

Disseny lògic de bases de dades

Xavier Burgués Illa

2. DISSENY LÒGIC: Transformació del model conceptual al model relacional

L'etapa de disseny lògic consisteix a obtenir un esquema lògic a partir de l'esquema conceptual generat en l'etapa anterior. L'esquema lògic és depenent del tipus de base de dades que haurem escollit, però serà independent de la implementació concreta del sistema gestor de bases de dades (SGBD).

Com ja hem dit anteriorment, en aquest mòdul ens centrarem en el cas de convertir un esquema conceptual expressat en un diagrama de classes UML a un esquema lògic per a un tipus de base de dades relacional.

A continuació, tot i que es pressuposa que l'estudiant coneix el model relacional i té coneixements bàsics de SQL, es fa esment dels conceptes més bàsics del model relacional i de com es representen en llenguatge SQL, per aquells elements que cal considerar per a fer la transformació de model conceptual al model lògic.

2.1 Conceptes previs del model relacional.

El model relacional representa la informació en base a un conjunt de relacions. Una **relació** es defineix com un conjunt d'**atributs**, cadascun amb un **domini** concret (el domini és el conjunt de valors que es poden assignar a l'atribut), essent un d'ells la clau primària. A aquesta descripció se l'anomena, **esquema d'una relació**.

Al conjunt de tots els esquemes de relació que descriuen la base de dades se l'anomena, **esquema de la base de dades**.

Donat l'esquema de la relació, podem crear **tuples** que conformen l'extensió de la relació; cada tupla dóna un valor del domini corresponent a cada atribut. Per exemple, podem definir una relació *Obra* que tingui un esquema amb dos atributs: un atribut *nom*, el domini del qual són les cadenes de caràcters, i un atribut *anyComp*, el domini del qual són els nombres. Per exemple, l'extensió de la relació *Obra* pot contenir les tuples <"Concert per a piano núm 3 de Mozart", 1779> i <"Parsifal", 1857>.

El model permet, a més, expressar restriccions que limiten els valors o combinacions de valors que poden prendre els atributs. Les més importants són:

- **Clau candidata.** Un atribut o grup d'atributs formen una clau candidata de la relació quan no hi pot haver dues tuples amb el mateix valor en aquell atribut o grup d'atributs. A més, no és possible assignar valor nul a aquests atributs. Per exemple, en una relació d'orquestres on cada tupla conté les dades d'una orquestra, l'atribut que correspon al número d'identificació fiscal es pot declarar com a clau candidata. És a dir, una clau candidata serveix per identificar cadascuna de les tuples de forma unívoca dins d'una relació.
- **Clau primària.** D'entre les claus candidates, se'n tria una que serà la que farem servir habitualment per identificar de forma unívoca una tupla de la relació. En llenguatge SQL es fa servir la restricció PRIMARY KEY per especificar la clau primària d'una relació.
- **Clau alternativa.** Cadascuna de les claus candidates que no és la clau primària rep el nom de clau alternativa. Seguint amb l'exemple anterior sobre la relació

d'orquestrres, si afegim un atribut amb el nom i cada orquestra té un nom diferent, aquest nom també pot ser clau candidata i, com que l'altra és la primària, aquesta seria una clau alternativa. En llenguatge SQL es fa servir la clàusula UNIQUE per especificar una clau alternativa en una relació.

