp**	**Ciudad**	**Provincia**
1	María	Lomas de Zamora	Buenos Aires
2	José	Vicente López	Buenos Aires
3	Ana	Vicente López	Buenos Aires

DFs:
- Legajo → Nombre_Emp, Ciudad
- Ciudad → Provincia

PK: {Legajo}

Análisis

- Está en 2FN? Sí
- ¿Cuáles son los atributos no primos? Nombre_Emp, Ciudad, Provincia
- Nombre_Emp y Ciudad dependen funcionalmente de Legajo en forma directa
- Por lo que vimos, Provincia depende funcionalmente de Legajo en forma indirecta.
- **No está en 3FN**

Tercera Forma Normal (3FN)

- Se basa en el concepto de dependencia funcional transitiva
- Una relación R está en **3FN** si:
  - Para cada dependencia **no trivial** DF:X → Y, **X es superclave, o Y es primo**
- Dicho de otra forma:
  - Está en 3FN si está en 2FN y todo atributo no primo de R **depende directamente** de todas las claves de R (no hay dependencias funcionales transitivas.
- Ej

Empleado_Supervisor		
Legajo	***Ciudad***	**Supervisor**
1	Lomas de Zamora	Zaldibar
1	Rosario	Lina
2	Vicente López	Santi
3	Vicente López	Ulises
3	Buenos Aires	Mara

DFs:
- Legajo, Ciudad → Supervisor
- Supervisor → Ciudad

PK: {Legajo, Ciudad}
CK: {{Legajo, Ciudad}, {Legajo, Supervisor})

Análisis
- Está en 2FN? Sí
- ¿Cuáles son los atributos no primos? Todos son primos
- Por lo tanto, toda parte derecha de las DFs son atributos primos
- **Está en 3FN**

Forma Normal Boyce-Codd (FNBC)

- Fue propuesta originalmente como una alternativa a la 3FN.
- Una relación R está en **FNBC** si:
  - Para cada dependencia **no trivial** DF:X → Y, **X es superclave**
- ¿FNBC es más estricta que 3FN?
  - Sí, porque exige la condición para todos los atributos, no sólo para los no primos
- Ej

Empleado_Supervisor

Legajo	***Ciudad***	**Supervisor**
1	Lomas de Zamora	Zaldibar
1	Rosario	Lina
2	Vicente López	Santi
3	Vicente López	Ulises
3	Buenos Aires	Mara

DFs:
- Legajo, Ciudad → Supervisor
- Supervisor → Ciudad

PK: {Legajo, Ciudad}
CK: {{Legajo, Ciudad}, {Legajo, Supervisor})

Análisis
- Está en 3FN? Sí
- En la primera DF, el antecesor es PK (y por lo tanto superclave)
- En la segunda DF, el antecesor no es superclave
- **No está en FNBC**

Forma Normal Boyce-Codd (FNBC)

- Fue propuesta originalmente como una alternativa a la 3FN.
- Una relación R está en **FNBC** si:
  - Para cada dependencia **no trivial** DF:X → Y, **X es superclave**
- ¿FNBC es más estricta que 3FN?
  - Sí, porque exige la condición para todos los atributos, no sólo para los no primos
- Ej

Empleado_Supervisor

Legajo	***Ciudad***	**Supervisor**
1	Lomas de Zamora	Zaldibar
1	Rosario	Lina
2	Vicente López	Santi
3	Vicente López	Ulises
3	Buenos Aires	Mara

DFs:

- Legajo, Ciudad → Supervisor

PK: {Legajo, Ciudad}
CK: {Legajo, Ciudad})

¿Qué pasaría si los supervisores pudiesen serlo en más de una ciudad? La segunda DF desaparece…

Análisis

- Está en 3FN? Sí
- En la única DF, el antecesor es PK (y por lo tanto superclave)
- **Está en FNBC**

Normalización

- **Normalización de datos**:
  - Proceso de transformación de esquemas de relación para alcanzar formas normales altas, de manera de minimizar la redundancia y las anomalías asociadas.

