Solución Taller de Dependencias Funcionales y Formas Normales
Gestión y Modelación de Datos
2013-1
Prof. María Constanza Pabón

Punto 1.

Dado R(A, B, C, D, E) y el conjunto de dependencias funcionales F = {A->BC, CD->E, B->D, E->A}, probar, utilizando los axiomas, que las siguientes dependencias se cumplen:

a) AD → E

1. B -> D (Dado en F)
2. BC -> DC (Por aumentatividad)
3. CD -> E (Dado en F)
4. BC -> E (Por transitividad en 2 y 3)
5. A -> BC (Dado en F)
6. A -> E (Por transitividad en 4 y 5)

b) AC → E

1. A -> BC (Dado en F)
2. AC -> BC (Por aumentatividad en 1)
3. B -> D (Dado en F)
4. BC -> DC (Por aumentatividad)
5. CD -> E (Dado en F)
6. BC -> E (Por transitividad en 4 y 5)
7. AC -> E (Por transitividad en 2 y 6)

Punto 2. (Tomado de Date, J.C. Database design and relational theory, 2012)

Diseñe una BD para representar información de empleados y programadores. Cada programador es un empleado, pero algunos empleados no son programadores. Los empleados tienen un número de empleado, nombre y salario. Los programadores son expertos en un (uno solo) lenguaje de programación.

Empleado (empID, empNombre, empSalario, empCargo, empLenguaje)

DF's: empID -> { empNombre, empSalario, empCargo, empLenguaje }
Llave: empID

Esta 4ª Forma Normal.

a) Qué diferencia hay si los programadores son expertos en muchos lenguajes de programación?

Empleado (empID, empNombre, empSalario, empCargo, empLenguaje)

DF's:
empID -> { empNombre, empSalario, empCargo }

empID ->-> empLenguaje

Llave { empID, empLenguaje }
No está en 2FN porque hay dependencia parcial de la llave en los atributos empNombre, empSalario, empCargo

Entonces se divide la tabla:

Empleado (empID, empNombre, empSalario, empCargo)
DF's: empID -> { empNombre, empSalario, empCargo }
Esta en 4FN.

Programadores(empID,Prog,Lenguaje)
DF´s: { empID, empLenguaje } -> { empID, empLenguaje }
Esta en 4FN

Punto 3. (Tomado de http://db4u.wikidot.com/)

Dada la siguiente relación que incluye un id de curso, nombre de profesor, fecha de contratación del profesor, libro de texto y año de publicación del libro de texto,

Course	Teacher	Hire date	Text	Copyright
BIT340	Moore	1992	340 Coursepack	2007
BIT340	Moore	1992	Access	2007
BIT340	Ravishankar	1986	340 Coursepack	2007
BIT340	Ravishankar	1986	Access	2007
BIT301	Moore	1992	301 Coursepack	2007
BIT301	Moore	1992	Excel	2007
BIT301	Moore	1992	being digital	2005
BIT301	Walls	1993	301 Coursepack	2007
BIT301	Walls	1993	Excel	2007
BIT301	Walls	1993	being digital	2005

a) Encuentre las dependencias funcionales

Course ->-> { Teacher, Text }
Teacher -> Hiredate
Text -> Copyright

b) Proponga una llave para la relación

{ Course, Teacher, Text }

c) En que forma normal está la relación?

En 1FN porque hay dependencias parciales de la llave (Teacher -> Hiredate,

Text -> Copyright)

d) Si no está en 4FN, entonces lleve el diseño hasta 4FN

Se divide la relación R (Course, Teacher, Hiredate, Text, Copyright)
En: R1 (Teacher, Hiredate), R2 (Text, Copyright), R3 (Course, Teacher, Text)
DF's de R1: Teacher -> Hiredate, R1 está en 4FN
DF's de R2: Text -> Copyright, R2 está en 4FN
DF's de R3: Course ->-> { Teacher, Text }, , R3 está en FNBC. No está en 4FN porque hay dependencia multivaluada entre course y teacher, y course y text.

Se divide R3 (Course, Teacher, Text)
En: R4 (Course, Teacher) y R5 (Course, Text).
DF's de R4: Course ->-> Teacher, R4 está en 4FN
DF's de R5: Course ->-> Text, R5 está en 4FN

Punto 4.

Dada la siguiente forma que corresponde a la factura generada por los gastos de hospitalización de un paciente en un hospital:

Patient bill						
Patient #: 12345			Date: 7/20/08			
Patient Name: Mary Baker			Date admitted: 7/14/08			
Patient Address: 300 Oak Street			Discharge date: 7/17/08			
City-State-Zip: Boulder, CO 80638						
Cost Center	Cost Name	Date Charged	Item Code	Desc	Charge	Bal Due
100	Room & Board	7/14/08	2000	Semi-prv room	200.00	
		7/14/08	2005	Television	5.00	
		7/15/08	2000	Semi-prv room	200.00	
		7/16/08	2000	semi-prv room	200.00	
				Subtotal		605.00
110	Laboratory	7/14/08	1580	Glucose	25.00	
		7/15/08	1585	Culture	20.00	
				Subtotal		45.00
125	Radiology	7/15/08	3010	X-ray chest	30.00	
				Subtotal		30.00
				Balance due		680.00

Proponga un diseño para una base de datos que permita almacenar la información de la factura. Haga el diseño partiendo de una Relación Universal (que incluye todos los datos) y normalicela hasta llegar a 4FN.

PatientBill (patient#, patientName, patientAddress, patientCity, patientState, patientZIP, date, dateAdmitted, dischargeDate, costCenter, costName, dateCharge, ItemCode, Desc, Charge)

DF's:
patient# -> { patientName, patientAddress, patientCity, patientState, patientZIP }
{ patient#, dateAdmitted } -> { date, dischargeDate }
costCenter -> costName
ItemCode -> { costCenter, costName }
ItemCode -> { dateCharge, Desc, Charge }
{ patient#, dateAdmitted } ->-> { costCenter, costName, dateCharge, ItemCode, Desc, Charge }

Llave: patient#, dateAdmitted, itemCode

PatientBill está en 1FN porque hay dependencias parciales:
patient# -> { patientName, patientAddress, patientCity, patientState, patientZIP }
{ patient#, dateAdmitted } -> { date, dischargeDate }
ItemCode -> { costCenter, costName, dateCharge, Desc, Charge }

PatientBill se divide en:
Patient (patient#, patientName, patientAddress, patientCity, patientState, patientZIP)
DF's: patient# -> { patientName, patientAddress, patientCity, patientState, patientZIP }
Patient está en 4FN

Item (ItemCode, Desc, Charge, dateCharge, costCenter, costName)
DF's: costCenter -> costName
ItemCode -> { costCenter, costName }
ItemCode -> { dateCharge, Desc, Charge }
Item está en 2FN. No está en 3FN porque hay dependencias transitivas: *ItemCode -> costName* resulta de aplicar la transitividad entre *ItemCode -> costCenter* y *costCenter -> costName*

billDates (patient#, dateAdmitted, dischargeDate, date)
DF's: { patient#, dateAdmitted } -> { date, dischargeDate }
billDates está en 4FN

PatientBill (patient#, dateAdmitted, ItemCode) --- Está en 4FN

Punto 5.

Una cadena de restaurantes requiere el diseño de una base de datos para guardar la información de sus recetas. Los restaurantes de la cadena están ubicados en varias ciudades y de ellos se desea guardar el nombre, la dirección, y la ciudad en que están ubicados. Cada restaurante tiene un Chef que trabajan en él. De los Chefs se requiere guardar el nombre y apellido, la nacionalidad y las recetas que ha creado. Cada receta tiene un código, un nombre, una lista de ingredientes (cada ingrediente tiene la cantidad necesaria) y una lista de pasos que describen como se prepara el plato. Los platos creados por un Chef se ofrecen en todos los restaurantes. El plato tiene un precio de venta que cambia en cada restaurante.

Proponga un diseño para una base de datos que permita almacenar la información de la factura. Haga el diseño partiendo de una Relación Universal (que incluye todos los datos) y normalicela hasta llegar a 4FN.