- **Clau forana.** Es pot especificar que un atribut o conjunt d'atributs d'una relació R1 formen una clau forana que referencia una relació R2 de l'esquema a través d'una clau candidata de R2. Aquesta clau candidata de R2 haurà d'estar formada per un conjunt d'atributs que es corresponen un a un amb els de la clau forana de R1. La declaració de clau forana implica que per a cada tupla t de R1, hi ha d'haver una tupla de R2 que té en els atributs de la clau candidata referenciada els mateixos valors que la tupla t té en els atributs de la clau forana. Alternativament, t pot tenir valors nuls als atributs de la clau forana. Per exemple, si hem definit les relacions *Obra* (amb un atribut *nomObra* i un atribut *comp*) i *Compositor* (amb un atribut *nomComp* - que és clau primària - i un atribut *anyNaix*) i diem que l'atribut *comp* d'*Obra* és clau forana que referencia *Compositor* a través de *nomComp*, per a cada obra que no tingui el valor nul a *comp*, ha d'existir un compositor amb aquest valor a l'atribut *nomComp*. En llenguatge SQL la restricció FOREIGN KEY permet especificar la clau forana d'una relació.
- **Valors nuls.** Tots aquells atributs que sempre han de tenir un valor concret es diu que no admeten valors nuls. En llenguatge SQL s'expressa mitjançant la restricció NOT NULL aplicada a la columna en qüestió.
- **Comprovació d'una condició.** És una restricció que verifica que el valor d'un o més atributs satisfà una expressió booleana que s'especifica en la declaració de la restricció. Per exemple, si la relació d'orquestrres que hem fet servir anteriorment té un atribut numèric anomenat *nombreMúsics* i volem que totes les orquestrres presents a la relació tinguin 30 músics o més, establim la restricció *Check(nombreMúsics >= 30)*. En llenguatge SQL fem servir la restricció CHECK seguida de l'expressió a satisfer.

Per claredat i concisió, expressarem un esquema del model relacional mitjançant una notació simplificada del llenguatge SQL standard:

- Les relacions les denotem a partir del nom seguit de la llista d'atributs entre parèntesis i separats per comes.
- Les claus primàries les denotem subratllant els atributs que les formen.
- Les claus alternatives les denotem subratllant amb línies discontinües els atributs que les formen.
- Les claus foranes les denotem com a fletxes que tenen l'origen al conjunt d'atributs que les formen i destinació al conjunt d'atributs que formen la clau referenciada. Aquesta notació és diferent de la notació utilitzada en altres textos sobre disseny de bases de dades, però és interessant ja que mostra gràficament i de manera més entenedora la transformació de l'esquema conceptual al model relacional. En cas de tenir un gran nombre de relacions pot


esdevenir confosa perquè pot implicar un gran nombre de fletxes que es creuen entre elles. En aquestes ocasions serà millor fer servir una notació textual que indiqui, en cada relació, quines claus foranes conté i a quina relació referencia cada clau forana.

- Farem servir el tipus de lletra **negreta** en els noms d'atribut que volem declarar NOT NULL.
- Farem servir el tipus de lletra *cursiva* en els noms d'atribut que volem declarar UNIQUE.

Per exemple, a continuació mostrem com expressar un model amb dues relacions, una d'obres i una de compositors, on les obres s'identifiquen per un codi de catalogació i contenen un nom popular que es pot repetir en compositors diferents, però no en obres d'un mateix compositor. Els compositors s'identifiquen pel seu nom i és obligatori assignar valor a l'atribut de data de naixement (*dataN*).

Obra(codi, nomPopular, autor, data)

Compositor(nom, **dataN**, dataD)



2.2 Impacte de l'ús dels valors nuls.

Per començar, farem una breu reflexió sobre els valors nuls, que es van incorporar al model relacional des d'un primer moment per poder expressar que una dada és desconeguda o que no és aplicable.

És important conèixer l'impacte que pot tenir l'existència de **valors nuls** perquè en fer la traducció de l'esquema conceptual a l'esquema relacional poden sorgir atributs, que els tipus d'entitat de l'esquema conceptual no tenien i que admeten valors nuls.

Recordem que els valors nuls constitueixen un mecanisme que soluciona el problema de representació de dades desconegudes o que no es poden aplicar. En general, no podem evitar l'existència de valors nuls.

Un mal ús dels valors nuls pot causar problemes, bàsicament, de dos tipus: en l'eficiència i en la construcció de consultes que manipulen atributs que contenen valors nuls.