- En líneas generales, se realizan test para verificar si satisface una determinada forma normal, y luego se procede a **descomponer** los esquemas de relación en relaciones más pequeñas que contienen subconjuntos de sus atributos relaciones

- *La manera de mejorar la calidad se basa en la descomposición de relaciones de mal diseño para alcanzar formas normales más altas*

Descomposición de relaciones

- Dado una relación $R(A_1, A_2, ..., A_n)$, se dice que dos relaciones $S(B_1, B_2, ..., B_m)$ y $T(C_1, C_2, ... C_k)$ son una **descomposición de una relación** R si:
  - $\{A_1, A_2, ..., A_n\} = \{B_1, B_2, ..., B_n\} \cup \{C_1, C_2, ... C_n\}$
  - $S = \pi_{B1, B2, ... Bm} (R)$
  - $T = \pi_{C1, C2, ... Ck} (R)$

Empleado

Legajo	**Nombre_Emp**	***Ciudad***	**Provincia**
1	María	Lomas de Zamora	Buenos Aires
1	María	Rosario	Santa Fe
2	José	Vicente López	Buenos Aires
3	Ana	Vicente López	Buenos Aires
3	Ana	Buenos Aires	CABA

Posibles descomposiciones binarias

Empleado

Legajo	**Nombre_Emp**	***Ciudad***
1	María	Lomas de Zamora
1	María	Rosario
2	José	Vicente López
3	Ana	Vicente López
3	Ana	Buenos Aires

Empleado_Pcia

Legajo	***Provincia***
1	Buenos Aires
1	Santa Fe
2	Buenos Aires
3	Buenos Aires
3	CABA

DFs:

- Legajo → Nombre_Emp
- Ciudad → Provincia

PK: {{Legajo, Ciudad}}

Empleado

Legajo	**Nombre_Emp**	***Ciudad***
1	María	Lomas de Zamora
1	María	Rosario
2	José	Vicente López
3	Ana	Vicente López
3	Ana	Buenos Aires

Ciudad_Pcia

Ciudad	**Provincia**
Lomas de Zamora	Buenos Aires
Rosario	Santa Fe
Vicente López	Buenos Aires
Buenos Aires	CABA

Descomposición de relaciones

- El concepto de descomposición **se generaliza a k relaciones**: todos los atributos del esquema de relación original deben aparecer en alguna de los esquemas de relación resultantes de la descomposición

Empleado			
Legajo	**Nombre_Emp**	***Ciudad***	**Provincia**
1	María	Lomas de Zamora	Buenos Aires
1	María	Rosario	Santa Fe
2	José	Vicente López	Buenos Aires
3	Ana	Vicente López	Buenos Aires
3	Ana	Buenos Aires	CABA

Posibles descomposición en 3 relaciones

Empleado	
Legajo	**Nombre_Emp**
1	María
2	José
3	Ana

Empl_Ciudad	
Legajo	***Ciudad***
1	Lomas de Zamora
1	Rosario
2	Vicente López
3	Vicente López
3	Buenos Aires

Ciudad_Pcia	
Ciudad	**Provincia**
Lomas de Zamora	Buenos Aires
Rosario	Santa Fe
Vicente López	Buenos Aires
Buenos Aires	CABA

DFs:

- Legajo → Nombre_Emp
- Ciudad → Provincia

PK: {{Legajo, Ciudad}}

Clausura de Dependencias Funcionales

- Dados R y un conjunto de dependencias funcionales F sobre R, X ⊆ R, Y ⊆ R, Z ⊆ R , W ⊆ R
- Una DF X → Y **se infiere de F**, si se mantiene para todo estado r(R).
- Usamos la notación F |= X → Y
- Se define como **Clausura de F** (y se denota como **F^+**) al conjunto de todas las dependencias funcionales de F junto con todas las dependencias funcionales que se infieren de F.
  - $F^+ = \{X \rightarrow Y \ / \ F \models X \rightarrow Y \}$

Empleado	
Legajo	**Nombre_Emp**
1	María
2	José
3	Ana

F: { Legajo → Nombre_Emp }

F^+: { Legajo → Nombre_Emp ,
Legajo → Legajo,
Nombre_Emp → Nombre_Emp,
{Legajo, Nombre_Emp} → Legajo,
{Legajo, Nombre_Emp} → Nombre_Emp,
{Legajo, Nombre_Emp} → Legajo, Nombre_Emp,
Legajo→ Legajo, Nombre_Emp
}

Clausura de Atributos

- Dados R y un conjunto de dependencias funcionales F sobre R, X ⊆ R
- Se define como **Clausura de X bajo F** (se denota como **X^+**) al conjunto de atributos de R que son determinados funcionalmente por X basados en F

Algorithm 15.1. Determining X^+, the Closure of X under F

Input: A set F of FDs on a relation schema R, and a set of attributes X, which is a subset of R.