Receta (idRestaurante, nombreRestaurante, ciudadRestaurante, direcciónRestaurante, chef, nombreChef, apellidoChef, nacionalidadChef, idReceta, nombreReceta, ingredienteReceta, cantidadIngrediente, nroPaso, pasoReceta, precio)
DF's:
idRestaurante -> { nombreRestaurante, ciudadRestaurante, direcciónRestaurante, idChef, nombreChef, apellidoChef, nacionalidadChef }
idChef -> { nombreChef, apellidoChef }
idReceta -> nombreReceta
idReceta -> { idChef, nombreChef, apellidoChef, nacionalidadChef }
{ idReceta, ingredienteReceta } -> cantidadIngrediente
{ idReceta, nroPaso } -> pasoReceta
{ idRestaurante, idReceta } -> precio
idChef ->-> { idReceta }
idReceta ->-> { ingredienteReceta, nroPaso }

Llave: idRestaurante, idReceta, ingredienteReceta, nroPaso

La relación Receta está en 1FN porque existen las siguientes dependencias parciales:

idRestaurante -> { nombreRestaurante, ciudadRestaurante, direcciónRestaurante, idChef, nombreChef, apellidoChef, nacionalidadChef }
idReceta -> { nombreReceta, idChef, nombreChef, apellidoChef, nacionalidadChef }
{ idReceta, ingredienteReceta } -> cantidadIngrediente
{ idReceta, nroPaso } -> pasoReceta
{ idRestaurante, idReceta } -> precio

Receta se divide en:

Restaurante (idRestaurante, nombreRestaurante, ciudadRestaurante, direcciónRestaurante, idChef, nombreChef, apellidoChef, nacionalidadChef)
DF's: idRestaurante -> { nombreRestaurante, ciudadRestaurante, direcciónRestaurante, idChef, nombreChef, apellidoChef, nacionalidadChef }
idChef -> { nombreChef, apellidoChef, nacionalidadChef }
Restaurante está en 2FN. No está en 3FN porque hay una dependencia transitiva: idRestaurante -> { nombreChef, apellidoChef, nacionalidadChef } se obtiene por transitividad entre idRestaurante -> { idChef } y idChef -> { nombreChef, apellidoChef, nacionalidadChef }

Receta (idReceta, nombreReceta, idChef)
DF's: idReceta -> { nombreReceta, idChef }
Receta está en 4FN.

IngredienteReceta (idReceta, ingredienteReceta, cantidadIngrediente)
DF's: { idReceta, ingredienteReceta } -> cantidadIngrediente
idReceta ->-> { ingredienteReceta }
IngredienteReceta está en 4FN.

PasoReceta (idReceta, nroPaso, pasoReceta)
DF's: { idReceta, nroPaso } -> pasoReceta
idReceta ->-> { nroPaso }
PasoReceta está en 4FN.

Precio (idRestaurante, idReceta, precio)
DF's: { idRestaurante, idReceta } -> precio
Precio está en 4FN.

Se normaliza Restaurante, se divide en:

Restaurante (idRestaurante, nombreRestaurante, ciudadRestaurante, direcciónRestaurante, idChef)
DF's: idRestaurante -> { nombreRestaurante, ciudadRestaurante, direcciónRestaurante, idChef }
Restaurante está en 4FN.

Chef (idChef, nombreChef, apellidoChef, nacionalidadChef)
DF's: idChef -> { nombreChef, apellidoChef, nacionalidadChef }
Chef está en 4FN.

Nota: observe que en este caso es posible tomar la decisión de des-normalizar Restaurante (es decir no hacer este ultimo paso de normalización que separa restaurante y chef), dado que se sabe por la definición del problema que cada restaurante tiene un solo chef)

Por lo tanto, el diseño en 4FN es:

Restaurante (idRestaurante, nombreRestaurante, ciudadRestaurante, direcciónRestaurante, idChef)
Chef (idChef, nombreChef, apellidoChef, nacionalidadChef)
Receta (idReceta, nombreReceta, idChef)
IngredienteReceta (idReceta, ingredienteReceta, cantidadIngrediente)
IngredienteReceta está en 4FN.
PasoReceta (idReceta, nroPaso, pasoReceta)
Precio (idRestaurante, idReceta, precio)