El problema d'eficiència es deriva de l'accés a files que contenen columnes amb valors nuls que es podrien haver evitat amb un disseny diferent de la base de dades. L'exemple següent ho il·lustra:

Obra(codi, autor, nom, nTenors, nBaixos, nContralts, nSopranos, nBarítons)

```
SELECT * FROM Obra WHERE nTenors >= 1
```

La relació *Obra* ens diu, per a cada obra, quants tenors, baixos, contralts, sopranos i barítons calen per interpretar-la (les columnes amb nom que comença per *n* tenen aquesta informació). Si considerem que hi haurà obres sense intervenció de veu,

aquestes obres tindran valors nuls en les columnes que comencen per *n*, ja que aquesta informació no es pot aplicar a obres instrumentals. La consulta de l'exemple, que pretén localitzar les obres on hi ha un o més tenors, haurà d'accedir a totes les obres instrumentals, i lògicament no retornarà cap d'aquestes tuples com a resultat de la consulta. Podem dissenyar un model relacional que eviti l'ús de valors nuls i també l'accés innecessari a tuples. Per exemple, separant les obres en dues relacions: una d'obres amb intervenció de veus i una altra d'obres sense intervenció de veus. Amb aquesta idea obtenim el model relacional següent:

```
ObraInstr(codi, autor, nom)
ObraCoral(codi, autor, nom, nTenors, nBaixos, nContralts, nSopranos,
nBarítons)
```

Amb aquest model relacional, la consulta ara la farem d'aquesta manera:

```
SELECT * FROM ObraCoral WHERE nTenors >= 1
```

La diferència d'accessos entre una alternativa i l'altra dependrà de la proporció d'obres corals respecte del nombre total d'obres. Com menys obres corals (i, per tant, més tuples amb valors nuls a la primera alternativa), més diferència hi haurà.

Així, doncs, quan hi ha columnes per als quals una proporció gran de les files poden tenir-hi valor nul, cal analitzar si les consultes poden ser penalitzades i, si s'escau, cal canviar el disseny, per eliminar la presència de valors nuls.

Pel que fa a la correcció de les consultes que involucren atributs que poden prendre valor nul, hem d'estar atents per assegurar-nos que la consulta retorna el resultat correcte, tant en el cas d'existir tuples amb valor nul, com en el cas contrari. Considerem el cas següent, on tant la relació de directors com la de compositors tenen una columna que indica de quin país és el director o compositor. Si volem obtenir els directors que són d'un país on no hi ha nascut cap compositor, podríem pensar en les dues consultes que es mostren a continuació:

```
Director(nom, país)
Compositor(nom, anyNaix, anyDef, país)
```

```
SELECT * FROM Director
WHERE país NOT IN (SELECT país FROM Compositor)
```

```
SELECT * FROM Director d
WHERE NOT EXISTS
(SELECT * FROM Compositor c WHERE d.país = c.país)
```

Les dues consultes retornen el mateix resultat sempre que la columna país no tingui valors nuls. Si un director té el valor nul a la columna *país*, la primera consulta no l'incorpora en el resultat i en canvi la segona sí.

Aquest és només un exemple per il·lustrar l'impacte que pot tenir l'existència de valors nuls en les consultes. Caldrà doncs tenir present aquest fet en totes les consultes que involucren els valors nuls en la condició de selecció, atributs d'agrupació (*GROUP BY*), funcions d'agregació (*MAX*, *SUM*, ...), etc.

Després dels preliminars anem a veure com transformar cadascun dels elements del model conceptual, expressats en llenguatge UML, al model lògic relacional.

2.3 Classes

El primer element del model conceptual que transformarem serà el concepte de classe.

Una classe de l'esquema conceptual es transforma, en general, en una relació del model relacional. Cada **atribut** de la classe esdevindrà una columna de la relació.

Suposem la classe de la figura 2.3.1:

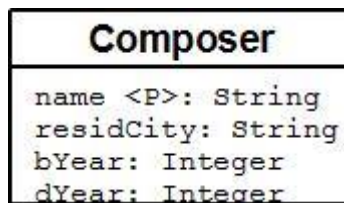


Fig. 2.3.1.