$X^+ := X$;
repeat
    old$X^+ := X^+$;
    for each functional dependency $Y \rightarrow Z$ in F do
        if $X^+ \supseteq Y$ then $X^+ := X^+ \cup Z$;
    until (X^+ = oldX^+);

Algoritmo tomado de Elmasri- Navathe "Fundamentals of Database Systems"

Empleado			
Legajo	**Nombre_Emp**	**Ciudad**	**Provincia**
1	María	Lomas de Zamora	Buenos Aires
2	José	Vicente López	Buenos Aires
3	Ana	Vicente López	Buenos Aires

DFs:
- Legajo → Nombre_Emp, Ciudad
- Ciudad → Provincia

¿Cuál sería la clausura de Legajo?

Conjuntos equivalentes de DFs

- Dados R y dos conjuntos de dependencias funcionales E y F sobre R, $X \subseteq R$
- Decimos que **F cubre a E**, si cada DF en E está en F^+

- Decimos que **E y F son equivalentes** si $E^+ = F^+$, o lo que es lo mismo, si E cubre a F y F cubre a E

- Calcular clausuras de DFs es costoso

- Una **forma alternativa** de verificar si **F cubre a E** es:
  - Para cada DF $X \rightarrow Y$ de E
    - Calcular X^+ respecto de F *(notar que se calcula la clausura con el otro conjunto de df)*
    - Verificar que cada $X^+ \supseteq Y$
  - Si se cumple en todos los casos, entonces F cubre a E

Proyección de DFs

- Dados R, un conjunto de dependencias funcionales F sobre R,

  y una descomposición ρ $(R_1, \ldots, R_k)$ de R

- Una **proyección de F** sobre un R_i es el conjunto de dependencias funcionales $X \rightarrow Y$ de F^+

  tal que $(X \cup Y) \subseteq R_i$

Cubrimiento Minimal

- Dados R, un conjunto de dependencias funcionales F sobre R
- Decimos que F es un **cubrimiento minimal** si
  - Toda dependencia funcional de F tiene un único atributo en el lado derecho
  - Para toda df $X \rightarrow A$, X no tiene atributos extraños (o redundantes).
    - Es decir, no podemos reemplazar $X \rightarrow A$ por $Y \rightarrow A$, con $Y \subset X$, y mantener un conjunto de DFs equivalente a F
  - No hay dfs redundantes.
    - Es decir, no podemos eliminar ninguna DF, y mantener un conjunto de DFs equivalente a F
- Para un conjunto de DFs puede haber más de un cubrimiento minimal diferente

✍ *En la práctica se verá un algoritmo para generar un cubrimiento minimal.*

Descomposición de relaciones

Propiedades deseables de las descomposiciones

Descomposición Sin Pérdida de Información (SPI)

- Garantiza que no se generen tuplas espurias al querer reconstruir la información de la relación original.
- Propiedad de Junta No Aditiva.

Formalmente…

Una descomposición

ρ $(R_1, \ldots, R_k)$ de R es SPI

Si $R = \bowtie \pi_{Ri}(R)$, $1 \leq i \leq n$

Descomposición Sin Pérdida de Dependencias Funcionales (SPDF)

- Asegura que cada dependencia funcional de la relación original está representada o se infiere a partir de las relaciones resultantes de la descomposición.
- Propiedad de Preservación de Dependencias

Formalmente…

Una descomposición ρ $(R_1, \ldots, R_k)$ de R es SPDF

Si $F^+ = (\cup \pi_{Ri}(F))^+$, $1 \leq i \leq n$

Es decir $\cup \pi_{Ri}(F)$ y F son equivalentes

Descomposición de relaciones

- Dado una relación R, **no siempre** se puede encontrar una descomposición que sea
  - SPI, SPDF y FNBC

- Sí se puede garantizar que sea
  - SPI, SPDF y 3FN

- Lo que no podemos permitir de una descomposición es que pierda información, es decir, **siempre tenemos que lograr descomposiciones SPI**

- Respecto de las otras dos propiedades, podemos tener que elegir entre:
  - Lograr FNBC con pérdida de DF
  - Lograr 3FN SPDF

Descomposición de Relaciones

¿cómo verificar si una descomposición binaria es SPI?

