**Desarrollo de Sistemas Transaccionales: Fase de Análisis y Especificación de Requerimientos.**

El desarrollo de cualquier sistema informático empieza con una fase de análisis y especificación de requerimientos. En esta fase se identifican los servicios que los futuros usuarios del sistema esperan de él. Esos servicios esperados son los *requerimientos funcionales*. También es necesario entender el *mundo* o contexto en el que se va a aplicar el sistema. Así mismo, hay que identificar las restricciones o condiciones que se imponen al sistema. A estas restricciones o condiciones las llamamos *requerimientos no funcionales*.

Un *requerimiento funcional* se describe a través de cuatro elementos:

* Un nombre
* Un resumen de la operación o servicio
* Las entradas que debe dar el usuario para que la aplicación pueda realizar la operación
* El resultado esperado de la operación

La comprensión del mundo o contexto en el que se va a aplicar el sistema se plasma en lo que se llama el *modelo del mundo* *del problema*, en el cual se identifican las entidades que participan, con sus principales atributos y relaciones. Este modelo del mundo se puede representar como un diagrama de clases UML (*Unified Modeling Language*) ó como un diagrama Entidad-Relación.

Los *requerimientos no funcionales* están muchas veces relacionados con restricciones sobre la tecnología que se debe usar, el volumen de los datos que se debe manejar o la cantidad de usuarios.

**Sistemas Transaccionales: El Modelo del Mundo del Problema**

En el modelo del mundo se utilizan tres elementos:

* Los *tipos de entidades* que nos interesan. Estas entidades pueden ser concretas (personas, vehículos) o abstractas (cuentas bancarias, cursos). En programación orientada por objetos los tipos de entidades se denominan *clases*. Les damos un nombre, que por convención debe tener la primera letra en mayúscula.
* *Atributos* de los tipos de entidades o clases. A cada atributo se le asocia un nombre significativo, que por convención comienza por una letra minúscula, y una descripción del conjunto de valores que puede tomar. En el lenguaje UML, una clase se representa con un cuadrado que tiene tres zonas: la primera, con el nombre de la clase o tipo de entidad; la segunda, con los atributos de la clase; la tercera, con los métodos o servicios que ofrece la clase. En el diagrama Entidad-Relación no aparece la tercera zona, pues se enfoca únicamente en la parte estructural (estática) del tipo de entidad.
* Las *Relaciones* entre entidades. Estas relaciones pueden ser: de *herencia* (especialización) ó de *asociación*. Las relaciones de *herencia* se dan entre dos clases o tipos de entidades, y se representan con una flecha (terminada en un triángulo) que va de la clase más especializada (subclase), a la clase más general. Las relaciones de *asociación* se pueden dar entre dos o más clases o tipos de entidades. Si son binarias (entre dos entidades), se representan como líneas sólidas que unen los cuadrados de las clases o tipos de entidades relacionadas. La asociación tiene un nombre. Este nombre sigue las convenciones del nombre de los atributos. Entre dos clases o tipos de entidades pueden existir varios tipos de asociaciones, cada una con su línea y su nombre. Si la asociación es de grado mayor que 2, por ejemplo ternaria, se representa por un rombo que está unido por líneas a los cajones correspondientes a las clases o tipos de entidades que participan en la asociación. En el rombo colocamos el nombre de la relación.

En ambos extremos de la línea que representa una asociación colocamos el rango de números permitidos para objetos o entidades relacionadas (1=exactamente uno; \* =cero o más; 1..\* = uno o más).

El siguiente ejemplo corresponde a un aspecto del modelo del mundo de una aplicación para manejar búsquedas en una tienda de música:

Persona

sexo

nacionalidad

ObraMusical

nombre

genero

Participante

nombre

tipo

Interpretacion

año\_grabacion

observaciones

Rol

nombre

desripcion

1..\* compuso \* \*-\*

1

tiene

\*

participa como

1 \* \* \*

1..\* \* \* 1 \* \*

\*

1

Allí aparecen cinco clases o tipos de entidades. La clase *Persona* es una subclase de *Participante*; es decir, toda persona es un participante y hereda todos los atributos de éste. Además de personas, los participantes pueden ser orquestas, estudios de grabación, etc. Una *Interpretacion* está asociada con una *ObraMusical* y exactamente con una. Una ObraMusical puede tener cero o más interpretaciones. Un ejemplo de obra musical podría ser *El Bolero* de Maurice Ravel. De esta obra puede haber muchas interpretaciones, hechas y grabadas por diferentes orquestas en diferentes épocas. La tienda de música que se está modelando puede tener algunas (o ninguna) o todas esas grabaciones. Maurice Ravel es una persona y también un participante. Es el compositor del Bolero. Una obra puede ser compuesta a varias manos, pero por lo menos tiene un compositor. Una persona no es obligatorio que haya compuesto, y puede haber compuesto varias obras. Roles pueden ser: director, cantante, patrocinador, etc. La asociación ternaria *participa\_como*, entre *Participante*, *Interpretacion* y *Rol* está representada por el rombo. Una interpretación puede aparecer en varias tripletas (por lo menos una), una por participante. En cada tripleta aparece exactamente un participante, una interpretación y un rol. Un rol no es obligatorio que aparezca en alguna tripleta. Tampoco un participante.

**Conversión de modelo conceptual a relacional**

En este capítulo se muestra cómo pasar el modelo del mundo del problema, representado en un diagrama de clases UML ó un modelo Entidad-Relación, a tablas relacionales.

Recordemos el ejemplo de modelo del mundo del capítulo 2:

Persona

sexo

nacionalidad

ObraMusical

nombre

genero

Participante

nombre

tipo

Interpretacion

año\_grabacion

observaciones

Rol

nombre

desripcion

1..\* compuso \* \*-\*

1

tiene

\*

participa como

1 \* \* \*

1..\* \* \* 1 \* \*

\*

1

En el modelo del mundo aparecen clases ó tipos de entidades, sus atributos, y una serie de relaciones entre clases (el concepto de relación entre clases del modelo conceptual es diferente al concepto de relación en el modelo de datos relacional; para evitar ambigüedades, en el modelo relacional hablaremos de tablas en lugar de relaciones).

Las relaciones entre clases pueden ser de *herencia* (especialización) ó de *asociación*. Las relaciones de asociación pueden ser binarias o n-arias. Las binarias pueden ser 1:1, 1: n, ó m: n. En el ejemplo anterior, la relación entre Persona y Participante es de herencia. La relación entre Persona y ObraMusical es de asociación, binaria m:n. La relación entre ObraMusical e Interpretación es de asociación, binaria 1:n. La relación entre Participante, Interpretación y Rol es de asociación, ternaria.

Si una asociación tiene atributos que la describan (atributos que son de la asociación y no de las clases que en ella participan), la asociación se representa por una clase (clase asociación) que estará relacionada con asociaciones binarias 1:n a cada una de las clases que participan en la asociación. Los atributos de la asociación se colocan como atributos de la clase asociación.

Las reglas de conversión a relacional son:

1. Cada clase es representada por una tabla, que tendrá un identificador único (llave primaria) asignado por el sistema (un consecutivo por jerarquía de herencia). Además del consecutivo asignado por el sistema, la tabla tendrá como atributos los atributos de la clase. El nombre de la tabla es el mismo nombre de la clase.
2. La relación de herencia se representa indicando que la llave primaria de la clase que hereda es además una llave foránea relacionada con la llave primaria de la tabla de la cual hereda. Para insertar una nueva fila en la tabla que hereda es obligatorio que primero se haya insertado la fila correspondiente en la tabla de la cual se hereda. En el ejemplo anterior, para insertar una fila en la tabla Persona primero se debe insertar como participante. Toda persona es un participante. La llave como persona es la misma que como participante.
3. Los consecutivos para llaves primarias son por jerarquía de clases. En el ejemplo anterior, habrá un consecutivo para la jerarquía Persona 🡪 Participante, un consecutivo para ObraMusical, un consecutivo para Interpretación, y un consecutivo para Rol.
4. Las asociaciones binarias 1:1 se representan poniendo en una de las dos tablas, la que tenga menos filas, una llave foránea hacia la otra tabla. Esa llave foránea no puede tener duplicados, lo cual se indica con un ND. La llave foránea puede ser opcional u obligatoria, dependiendo de lo que se indique en el diagrama del modelo del mundo.
5. Las asociaciones binarias 1:n se representan poniendo en la tabla hijo, la que está del lado del n, una llave foránea hacia la tabla padre, la del lado del 1. la llave foránea puede ser opcional u obligatoria, dependiendo de lo que se indique en el diagrama del modelo del mundo.
6. Las asociaciones binarias m:n se representan con una tabla que tiene como atributos las llaves primarias de cada una de las tablas correspondientes a las clases que participan en la asociación. El nombre de la tabla es el nombre de la asociación. La llave primaria es la combinación de las llaves de las tablas correspondientes a las clases participantes. Cada componente de la llave primaria es a su vez una llave foránea.
7. Las asociaciones n-arias, con n > 2, se representan por una tabla que tiene como atributos las llaves primarias de las tablas participantes en la asociación. La llave primaria de la tabla es la combinación de las llaves individuales, las cuales a su vez son llaves foráneas.
8. Las clases asociación, aquellas introducidas cuando una asociación tiene atributos, se representan por una tabla que tiene como atributos las llaves de las clases participantes más los atributos de la asociación. La llave primaria de esta tabla asociación será la combinación de las llaves primarias participantes.

A continuación se presentan las tablas relacionales correspondientes al modelo del mundo que nos ha servido de ejemplo:

PARTICIPANTE

|  |  |  |
| --- | --- | --- |
| Id\_participante | Nombre | Tipo |
| PK | NN | NN |

PERSONA

|  |  |  |
| --- | --- | --- |
| Id\_persona | Sexo | Nacionalidad |
| PK, FK | NN | NN |

OBRAMUSICAL

|  |  |  |
| --- | --- | --- |
| Id\_obra | Nombre | Genero |
| PK | NN | NN |

INTERPRETACION

|  |  |  |  |
| --- | --- | --- | --- |
| Id\_interpretacion | Año\_grabacion | Observaciones | Id\_obra |
| PK | NN |  | FK, NN |

ROL

|  |  |  |
| --- | --- | --- |
| Id\_rol | Nombre | Descripcion |
| PK | NN |  |

COMPUSO

|  |  |
| --- | --- |
| Id\_persona | Id\_obra |
| PK, FK1 | PK, FK2 |

PARTICIPA\_COMO

|  |  |  |
| --- | --- | --- |
| Id\_participante | Id\_interpretacion | Id\_rol |
| PK, FK1 | PK, FK2 | PK, FK3 |

Los nombres de las llaves foráneas no necesariamente deben ser iguales a los de las llaves primarias asociadas. Por ejemplo, la llave primaria de Persona la llamamos Id\_persona, aunque la llave primaria asociada se llama Id\_participante.

**Normalización**

Una forma de medir la calidad de un diseño de base de datos relacional es utilizar los conceptos básicos de normalización propuestos por E. F. Codd como parte de su modelo relacional.

Imaginemos los datos que una empresa quisiera tener sobre sus empleados: nombres y apellidos, fecha y lugar de nacimiento, sexo, estado civil, número del documento de identidad, dirección, estudios, cargos, hijos, proyectos en los que ha participado, etc.

Los primeros atributos – nombres, apellidos, fecha y lugar de nacimiento, sexo, estado civil, documento y dirección, son mono-valores. Los restantes – estudios, cargos, etc – son multi-valor: de cada uno puede tener cero o más; además, cada uno de ellos es a su vez compuesto: de cada estudio interesa saber la institución y el país donde lo realizó, el año de culminación, el nivel y el título obtenido; de cada cargo se quiere tener la empresa, el nombre del cargo, el periodo de tiempo en el que desempeñó ese cargo, etc.

Si guardamos toda la información de los empleados en una sola tabla, cada fila representando a un empleado, tendríamos que esas filas serían complejas: campos mono-valor mezclados con campos que a su vez pueden ser tablas.