Veiem que aquesta classe té els atributs nom (*name*), que és clau primària, ciutat de residència (*residCity*) que és una cadena de caràcters, any de naixement (*bYear*) i any de defunció (*dYear*) que són enters. Aquesta classe es representa en el model lògic com la relació que es mostra a continuació:

Composer(name, residCity, bYear, dYear)

2.3.1 Atributs multivaluats

La transformació dels atributs multivaluats requereix una anàlisi més detallada. El model relacional no suporta directament aquesta possibilitat, però hi ha dues maneres de representar atributs multivaluats:

- Per columnes. La representació per columnes consisteix en definir en l'esquema relacional tantes columnes com el nombre màxim de valors pugui prendre l'atribut multivaluat de l'esquema conceptual. Aquesta representació requereix conèixer el nombre màxim de valors, informació que no sempre està definida.
- Per files. Aquesta segona representació consisteix a representar cada valor d'un atribut multivaluat com una fila o tupla d'una nova relació en l'esquema relacional.

Posem, per exemple, que la classe *Composer* ara té, en comptes de l'atribut *residCity*, un atribut *residCities* multivaluat tal com es mostra a la figura 2.3.2.

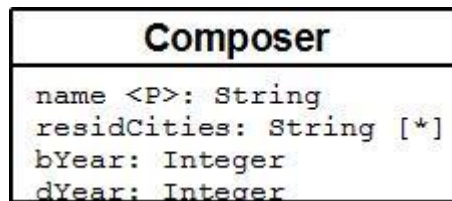


Fig. 2.3.2

Suposant que cada compositor pot tenir com a màxim cinc ciutats de residència, podem representar aquest atribut multivaluat mitjançant dos possibles models lògics:

1) Per columnes:

Composer(name, bYear, dYear, city1, city2, city3, city4, city5)

2) Per files:

Composer(name, bYear, dYear)

CitiesComp(comp, n, city)

Fixem-nos que, com hem dit, per fer la representació per columnes necessitem saber el nombre màxim de ciutats en què pot haver viscut un compositor, i que per cada compositor deixarem amb valor nul els atributs de ciutat que no siguin necessaris. Si coneixem aquest nombre màxim i la majoria de compositors deixen pocs o cap valor nul, aquesta pot ser una bona representació. En altre cas, probablement serà millor la representació per files. També cal tenir present que en aquesta segona representació caldrà efectuar operacions de combinació (*Join*) per recuperar les ciutats d'un compositor, cosa que no caldrà fer si s'utilitza la primera representació.

2.4 Associacions

Per a fer la transformació de les associacions al model relacional cal fixar-se en: l'aritat o grau i les multiplicitats. Respecte del grau, distingirem entre grau 2 o més i respecte a la multiplicitat distingirem si el mínim és 0 o més i si el màxim és 1 o més. També estudiarem els casos particulars d'associacions reflexives i de composicions.

Tota associació es pot representar amb una nova relació que té com a clau primària la concatenació de claus primàries de les relacions que representen les classes que participen en l'associació. A més, aquesta nova relació tindrà una clau forana per cada classe de l'associació.

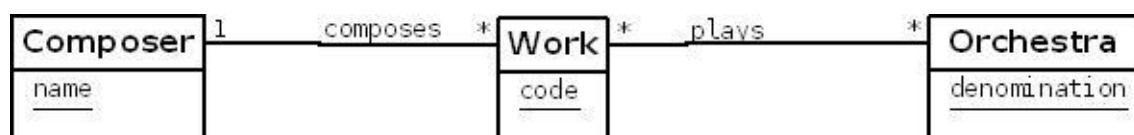


Fig. 2.4.1. Un esquema amb tipus de relacions binàries

Per exemple, l'associació interpreta (*plays*) de la figura 2.4.1 es pot transformar al model relacional de la manera següent:

