



Capitulo 4



Diseño de una Bases de Datos

Agenda

- 4.1.- Modelo Relacional
 - 4.1.1.- Introducción
 - 4.1.2.- Conceptos
 - 4.1.3.- Propiedades de la relaciones
 - 4.1.4.- Restricciones del modelo relacional
- 4.2.- Transformación del diagrama ER al relacional
- 4.3.- Normalización
 - 4.3.1.- Anomalías
 - 4.3.2.- Definiciones
 - 4.3.3.- Dependencias funcionales
 - 4.3.4.- Formas Normales
 - 4.3.5.- Otras formas normales
 - 4.3.6.- Ejemplos

4.1.1 El modelo de datos relacional

- ▶ Fue introducido por Codd en 1970.
- ▶ Está basado en estructuras de datos simples y uniformes y tiene una sólida base teórica.
- ▶ Representa a la base de datos como una colección de relaciones.
- ▶ Cuando una relación es considerada como una tabla de valores, cada renglón en la tabla representa una colección de valores de datos relacionados.
- ▶ Los nombres de la tabla y de las columnas ayudan a interpretar el significado de los valores en cada renglón.

4.1.2 Conceptos

- ▶ Todos los valores en una columna son del mismo tipo de datos.
- ▶ En esta terminología, un renglón es llamado un tupla, una columna es llamada un atributo, y la tabla es llamada una relación.
- ▶ El tipo de dato que describe los tipos de valores que pueden aparecer en cada columna es llamado el dominio.
- ▶ El dominio $-dom(A)-$ es un conjunto de valores atómicos; esto es, cada valor es indivisible.

Definiciones

- ▶ Un esquema de una relación se denota por $R(A_1, A_2, \dots, A_n)$, en donde R es la relación y A_1, A_2, \dots, A_n es la lista de atributos.
- ▶ El grado de la relación es el número de atributos del esquema de la relación.
- ▶ Una instancia de una relación $-r(R)-$ es un conjunto de n -tuplas $r = \{t_1, t_2, \dots, t_n\}$
- ▶ Cada tupla t es una lista de valores ordenados $t = \langle v_1, v_2, \dots, v_n \rangle$ donde cada valor v es un elemento de $dom(A)$ o un valor nulo.
- ▶ La cardinalidad de una relación es el numero de tuplas de que está compuesta y se denota $card(R)$.

Ejemplo

EMPLEADO

NOMBRE	<u>NSS</u>	FNAC	DIRECCION	NoD
John Smith	123456789	09-JAN-55	731 Fondren, Houston	5
Franklin Wong	333444555	08-DEC-45	638 Voss, Houston	5
Alicia Zelaya	999887777	19- JUL-58	3321 Castle, Spring	4
James Borg	888665555	10-NOV-27	450 Stone, Houston	4

DEPARTAMENTO

NOMBDEP	<u>NoDEP</u>	NSSJEFE	FINICIOJEFE
Reserach	5	333444555	22-MAY-78
Administration	4	987654321	01-JAN-85

PROYECTO

NOMBP	<u>NoPROY</u>	LUGARP	NoDEP
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5

DEPENDIENTE

<u>NSSEMP</u>	<u>NOMBDEP</u>	SEXO	FNAC
333444555	Alice	F	05-APR-76
333444555	Theodore	M	25-OCT-73
999887777	Joy	F	03-MAY-48
888665555	Michael	M	01-JAN-78

4.1.3 Tipos de llaves

▶ Superllave

- ▶ Es el conjunto de uno o más atributos, los cuales, tomados como conjunto, permiten identificar de manera única a una tupla en la relación.

▶ Llave primaria

- ▶ Es la llave candidata que es elegida por el diseñador de la base de datos como la llave principal.

▶ Llave candidata (alternativa)

- ▶ En una tabla puede que tengamos más de una columna que puede ser llave primaria por sí misma. En ese caso se puede escoger una para ser la llave primaria y las demás llaves serán llaves candidatas.

▶ Llave ajena (foreign key o llave foránea)

- ▶ Es aquella columna que existiendo como dependiente en una tabla, es a su vez llave primaria en otra tabla.

▶ Llave compuesta

- ▶ Es una llave que está compuesta por más de una columna.



4.1.3 Propiedades de las relaciones

- ▶ **Orden de las tuplas en una relación.-** Matemáticamente, los elementos de una relación no tienen un orden entre ellos; una relación intenta representar hechos a un nivel abstracto.
- ▶ **Orden de valores dentro de una tupla.-** Una tupla es una lista ordenada de n valores. Sin embargo, a nivel lógico, si la correspondencia atributo-valor se mantiene, esto no es tan importante.
- ▶ **Valores en las tuplas.-** Cada valor en las tuplas es atómico; por lo tanto, no se permiten atributos multivalores o compuestos (primera forma normal).
- ▶ **Interpretación de una relación.-** Un esquema de una relación puede ser interpretado como una declaración o como un tipo de aserción. El modelo relacional representa hechos como entidades o relaciones uniformemente como relaciones.

4.1.4 Restricciones en el modelo relacional (1)

- ▶ Un esquema de base de datos relacional S es un conjunto de esquemas relacionales $S = \{R_1, R_2, \dots, R_n\}$ y un conjunto de restricciones IC .
- ▶ Una instancia de base de datos relacional es un conjunto de instancias de relaciones $DB = \{r_1, r_2, \dots, r_n\}$ tal que cada r_i es una instancia de R_i y tal que las r_i relaciones satisfacen las restricciones de integridad especificadas en IC .
- ▶ La restricción de **integridad de entidad** establece que ningún valor de llave primaria puede ser nulo.

Restricciones en el modelo relacional (2)

- ▶ Los atributos con la propiedad de que no existan dos tuplas de una instancia de relación r de R tengan la misma combinación de valores para esos atributos. Esto es:

$t_1[K] \neq t_2[K]$ donde K es el subconjunto de atributos mencionado.

- ▶ Este subconjunto de atributos es llamado la *superllave* del esquema de la relación R .
- ▶ Toda relación tiene al menos una superllave.
- ▶ Una *llave* K es una superllave de R con la propiedad adicional de no redundancia. Una llave es una superllave mínima.
- ▶ Cuando una relación tenga más de una llave, cada atributo es llamado una llave *candidata*, pero solo una es elegida como llave *primaria*.

Restricciones en el modelo relacional (3)

- ▶ **Restricciones de dominio.-** Esta restricción especifica que el valor de cada atributo debe ser un valor atómico para el dominio $dom(A)$ para ese atributo. Los tipos de datos asociados con los dominios típicamente incluyen tipos de datos numéricos para enteros (cortos y largos) y números reales (flotante y flotante de doble precisión). También son aceptados los tipo carácter (fijo y variable), así como las fechas, tiempo, estampa de tiempo y moneda.