En lugar de tener todo en una tabla, se puede dividir la información en varias tablas: una que contenga todos los atributos mono-valor, y una por cada atributo multi-valor. Ejemplo:

EMPLEADO

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Id\_empleado | Nombres | Apellidos | Fecha\_nac | Lugar\_nac | Sexo | Estado\_civil |  |
| PK | NN | NN | NN | NN | NN | NN |  |

ESTUDIOS\_EMPLEADO

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Id\_empleado | Institucion | pais | año | nivel | titulo |
| PK, FK | NN | NN | NN | NN | PK |

CARGOS\_EMPLEADO

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Id\_Empleado | Empresa | Cargo | Fecha\_desde | Fecha\_hasta |
| PK, FK | NN | PK | PK |  |

HIJOS\_EMPLEADO

|  |  |  |  |
| --- | --- | --- | --- |
| Id\_empleado | Nombre\_hijo | Sexo\_hijo | Fecha\_nacimiento |
| PK, FK | NN | NN | PK |

En estas tablas, cada fila es una agregación de campos atómicos, con la excepción de las fechas, que se pueden descomponer en año, mes y día.

Toda tabla compleja, como sería el caso de tener todos los datos de los empleados en una sola tabla, se puede descomponer en un conjunto de tablas sencillas (con filas de campos atómicos), tal y como se hizo en el ejemplo. Los lenguajes para manejar tablas no complejas son más sencillos. Por esa razón, Codd en su modelo relacional propuso trabajar únicamente con tablas sencillas, a las que llamó normalizadas (las complejas son no normalizadas).

Todas las tablas sencillas están en lo que Codd llamó la *primera forma normal* (1FN): sus filas son agregaciones de campos atómicos (con la excepción de las fechas).

Además de la primera forma normal, Codd definió otras formas normales – la segunda, la tercera, la de Boyce-Codd, la cuarta – que tienen que ver con redundancia y con anomalías de inserción y de supresión. Retomando la base de datos de bebedores, bares y cervezas del capítulo 3, supongamos que la tabla GUSTA tiene ahora los siguientes atributos:

GUSTA

|  |  |  |  |
| --- | --- | --- | --- |
| Id\_bebedor | Id\_cerveza | Nombre\_cerveza | Grado\_alcohol |
|  |  |  |  |

Supongamos que sobre esta tabla se hacen las siguientes aserciones:

* A un identificador de cerveza le corresponde un solo nombre
* No hay dos cervezas con el mismo nombre
* A un identificador de cerveza le corresponde un solo grado de alcohol
* Varias cervezas pueden tener el mismo grado de alcohol
* A un bebedor le pueden gustar varias cervezas
* Una cerveza le puede gustar a varios bebedores.

Algunas de estas aserciones, que son reglas de integridad (también son reglas del negocio), se pueden expresar como *dependencias funcionales*, de la siguiente manera:

* Id\_cerveza 🡪 Nombre\_cerveza
* Nombre\_cerveza 🡪 Id\_cerveza
* Id\_cerveza 🡪 Grado\_alcohol

Una dependencia funcional X 🡪 Y, donde X e Y pueden ser compuestas, expresa que a un valor de X le corresponde un solo valor de Y (X determina funcionalmente a Y). Otra forma de expresarlo es que si esa dependencia funcional es cierta sobre una tabla T, entonces si dos filas de T tienen el mismo valor para X, obligatoriamente tienen el mismo valor para Y.

Las dependencias funcionales se utilizan para definir formalmente el concepto de llave de una tabla. Recuerde que una tabla puede tener varias llaves candidatas. Cuando una de esas candidatas se escoge como la primaria, las otras se consideran alternas.

Si A1, A2, …, An son los atributos de una tabla T, si al conjunto de atributos A1, A2, … , An lo llamamos U (U={ A1, A2, … , An}), y si X es un subconjunto de U, entonces X es una llave de T si:

1. X 🡪 U es cierta (si dos filas tienen el mismo valor para X entonces las dos filas son iguales, y en efecto son la misma fila).
2. No existe Y subconjunto propio de X tal que Y 🡪 U; es decir, una llave es el mínimo conjunto de atributos que cumplen la función de diferenciar. Si X cumple la primera condición y no la segunda, decimos que X es una *superllave* y no una llave, porque contiene a una llave.

La tabla GUSTA extendida con el nombre y grado de alcohol de la cerveza tiene dos llaves candidatas:

1. La combinación (Id\_bebedor, Id\_cerveza)
2. La combinación (Id\_bebedor, Nombre\_cerveza)

Sobre esa tabla se cumple que:

1. Id\_bebedor, Id\_cerveza 🡪 Id\_bebedor, Id\_cerveza, Nombre\_cerveza, Grado\_alcohol
2. Id\_bebedor, Nombre\_cerveza 🡪 Id\_bebedor, Id\_cerveza, Nombre\_cerveza, Grado\_alcohol

Lo anterior se puede demostrar usando unas reglas de inferencia que no vamos a presentar aquí (los axiomas de Armstrong).

A los atributos de una tabla que hacen parte de alguna de sus llaves candidatas, los llamamos atributos *primos*. Los que no hacen parte de ninguna llave se llaman *no primos*.

Una dependencia X 🡪 Y es *trivial* si Y es un subconjunto de X.

Una dependencia funcional X 🡪 Y es *parcial* si:

1. No es trivial
2. Y es no primo
3. X es un subconjunto propio de alguna llave candidata

Una dependencia funcional X 🡪 Y es *transitiva* si:

1. No es trivial
2. Y es no primo
3. X no es una superllave, no es una llave, no es un subconjunto propio de alguna llave.

Cuando en una tabla hay dependencias parciales ó transitivas, es porque hay redundancia y, posiblemente, anomalías de inserción y de supresión.

En la tabla GUSTA extendida hay redundancia: la asociación Id\_cerveza – Nombre\_cerveza y Grado\_alcohol se hace tantas veces como bebedores haya a quienes les gusta una cerveza. Además, si la única tabla donde se guarda esta información de las cervezas es GUSTA, hay también anomalías de inserción y de supresión: No se puede ingresar en la base de datos información de una cerveza hasta que no haya al menos un bebedor a quien le guste (anomalía de inserción); Si solo había un bebedor a quien le gustaba una cerveza, y se elimina ese bebedor, se pierde la información de la cerveza (anomalía de supresión).

El diseño correcto para la base de datos de bebedores es el que presentamos en el capítulo 3: Hay una tabla de cervezas, y en GUSTA solo está el Id\_bebedor y el Id\_cerveza. No hay redundancia ni anomalías en ese caso.

Las tablas que no tienen redundancia ni anomalías (ignorando aquí dependencias multi-valor y cuarta forma normal), están en la *forma normal de Boyce-Codd* (BCFN). Una tabla está en la BCFN si, además de estar en la 1FN, todas las dependencias funcionales no triviales que aplican sobre esa tabla en su parte izquierda tienen una llave ó una superllave.

GUSTA extendida está en la 1FN pero no está en la BCFN porque hay al menos tres dependencias funcionales no triviales que en su parte izquierda no tienen una llave. Ellas son:

* Id\_cerveza 🡪 Nombre\_cerveza
* Nombre\_cerveza 🡪 Id\_cerveza
* Id\_cerveza 🡪 Grado\_alcohol

Una tabla está en la *segunda forma normal* (2FN) si, además de estar en la 1FN, no tiene dependencias parciales (puede tener transitivas).

Una tabla está en la *tercera forma normal* (3FN) si, además de estar en la 2FN, no tiene dependencias transitivas; es decir, no tiene dependencias parciales ni transitivas.

GUSTA extendida tiene una dependencia parcial: Id\_cerveza 🡪 Grado\_alcohol. Por lo tanto no está en la 2FN y por consiguiente tampoco en la 3FN. Está en la 1FN.

Si una tabla tiene una sola llave, o varias llaves simples, o varias llaves compuestas disjuntas, si está en la 3FN también está en la BCFN. La situación en que la tabla está en la 3FN y no en la BCFN solo se puede dar si tiene varias llaves y estas son compuestas con intersección no vacía.

Un buen diseño relacional es aquel en el que todas las tablas están en la BCFN.

*Ejercicio:*

Suponga una tabla T con atributos A, D, P, J y las siguientes dependencias funcionales:

P, A 🡪 D

D, A 🡪 J

¿Cuál es la llave de esa tabla?. ¿En qué forma normal se encuentra?