- Dado una descomposición binaria ρ (R_1, R_2) de R y un conjunto F de DF:
  - ρ es SPI si y solo si

    $(R_1 \cap R_2) \rightarrow (R_1\text{-}R_2)$ está en F^+

    ó

    $(R_1 \cap R_2) \rightarrow (R_2\text{-}R_1)$ está en F^+

- Dicho de otra forma, ρ es SPI si y solo si $(R_1 \cap R_2)$ es superclave de R_1 o de R_2

- ¿y si la descomposición fuera n-aria con n>2? Hay un algoritmo, llamado Algoritmo Tableau.

✍ *Para quien quiera profundizar sobre este algoritmo Tableau, está en el apunte de Normalización (material complementario). No se verá en clase ni se tomará en los exámenes.*

Descomposición de Relaciones

¿cómo verificar si una descomposición es SPDF?

- Calcular clausuras de DFs es muy costoso.
- Hay un **algoritmo alternativo**, que requiere calcular clausuras de atributos.
- Dado una descomposición ρ (R_1, R_2, …R_n)de R y un conjunto F de dependencias funcionales:
  - Para cada dependencia $\alpha \rightarrow \beta$ de F, aplicar el algoritmo siguiente:

```
result = α
repeat
    for each R_i in the decomposition
        t = (result ∩ R_i)^+ ∩ R_i
        result = result ∪ t
until (result does not change)
```

Algoritmo tomado de Silberschatz-Korth-Sudarshan "Database System Concepts "

Notar que el ciclo "repeat" podría frenar también cuando $\beta \subseteq$ result

  - Si el resultado result contiene a β, y esto se verifica para todas las dependencias de F, es SPDF.
  - Caso contrario, no es SPDF

✍ *Para quien quiera profundizar sobre este algoritmo, está en el apunte de Normalización (material complementario). No se verá en la práctica ni se tomará en los exámenes.*

Descomposición de relaciones

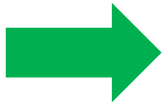
- Verifiquemos SPI y SPDF en las descomposiciones de ejemplo

Empleado

Legajo	**Nombre_Emp**	***Ciudad***	**Provincia**
1	María	Lomas de Zamora	Buenos Aires
1	María	Rosario	Santa Fe
2	José	Vicente López	Buenos Aires
3	Ana	Vicente López	Buenos Aires
3	Ana	Buenos Aires	CABA

DFs:
- Legajo → Nombre_Emp
- Ciudad → Provincia

PK: {{Legajo, Ciudad}}



Empleado_Ciudad

Legajo	**Nombre_Emp**	***Ciudad***
1	María	Lomas de Zamora
1	María	Rosario
2	José	Vicente López
3	Ana	Vicente López
3	Ana	Buenos Aires

Empleado_Pcia

Legajo	***Provincia***
1	Buenos Aires
1	Santa Fe
2	Buenos Aires
3	Buenos Aires
3	CABA

Es SPI?

Empleado ∩ Empleado_Pcia = {Legajo}

No es superclave de Empleado_Ciudad ni de Empleado_Pcia

No es SPI

Empleado_Ciudad ⋈ Empleado_Pcia

Legajo	Nombre_Emp	Ciudad	Provincia
1	María	Lomas de Zamora	Buenos Aires
1	María	Lomas de Zamora	Santa Fe
1	María	Rosario	Buenos Aires
1	María	Rosario	Santa Fe
2	José	Vicente López	Buenos Aires
3	Ana	Vicente López	Buenos Aires
3	Ana	Vicente López	CABA
3	Ana	Buenos Aires	Buenos Aires
3	Ana	Buenos Aires	CABA

Es SPDF?

Legajo → Nombre_Emp se proyecta a Empleado_Ciudad

Ciudad → Provincia no se proyecta ni se puede inferir

No es SPDF

Descomposición de relaciones

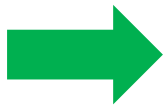
- Verifiquemos SPI y SPDF en las descomposiciones de ejemplo

Empleado

Legajo	**Nombre_Emp**	***Ciudad***	**Provincia**
1	María	Lomas de Zamora	Buenos Aires
1	María	Rosario	Santa Fe
2	José	Vicente López	Buenos Aires
3	Ana	Vicente López	Buenos Aires
3	Ana	Buenos Aires	CABA

DFs:
- Legajo → Nombre_Emp
- Ciudad → Provincia

PK: {{Legajo, Ciudad}}



Empleado_Ciudad

Legajo	**Nombre_Emp**	***Ciudad***
1	María	Lomas de Zamora
1	María	Rosario
2	José	Vicente López
3	Ana	Vicente López
3	Ana	Buenos Aires

Ciudad_Pcia

Ciudad	**Provincia**
Lomas de Zamora	Buenos Aires
Rosario	Santa Fe
Vicente López	Buenos Aires
Buenos Aires	CABA

Es SPI?

Empleado ∩ Ciudad_Pcia = {Ciudad}
Es superclave de Ciudad_Pcia
Es SPI

Empleado_Ciudad ⋈ Ciudad_Pcia

Legajo	Nombre_Emp	Ciudad	Provincia
1	María	Lomas de Zamora	Buenos Aires
1	María	Rosario	Santa Fe
2	José	Vicente López	Buenos Aires
3	Ana	Vicente López	Buenos Aires
3	Ana	Buenos Aires	CABA

Es SPDF?

Legajo → Nombre_Emp se proyecta a Empleado_Ciudad

Ciudad → Provincia se proyecta a Ciudad_Pcia

Es SPDF

Descomposición de relaciones

- Verifiquemos SPI y SPDF en las descomposiciones de ejemplo

Empleado

Legajo	**Nombre_Emp**	***Ciudad***	**Provincia**
1	María	Lomas de Zamora	Buenos Aires
1	María	Rosario	Santa Fe
2	José	Vicente López	Buenos Aires
3	Ana	Vicente López	Buenos Aires
3	Ana	Buenos Aires	CABA

DFs:
- Legajo → Nombre_Emp
- Ciudad → Provincia

PK: {{Legajo, Ciudad}}

Empleado

Legajo	**Nombre_Emp**
1	María
2	José
3	Ana

Empl_Ciudad

Legajo	***Ciudad***
1	Lomas de Zamora
1	Rosario
2	Vicente López
3	Vicente López
3	Buenos Aires

Ciudad_Pcia

Ciudad	**Provincia**
Lomas de Zamora	Buenos Aires
Rosario	Santa Fe
Vicente López	Buenos Aires
Buenos Aires	CABA

Es SPI?

Empleado_Ciudad y Ciudad_Pcia era SPI

Veamos ahora Empleado y Empl_Ciudad como descomposición binaria de Empleado_Ciudad
Empleado ∩ Empl_Ciudad = {Legajo}
Es superclave de Empleado (Legajo, Nombre_Emp)

Es SPI

Empleado_Ciudad

Legajo	**Nombre_Emp**	***Ciudad***
1	María	Lomas de Zamora
1	María	Rosario
2	José	Vicente López
3	Ana	Vicente López
3	Ana	Buenos Aires

Es SPDF?

Legajo → Nombre_Emp se proyecta a Empleado

Ciudad → Provincia se proyecta a Ciudad_Pcia

Es SPDF

Normalización

- Veremos ahora un ejemplo de **normalización a FNBC**, basado en sucesión de **descomposiciones binarias**
- Como ejemplo utilizaremos la siguiente relación:

ALUMNO_CARRERA

DNI	Nombre	Apellido	Teléfono	Ciudad	Provincia	Cod_Carrera	Carrera	Tutor
43432	Juan	A	001122	Buenos Aires	CABA	F	Física	Sandra
5244	Maria	B	112233	Lomas de Zamora	Buenos Aires	M	Matematica	José
4524	Pedro	C	223344	Rosario	Santa Fe	C	Computación	Marina
6534	Iris	E	445566	Vicente López	Buenos Aires	B	Biologia	Emilie
6534	Iris	E	445566	Vicente López	Buenos Aires	Q	Quimica	Alex
635634	Cris	F	667788	Chivilcoy	Buenos Aires	D	Datos	Dani
635634	Cris	F	667788	Chivilcoy	Buenos Aires	C	Computación	Jorge
4544	Carle	H	990011	Santa Rosa	La Pampa	C	Computación	Jorge

- Están identificadas las siguientes dependencias funcionales:
  - DNI → Nombre, Apellido, Teléfono, Ciudad
  - Ciudad → Provincia
  - Cod_Carrera → Carrera
  - DNI, Cod_Carrera → Tutor
  - Tutor → Cod_Carrera /** cada tutor corresponde a una única carrera **/
- PK = {{DNI, Cod_Carrera}}
- **¿Cuales son las claves candidatas? ¿Hay alguna aparte de la PK?**

Normalización

- Como ejercicio **buscaremos las claves** de la relación, a partir de sus dependencias funcionales

  DFs: {DNI → Nom, Ape, Tel, Ciu ; Ciu → Prov ; CC → Car ; Tut → CC ; {DNI, CC} → Tut }

- ❑ Seleccionamos los atributos que no están en ninguna parte derecha: sólo DNI
- ❑ Calculamos $\{DNI\}^+ = \{DNI, Nom, Ape, Tel, Ciu, Prov\}$ – No están todos los atributos
- ❑ Agregamos un atributo (hay que probar todos) y buscamos clausura
- ❑ $\{DNI, CC\}^+ = \{DNI, CC, Nom, Ape, Tel, Ciu, Prov, Car, Tut\}$ – **Están todos los atributos, es clave**
- ❑ $\{DNI, Car\}^+ = \{DNI, Car, Nom, Ape, Tel, Ciu, Prov\}$ – No están todos los atributos
- ❑ $\{DNI, Tut\}^+ = \{DNI, Tut, Nom, Ape, Tel, Ciu, Prov, CC, Car\}$ – **Están todos los atributos, es clave**

Las CK son {DNI, CC} y {DNI, Tut}

Normalización

- Veremos ahora un ejemplo de normalización a FNBC, basado en sucesión de descomposiciones binarias
- Como ejemplo utilizaremos la siguiente relación:

ALUMNO_CARRERA

DNI	Nombre	Apellido	Teléfono	Ciudad	Provincia	Cod_Carrera	Carrera	Tutor
43432	Juan	A	001122	Buenos Aires	CABA	F	Física	Sandra
5244	Maria	B	112233	Lomas de Zamora	Buenos Aires	M	Matematica	José
4524	Pedro	C	223344	Rosario	Santa Fe	C	Computación	Marina
6534	Iris	E	445566	Vicente López	Buenos Aires	B	Biologia	Emilie
6534	Iris	E	445566	Vicente López	Buenos Aires	Q	Quimica	Alex
635634	Cris	F	667788	Chivilcoy	Buenos Aires	D	Datos	Dani
635634	Cris	F	667788	Chivilcoy	Buenos Aires	C	Computación	Jorge
4544	Carle	H	990011	Santa Rosa	La Pampa	C	Computación	Jorge

- Están identificadas las siguientes dependencias funcionales:
  - DNI → Nombre, Apellido, Teléfono, Ciudad
  - Ciudad → Provincia
  - Cod_Carrera → Carrera
  - DNI, Cod_Carrera → Tutor
  - Tutor → Cod_Carrera
- PK = {{DNI, Cod_Carrera}}
- CK = {{DNI, Cod_Carrera}, {DNI, Tutor}}

***FNBC**: Para cada dependencia no trivial DF:X → Y, X es superclave*

Normalización

(*DNI*, Nom, Ape, Tel, Ciu, Prov, *CC*, Car, Tut)
{DNI → Nom, Ape, Tel, Ciu ; Ciu → Prov ; CC → Car ; Tut → CC ; {DNI, CC} → Tut }

Ciu → Prov

(*Ciu*, Prov)

(*DNI*, Nom, Ape, Tel, Ciu, *CC*, Car, Tut)
{DNI → Nom, Ape, Tel, Ciu ; CC → Car ; Tut → CC ; {DNI, CC} → Tut }

CC → Car

(*CC*, Car)

(*DNI*, Nom, Ape, Tel, Ciu, *CC*, Tut)
{DNI → Nom, Ape, Tel, Ciu ; Tut → CC ; {DNI, CC} → Tut }

DNI → Nom, Ape, Tel, Ciu

(*DNI*, Nom, Ape, Tel, Ciu)

(*DNI*, *CC*, Tut)
{Tut → CC ; {DNI, CC} → Tut }

Tut → CC

(*Tut*, CC)

(*DNI*, *Tut*)

***FNBC**: Para cada dependencia no trivial DF:X → Y, X es superclave*

Hasta acá está en 3FN

Este algoritmo que aplicamos logra descomposiciones FNBC y SPI, sin garantizar SPDF. La descomposición resultante es dependiente del orden de elección de DFs.

✍ *En la práctica verán también el algoritmo para lograr descomposiciones 3FN, SPI y SPDF*

Notar que se perdió la DF {DNI,CC} → Tut

Normalización

- Proyectando el estado de Relación sobre la descomposición quedaría

Ciudad

Ciudad	Provincia
Buenos Aires	CABA
Lomas de Zamora	Buenos Aires
Rosario	Santa Fe
Vicente López	Buenos Aires
Chivilcoy	Buenos Aires
Santa Rosa	La Pampa

Carrera

Cod_Carrera	Carrera
F	Física
M	Matematica
C	Computación
B	Biologia
Q	Quimica
D	Datos

Alumno

DNI	Nombre	Apellido	Teléfono	Ciudad
43432	Juan	A	001122	Buenos Aires
5244	Maria	B	112233	Lomas de Zamora
4524	Pedro	C	223344	Rosario
6534	Iris	E	445566	Vicente López
635634	Cris	F	667788	Chivilcoy
4544	Carle	H	990011	Santa Rosa

Tutor	Carrera
Sandra	Física
José	Matematica
Marina	Computación
Emilie	Biologia
Alex	Quimica
Dani	Datos
Jorge	Computación

DNI	*Tutor*
43432	Sandra
5244	José
4524	Marina
6534	Emilie
6534	Alex
635634	Dani
635634	Jorge
4544	Jorge

Algo de diseño

- Atributos atómicos… ¿Qué pasa con un atributo que contenga valores como “LCC107”, donde LCC representa la carrera?
- Nombres de relaciones… ¿Empleado_Proyecto vs Trabaja_En? ¿Empleado_Empleado vs Es_Jefe_De?
- Nombres de atributos… campo1, campo2, campo3… ¿semántica?
- Nombres de claves foráneas… ¿id_empl1, id_empl2? ¿roles?
- Datos asociados a rangos temporales… por ej anualizados
  - ¿una relación con un atributo año? ¿una relación por año? ¿una relación con un atributo por año?
- Datos temporales? O sea, con vigencia temporal, manteniendo historia
- Datos auditables?

- Common Data Models… ejemplo Contact

Desnormalización

- **Desnormalización de datos**: proceso de almacenar una junta de relaciones de alto grado de normalización como una relación de menor grado de normalización
- La desnormalización debe ser un proceso controlado y con justificación por algún motivo de relevancia (por ej, performance)

Bibliografía del tema

- *Elmasri, Navathe (2016) Fundamentals of Database Systems, 7th Edition. Pearson. Cap. 14 y 15*
- *García Molina H., Ullman J. & Widom J. (2009) Database Systems: The Complete Book, (2da. Ed.), Pearson, Cap. 3*
- *Silberschatz A., Korth H. & Sudarshan S.(2020) Database System Concepts (7ma. Ed.), Mc.Graw Hill, Cap. 7*
- *Ramakrishnan r. & Gehrke J. (2003) Database System Concepts (3ra. Ed.), Mc.Graw Hill, Cap. 19*

Dudas

Muchas gracias!