Tipos de datos del SQL

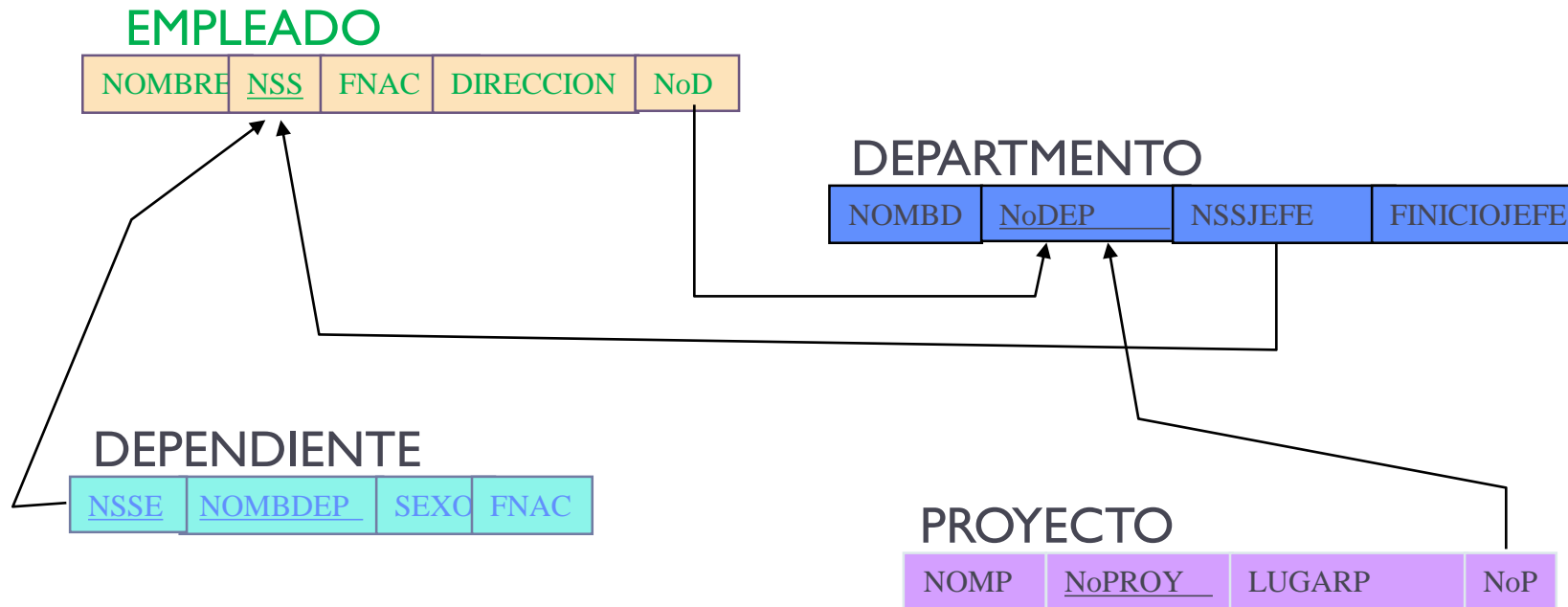
<u>Tipo de dato</u>	<u>Sinónimos</u>	<u>Tamaño</u>	<u>Descripción</u>
BINARY	VARBINARY, BINARY VARYING, BIT VARYING	1 byte por carácter	Se puede almacenar cualquier tipo de datos en un campo de este tipo. Los datos no se traducen (por ejemplo, a texto).
BIT	BOOLEAN, LOGICAL, LOGICAL1, YESNO	1 byte	Valores Sí y No, y campos que contienen solamente uno de dos valores.
TINYINT	INTEGER1, BYTE	1 byte	Un número entero entre 0 y 255.
COUNTER	AUTOINCREMENT		Se utiliza para campos contadores cuyo valor se incrementa automáticamente al crear un nuevo registro.
MONEY	CURRENCY	8 bytes	Un número entero comprendido entre – 922.337.203.685.477,5808 y 922.337.203.685.477,5807.
DATETIME	DATE, TIME	8 bytes	Una valor de fecha u hora entre los años 100 y 9999
UNIQUEIDENTIFIER	GUID	128 bits	Un número de identificación único utilizado con llamadas a procedimientos remotos.
DECIMAL	NUMERIC, DEC	17 bytes	Un tipo de datos numérico exacto con valores comprendidos entre 10^{28} - 1 y -10^{28} - 1. Puede definir la precisión (1 - 28) y la escala (0 - precisión definida). La precisión y la escala predeterminadas son 18 y 0, respectivamente.
REAL	SINGLE, FLOAT4, IEEE SINGLE	4 bytes	Un valor en punto flotante de precisión simple con un rango de -3.402823×10^{38} a $-1.401298 \times 10^{-45}$ para valores negativos, 1.401298×10^{-45} a 3.402823×10^{38} para valores positivos, y 0.
FLOAT	DOUBLE, FLOAT8, IEEE DOUBLE, NUMBER	8 bytes	Un valor en punto flotante de doble precisión con un rango de $-1.79769313486232 \times 10^{308}$ a $-4.94065645841247 \times 10^{-324}$ para valores negativos, $4.94065645841247 \times 10^{-324}$ a $1.79769313486232 \times 10^{308}$ para valores positivos, y 0.
SMALLINT	SHORT, INTEGER2	2 bytes	Un entero corto entre – 32.768 y 32.767.
INTEGER	LONG, INT, INTEGER4	4 bytes	Un entero largo entre – 2.147.483.648 y 2.147.483.647.
IMAGE	LONG BINARY	Lo que se requiera	Desde cero hasta un máximo de 2.14 gigabytes. Se utiliza para objetos OLE.
TEXT	LONGTEXT, LONGCHAR, MEMO, NOTE, NTEXT	2 bytes por carácter.	Desde cero hasta un máximo de 2.14 gigabytes.
CHAR	TEXT(n), ALPHANUMERIC CHARACTER, STRING VARCHAR, CHARACTER VARYING, NCHAR	2 bytes por carácter.	Desde cero a 255 caracteres.



Restricciones en el modelo relacional (4)

- ▶ La restricción de **integridad referencial** está especificada entre dos relaciones y es usada para mantener la consistencia entre tuplas de las dos relaciones.
- ▶ Un conjunto de atributos *FK* en un esquema de una relación es una llave *foránea* si:
 - 1.- Los atributos en *FK* tienen el mismo dominio que los atributos de llave primaria del esquema de la relación referenciada.
 - 2.- Un valor de *FK* en una tupla siempre ocurre como un valor de *PK* para alguna tupla en la relación referenciada o es nulo.

Ejemplo



Reglas de Codd (1)

- ▶ Codd se percató de que existían bases de datos en el mercado las cuales decían ser relacionales, pero lo único que hacían era guardar la información en las tablas, sin estar estas tablas literalmente normalizadas; entonces publicó 12 reglas que un verdadero sistema relacional debería tener, aunque en la práctica algunas de ellas son difíciles de realizar. Un sistema podrá considerarse "más relacional" cuanto más siga estas reglas.
- ▶ **Regla No. 1 - La Regla de la información**
 - ▶ Toda la información en un RDBMS está explícitamente representada de una sola manera por valores en una tabla.
 - ▶ Cualquier cosa que no exista en una tabla no existe del todo. Toda la información, incluyendo nombres de tablas, nombres de vistas, nombres de columnas, y los datos de las columnas deben estar almacenados en tablas dentro de las bases de datos. Las tablas que contienen tal información constituyen el Diccionario de Datos. Esto significa que todo tiene que estar almacenado en las tablas.
 - ▶ Toda la información en una base de datos relacional se representa explícitamente en el nivel lógico exactamente de una manera: con valores en tablas. Por tanto los metadatos (diccionario, catálogo) se representan exactamente igual que los datos de usuario. Y puede usarse el mismo lenguaje (ej. SQL) para acceder a los datos y a los metadatos (regla 4).
- ▶ **Regla No. 2 - La regla del acceso garantizado**
 - ▶ Cada ítem de datos debe ser lógicamente accesible al ejecutar una búsqueda que combine el nombre de la tabla, su clave primaria, y el nombre de la columna.
 - ▶ Esto significa que dado un nombre de tabla, dado el valor de la clave primaria, y dado el nombre de la columna requerida, deberá encontrarse uno y solamente un valor. Por esta razón la definición de claves primarias para todas las tablas es prácticamente obligatoria.



Reglas de Codd (2)

- ▶ **Regla No. 3 - Tratamiento sistemático de los valores nulos**
 - ▶ La información inaplicable o faltante puede ser representada a través de valores nulos
 - ▶ Un RDBMS (Sistema Gestor de Bases de Datos Relacionales) debe ser capaz de soportar el uso de valores nulos en el lugar de columnas cuyos valores sean desconocidos.
 - ▶ • Se reconoce la necesidad de la existencia del valor nulo, el cual podría servir para representar, o bien, una información desconocida (ejemplo, no se sabe la dirección de un empleado), o bien una información que no aplica (a un empleado soltero no se le puede asignar un nombre de esposa). Así mismo, consideremos el caso de un alumno que obtiene 0 puntos en una prueba y el de un alumno que no presentó la prueba.
 - ▶ • Hay problemas para soportar los valores nulos en las operaciones relacionales, especialmente en las operaciones lógicas, para lo cual se considera una lógica trivaluada, con tres (no dos) valores de verdad: Verdadero, Falso y null. Se crean tablas de verdad para las operaciones lógicas:

$\text{null AND null} = \text{null}$

$\text{Verdadero AND null} = \text{null}$

$\text{Falso AND null} = \text{Falso}$

$\text{Verdadero OR null} = \text{Verdadero, etc.}$



Reglas de Codd (3)

- ▶ **Regla No. 4 - La regla de la descripción de la base de datos**
 - ▶ La descripción de la base de datos es almacenada de la misma manera que los datos ordinarios, esto es, en tablas y columnas, y debe ser accesible a los usuarios autorizados.
 - ▶ La información de tablas, vistas, permisos de acceso de usuarios autorizados, etc, debe ser almacenada exactamente de la misma manera: En tablas. Estas tablas deben ser accesibles igual que todas las tablas, a través de sentencias de SQL (o similar).
- ▶ **Regla No. 5 - La regla del sub-lenguaje Integral**
 - ▶ Debe haber al menos un lenguaje que sea integral para soportar la definición de datos, manipulación de datos, definición de vistas, restricciones de integridad, y control de autorizaciones y transacciones.
 - ▶ Esto significa que debe haber por lo menos un lenguaje con una sintaxis bien definida que pueda ser usado para administrar completamente la base de datos.
- ▶ **Regla No. 6 - La regla de la actualización de vistas**
 - ▶ Todas las vistas que son teóricamente actualizables, deben ser actualizables por el sistema mismo.
 - ▶ La mayoría de las RDBMS permiten actualizar vistas simples, pero deshabilitan los intentos de actualizar vistas complejas.
- ▶ **Regla No. 7 - La regla de insertar y actualizar**
 - ▶ La capacidad de manejar una base de datos con operandos simples aplica no sólo para la recuperación o consulta de datos, sino también para la inserción, actualización y borrado de datos.
 - ▶ Esto significa que las cláusulas para leer, escribir, eliminar y agregar registros (SELECT, UPDATE, DELETE e INSERT en SQL) deben estar disponibles y operables, independientemente del tipo de relaciones y restricciones que haya entre las tablas o no.



Reglas de Codd (4)

▶ Regla No. 8 - La regla de independencia física

- ▶ El acceso de usuarios a la base de datos a través de terminales o programas de aplicación, debe permanecer consistente lógicamente cuando quiera que haya cambios en los datos almacenados, o sean cambiados los métodos de acceso a los datos.
- ▶ El comportamiento de los programas de aplicación y de la actividad de usuarios vía terminales debería ser predecible basados en la definición lógica de la base de datos, y éste comportamiento debería permanecer inalterado, independientemente de los cambios en la definición física de ésta.

▶ Regla No. 9 - La regla de independencia lógica

- ▶ Los programas de aplicación y las actividades de acceso por terminal deben permanecer lógicamente inalteradas cuando quiera que se hagan cambios (según los permisos asignados) en las tablas de la base de datos.
- ▶ La independencia lógica de los datos especifica que los programas de aplicación y las actividades de terminal deben ser independientes de la estructura lógica, por lo tanto los cambios en la estructura lógica no deben alterar o modificar estos programas de aplicación.

▶ Regla No. 10 - La regla de la independencia de la integridad

- ▶ Todas las restricciones de integridad deben ser definibles en los datos, y almacenables en el catálogo, no en el programa de aplicación.
- ▶ Las reglas de integridad
- ▶ Ningún componente de una clave primaria puede tener valores en blanco o nulos (ésta es la norma básica de integridad).
- ▶ Para cada valor de clave foránea deberá existir un valor de clave primaria concordante. La combinación de estas reglas aseguran que haya integridad referencial.



Reglas de Codd (5)

▶ Regla No. 11 - La regla de la distribución

- ▶ El sistema debe poseer un lenguaje de datos que pueda soportar que la base de datos esté distribuida físicamente en distintos lugares sin que esto afecte o altere a los programas de aplicación.
- ▶ El soporte para bases de datos distribuidas significa que una colección arbitraria de relaciones, bases de datos corriendo en una mezcla de distintas máquinas y distintos sistemas operativos y que esté conectada por una variedad de redes, pueda funcionar como si estuviera disponible como en una única base de datos en una sola máquina.

▶ Regla No. 12 - Regla de la no-subversión

- ▶ Si el sistema tiene lenguajes de bajo nivel, estos lenguajes de ninguna manera pueden ser usados para violar la integridad de las reglas y restricciones expresadas en un lenguaje de alto nivel (como SQL).
- ▶ Algunos productos solamente construyen una interfaz relacional para sus bases de datos No relacionales, lo que hace posible la subversión (violación) de las restricciones de integridad. Esto no debe ser permitido.

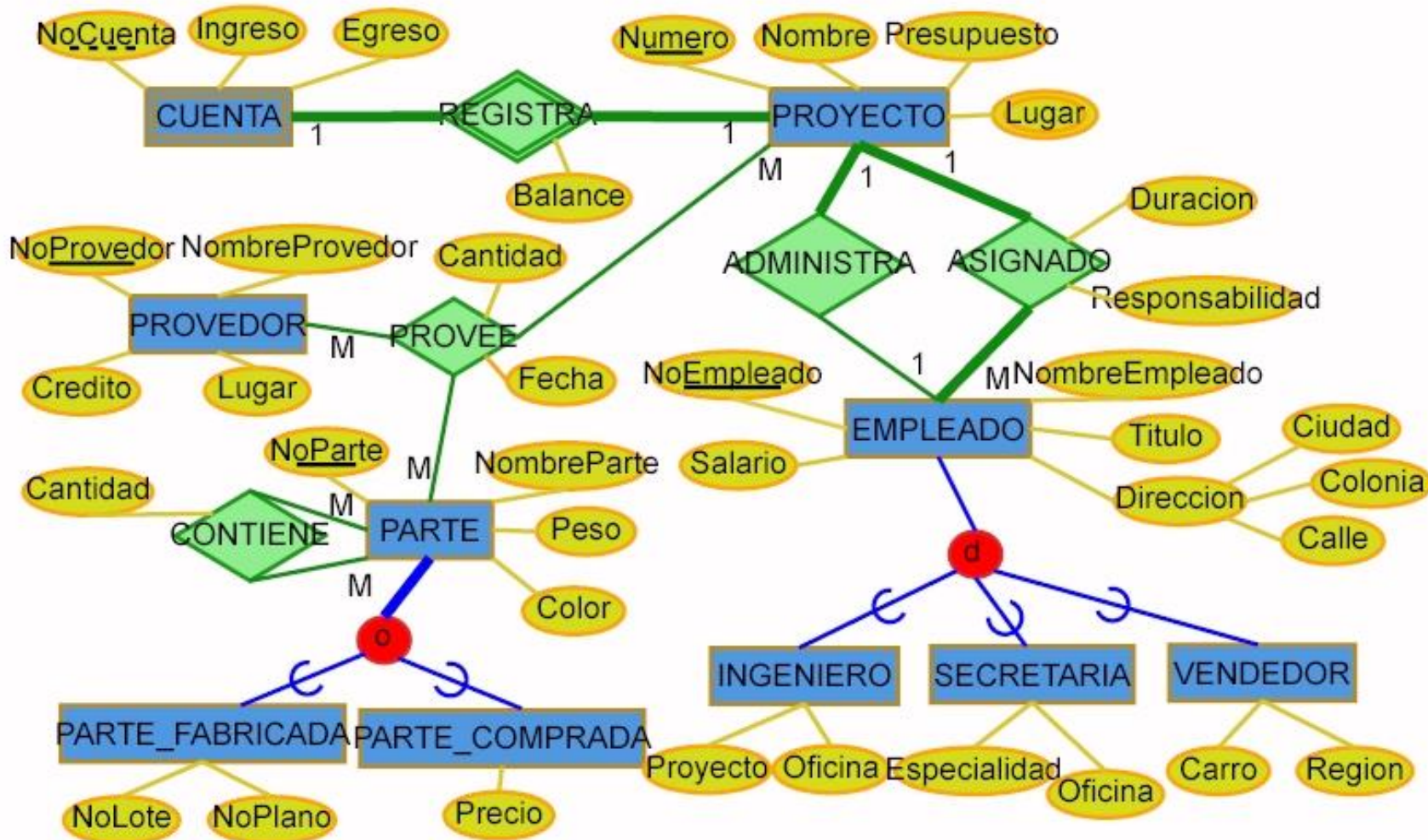


Principios de diseño relacional

- ▶ Las relaciones deben tener una unidad semántica
- ▶ La repetición de información debe de ser evitada
 - ▶ Anomalías: de inserción, de eliminación y de actualización
- ▶ Evitar los valores nulos tanto como sea posible
 - ▶ Es difícil la interpretación
 - ▶ No se sabe, no tiene importancia, es conocido pero no disponible, no aplica
 - ▶ Produce dificultades en las reuniones
- ▶ Tratar de evitar las reuniones no necesarias



4.2 Transformación de modelo EA a Relacional



Paso 1- Manejo de entidades

- ▶ Para cada tipo de entidad regular E en el esquema EA, cree una tabla R.
 - ▶ Incluya como atributos de R sólo los atributos simples de E.
 - ▶ Para los atributos compuestos de E, simplemente incluya sus atributos simples constitutivos en R.
 - ▶ El atributo identificador de E se vuelve la llave primaria de R. Si hay más de un atributo llave de E, entonces elegir uno como la llave primaria de R.



Paso 1 - Ejemplo

- ▶ Cree los siguientes esquemas relacionales.

- ▶ Las llaves están subrayadas.

Empleado(NoEmpleado, NombreEmpleado, Titulo, Salario, Calle, Colonia, Ciudad)

Proyecto (Numero, Nombre, Presupuesto)

Provedor(NoProvedor, NombreProvedor, Credito, Lugar)

Parte (NoParte, NombreParte, Peso, Color).



Paso 2 – Entidades débiles

- ▶ Para cada tipo de la entidad débil W asociada con el tipo de entidad fuerte E en el esquema EA, crear una tabla R .
 - ▶ Los atributos de R son los atributos simples de W (o las versiones simplificadas de atributos compuestos).
 - ▶ Incluya entre los atributos de R todos los atributos importantes de entidad fuerte E . Estos sirven como llaves foráneas de R .
 - ▶ La llave primaria de R es la combinación de la llave primaria de E y la llave parcial de W .



Paso 2 - Ejemplo

- ▶ **Ejemplo:**

- ▶ Crear la relación *Cuenta* como sigue:

- ▶ Cuenta(NumProy, NoCuenta, Ingreso, Egreso)



Llave foránea



Paso 3 – Relaciones 1:1

- ▶ Para cada relación 1:1 R en el esquema EA, donde las dos entidades relacionadas son E_1 y E_2 , establezca las tablas S y T que correspondan a E_1 y E_2 respectivamente.
 - ▶ Elija una de las tablas, preferentemente una cuya participación en R sea total (digamos S). Incluya en S la llave primaria de T como llave foránea.
 - ▶ También, si hay atributos en la tabla R , inclúyalos en S .
 - ▶ Se pueden renombrar los atributos para estas acciones.



Paso 3 - Ejemplo

- ▶ Para la relación 1:1 de *Administra* entre las entidades *Empleado* y *Proyecto*:
 - ▶ Elija *Proyecto* como S, ya que su participación en la relación *Administra* es total (cada proyecto tiene un administrador, pero cada empleado no necesita administrar un proyecto). Entonces, incluya en *Proyecto* la llave primaria de *Empleado*.
 - ▶ Proyecto(Numero, Nombre, Presupuesto, Manager)
- ▶ Para la asociación 1:1 de *Registra* entre las entidades *Proyecto* y *Cuenta*:
 - ▶ Elija *Cuenta* como S (nota: *Cuenta* es una entidad débil, así que es la única opción que tiene sentido)
 - ▶ Incluya *Numero* en *Cuenta* (que fue hecho en el paso 2) y Balance
 - ▶ Cuenta(NumeroProy, NoCuenta, Ingreso, Egreso, Balance)



Paso 4 – Relaciones 1:N

- ▶ Para cada relación 1:N binaria regular (no débil) en el esquema EA, identificar la tabla S que corresponda al tipo de entidad en el lado N de la relación. Deje la otra tabla en el lado 1 (uno) como T.
- ▶ Incluya en S como llave foránea la llave primaria de T.
- ▶ Si hay atributos en la relación R, inclúyalos en S también.



Paso 4 - Ejemplo

- ▶ Tenemos sólo la relación *Asignado* a considerar. Está definida entre *Proyecto* y *Empleado*
 - ▶ El lado N de la relación es *Empleado*; el lado I es *Proyecto*
 - ▶ Incluya en *Empleado*
 - ▶ La llave primaria (*Numero*) de *Proyecto*
 - ▶ Los atributos de la relación *Asignado* (*Duracion* y *Responsabilidad*)
- Empleado(NoEmpleado, NombreEmpleado, Titulo, Salario, Calle, Colonia, Ciudad, **NumProy, Duracion, Responsabilidad**)



Paso 4 – Relaciones 1:N

- ▶ Si esto es un problema, entonces crear una nueva tabla S correspondiente a la relación R e incluya en S la llave primaria de las dos entidades tal que R ligue en adición a sus propios atributos. La llave primaria de S es la combinación de las llaves primarias de las dos entidades.
- ▶ En nuestro caso, tendremos
Asignado(NoEmpleado, NumProyecto, Duracion, Responsabilidad)



Paso 5 – Relaciones M:N

- ▶ Para cada relación binaria M:N conectando entidades E_1 y E_2 en el esquema EA, crear una tabla S.
 - ▶ Incluya como llaves foráneas de S, las llaves primarias de las dos tablas que correspondan a E_1 y E_2 .
 - ▶ Estos atributos, juntos, forman la llave primaria de S.
 - ▶ También incluya en S cualquier atributo de la relación R.



Paso 5 - Ejemplo

- ▶ Tenemos una relación M:N a considerar: *Contiene*, la cual es una relación recursiva sobre la entidad *Parte*.
 - ▶ Creamos la siguiente relación:
Contiene(NoParte1, NoParte2, Cantidad)



Paso 6 – Atributos multivalor

- ▶ Para cada atributo multivalor A , crear una nueva tabla R .
 - ▶ Estos atributos de R son A (si componen, entonces los componentes simples)
 - ▶ También incluya en R la llave primaria K de la entidad que contiene A .
 - ▶ La llave primaria de R entonces serán K y A juntas.



Paso 6 - Ejemplo

- ▶ En nuestro ejemplo, tenemos que crear una nueva tabla para el atributo multivalor *Lugar* en *Proyecto*.
 - ▶ Esta relación es creada como sigue:
Lugar(NoProyecto, NombreLugar)



Paso 7 – Relaciones de orden mayor

- ▶ Para cada relación de orden mayor conectadas como E_1, E_2, \dots, E_n en el esquema EA, crear una tabla S.
 - ▶ Incluya en S las llaves primarias de las tablas correspondientes a E_1, E_2, \dots, E_n
 - ▶ También incluya en S cualquier atributo de R.
 - ▶ La llave primaria de S es la combinación de las llaves primarias de las tablas correspondientes a E_1, E_2, \dots, E_n .



Paso 7 - Ejemplo

- ▶ La única asociación de orden mayor es *Provee* entre *Proveedor*, *Proyecto* y *Parte*.

- ▶ Crear la tabla *Provee*

Provee(NoParte, NoProy, NoProvedor, Cantidad, Fecha)



Paso 8 - Especialización

- ▶ Para cada especialización con m subclases $\{S_1, \dots, S_m\}$ y un superclase generalizada C , donde los atributos de C son $\{k, A_1, \dots, A_n\}$ (k es la llave primaria), convierta de acuerdo con lo siguiente:
 - ▶ ① Caso general:
 - ▶ Crear una tabla T para C con atributos $\{k, A_1, \dots, A_n\}$ y use k como la llave primaria.
 - ▶ Crear una tabla U_i para cada S_i . Incluya en U_i todos los atributos de S_i y k . Use k como la llave primaria de U_i .



Paso 8 - Especialización

- ▶ ② No hay asociación de superclase:

- ▶ Crear una tabla U_i para cada S_i . Incluya en U_i todos los atributos de S_i y $\{k, A_1, \dots, A_n\}$. Use k como la llave primaria de U_i .

- ▶ ③ Para subclases disjuntas:

- ▶ Crear una tabla única U la cual contiene todos los atributos de todos los S_i y $\{k, A_1, \dots, A_n\}$ y t . Use k como la llave primaria de U . El atributo t indica el tipo de atributo de acuerdo a qué especialización es hecha.



Paso 8 - Especialización

- ▶ ④ Para subclases traslapadas:
 - ▶ Crear una tabla única U la cual contiene todos los atributos de todos los S_i y $\{k, A_1, \dots, A_n\}$ y $\{t_1, \dots, t_m\}$. Use k como la llave primaria de U . Los atributos t_i son evaluados de forma booleana, indicando si la tupla pertenece a la subclase S_i .
 - ▶ Nota: se pueden generar un largo número de valores nulos en la relación.



Paso 8 - Ejemplo

► Especialización de *Empleado*

- La relación *Empleado* ya existe; la opción 2 no es válida
- La especialización es de disyunción; la opción 4 no es válida
- Las opciones 1 o 3 son posibles:

- Opción 1:

Ingeniero(NoEmpleado, Proyecto, Oficina)

Secretaria(NoEmpleado, Oficina, Especialidad)

Vendedor(NoEmpleado, Carro, Region)

- Opción 3:

Empleado(NoEmpleado, NombreEmpleado, Titulo, Salario, Calle, Colonia, Ciudad, NumProy, Duracion, Responsabilidad, **Tipo**, Proyecto, Oficina, Especialidad, Carro, Region)



Paso 8- Ejemplo

► Especialización de Parte

- La relación *Parte* ya existe; la opción 2 no es válida
- La especialización está traslapada; la opción 3 no es válida
- Las opciones 1 o 4 son posibles:

- Opción 1:

Parte_Fabricada(NoParte, NoLote, NoPlano)

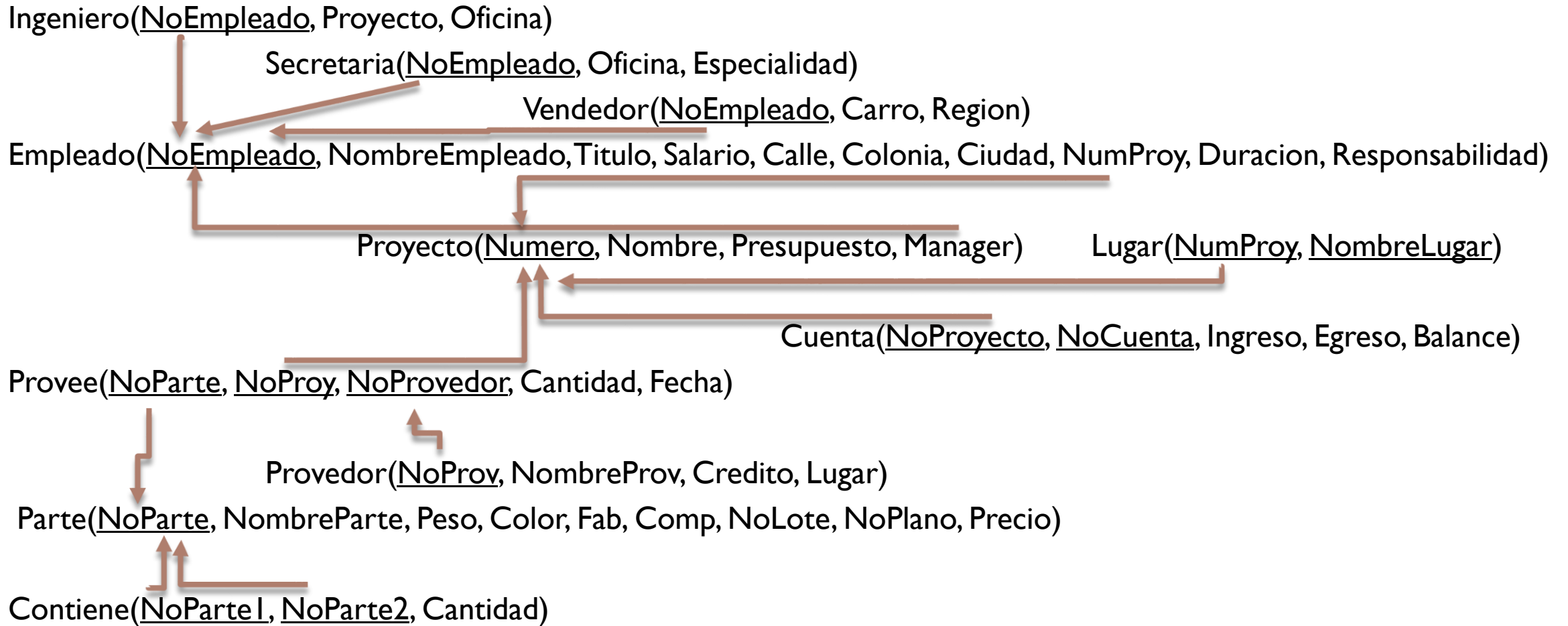
Parte_Comprada(NoParte, Precio)

- Opción 4:

Parte(NoParte, NombreParte, Peso, Color, **Fab**, **Comp**, NoLote, NoPlano, Precio)



Conjunto final de tablas



4.3 Normalización

- ▶ Es el proceso de separación conceptual, el cual produce un esquema por refinamientos subsecuentes y descomposiciones
- ▶ No se combinan conjuntos no relacionados de hechos en una tabla; cada relación debe contener un conjunto de hechos independiente
- ▶ Surge del artículo “Further Normalization of the DataBase Relational Model” de E. F. Codd, en 1972
- ▶ Se pretende tomar cada relación individualmente y “mejorarla” en términos de las características deseadas
 - ▶ Formas Normales
 - ▶ Sólo valores atómicos (1FN)
 - ▶ Dependencias funcionales (2FN, 3FN, BCFN)
 - ▶ Dependencias multivaluadas (4FN)
 - ▶ Dependencias de reunión (5FN)



4.3.1 Anomalías

▶ **Inserción**

- ▶ Es difícil (o imposible) almacenar información de un nuevo proyecto a menos de que haya un empleado asignado a él.

▶ **Eliminación**

- ▶ Si un empleado deja la compañía, y es el único asignado a un proyecto, no puede ser eliminado porque se perdería la información acerca del proyecto

▶ **Modificación**

- ▶ Si un atributo de un proyecto es modificado, todas las tuplas de todos los empleados que trabajan en ese proyecto necesitan ser modificadas



Suponer...

- Para guardar los empleados y los proyectos involucrados...

Emp_proy(NoEmp, NombEmp, Titulo, Salario, NoProy, NombProy, Presup, Duracion, Resp)

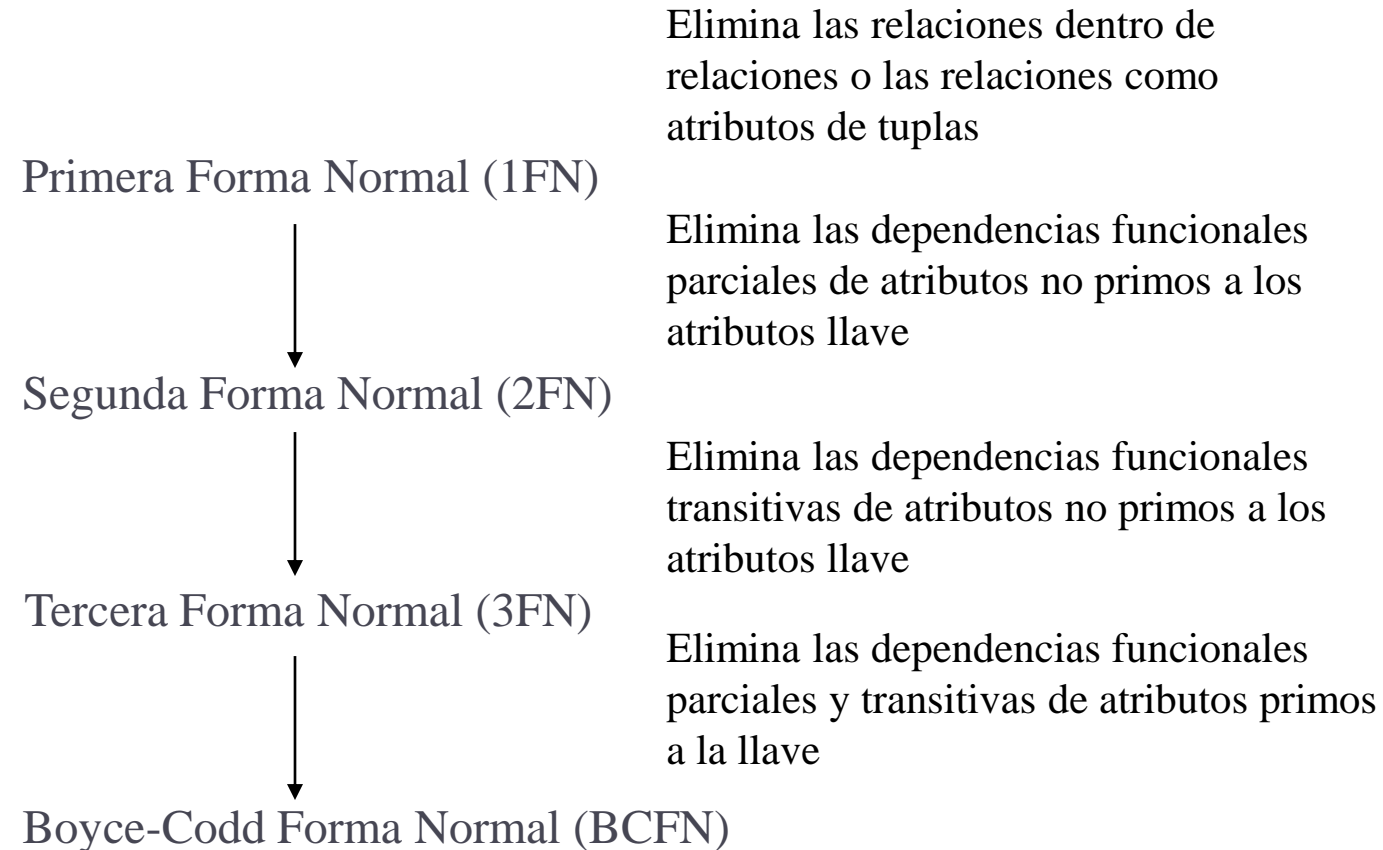
NoEmp	NombEmp	Titulo	Salario	NoProy	NombProy	Presup	Duracion	Resp
E1	J. Doe	Eléctrico Ing.	40000	J1	Instrumentacion	150000	12	Administrador
E2	M. Smith	Analista Sistemas	34000	J1	Instrumentacion	150000	24	Analista
E2	M. Smith	Analista Sistemas	34000	J2	Desarrollador Database	135000	6	Analista
E3	A. Lee	Ing. Mecanico	27000	J3	CAD/CAM	250000	10	Consultor
E3	A. Lee	Ing. Mecanico	27000	J4	Mantenimiento	310000	48	Programador
E4	J. Miller	Programador	24000	J2	Desarrollador Database	135000	18	Analista
E5	B. Casey	Analista Sistemas	34000	J2	Desarrollador Database	135000	24	Administrador
E6	L. Chu	Ing. Eléctrico	40000	J4	Mantenimiento	310000	48	Administrador
E7	R. Davis	Ing. Mecanico	27000	J3	CAD/CAM	250000	36	Ingeniero
E8	J. Jones	Sistemas Anal	34000	J3	CAD/CAM	250000	40	Administrador

Repetición de la información

- ▶ Los valores de los atributos *Título*, *Salario* y *Presup* están repetidos para cada proyecto en que un empleado esté involucrado
 - ▶ Desperdicio de espacio
 - ▶ Actualizaciones complicadas

NoEmp	NombEmp	Título	Salario	NoProy	NombProy	Presup	Duracion	Resp
E1	J. Doe	Eléctrico Ing.	40000	J1	Instrumentacion	150000	12	Administrador
E2	M. Smith	Analista Sistemas	34000	J1	Instrumentacion	150000	24	Analista
E2	M. Smith	Analista Sistemas	34000	J2	Desarrollador Database	135000	6	Analista
E3	A. Lee	Ing. Mecanico	27000	J3	CAD/CAM	250000	10	Consultor
E3	A. Lee	Ing. Mecanico	27000	J4	Mantenimiento	310000	48	Programador
E4	J. Miller	Programador	24000	J2	Desarrollador Database	135000	18	Analista
E5	B. Casey	Analista Sistemas	34000	J2	Desarrollador Database	135000	24	Administrador
E6	L. Chu	Ing. Eléctrico	40000	J4	Mantenimiento	310000	48	Administrador
E7	R. Davis	Ing. Mecanico	27000	J3	CAD/CAM	250000	36	Ingeniero
E8	J. Jones	Sistemas Anal	34000	J3	CAD/CAM	250000	40	Administrador

4.3.2 Pasos de la normalización



Formas Normales

Forma normal	Prueba	Normalización
Primera (1FN)	La relación no debe tener atributos no atómicos o relaciones anidadas	Se forman nuevas relaciones para cada valor no atómico o relación anidada
Segunda (2FN)	Para relaciones donde la llave primaria contiene varios atributos, los atributos no llave no deben ser funcionalmente dependientes en una parte de la llave primaria	Se descompone y establece una nueva relación para cada llave parcial con sus atributos dependientes. Se debe asegurar mantener la relación con la llave primaria y cualquier atributo que sea dependiente funcional completo en este
Tercera (3FN)	La relación no debe tener atributos no llave funcionalmente determinados por otro atributo no llave (o por un conjunto de atributos no llave). Esto es, no debe haber dependencia transitiva del atributo no llave en la llave primaria	Se descompone y establece una relación que incluya los atributos no llaves que funcionalmente determinen otros atributos no llave

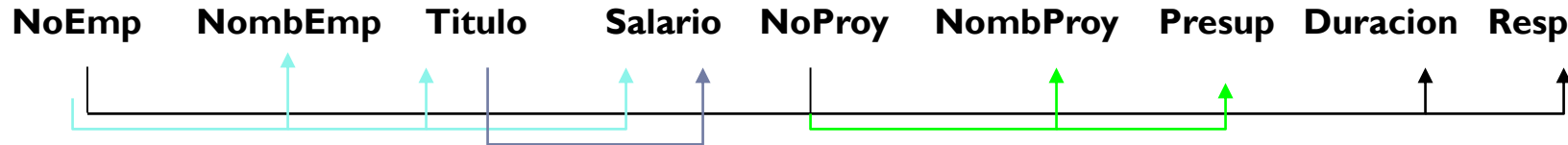
Criterios de la normalización

- ▶ ¿Como descomponer un esquema en la forma normal deseable?
- ▶ ¿Que criterio debe seguir el esquema descompuesto de manera de conservar la semántica del esquema original?
 - ▶ **Resconstructibilidad:** recuperar la relación original sin tener reuniones innecesarias
 - ▶ **Descomposición sin pérdidas:** no hay perdida de información.
 - ▶ **Conservación de dependencias:** las restricciones que estén en la relación original deben forzarse mediante las restricciones definidas en las relaciones descompuestas.
 - ▶ Sea F_i el conjunto de dependencias F^+ que incluyan sólo atributos en R_i . Preferentemente la descomposición debe preservar las dependencias, esto es, $(F_1 \cup F_2 \cup \dots \cup F_n)^+ = F^+$
 - ▶ De otra forma, la validación de actualizaciones para la violación de las dependencias funcionales puede requerir del cálculo de uniones, lo cual es costoso.
- ▶ ¿Que sucede con las consultas?
 - ▶ El tiempo de procesamiento se incrementa debido a las reuniones



4.3.3 Dependencias Funcionales

- ▶ El atributo B de una relación R es funcionalmente dependiente del atributo A de R sí, en cada instante, cada valor de A está asociado con no más de un valor de B dentro de la relación R
- ▶ Definición:
 - ▶ Dada una relación R definida sobre $U=\{A_1, A_2, \dots, A_n\}$ donde $X \subseteq U, Y \subseteq U$. Si, para todos los pares de tuplas t_1 y t_2 en cualquier instancia legal del esquema de la relación R , $t_1[X] = t_2[X] \Rightarrow t_1[Y] = t_2[Y]$, entonces hay una dependencia funcional $X \rightarrow Y$ en R
- ▶ Ejemplo:
 - ▶ En la relación EMP_PROY
 - ▶ $(\text{NoEmp}, \text{NoProy}) \rightarrow (\text{Duracion}, \text{Resp})$
 - ▶ $\text{NoEmp} \rightarrow (\text{NombEmp}, \text{Titulo}, \text{Salario})$
 - ▶ $\text{NoProy} \rightarrow (\text{NombProy}, \text{Presup})$
 - ▶ $\text{Titulo} \rightarrow \text{Salario}$



Dependencias funcionales

- ▶ K es una superllave de un esquema de una relación R , sí y sólo sí $K \rightarrow R$
- ▶ K es una llave candidata de R sí y sólo sí
 - ▶ $K \rightarrow R$, y
 - ▶ para $A \subset K, A \not\rightarrow R$
- ▶ Las dependencias funcionales permiten expresar restricciones que no pueden ser expresadas usando superllaves.
- ▶ **Atributos**
 - ▶ Un atributo primo es miembro de cualquier llave
 - ▶ Un atributo no primo es un atributo que no es miembro de una llave
- ▶ Una dependencia funcional es trivial si es satisfecha para todas las relaciones.
 - ▶ En general, $\alpha \rightarrow \beta$ es trivial si $\beta \subseteq \alpha$



Dependencias funcionales

- ▶ Dependencia funcional completa (DFC)
 - ▶ Una DF es completa si X es mínimo, esto es, quitando cualquier atributo A de X significa que la dependencia no existe más
- ▶ Dependencia funcional parcial (DFP)
 - ▶ Si se elimina algún atributo A de X , la dependencia aún permanece
- ▶ Dependencia transitiva
 - ▶ Si existe un conjunto de atributos Z de X , que no es llave candidata ni es un subconjunto de cualquier llave de X , y $X \rightarrow Z$ y $Z \rightarrow Y$, entonces $X \rightarrow Y$



Reglas de Inferencia para dependencias funcionales

- ▶ Es posible inferir DF de un conjunto de DF (F) si $X \rightarrow Y$ cumple para cada estado de una relación
- ▶ La cerradura F^+ de F es el conjunto de todas las DF que pueden ser inferidas de F .
 - ▶ **R1**. Regla Reflexiva: Si $X \supseteq Y$, entonces $X \rightarrow Y$
 - ▶ **R2**. Regla de aumentación: $\{X \rightarrow Y\} \models XZ \rightarrow YZ$
 - ▶ **R3**. Regla transitiva: $\{X \rightarrow Y, Y \rightarrow Z\} \models X \rightarrow Z$
 - ▶ **R4**. Regla de descomposición: $\{X \rightarrow YZ\} \models X \rightarrow Y$
 - ▶ **R5**. Regla aditiva: $\{X \rightarrow Y, X \rightarrow Z\} \models X \rightarrow YZ$
 - ▶ **R6**. Regla pseudotransitiva: $\{X \rightarrow Y, WY \rightarrow Z\} \models WX \rightarrow Z$
- ▶ Las reglas $R1$ a $R3$ son conocidas como los axiomas de Argmstrong



Cerradura de un descriptor

- ▶ **F^+** : Conjunto de todos los atributos determinados funcionalmente.
($F \rightarrow F^+$)
- ▶ **Algoritmo de Cierre de un descriptor respecto a un DF:**
 - ▶ **Entrada:** R, DF. (suponemos que los atributos de partida son todos los contenidos en DF y R es un subconjunto de ellos)
 - ▶ **Salida:** R^+
 - ▶ **Proceso:**
 - 1) $R^+ := R$
 - 2) Repetir (hasta que no se añadan más atributos a R^+):
 - 2.1) Para cada $X \rightarrow Y$ en DF:
Si $X \subseteq R^+$ e $Y \not\subseteq R^+$ entonces $R^+ := R^+ \cup Y$



4.3.4 Formas Normales

- ▶ Inicialmente, Codd propuso en 1970 tres formas normales basadas en las DF: **primera** (1FN), **segunda** (2FN) y **tercera** forma normal (3FN), CODD (1970).
- ▶ Debido a que en 3FN aún persisten algunos problemas en las relaciones, en 1974 Codd y Boyce introdujeron una definición más restrictiva de la tercera forma normal, que se denominó Forma Normal de **Boyce-Codd** (FNBC).
- ▶ En 1977 y 1979 Fagin introduce la **cuarta** (4FN) y **quinta** (5FN) formas normales respectivamente. Ambas están basadas en otro tipo de dependencias distintas de las funcionales: las dependencias multivaluadas (4FN) y las dependencias de combinación (5FN).




Primera Forma Normal (1FN)

- ▶ Todos los valores de los atributos son atómicos
- ▶ Una relación en 1FN no puede tener un valor de atributo que sea:
 - ▶ Un conjunto de valores
 - ▶ Una tupla de valores (relación anidada)
- ▶ Es una suposición estándar en los SABD relacionales
- ▶ En los SABD orientados a objetos esta suposición es relajada



Ejemplo

- ▶ Supongamos:
Departamentos(DNomb, DNo, Locaciones)
en donde *Locaciones* es un atributo multivalor
- ▶ Tres posibles soluciones:
 - ▶ Quitar el atributo *Locaciones* y ponerlo en otra relación, junto con la llave primaria *DNomb*
 - ▶ Repetir cada tupla de *Departamentos* para cada valor de *Locaciones* (esto viola la no duplicidad de la llave primaria)
 - ▶ Establecer un nuevo atributo (digamos *Locacion 1*, ..., *Locacion N*) para cada valor de *Locaciones* (esto es impráctico, porque no se sabe cuál es el número máximo de valores, además de que generan valores nulos)
- ▶ Solución:
 - ▶ Se toma la primera opción y se tiene:
Departamentos(DNomb, DNo)

Locaciones(DNomb, Loc)



Segunda Forma Normal (2FN)

- ▶ Un esquema de una relación está en *2FN* si está en *1FN* y si cada atributo no primo *A* en *R* no es parcialmente dependiente (o es completamente dependiente) de cualquier llave candidata de *R*
- ▶ Las dependencias funcionales parciales causan problemas
- ▶ La *2FN* es importancia histórica, ya que es sobrepuesta por la *3FN*
- ▶ En el ejemplo, *Emp_Proj* no está en *2FN*, por lo que se descompone como:

Empleado(NoEmp, NombEmp, Titulo, Salario)

Asignado(NoEmp, NoProy, Duracion, Resp)

Proyecto(NoProy, NombProy, Presup)



Ejemplo

- ▶ Suponga la definición de la tabla matricula como sigue:

Matricula (cveAlumno, apellidos, nombre, nota, IdCurso, curso, lugar, aula)

- ▶ En donde

- ▶ El atributo *IdCurso* es el identificador de los cursos donde esta matriculado el alumno
 - ▶ El atributo *cveAlumno* es el identificador de los datos del alumno
 - ▶ El atributo *nota* es la evaluación asignado al alumno con respecto al curso.
 - ▶ El atributo *aula* representa el aula donde se imparte el curso
 - ▶ El atributo *lugar* representa los lugares de estudio en donde se imparten los cursos.
- ▶ Dado que existe una dependencia funcional no completa entre atributos que no forman parte de la llave, no se encuentra en la 2FN



Dependencias funcionales

- ▶ Dado que, de acuerdo al problema, se tiene que el identificador del alumno determina los datos del propio alumno, se puede escribir:
 - ▶ $cveAlumno \rightarrow \text{apellidos, nombre}$
- ▶ De igual forma, y el identificador del curso determina el curso, por lo que se tiene:
 - ▶ $idCurso \rightarrow \text{curso}$
- ▶ Y dado que en conjunto el alumno y el curso determinan la calificación obtenida, se tiene:
 - ▶ $cveAlumno, idCurso \rightarrow \text{nota}$
- ▶ Por lo que el conjunto de dependencias funcionales será:
 - ▶ $cveAlumno \rightarrow \text{apellidos, nombre}$
 - ▶ $idCurso \rightarrow \text{curso}$
 - ▶ $cveAlumno, idCurso \rightarrow \text{nota, lugar, aula}$



Ejemplo

- ▶ Se descompone la relación en:

Imparte (IdCurso, curso)

Matricula (cveAlumno, IdCurso, apellidos, nombre, nota, lugar, aula)

- ▶ De aquí, *Imparte* queda en 2FN, en donde el atributo primo *curso* depende completamente de la llave.
- ▶ Sin embargo, se observa que en *Matricula* existe la DF $cveAlumno \rightarrow apellidos, nombre$, por lo que se procede a descomponer la relación en:

Alumno (cveAlumno, apellidos, nombre)

Matricula (cveAlumno, IdCurso, nota, lugar, aula)

- ▶ Lo que finalmente produce:

Imparte (IdCurso, curso)

Matricula (cveAlumno, IdCurso, nota, lugar, aula)

Alumno (cveAlumno, apellidos, nombre)

- ▶ que cumple la 2FN



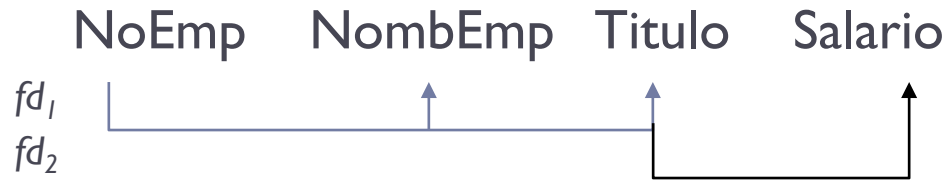
Tercera Forma Normal (3FN)

- ▶ Una relación está en *3FN* si está en *2FN* y si no hay atributos no primos de *R* transitivamente dependientes en la llave primaria
- ▶ Lo necesario es eliminar las dependencias transitivas (la ausencia de dependencias transitivas garantiza la ausencia de dependencias funcionales parciales)
- ▶ Formalmente:
 - ▶ Un esquema de una relación *R* definido sobre $U=\{A_1, A_2, \dots, A_n\}$ está en *3FN* si para todas las dependencias funcionales que estén en *R* de la forma $X \rightarrow Y$, donde $X \subseteq U$ y $Y \subseteq U$, al menos una de las siguientes opciones se cumple:
 - ▶ $X \rightarrow Y$ es una dependencia funcional trivial (esto es, $Y \subseteq X$)
 - ▶ *X* es una superllave de *R*
 - ▶ *X* no es superllave y cada atributo $A \in (Y - X)$ está contenido en una clave candidata de *R*



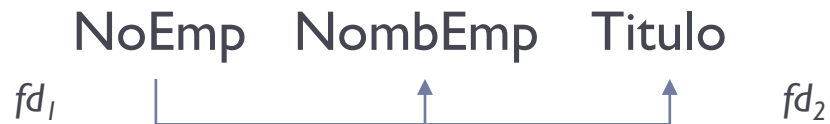
Ejemplo de 3FN

Empleado



- ▶ *Empleado* no está en 3FN por fd_2
 - ▶ *Titulo* \rightarrow *Salario* pero *Titulo* no es una superllave y *Salario* es no primo
 - ▶ El problema es que *NoEmp* transitivamente determina *Salario*
- ▶ Solución:

Empleado



Pago



Ejemplo

- ▶ Regresando al ejemplo de la 2FN, observemos la relación

Matricula:

Matricula (cveAlumno, IdCurso, nota, lugar, aula)

- ▶ Si se considera que cada aula esta ubicada en un sólo lugar, se tiene una dependencia funcional entre *lugar* y *aula*, además de la DF completa entre estos y la clave primaria de *Matricula*.
- ▶ *cveAlumno, IdCurso* \rightarrow *nota*
- ▶ *lugar* \rightarrow *aula*



Ejemplo

- ▶ De aquí, se descompone en las siguientes relaciones:

Ubicación (lugar, aula)

Matricula (cveAlumno, IdCurso, nota, lugar)

- ▶ Lo cual elimina las DF transitivas. Las relaciones finales quedan como sigue:

Imparte (IdCurso, curso)

Alumno (cveAlumno, apellidos, nombre)

Ubicación (lugar, aula)

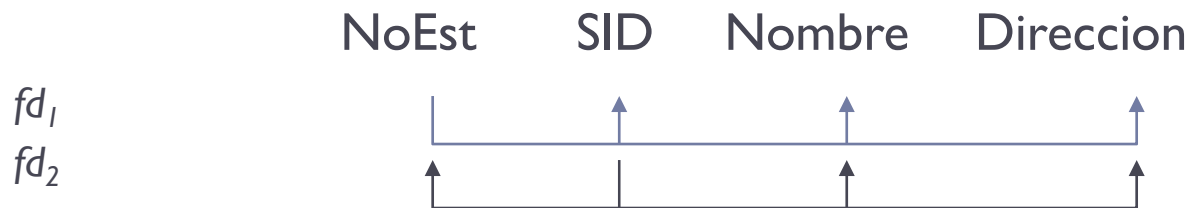
Matricula (IdCurso, cveAlumno, nota, lugar)



Normalización

- ▶ Las formas normales fueron originalmente definidas en términos de sus llaves primarias (una relación está en 3FN si no hay dependencias parciales o transitivas en la llave primaria)
- ▶ La siguiente relación no satisface la definición de 3FN (donde SID es un número de seguro social único)

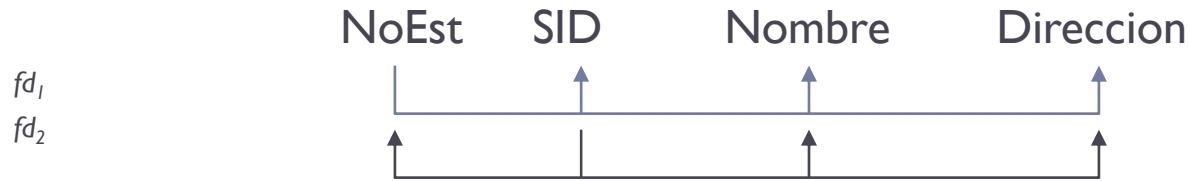
Estudiante



Normalización

- ▶ Sin embargo, no parece que haya nada mal en el esquema de la relación (no conlleva a ninguna anomalía)

Estudiante



- ▶ La razón de la falta de anomalías es que SID es una llave candidata, y cualquier atributo funcionalmente dependiente completo en la llave primaria también será completamente funcionalmente dependiente de la llave candidato (o llaves)



Forma Normal de Boyce-Codd

- ▶ Todavía se pueden seguir teniendo dependencias transitivas en 3FN si los atributos dependientes son primos
- ▶ Un esquema de relación R está en BCFN si para cada dependencia funcional no trivial $X \rightarrow Y$, X es una superllave
- ▶ Propiedades:
 - ▶ Todos los atributos no primos son completamente dependientes en cada llave
 - ▶ Ningún atributo es dependiente no trivial en cualquier conjunto de atributos no primos



Forma Normal de Boyce-Codd

- ▶ Una definición más fuerte de la 3FN (referenciada como BCFN) es:
 - ▶ Una relación R está en BCFN si para cualquier dependencia funcional $X \rightarrow A$ en R , entonces X es una superllave de R
- ▶ Mientras que la 3FN permite la dependencia funcional de la forma siguiente, BCFN no permite la existencia de tales dependencias funcionales

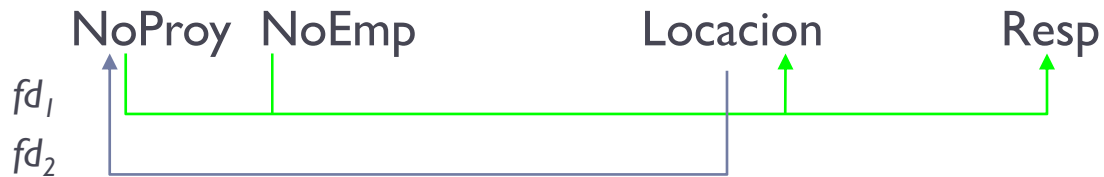
Relación



Ejemplo de BCFN

- ▶ Asuma la siguiente definición de la relación *Proyecto* con:
 - ▶ Cada empleado en un proyecto tiene una única localización y responsabilidad con respecto a ese proyecto, y
 - ▶ Sólo un proyecto puede ser encontrado en cada localización
- ▶ Las DF podrían ser:

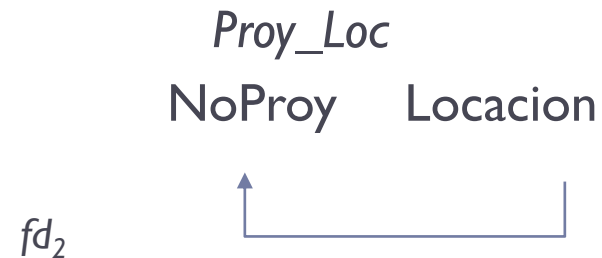
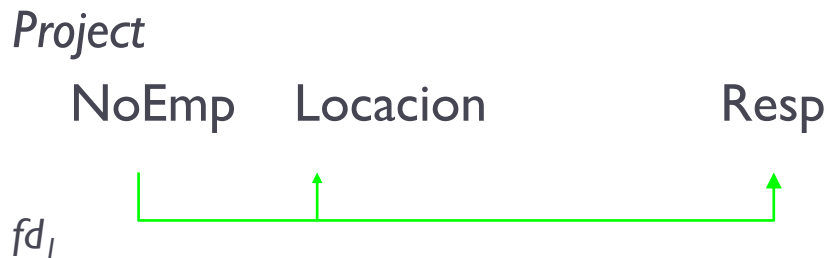
Proyecto



que deja *Project* en 3FN pero no en BCFN

Descomposición en BCFN

- ▶ fd_1 cumple con 3FN, porque *NoProy* y *NoEmp* es la superllave
- ▶ fd_2 cumple con 3FN, porque *NoProy* es primo; sin embargo *Locacion* no es una superllave
- ▶ Solución:



Ejemplo

- ▶ Suponga la siguiente relación Matricula:
Matricula (cveAlumno, IdCurso, apellidos, nombre, nota, curso, lugar, aula)
- ▶ en donde se considera a *IdCurso*, *apellidos*, *nombre* como llave candidata.
- ▶ De aquí, hay dos determinantes funcionales:
cveAlumno, *IdCurso* y *IdCurso*, *apellidos*, *nombre*.
- ▶ Las DF encontradas son:
 1. $\text{IdCurso} \rightarrow \text{curso}$
 2. $\text{cveAlumno}, \text{IdCurso} \rightarrow \text{lugar}$
 3. $\text{cveAlumno}, \text{IdCurso} \rightarrow \text{aula}$
 4. $\text{cveAlumno}, \text{IdCurso} \rightarrow \text{nota}$
 5. $\text{apellidos}, \text{nombre}, \text{IdCurso} \rightarrow \text{lugar}$
 6. $\text{apellidos}, \text{nombre}, \text{IdCurso} \rightarrow \text{aula}$
 7. $\text{apellidos}, \text{nombre}, \text{IdCurso} \rightarrow \text{nota}$
 8. $\text{lugar} \rightarrow \text{aula}$
 9. $\text{cveAlumno}, \text{IdCurso} \rightarrow \text{apellidos}, \text{nombre}, \text{IdCurso}$



Ejemplo

- ▶ Tomando en consideración las DF 1 y 8, se descompone la relación *Matricula* de la siguiente forma:

Imparte (IdCurso, curso)

Ubicación (lugar, aula)

Matricula (cveAlumno, IdCurso, apellidos, nombre, nota, lugar)

- ▶ Lo cual las deja en 3FN y BCFN, excepto la relación *Matricula* que no está en la BCFN, pues se tiene la DF $cveAlumno \leftrightarrow apellidos, nombre$. De aquí que *Matricula* se descompone de la sig. forma:

Alumno (cveAlumno, apellidos, nombre)

Matricula (cveAlumno, IdCurso, nota, lugar)



4.3.5 Otras formas normales

► Cuarta Forma Normal (4FN)

- Un esquema relacional R está en la 4FN con respecto a un conjunto de dependencias F (incluyendo las dependencias funcionales y multivaluadas) si, para cada dependencia multivaluada no trivial $X \twoheadrightarrow Y$ en F , X es una superllave de R .
- Una MVD $X \twoheadrightarrow Y$ se dice que es trivial si
 - a) Y es un subconjunto de X , o
 - b) $X \cup Y = R$
- Si no se cumple cualquiera de las condiciones anteriores, se dice que es una MVD no trivial.



Dependencias Multivaluadas

- ▶ En algunos casos pueden existir restricciones que no pueden ser expresadas como dependencias funcionales.
- ▶ Las dependencias multivaluadas son consecuencia de la 1FN, ya que no se permite en un atributo de una tupla que exista un conjunto de valores.
- ▶ Si en una relación se tienen dos o más atributos multivaluados independientes, será un problema el repetir cada valor de un atributo con cada valor del otro atributo, para mantener la independencia entre ellos.



Definición

- ▶ Existe una dependencia multivaluada (MVD) $X \twoheadrightarrow Y$ en una relación R , donde X y Y son subconjuntos de R , si y solo si cada valor de X tiene asignado un conjunto bien definido de valores de Y y este conjunto es independiente de cualquier valor que tome otro atributo $Z \in R$, el cual depende del valor de X .
- ▶ Las dependencias multivaluadas representan la independencia existente entre dos conjuntos Y y Z , la cual esta correlacionada por la dependencia que tiene cada uno de estos conjuntos con el conjunto X del cual dependen ambos de forma multivaluada.



Dependencias Multivaluadas

- ▶ Hay esquemas de bases de datos en BCFN que no parecen estar suficientemente normalizadas
- ▶ Considere una base de datos:
 $clases (\underline{curso}, \underline{maestro}, \underline{libro})$
tal que $(c,m,l) \in clases$ significa que m está calificado para enseñar c , y l es un texto requerido para c
- ▶ La base de datos supone que lista para cada curso el conjunto de maestros, cualquiera que pueda ser el instructor del curso, y el conjunto de libros que sean requeridos por el curso (no importa quién lo enseñe).



<i>curso</i>	<i>maestro</i>	<i>libro</i>
Base de datos	Avi	DB Concepts
Base de datos	Avi	Ullman
Base de datos	Hank	DB Concepts
Base de datos	Hank	Ullman
Base de datos	Sudarshan	DB Concepts
Base de datos	Sudarshan	Ullman
Sistemas operativos	Avi	OS Concepts
Sistemas operativos	Avi	Shaw
Sistemas operativos	Jim	OS Concepts
Sistemas operativos	Jim	Shaw

classes

- ▶ Ya que no hay dependencias no triviales, y (*curso, maestro, libro*) es la única llave, la relación está en BCFN.
- ▶ Existen anomalías de inserción – si Sara es una nueva maestra que puede enseñar bases de datos, es necesario insertar dos nuevas tuplas:

(Base de datos, Sara, DB Concepts)

(Base de datos, Sara, Ullman)

- De aquí, es mejor descomponer *clases en*:

<i>curso</i>	<i>maestro</i>
Base de datos	Avi
Base de datos	Hank
Base de datos	Sudarshan
Sistemas operativos	Avi
Sistemas operativos	Jim

enseña

<i>curso</i>	<i>libro</i>
Base de datos	DB Concepts
Base de datos	Ullman
Sistemas operativos	OS Concepts
Sistemas operativos	Shaw

texto

Estas dos relaciones están en la Cuarta Forma Normal (4FN)

Descomposición por reunión sin pérdidas

- ▶ Todos los atributos del esquema original R deben de aparecer en la descomposición R_1, R_2 :
 - ▶ $R = R_1 \cup R_2$
- ▶ Descomposición sin pérdidas:
 - ▶ Para todas las posibles relaciones r en el esquema R :
 - ▶ $r = \Pi_{R_1}(r) \bowtie \Pi_{R_2}(r)$
 - ▶ Una descomposición de R en R_1 y R_2 es sin pérdidas si y sólo si al menos una de las siguientes dependencias está en F^+ :
 - ▶ $R_1 \cap R_2 \rightarrow R_1$
 - ▶ $R_1 \cap R_2 \rightarrow R_2$

A	B
X	1
X	2
Y	1

r

A
X
Y

$\Pi_A(r)$

B
1
2

$\Pi_B(r)$

A	B
X	1
X	2
Y	1
Y	2

$\Pi_A(r) \bowtie \Pi_B(r)$

Quinta Forma Normal (5FN)

- ▶ Pueden existir casos en el que haya más de dos esquemas relacionales en la descomposición para satisfacer la 4FN, y además, puede ser que no haya dependencias funcionales en R tal que viole alguna forma normal hasta la BCFN, y que no haya MVD no triviales que violen la 4FN.
- ▶ Esto lleva a la descomposición debido a las dependencias de reunión. Es importante notar que esta dependencia es muy difícil de detectar en la práctica, y por ello, la normalización en la 5FN es muy rara vez aplicada.
- ▶ Un esquema relacional está en 5FN con respecto a un conjunto F de dependencias funcionales, multivaluadas y de reunión si, para cada dependencia de reunión no trivial $JD(R_1, R_2, \dots, R_n)$ en F , cada R_i es una superllave de R .



Quinta Forma Normal (5FN)

- ▶ Un esquema de una relación R está en 5FN con respecto a un conjunto D de dependencias funcionales, multivaluadas, y de reunión si para todas las dependencias de reunión en D^+ de la forma $*(R_1, R_2, \dots, R_n)$ donde $R_i \subseteq R$ y $R = R_1 \cup R_2 \cup \dots \cup R_n$ al menos una de las siguiente condiciones se cumple:
 - ▶ $*(R_1, R_2, \dots, R_n)$ es una dependencia de reunión trivial.
 - ▶ Cada R_i es una superllave para R .
- ▶ Desde que cada dependencia multivaluada está también como una dependencia de reunión, cada esquema en 5FN también está en 4FN.



Ejemplo

- ▶ Considere un esquema para prestamos dado por :
Prestamos-info (sector, nombre_cliente, numero_prestamo, cantidad).
- ▶ Cada préstamo tiene uno o más clientes, está en uno o más sectores y tiene una cantidad de préstamo; estas relaciones son independientes, y por esto se tiene una dependencia de reunión
- ▶ $\ast(=(\text{numero_prestamo}, \text{sector}), (\text{numero_prestamo}, \text{nombre_cliente}), (\text{numero_prestamo}, \text{cantidad}))$
- ▶ *Prestamos-info* no está en 5FN con respecto al conjunto de dependencias de reunión. Para ponerlo en 5FN, se debe descomponer en los siguientes tres esquemas especificados por la dependencia de reunión:
Prestamos-sector (numero_prestamo, sector)
Prestamos-cliente (numero_prestamo, nombre_cliente)
Prestamos-cantidad (numero_prestamo, cantidad)



Metas del diseño

- ▶ Las metas para un buen diseño de una base de datos relacional son:
 - ▶ Deben estar en 3FN o BCFN.
 - ▶ Reuniones sin perdidas.
 - ▶ Conservación de dependencias.
- ▶ Si no es posible obtener esto, se debe aceptar:
 - ▶ Falta de conservación de dependencias, o
 - ▶ Redundancia de datos debido la falta de la 3FN.
- ▶ Interesantemente, el SQL no proporciona una forma directa de especificar las dependencias funcionales más que las superllaves.
- ▶ Se pueden especificar las DF usando aserciones, pero son costosas de probar.
- ▶ Incluso si se tienen descomposiciones con conservación de dependencias, usando SQL no es posible comprobar eficientemente una dependencia funcional cuyo lado izquierdo no es una llave.



Modelo EA y normalización

- ▶ Cuando un diagrama E-A es diseñado cuidadosamente, identificando todas las entidades correctamente, las tablas generadas por el diagrama E-A no necesitan mayores normalizaciones.
- ▶ Sin embargo, en el diseño real puede haber DF de los atributos no llave de una entidad hacia los otros atributos de la entidad.
- ▶ Ej., la entidad *empleado* con los atributos *NumDepartamento* y *DireccDepartmento*, y una DF *NumDepartamento* → *DireccDepartmento*
 - ▶ Un buen diseño deberá hacer *departamento* una entidad.

